

# *Ab Initio* Study of the Si(111)-(7×7) Surface Reconstruction: A Challenge for Massively Parallel Computation

by

Karl Daniel Brommer

B.S., Cornell University (1981)

S.M., Massachusetts Institute of Technology (1988)

Submitted to the Department of Physics  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993



©Massachusetts Institute of Technology, 1993

Signature of Author .....

Department of Physics  
May 5, 1993

Certified by ....................

John D. Joannopoulos  
Professor of Physics  
Thesis Supervisor

Accepted by ...............

George F. Koster  
Chairman, Departmental Committee on Graduate Students

1 ARCHIVES

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

[JUN 02 1993

# ***Ab Initio* Study of the Si(111)-(7×7) Surface Reconstruction: A Challenge for Massively Parallel Computation**

by

Karl Daniel Brommer

Submitted to the Department of Physics  
on May 5, 1993, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

## **Abstract**

This thesis presents the first *ab initio* calculation of the Si(111)-(7×7) surface reconstruction, perhaps the most complex and widely studied surface of a solid. The large number of atoms in the unit cell has up to now defied any complete and realistic treatment of its properties. In this thesis, we exploit the power of massively parallel computation to investigate the surface reconstruction with a supercell geometry containing 700 effective atoms. These calculations predict the fully relaxed atomic geometry of this system; allow construction of theoretical STM images as a function of bias voltages; and predict the energy difference between the (7×7) and (2×1) reconstructions. The diversity of dangling bond sites on the (7×7) surface provides an optimal system for investigating chemical reactivity. A detailed study of the electronic surface states is presented, showing that the interpretation of the surface chemical reactivity in terms of newly developed theories of local softness is consistent with chemisorption experiments. We conclude with predictions of results for surface reactions involving a large variety of atoms and molecules. The method of computing electronic structure on a massively parallel computer is fully described, including a discussion of how the calculations would be improved through implementation on a more modern parallel computer. The results demonstrate that the state of the art in *ab initio* quantum-mechanical computation of electronic structure has been raised to a new echelon as the study of systems involving thousands of atoms is now possible.

Thesis Supervisor: John D. Joannopoulos  
Title: Professor of Physics

## Acknowledgments

Many people should be thanked for the success of the research described in this report, more than will fit in this short space. My thesis advisor John Joannopoulos provided world-class research opportunities and patient guidance while I struggled to take advantage of them. John Negele introduced me to stochastic computation and the Connection Machine, both of which improved my career. The third member of my committee, Gerry Sussman introduced me to LISP and the recursive solution of logic problems, two of the most powerful tools I now have in my kit.

Brond Larson of Thinking Machines was involved in many of the hands-on activities related to the Connection Machine. Among many members of the Joannopoulos group, Tomás Arias, Kyeong-jae Cho, Arnaldo dal Pino, Marcelo Galván, Bob Meade, Mark Needels, Andrew Rappe and Jing Wang gave valuable advice and encouragement. Jim Reardon and Dave Philimore at Thinking Machines helped with the Connection Machine. My supervisors at Lockheed Sanders, especially Henry Mullaney, Preston Kohn, Jack McCarron, Walt Zandi, Jim Queenan and Milt Janosky have supported these studies at MIT.

The work described in this report was directly or indirectly funded by DARPA, NADC, RADC, ONR, NRL and various other government agencies. An NSF fellowship funded full-time studies at MIT from 1981 to 1983. Lockheed Sanders funded tuition and indirect costs. The Pittsburgh and Los Alamos Supercomputer Centers provided computer time and technical assistance. Without the informal hands-on spirit at the Pittsburgh Supercomputer Center, we never would have been the first team to demonstrate the power of parallel processing in electronic structure calculations.

Every federal taxpayer in the United States of America ultimately paid for this research. I owe them more than an expression of gratitude for the training they provided. I'm trying as hard as I can to do science that creates jobs, a better way of life for citizens of the United States and a good return on their investment.

My children, Dieter and Tracey, contributed more to this project than anyone could reasonably ask. They sacrificed countless, irreplaceable hours with their father while he chased his selfish dream. My wife and best friend Connie contributed still more. Without her constant support and encouragement, I would not be writing this thesis. When she was eight months pregnant with Dieter, I had to tell her that the research topic I was pursuing had reached a dead end and that I would have to start over on something else. Even then, she never hesitated in encouraging me not to give up. She is truly a blessing from heaven and I love her with all of my heart.

*This thesis is dedicated to Dieter, Tracey and Connie  
with love and thanks for living with me  
through the entire story of the Silicon - (7×7).*

*...and you will know the truth,  
and the truth will make you free.*

John 8:32



# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
<b>2</b>	<b>Parallelism in Scientific Hardware and Software: Real and Apparent Concurrency</b>	<b>17</b>
2.1	Concurrent Hardware . . . . .	18
2.2	Software . . . . .	37
2.3	Massively Parallel Hardware: The Connection Machine CM-2 . . . . .	47
<b>3</b>	<b>A Massively Parallel Implementation of the Car-Parrinello Algorithm</b>	<b>51</b>
3.1	Algorithm Description . . . . .	53
3.2	Parallel Implementation of Car-Parrinello Algorithm . . . . .	60
3.3	Performance . . . . .	78
<b>4</b>	<b>Application to the Si(111)-(7 × 7) Surface Reconstruction</b>	<b>85</b>
4.1	DAS Model . . . . .	87
4.2	Method of Calculation . . . . .	87
4.3	Surface Energy . . . . .	90
4.4	Ionic Coordinates . . . . .	90

4.5	Overview of Electronic Surface States . . . . .	96
<b>5</b>	<b>Surface Reactivity of the Si(111)-(7×7) Reconstruction: A Density Functional-Theoretic Analysis</b>	<b>107</b>
5.1	Local Properties of Electronic Surface States . . . . .	109
5.2	Local Softness and Surface Reactivity . . . . .	116
5.3	Chemical Reactivity of Si(111)-(7×7) - Local Softness and Charge Capacity	122
5.4	Conclusions . . . . .	130
<b>6</b>	<b>A New Quantum Monte Carlo Method for Fermions</b>	<b>133</b>
6.1	Conventional Path Integral Monte Carlo . . . . .	135
6.2	Generalized Markov Process . . . . .	140
6.3	Discussion . . . . .	149
<b>7</b>	<b>Conclusion</b>	<b>153</b>

# List of Figures

2.1	Integrated circuit complexity vs. time. . . . .	20
2.2	Parallelism within functional units. . . . .	21
2.3	A process $P$ in unsegmented form and pipelined form. . . . .	22
2.4	Pipelined concurrency within a floating point adder. . . . .	23
2.5	Partition of a uniprocessor into a controller, memory and adder. . . . .	24
2.6	The Convex C120 functional block diagram. . . . .	25
2.7	Matrix multiplication program in a serial programming language. . . . .	26
2.8	SIMD architecture. . . . .	28
2.9	Illiad IV functional block diagram. . . . .	30
2.10	Sonar application graph for MIMD architecture. . . . .	34
2.11	Allocation of partitioned modules on a MIMD computer. . . . .	35
2.12	Loss due to contention on a shared bus architecture. . . . .	35
2.13	Simple hardware model suggested by APL functional forms. . . . .	43
2.14	Connection machine CM-2 programmer's model . . . . .	48
2.15	Connection machine CM-2 data processing node . . . . .	49
3.1	Car-Parrinello algorithm summary. . . . .	61

3.2	Memory utilization for the Si(111)- $7 \times 7$ calculation with 896 bands at a 8 Ry cutoff energy. . . . .	66
3.3	Distribution of the wavefunction across the processing array. . . . .	68
3.4	Distribution of the charge density across the processing array. . . . .	69
3.5	Algorithm performance for a 8 Ry Si(111)- $7 \times 7$ calculation on the CM-2. .	79
3.6	Allocation of processing load and processing time among the major computational functions. . . . .	80
3.7	Processing time and processing load vs. number of bulk Si atoms at 12 Ry cutoff energy. . . . .	81
3.8	Processing rate vs. $\log_2(N)$ , where $N$ = number of bulk Si atoms. . . . .	82
3.9	Processing time and processing load vs. cutoff energy for 256 bulk Si atoms.	83
3.10	Processing rate vs. cutoff energy for 256 bulk Si atoms. . . . .	83
4.1	The DAS model for the Si(111)-( $7 \times 7$ ) surface reconstruction. . . . .	88
4.2	Vertical cross section through the plane of the long diagonal of the ( $7 \times 7$ ) unit cell with periodic extensions illustrating the scope of the calculation. .	89
4.3	The surface energies of $\pi$ -bonded ( $2 \times 1$ ) and ( $7 \times 7$ ) reconstructions vs. plane-wave cutoff energy computed for four-bilayer slabs at equivalent k-points. .	91
4.4	Labels for various vertical displacements of interest. . . . .	93
4.5	STM images of the Si(111)-( $7 \times 7$ ) surface reconstruction. . . . .	97
4.6	Theoretical STM depth vs position through the long diagonal. . . . .	99
4.7	Difference in STM height for faulted and unfaulted corner adatoms vs. bias voltage. . . . .	100
4.8	Occupied electronic states within 0.2 V of the Fermi energy through the long diagonal plane. . . . .	101
4.9	Occupied electronic states between 0.9 and 0.7 V below the Fermi energy. .	101
4.10	Occupied electronic states between 1.7 and 1.6 V below the Fermi energy.	102

4.11	Surface contours near the corner hole comparing the three occupied surface states. . . . .	103
4.12	Contours comparing unoccupied surface states. . . . .	105
5.1	Si(111)-(7×7) surface dangling bond sites. . . . .	109
5.2	Contour plots of the local density of states for several values of energy in a plane perpendicular to the surface along the longest diagonal. . . . .	111
5.3	Density of states at different dangling bonds averaged over faulted and unfaulted halves of the unit cell. . . . .	113
5.4	Local density of states for corner and center adatoms averaged over faulted and unfaulted halves of the unit cell. . . . .	115
5.5	Differences between faulted and unfaulted states:(a) faulted and unfaulted corner adatoms; (b) unfaulted and faulted center adatoms; (c) faulted and unfaulted rest atoms . . . . .	115
5.6	Schematic illustration of the interaction of the electronic system with a charge reservoir. . . . .	121
5.7	Schematic illustration of the chemical potential as a function of the number of electrons. . . . .	121
5.8	Local softness for a plane parallel to the surface showing hard and soft channels. . . . .	124
6.1	Sketch of the branched random walk which samples the ground state distribution for a particle in the potential $V(x)$ . . . . .	137
6.2	Sketch of sequence of configurations obtained for the first odd state of a harmonic oscillator. . . . .	140
6.3	Self-consistent solutions $\psi^-(x)$ to the coupled equations for the first odd state of a harmonic oscillator. . . . .	143
6.4	Distribution of points sampling the positive region of the first odd eigenstate of the two-dimensional harmonic oscillator. . . . .	148

# List of Tables

3.1	Summary of arrays used in the Car-Parrinello implementation. . . . .	58
3.2	Array size abbreviations . . . . .	59
3.3	Array distributions . . . . .	67
3.4	Approximate allocation of processing load among the major computational functions. . . . .	78
4.1	Relaxed atomic positions for the adatom layer and the first three surface layers for the <i>ab initio</i> $7\times 7$ calculation. . . . .	92
4.2	Average surface layer displacements. . . . .	94
4.3	Average vertical ionic displacements. . . . .	95
4.4	Difference in vertical displacements for faulted and unfaulted halves. . . . .	96
5.1	Charge capacity for different atoms on the surface. . . . .	123
5.2	Charge capacity order for different sites on the Si(111)-( $7\times 7$ ) reconstruction. . . . .	123
5.3	Differences in charge capacity between faulted and unfaulted halves . . . . .	123
5.4	Chemical reactivity of the Si(111)-( $7\times 7$ ) reconstruction . . . . .	125
5.5	Global softness and relative electronegativity $\eta$ of reactants . . . . .	127
5.6	Predicted reactivity of the Si(111)-( $7\times 7$ ) reconstruction. . . . .	128

# Chapter 1

## Introduction

The  $(7\times 7)$  reconstruction of Si(111) is perhaps the most complex and widely studied surface of a solid. Since its discovery through low-energy electron diffraction more than thirty years ago, an enormous amount of effort has been expended to elucidate the properties of this important surface. Based on this work, it is now generally accepted that the geometry of the  $(7\times 7)$  reconstruction is described by a dimer-adatom-stacking fault (DAS) model as proposed by Takayanagi *et al.* The complexity of this geometry, however, has defied any complete and realistic theoretical treatment of its properties. The only progress that could be made theoretically was by isolating and modelling bits and pieces of the surface. The only attempt at a complete work has been using an empirical tight binding model to study the Si(111)- $(7\times 7)$  reconstruction in a supercell geometry with 196 atoms. In this thesis, we exploit the power of the state of the art in parallel computation to demonstrate the feasibility of performing *ab initio* calculations with supercells approaching one thousand atoms. Specifically, we have performed the first *ab initio* calculation of the Si(111)- $(7\times 7)$  reconstruction using a supercell geometry with 700 effective atoms.

By applying the tool of massively parallel computation to electronic structure calculations, it is possible to investigate many aspects of the Si(111)-(7×7) reconstruction. The energy per surface atom is a crucial starting point. It is known experimentally that the (111) surface of silicon cleaves into a metastable (2×1) phase. It is only through annealing at 600 degrees Kelvin that the surface reconstructs into the complicated (7×7) phase. The energy difference per surface atom between these two phases must be very small. Computing this energy difference is a good test of the accuracy of the pseudopotential theory and its computational implementation.

Having verified the fundamental accuracy of computation, it is possible to use the computed wave function to determine forces on atoms. At the time the investigation described in this thesis was begun, the general structure of the (7×7) reconstruction was known, but uncertainties in the position of any atom in the system were on the order of 0.5 Angstroms. We began our investigation of the system by placing atoms in approximately correct positions guessed from the lattice geometry of bulk silicon. By continuing to iteratively compute forces on atoms, relax atoms in response to forces and update the wave function until equilibrium is achieved, the ionic positions were determined with roughly an order of magnitude more accuracy,  $\sim .05\text{\AA}$ .

While accurate knowledge of the coordinates of the atoms near the silicon surface is valuable for comparing various experimental and empirical techniques previously applied to the problem, it is crucial for resolving an unsolved controversy related to the interaction of the Si(111) surface with Scanning Tunneling Microscopy (STM) probes. STM is a recently discovered technique for imaging individual surface atoms by measuring the atomic-scale deflection of an electrically biased metal probe as it is moved across the surface of a spec-



imen. The Si(111) surface is probably the most well-known STM image because of the complexity of its structure and the esthetically pleasing pattern arising from the large horizontal separation of adatoms on the surface, roughly twice the normal separation of atoms in bulk silicon. What is unexplained about these images, however, is why one side of the  $(7\times 7)$  unit cell appears significantly higher than the other. Since STM signals are sensitive both to heights of surface atoms and the amount of electronic charge on surface atoms, it is difficult to experimentally separate the two contributions. Accurate knowledge of the surface electronic wave function is necessary to determine whether the Si(111) surface is structurally asymmetric or whether a mass redistribution of electronic charge causes the effect.

This thesis will address issues related to the electronic surface states in detail, including the STM asymmetry about the short diagonal of the unit cell. By comparison to STM experiment, it is possible to characterize the performance of contemporary STM probes. While the horizontal resolution of STM probes is well-established through comparison with X-ray data, the ability of STM to image complicated three-dimensional structures varying on a sub-nanometer length scale is largely unknown because no other experimental technique has this capability to work at this experimentally small length scale. This same length scale is actually large for theory. No first-principles theoretical calculations have been performed at this length scale because of the computational difficulties until now. By comparing theory to experiment for the deep corner hole of the Si(111) surface unit cell, it is possible to determine how accurately STM can characterize three-dimensional sub-nanometer structures. This issue has technological ramifications as many countries are now researching sub-nanometer electronic devices.

While bulk silicon is an electronic semiconductor with a gap at the Fermi energy, a cleaved silicon surface can be metallic, since the broken bonds can contribute free electrons to the surface layer. As a result, silicon surfaces can be highly reactive and prone to interact and bond with any impurities present in the vacuum system used for semiconductor processing or experimental investigation. Within the past five years, it is possible to electronically image impurities on the Si(111) surface using STM. The results of these investigations show that various impurities exhibit strong preferences regarding which of the different dangling bond sites on the Si(111) surface favor chemisorption. This picture is complicated because impurities with different electronegativities and chemical hardnesses prefer different surface bonding sites. Traditional first-order theories of chemisorption do not apply to the silicon surface because each bonding site consists of a silicon atom with the same electronegativity.

The chemical behavior of Si(111) can only be explained through a second-order chemisorption theory which distinguishes the subtle differences between the nineteen closely related yet distinct bonding sites. This thesis presents a calculation of the local softness and charge capacity of each bonding site on the surface. The results will show that charge capacity accurately explains the one-reactant chemisorption experiments performed to date. Our local softness calculations also suggest reinterpretations of multi-reactant chemisorption experiments that have been reported before our theoretical picture emerged. Having demonstrated the accuracy and utility of chemical softness analysis for this system, we conclude with a prediction of how many reactants will behave on silicon surfaces without resorting to expensive experiments or first-principles calculations.

The chapters of this thesis are organized in a logical progression starting from general aspects of parallel computing through a detailed discussion of the specific physical results that

were obtained for the  $(7\times 7)$  system. Chapter 2 presents a survey of concurrent computer technology written from the perspective of an industrial practitioner rather than a computer scientist. It shows what tools are currently available for scientific computation with emphasis on the massively parallel technology used to investigate Si(111). We also discuss how more advanced computer architectures would facilitate the study of more complicated material systems.

Chapter 3 describes how a typical implementation of the Car-Parrinello algorithm used by dozens of electronic structure theorists was modified to run on a massively parallel computer. The details regarding performance and bottlenecks for various sizes and types of material systems are discussed.

Chapter 4 presents the new physical results for the Si(111)- $(7\times 7)$  surface reconstruction. Having obtained the wave function and relaxed ionic coordinates for this system, the surface energy, structural properties and electronic states follow directly. These results are presented and compared with previous theory and experiment whenever previous data exists. New results are presented for the details of electronic states, particularly below the surface adatom layer where experimental investigation is difficult.

The diversity of dangling bond sites on the  $(7\times 7)$  surface provides an optimal system for investigating chemical reactivity. The different reactive sites on this surface are identical to first order in terms of classical electronegativity analysis and yet exhibit distinctly different chemical behavior. A detailed study of the chemical physics is presented in Chapter 5, showing that the interpretation of the surface chemical reactivity in terms of newly developed theories of local softness is consistent with chemisorption experiments. The complexity of the  $(7\times 7)$  surface reconstruction provides a strong test of higher-order terms in expansions

of the chemical potential within the local density approximation. Predictions of surface chemisorption with respect to reactants which have not yet been experimentally tested are presented.

The calculations presented thus far in the thesis were performed within the context of the Local Density Approximation, a one-electron approximation to the true many-body wave function of an electronic system. A brief review of alternative methods for more accurate electronic structure calculations is the subject of chapter 6, including a new stochastic method for computing the wave function of many-fermion systems. The more exact techniques are limited to problems involving two or three electrons. As this thesis shows, it is remarkable that the Local Density Approximation can successfully analyze systems involving thousands of electrons using the one-electron approximation.

This  $7\times 7$  study shows what is now possible with massively parallel computing. The unprecedented complexity of the material system under investigation allowed us to resolve outstanding questions involving the electronic structure of nanometer-scale systems of current technological interest, including chemical, structural and electronic properties as they relate to a variety of experimental probes and theoretical techniques. Popularized accounts of some of our results [1] appeared in *Science* [2] and *Physics World* [3]. But the hardware is already obsolete and the algorithm is obsolescent. Current parallel hardware, including the CM-5 [4], and more advanced electronic structure algorithms [5], including those designed specifically for parallel computation should greatly boost performance. The uniformity and extensibility of massively parallel architectures means that larger machines should be forthcoming, resulting in still more exciting scientific results in the future.

## Chapter 2

# Parallelism in Scientific Hardware and Software: Real and Apparent Concurrency

One of the more subtle questions raised in this report is not related to how one works with large numbers of atoms using *ab initio* quantum-mechanical techniques. Rather, it is why it has taken so long for supercomputers other than vector-pipelined architectures to be useful in high-performance computation. Answering this question requires some understanding of computer architecture including the limitations of contemporary digital circuit technology. Before addressing algorithmic implementation and physical results in subsequent chapters, we'll spend this chapter considering what kind of hardware makes a computer fast and how one uses software to exploit the speed. In addition to providing clues for effective use of existing parallel supercomputers, a knowledge of computer hardware enables scientific programmers to anticipate what sorts of architectures should be available soon for even more advanced calculations.

When designing a faster computer, an architect uses two non-exclusive approaches.

One is to select fast circuit technology so that the computer executes digital operations at a relatively high clock frequency. The advantage of this approach is its relative simplicity in terms of parts count and communication requirements. The disadvantage is that fast components tend to be more expensive, more difficult to cool and less dense in terms of gates per integrated circuit package. As a result, computers using fast circuit technology tend to cost more, dissipate more heat and occupy more space than otherwise equivalent counterparts.

There is a second way to speed up computers even with a fixed operating speed for individual components. This option is parallelism. In the context of computation, parallelism is simply defined as doing more than one thing at a time. There are many different ways of designing parallelism into a computer architecture. Most methods of parallel execution in computers do not involve performing identical operations simultaneously so perhaps a more general word is concurrency.

All computers exploit hardware concurrency to differing extents. In many cases the concurrency is so efficiently designed that users take advantage of it transparently. In the domain of high performance computation, this is rarely the case. Effective use of new and advanced computer architectures requires understanding concurrent hardware and how to use it.

## **2.1 Concurrent Hardware**

Concurrency in computation is not a new idea. It seems to have occurred, in one form or another, to nearly all the pioneers of computing from the earliest times [6]. However

most of the good ideas could not be implemented in the past due to hardware limitations. While this is still a problem, modern integrated circuits offer an unprecedented freedom for implementing advanced computer designs. Figure 2.1 shows that the device density of integrated circuit components has rapidly increased over three decades [7]. However, it is only since about 1975 or 1980 that devices with functionality comparable to an 8-bit microprocessor could be designed into a single integrated circuit package. Many forms of concurrency require complexity well beyond this scale.

The idea of parallelism appeared almost as soon as the first computer hardware [6]. In the 1830's, Charles Babbage designed the Analytical Engine, one of the first computers capable of automatic multistep calculation. Although Babbage never mentions parallelism in his writing, the possibility is mentioned explicitly by Menabrea, one of his colleagues, following a discussion with Babbage. In the Bibliothèque Universelle de Genève, one can read an account of this machine by Menabrea[8]:

D'ailleurs, lorsque l'on devra faire une longue série de calculs identiques, comme ceux qu'exige la formation de tables numériques, on pourra mettre en jeu la machine de manière à donner plusieurs résultats à la fois, ce qui abrégera de beaucoup l'ensemble des opérations.

A modern computer incorporates many levels of functionality where concurrency can be used. We categorize them as follows:

1. within functional units
2. within processing elements

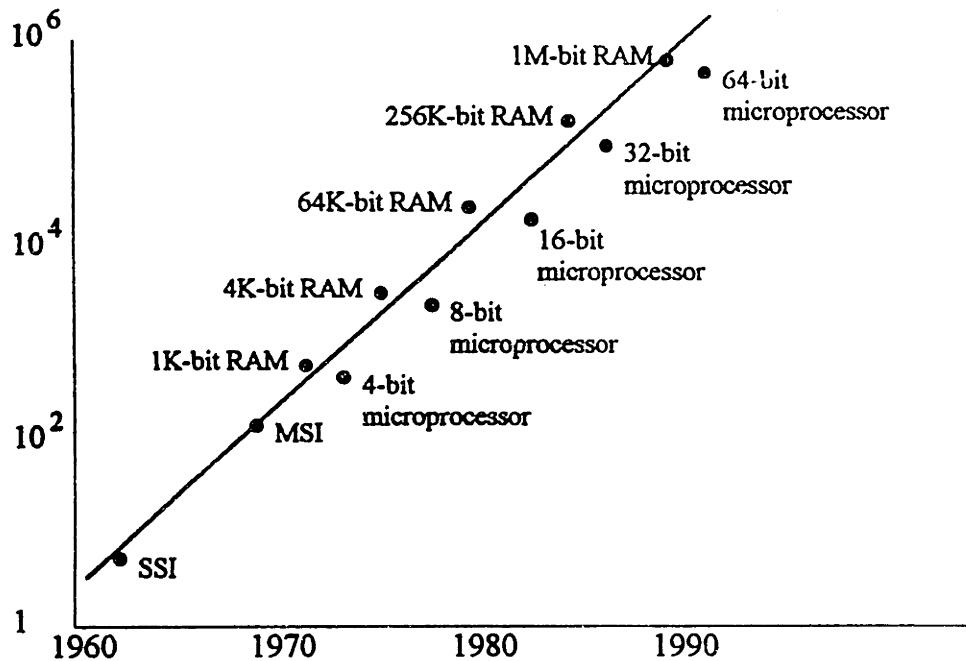


Figure 2.1: Integrated circuit complexity vs. time.

3. within uniprocessors

4. many-processor systems

It is important to understand some of the opportunities for designing concurrency into each level, especially the forms of concurrency used in high-performance scientific computers.

**Within functional units.** Logical and arithmetic operations can be applied to operands in a bit-serial mode or to whole operands concurrently. Figure 2.2 compares circuits using bit-serial and bit-parallel operations [9]. The major difference is that a bit-parallel computer requires wide, generally 16, 32, 48 or 64-bit data paths. Less obvious is the complexity of the carry look-ahead circuitry and pipelining necessary to perform operations ranging from integer addition to floating point multiplication and addition in bit-parallel modes. However, these functions have been designed for decades and are now available as standard IC components or macrocell libraries for VLSI circuit design. The main cost of parallelism



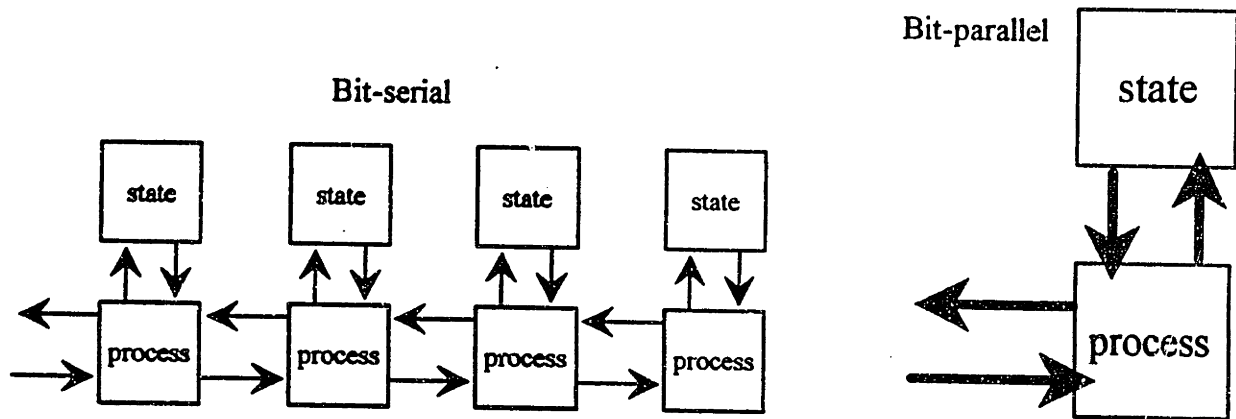


Figure 2.2: Parallelism within functional units.

at the bit level is the increased gate count to apply operations to bit-parallel operands and increased pin count on IC packages to connect the wider data paths between different devices. The main benefit is an acceleration of computer operations by at least a factor of  $b/n$ , where  $b$  is the number of bits in the operand and  $n$  is the number of clock cycles required to apply to operation concurrently to every bit in the operand.

**Within processing elements.** The discussion of concurrency with a single functional unit ended with the statement that the speedup is *at least* a factor of  $b/n$ . A speedup by a factor of  $b$  can be obtained by using concurrency within a single processing element in the form of pipelining. In many computational processes the total process can be partitioned into a number of discrete steps or segments. Figure 2.3 shows schematically the difference between some unsegmented process  $P$  and the same process separated into six sequential segments. There is no reason why the total time  $T$  for an unsegmented process  $P$  should differ from the sum of the separate segment times  $t_1 + t_2 + t_3 + t_4 + t_5 + t_6$ . The idea

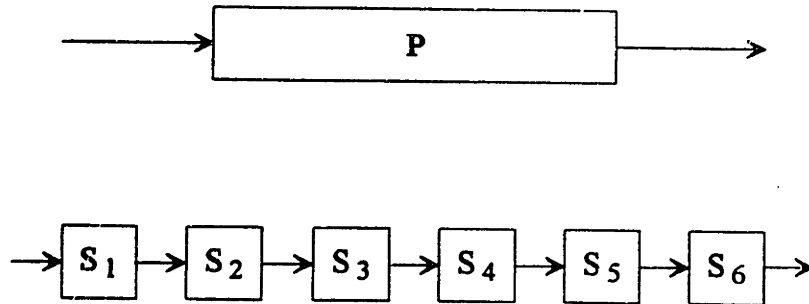


Figure 2.3: A process  $P$  in unsegmented form and pipelined form.

of pipelining is simply that if all the segments  $S$  are implemented by physically separate subunits, then they can operate together, and several processes may proceed concurrently in an overlapped fashion. Figure 2.3 represents thus a processing pipe in which there may be up to six concurrent processes  $P$  at any given instant.

A very important example of pipelined concurrency within a functional unit occurs in a pipelined floating point arithmetic logic unit (ALU). A functional block diagram of a typical ALU is shown in figure 2.4. The concurrent segments break down as follows:

- $S_1$  Move input operands to the input registers
- $S_2$  Subtract the exponents
- $S_3$  Shift the smaller operand's mantissa to align
- $S_4$  Add the aligned mantissas
- $S_5$  Normalize the result
- $S_6$  Move the result to the output register

If two long vectors are to be added together as  $\mathbf{a} = \mathbf{b} + \mathbf{c}$  with this pipelined adder, the timing is as shown in the right column of the figure. When element  $a_j$  is written to

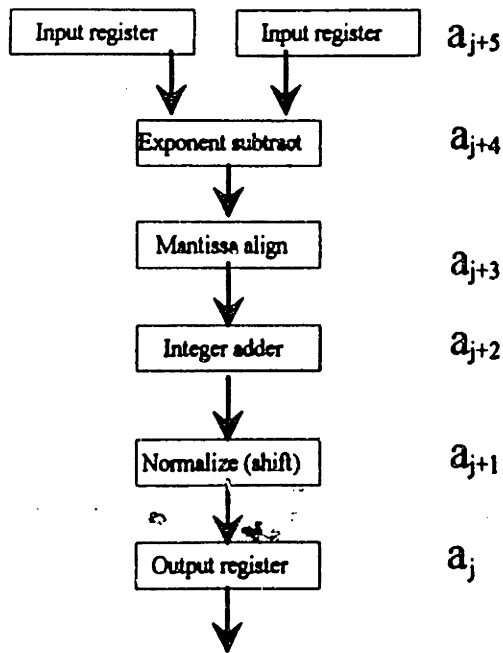


Figure 2.4: Pipelined concurrency within a floating point adder.

the output register, element  $a_{j+5}$  is written to the input register. It takes six clock cycles, the pipeline latency, for each input operand to generate an output from the pipe. However, outputs appear every clock cycle as long as the pipeline is kept full. Whenever the number of elements  $N$  in each vector is much larger than the pipeline latency, this pipelined device speeds up processing by a factor of six compared to a similar non-pipelined functional unit.

Referring back to the example of bit-serial arithmetic, some number  $b$  of microinstruction cycles are required within units like the adder in figure 2.4. It is possible to further speed the processing by pipelining the adder so that up to  $b$  operands are processed concurrently through the ALU microsequence. The cost of this concurrency is a longer pipeline latency and more registers for clocking intermediate adder stages.

**Within uniprocessors.** It is generally possible to find several independent tasks that can be performed concurrently by different functional units within a uniprocessor. There are two important examples of concurrency within uniprocessors. The first is known as

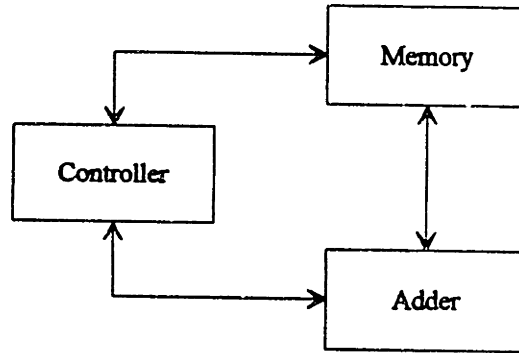


Figure 2.5: Partition of a uniprocessor into a controller, memory and adder.

instruction pre-fetch. Every uniprocessor can be partitioned into the three modules shown in figure 2.5. The controller must fetch instructions from program memory and set up the processing unit to execute the instruction on appropriate data elements stored in memory. During the time while the processing unit is executing instruction  $i_n$ , the controller can fetch an instruction  $i_{n+m}$ , where  $m$  is the length of the instruction pipeline. This technique minimizes the amount of overhead time required to set up the processing unit for execution. Instruction pre-fetch is discussed in great detail in reference [10].

Instruction pre-fetching is particularly important in vector-processing architectures where the complexity of the different functional units may require numerous clock cycles to initialize a given instruction. Long instruction overhead times will otherwise destroy performance during programs with short vector lengths.

Vector-oriented architectures generally include more specialized processing units within the uniprocessor. Consider, for example the functional block diagram of the Convex C120

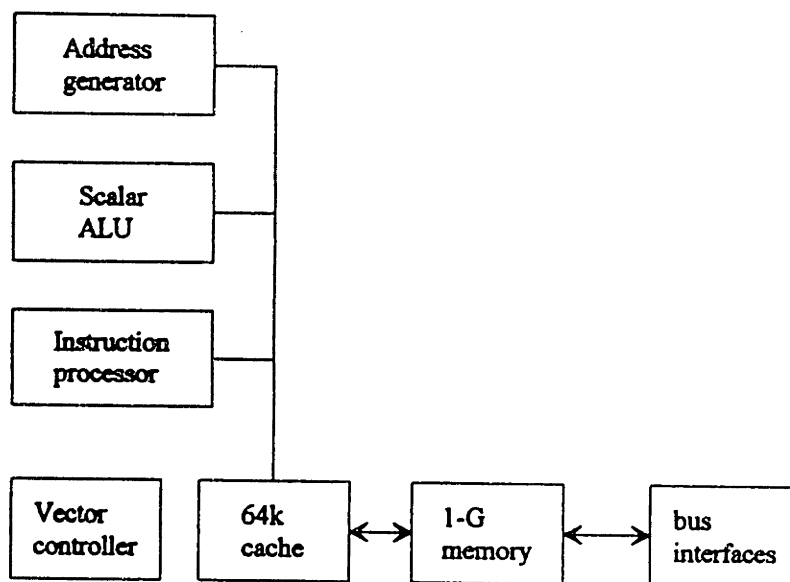


Figure 2.6: The Convex C120 functional block diagram.

supercomputer shown in figure 2.6 taken from reference [11]. The main functional units are for instruction processing, vector-pipelined arithmetic, scalar arithmetic and address translation. The address translation unit computes addresses of each input and output operand for the vector pipeline. Understanding its function is best done by looking at a typical vector operation. A conventional matrix multiplication, for example

$$a_{ij} = \sum_{k=0}^{N-1} b_{jk} c_{ki} \quad (2.1)$$

is written in a scalar programming language such as C in figure 2.7.

The C programming language was chosen because it maps closely into uniprocessor assembly language, providing a detailed description of the hardware operations necessary to multiply matrices. The program ravelles all matrices into one-dimensional arrays to accelerate the address arithmetic, as any good optimizing compiler would try to do. From the large number of lines in the program relative to the single multiply operator in the code, we see that a uniprocessor must serially execute a sequence of instructions to do

```

matrix_multiply(b, c, a, bcol, brow, crow) /* A <- B * C */
float b[], c[], a[]; /* input matrices written as 1-d arrays for speed */
short bcol, brow, crow; /* matrix dimensions */
{
    short i, j, k; /* indices */
    float sum; /* accumulator */
    float *bptr, *ptr; /* pointers used to accelerate address arithmetic */
    float *resetb, *resetc;

    resetb = b; /* initialize pointers to beginning of matrices */
    resetc = c;
    for(i=0; i<bcol; i++) {
        for(k=0; k<crow; k++) {
            bptr = resetb; /* repeat the left matrix B */
            cptr = resetc++; /* move to next column in C */
            sum = 0;

            /* INNER LOOP: if j = B's rowlength, branch */
            for(j=0; j<brow; j++) {
                sum += (*bptr++) * (*cptr); /* arithmetic function */
                cptr += crow; /* increment C pointer to move down column */
            } /* branch back */
            *a++ = sum; /* update result and increment A pointer */
        }
        resetb += brow;
        resetc = c;
    }
}

```

Figure 2.7: Matrix multiplication program in a serial programming language.

the matrix multiplication. The three for loops are compiled into machine instructions to reset, increment and test counters to determine when to branch in each loop. All of the other lines except two perform address arithmetic to fetch and store matrix elements. The `sum=0` line initializes the pipelined ALU by clearing the accumulator. Exactly one line, `sum += (*bptr++)*(*cptr)` does useful floating point arithmetic. Without a separate address arithmetic unit to compute new input and output operand addresses such as `cptr += crow`, the arithmetic unit would constantly alternate between computing floating point data and integer address arithmetic. Since these two operations are functionally independent, it is possible to dedicate a address generation unit to concurrently execute the address arithmetic, with the floating point pipeline allocated exclusively for floating point operations. While the address and floating point units execute concurrently, the control functions such as `for(j=0; j<brow; j++)` are processed in the instruction processor and scalar arithmetic unit. Thus concurrency allows a relatively simple supercomputing architecture to execute multiple address computations to fetch operands, a decrement, test and branch to determine loop limits in the time required for a single floating point operation. This type of concurrency is necessary so that the matrix multiplication is driven by the memory access rate and floating point multiply/accumulate rate rather than the time required to serially execute the control instructions and address arithmetic specified in the C program.

With this description of a simple vector-pipelined architecture, we have now summarized every type of concurrency that has been used in electronic structure calculations prior to the results in this thesis. Reference [12] describes a much more complicated supercomputer system designed along the same types of concurrency, the Cray-1. It is an important example because computers in the Cray or IBM 3090 performance class are generally used for research-level problems in electronic structure. Recently, more advanced levels of parallelism

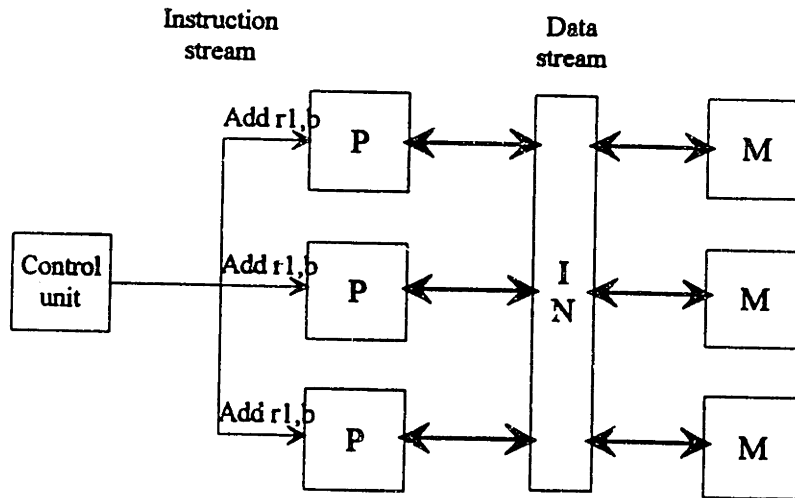


Figure 2.8: SIMD architecture.

have become available to go beyond the performance possible with the vector-pipelined approach.

**Many-processor systems.** This category of concurrency describes systems where a computer contains several processors, often sharing a single memory with some means of intercommunication. There is a great difference between computers where all processors are identical and operate in lockstep and where they differ. A commonly used taxonomy was introduced by Flynn [13]. According to Flynn, many-processor systems are divided into two broad categories, those which concurrently execute the same instruction in every processor on different data (Single-Instruction, Multiple-Data or SIMD) and those which concurrently execute different instructions on different data (Multiple-Instruction, Multiple-Data or MIMD).

The earliest massively parallel computers fall into the SIMD category. A typical SIMD architecture known as a processor array is shown in figure 2.8. SIMD architectures typically



employ a central control unit, multiple processors and an interconnection network for either processor-to-processor or processor-to-memory communications [14].

The first published idea of using regular arrays of processors to obtain high throughput appears to go back to Unger in 1958 [15]. He described a processor array designed for pattern recognition. Each node was a single-bit processor that performed logical xor and and functions between six bits of memory and a single-bit accumulator. Each node required about forty transistors. Each processing node was connected to its four nearest neighbors. This paper is a remarkable prediction of the processor array connectivity of subsequent massively parallel machines, including the Connection Machine CM-2. It also predicts the important role of monolithic digital integrated circuit technology in building a machine with so many identical components.

The Illiac IV was the first processor array to actually be built [16]. It comprised an 8x8 array of 64-bit floating point processors. With a 13 MHz clock, the maximum processing speed was 15 MFLOPS. An Illiac IV block diagram is shown in figure 2.9 [17]. The central control unit interprets instructions as in a conventional computer. The processor array simultaneously executes array-oriented commands whenever they can be scheduled. The processors are each connected to private memory and are interconnected by a communications network. In this system, a single, identical instruction will simultaneously execute on multiple data streams. Processing elements in each array are connected to their four nearest neighbors, the same topology suggested by Unger [15]. The control unit must control the processor array in addition to executing all of the data processing functions found in a standard uniprocessor.

Only one Illiac IV was built. It was tested on a wide variety of applications [18],

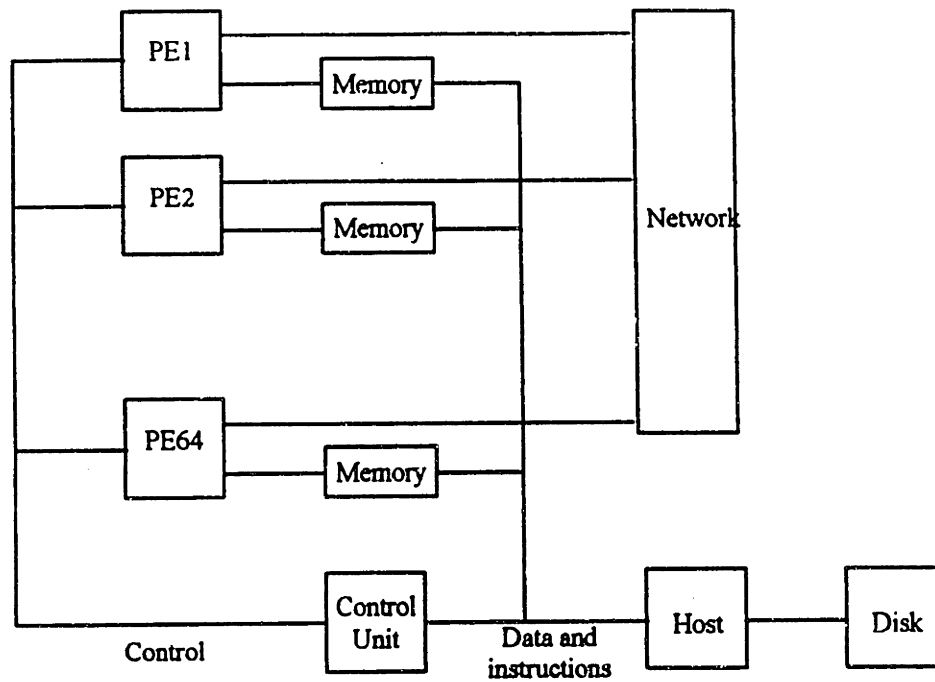


Figure 2.9: Illiac IV functional block diagram.

spawning much research into algorithms for parallel computers. In these respects it laid the groundwork for the current generation of parallel machines, which shares some of the same limitations. For example, the operating system software is similar to Connection Machine CM-2 in that one user attaches to the machine, uses it exclusively and then releases to the next user in the queue.

The main limitations of the architecture are overhead latency time for initialization of the processor array instructions by the control unit and limited inter-processor communication bandwidth. A long setup latency means that the processor will only efficiently execute programs where most of the processing operations apply to long vectors. Otherwise, the execution time is driven by the amount of time spent by the control processor executing data processing parts of the code and the time spent initializing the massively parallel array for relatively short vector instructions.

However, the principal difficulty of the Illiac IV was not the aggregate processor power

available, but that it was difficult to get the required operands in and out of the processing elements fast enough. There is only one channel between the control processor and its 64 slave processors. Transfer of data to all 64 processors must be performed serially over this channel at rates governed by the speed of the serial processor. Each processing element has direct access only to its corresponding 4096-word memory. As long as operations can be performed in the PE memories, concurrency is obtained and the system throughput is relatively high. Whenever processors must communicate, either among each other or to fetch more data from the control processor, the throughput drops to uniprocessor rates. In this sense, the concurrency of the massively parallel architecture is merely apparent, because the memory-processor bottleneck resulted in an unbalanced design.

A typical illustration of the severity of this bottleneck can be found in reference [19], a comparison of a vector-pipelined computer similar to figure 2.6 with the Illiac IV done in 1976. Both processors ran at a similar clock rate, so the major difference was the comparison of massive parallelism with a single vector-pipelined architecture. The test problem was taken from naval underwater surveillance: compute the coherence function [20] between two sonobuoy sensors

$$\gamma_{x,y}^2(\tau, \theta) = \frac{|\sum_{k=1}^N e^{ik\theta} X_k Y_k^*(\tau)|^2}{(\sum_{k=1}^N X_k X_k^*)(\sum_{k=1}^N Y_k(\tau) Y_k^*(\tau))} \quad (2.2)$$

This calculation is more parallel than the typical electronic structure calculation, consisting of dot products and the Fast Fourier transform. It is applied repetitively to data sets differing only in time delay  $\tau$ . Despite its 64 parallel processors, the Illiac IV was actually a factor of five times slower than the vector-pipelined machine. The reason was the memory-processor bottleneck and the significant overhead in the control processor - processor array interface. The 64 parallel processors constitute an apparent concurrency whenever

they sit idle waiting for initialization from the control unit. Moreover, the vector-pipelined computer fit in a single 19-inch equipment rack suitable for installation on an aircraft or submarine, while the Illiac IV occupied an entire room. This example shows the perils of taking concurrency too far and the difficulties in making massively parallel computation live up to its potential.

The second type of concurrency among multiple processors, the MIMD architecture, goes beyond the SIMD approach in complexity in the sense that one may now assign completely different programs to different computers interconnected according to some communications network. A review of various MIMD connectivities may be found in [14]. While the MIMD approach is still new to scientific computation, it has been explored in signal processing for at least a decade.

The programming issue is one of partitioning. The algorithm must be divided into several individual subtasks. The representation of the partitioned algorithm is invariably a graph, usually similar to a signal processing flow graph [21], although oftentimes at a higher level than the multiplies and adds common to many signal flow graphs from the 1970's. The objective is to construct the graph with a sufficient number of nodes to spread across a MIMD architecture, with the nodes chosen to minimize inter-node communication. This technique is commonly applied in the real-time signal processing community. A sonar example is shown in figure 2.10 [22]. In this example, the following subroutines must be executed:

**Prefilter** Hilbert transform data from real to complex

**AGC** - Compensate long term variation in input level

**Octave Filter** - Separate data into octave bands

**Spectrum Analyze** - Separate octave bands into narrow band channels

**Linear Detect** - Compute magnitude of narrow band channels

**Short Term Integrate** - Slow outputs of higher octaves to rate of lowest octave

**Long Term Integrate** - Apply additional smoothing to narrowband detected data

**Noise Mean Estimation** - Establish background level

**Threshold** - Separate signals from background

**Format** - Pack data for transmission to display

In the figure, each module is assigned one or more nodes on the graph, with lines connecting nodes that must transfer data. When a module is assigned to more than one node, the user expects that module to run concurrently on multiple nodes in SIMD fashion. In this example, three nodes execute Input, Prefilter, and AGC in a MIMD pipeline. At this point data is passed to three Octave-Spectrum Analyze-Linear Detect pipelines which form a SIMD node cluster within the overall MIMD architecture. The back-end processing is once again MIMD with one node per processor.

The crucial issues in getting an algorithm to run effectively on a MIMD computer involve balancing processing and minimizing communication. Each node on the graph must perform close to the same amount of computation, since the most time-consuming node will be the gating element on system throughput. Thus algorithms which are partitionable into a large number of graph nodes will run faster than those which are not. However, the number of graph nodes is limited by the communication capacity of the bus network

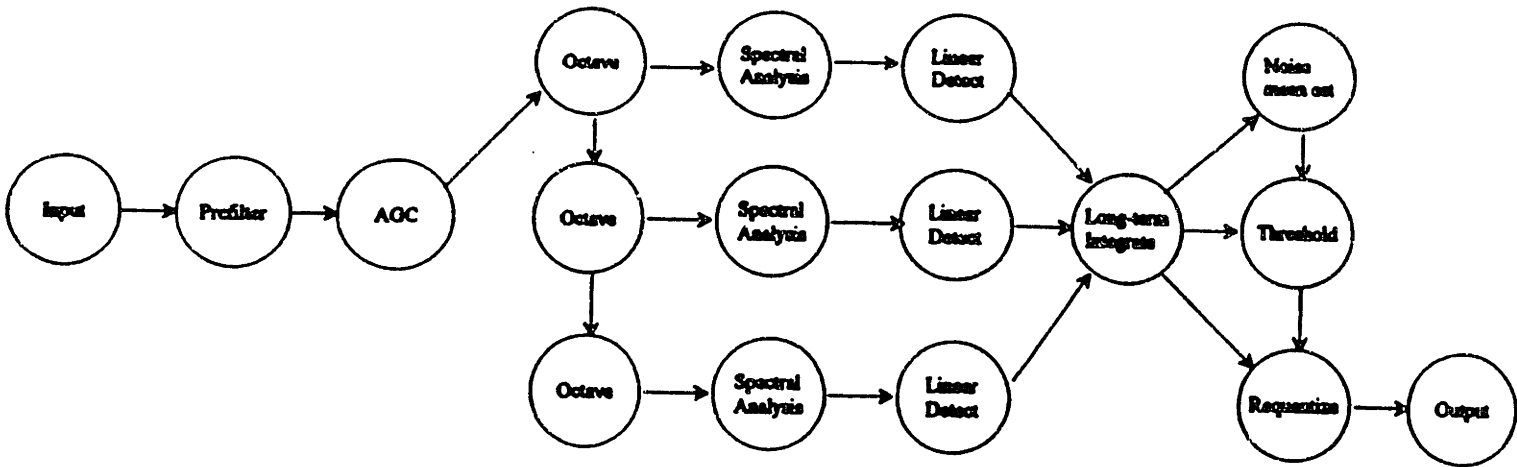


Figure 2.10: Sonar application graph for MIMD architecture.

connecting the processors. It does no good to minimize the amount of computation time in each node if the resulting graph raises communication levels to the point where nodes must sit idle waiting for a bus before communicating to other nodes.

Figure 2.11 shows an example of MIMD hardware partitioned for a real-time signal processing application [23]. The top diagram shows a set of identical uniprocessor nodes communicating over a global bus. The bottom diagram shows how communication is routed among the processors for a specific application in speech recognition. This particular example shows an apparent concurrency in the sense that while the mapping implies that dedicated communication channels exist, the actual communication will be done serially over a single shared bus. Markov calculations by the author[24] have shown that this shared-bus approach works without excessive delays due to bus contention within the parameters shown in figure 2.12.

If communication requirements exceed the shared-bus bandwidth, it is clear how to

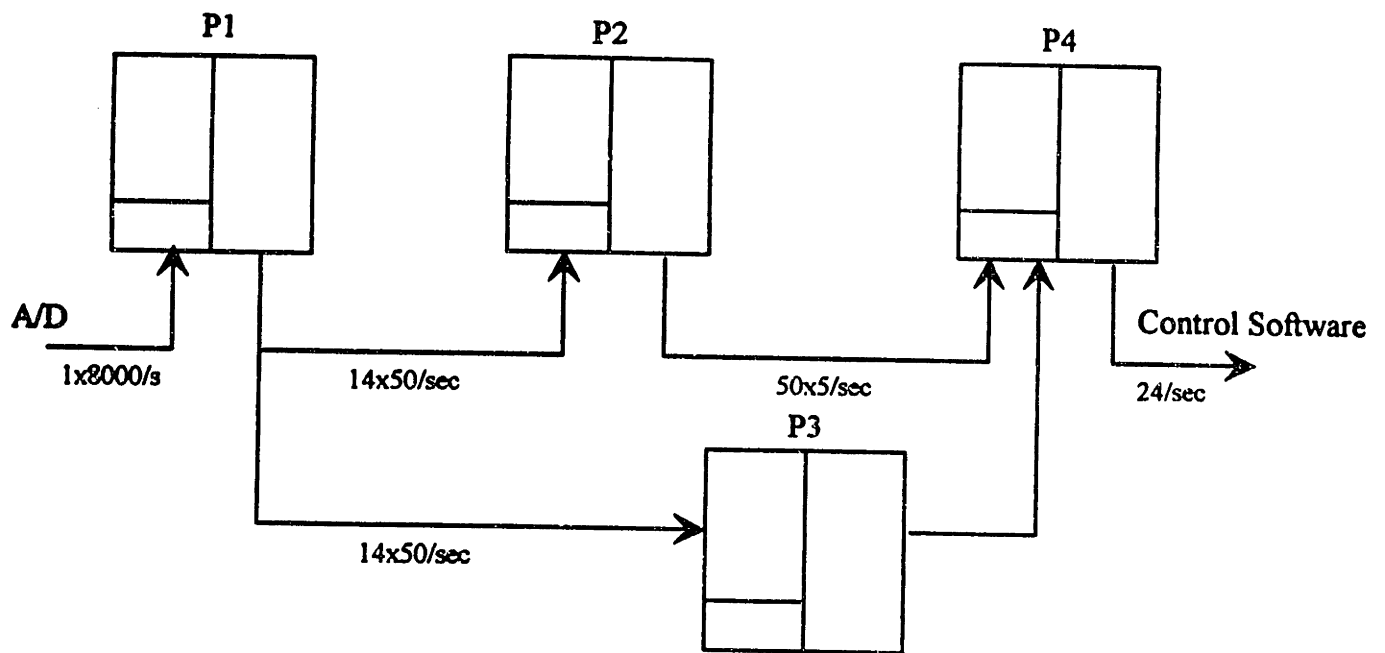


Figure 2.11: Allocation of partitioned modules on a MIMD computer.

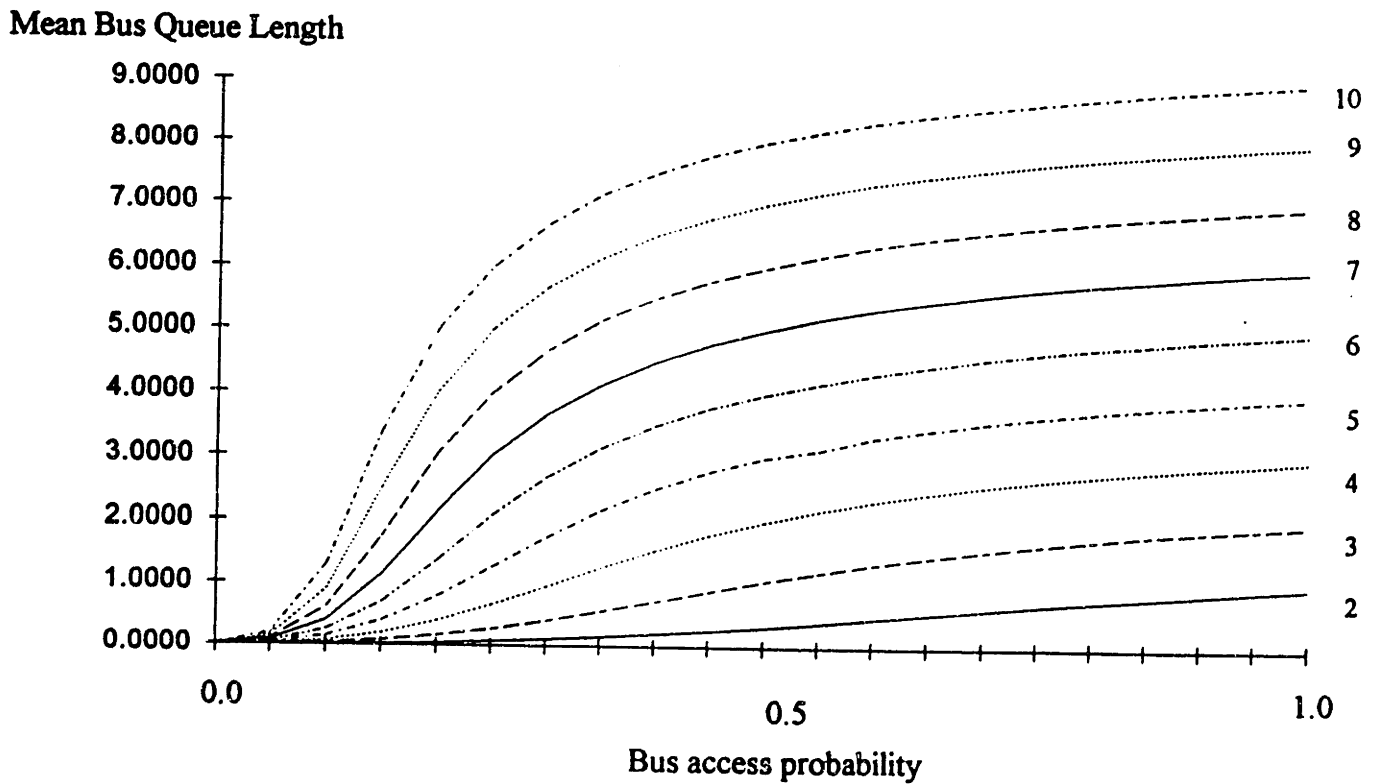


Figure 2.12: Loss due to contention on a shared bus architecture.

regain concurrency in this MIMD example. One would simply use a multi-ported uniprocessing node and allocate direct bus interconnects between the nodes. The cost of this approach is increased complexity due to the additional connectivity as well as less reliability due to vulnerability to single-point failures in the dedicated communication channels. Reliability is an important issue in any large computer system. It is intuitively clear that a MIMD architecture is more reliable than a SIMD architecture, since the MIMD architecture encompasses a heterogeneous pool of computer resources which can be mapped in a variety of ways by different users on different applications, while the SIMD architecture presents the same array of  $N$  fixed, identical processors to every user.

While no MIMD algorithm of any sort has yet been used for an electronic structure calculation, there is one advanced feature of MIMD architectures that may soon be valuable for scientific applications: the ability to insert special-purpose nodes into the heterogeneous architecture. For example, one may wish to install a special node dedicated to high-speed computation of three-dimensional Fourier transforms *in hardware*. Recent advances in silicon compilers [25] make it more cost-effective for users to permanently program hardware. The program is written in a hardware description language. The program is then “compiled” at a VLSI foundry, with the resulting “object code” shipped back in the form of an integrated circuit dedicated to executing the function. Every year brings new improvements in the usability of hardware programming languages and reductions in the costs of the resulting dedicated integrated circuits. It is easy to imagine how one could convert parts of large electronic structure calculations from Fortran into a hardware language so that several crucial subroutines run on dedicated nodes in a MIMD. At this point, however, we are well beyond the level of concurrency used for the electronic structure calculations in this thesis. Before we can begin to describe the parallel Car-Parrinello algorithm, we still



must consider a computer issue: how do scientific users program the concurrency?

## 2.2 Software

The language of choice for scientific programming has always been a version of Fortran. The biggest disadvantage of using Fortran on a parallel architecture is its complete lack of array-oriented constructs. All of the primitive arithmetic operators, including addition, subtraction, multiplication, and division, apply to scalar operands. Unfortunately this complaint applies to every major programming language, including C, Pascal, Ada or even Cobol. We saw what happens to matrix multiplication when it is described in C, with all indexing and iteration explicitly specified in figure 2.7. Most the code has nothing to do with multiplication. The challenge for a parallel programmer is to take a program like this one which is written in a common programming language and make it run efficiently on parallel hardware.

The simplest way out of this bind would be to have a compiler that is capable of “vectorizing” existing problems. This technique is commonly employed in the vector-pipelined supercomputer industry. One must learn which language constructs are vectorizable and which are not. While the optimizing compilers for mature systems like the Cray are remarkably good, often times those for new machines are not. Automatically optimized programs typically run far slower than the peak rate of the target vector machine [26].

Another common technique is for the vendor to supply a subroutine library of hand-optimized functions. The problem here is always one of versatility. Subroutine calls are limited since they cannot be manipulated with the same facility as a complete programming

language. When a given subroutine does not exist, it must usually be written in the less efficient high-level language since combining subroutines is rarely effective.

A realistic alternative for many embedded signal processing applications is to hand-microcode the problem. The time and expense is justified for numerically intensive programs that will run billions of times. An avionic signal processing program is a typical example. However, microprogramming is labor-intensive and requires specialized skills. It is rarely feasible for scientific programming.

Some vendors have designed scientific machines that directly execute high-level language constructs. The design of an APL computer is described in reference [27]. The Analogic AP500 array processor was based on a similar concept. This array processor was connected to an IBM PC workstation host and programmed in a machine language identical to APL [28]. The machine was commercially successful. Unfortunately, machines of this class are rarely designed and manufactured.

All of these programming methods would be unnecessary if a true parallel language existed based on a hardware model that mapped straightforwardly onto real parallel architectures. A detailed and famous review of the flaws of Fortran was given by its inventor Backus in 1978 [29]. In this review, he traces the problem back to its underlying machine model. Any programming language must have an underlying machine model if it is to be efficiently implemented.

Fortran, which was invented in the 1950's and has hardly been improved by existing well-known computer languages, has as its intellectual parent, the von Neumann computer architecture. This computing hardware model was developed in the 1940's before the exis-

tence of transistors, at time when each logic gate was implemented on a single circuit board. While this architecture was a great engineering advance at its time, Backus criticized the computing industry for not moving on to more advanced models in the forty years since its introduction in his famous discussion of the *von Neumann bottleneck*:

In its simplest form a von Neumann computer has three parts: a central processing unit (or CPU), a store and a connecting tube that can transmit a single word between the CPU and the store (and send an address to the store). I propose to call this tube the *von Neumann bottleneck*. The task of a program is to change the contents of the store in some major way; when one considers that this task must be accomplished entirely by pumping single words back and forth through the von Neumann bottleneck, the reason for its name becomes clear.

Ironically, a large part of the traffic in the bottleneck is not useful data but merely names of data, as well as operations and data used only to compute such names. Before a word can be sent through the tube its address must be in the CPU; hence it must either be sent through the tube from the store or be generated by some CPU operation. If the address is sent from the store, then *its* address must either have been sent from the store or generated in the CPU and so on. If, on the other hand, the address is generated in the CPU, it must be generated either by a fixed rule (e.g. "add 1 to the program counter") or by an instruction that was sent through the tube, in which case *its* address must have been sent... and so on.

Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the von Neumann

bottleneck. Not only is this tube a literal bottleneck for the data traffic of a problem, but, more importantly, it is an intellectual bottleneck that has kept us tied to word-at-a-time thinking instead of encouraging us to think in terms of the larger conceptual units of the task at hand. Thus programming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself but where to find it.

We saw these details in the matrix multiply program written in C, where all but one line of code went toward bookkeeping related to address arithmetic and iteration. We also pointed out that in a serial uniprocessor, each of these instructions must be executed in sequence because each instruction requires the same communication channel between the CPU and store, the channel Backus calls the von Neumann bottleneck. Backus goes on to say that conventional programming languages are basically high level, complex versions of von Neumann computers:

Von Neumann programming languages use variables to imitate the computer's storage cells; control statements elaborate its jump and test instructions; and assignment statements imitate its fetching, storing, and arithmetic. The assignment statement is the von Neumann bottleneck of programming languages and keeps us thinking in word-at-a-time terms in much the same way the computer's bottleneck does.

Consider a typical program; at its center are a number of assignment statements containing some subscripted variables. Each assignment statement produces a one-word result. The program must cause these statements to be executed

many times, while altering subscript values in order to make the desired overall change in the store, since it must be done one word at a time. The programmer is thus concerned with the flow of words through the assignment bottleneck as he designs the nest of control statements to cause the necessary repetitions.

In fairness to Fortran, when it was invented it was never intended to serve as a standard for over one-third of a century. So what's the alternative? Backus goes on to discuss functional programming languages, which explicitly address the issue of removing the assignment bottleneck. Rather than discussing the details of functional languages which are largely unavailable on parallel hardware, let's discuss some of the evolutionary features of existing languages.

Kenneth Iverson's APL removes the von Neumann bottleneck in many array-oriented operations [30]. The array operators are applied in right-to-left order to input arrays and stored in output arrays. It allows users to apply arithmetic primitives to large data collections (arrays) rather than a single word at a time. For example, if one writes

$$A \leftarrow B + C$$

corresponding elements of *arrays*  $B$  and  $C$  are added together, irrespective of dimensionality and rank of  $B$  and  $C$ . All operations on individual elements are extended to component-by-component operations on aggregates. Backus calls this convention, a *functional form*, a method of combining existing functions to make new ones. In this example, the functional form is the implicit extension of elementary operations to vectors, and the existing functions are scalar addition, multiplication and so on.

Iverson introduced several other functional forms into APL, which he calls *operators* [31].

To illustrate this concept, consider the *reduction* operator. The  $\odot$ -reduction of a vector  $\mathbf{x}$  is denoted by  $\odot/$  and defined as

$$z \leftarrow \odot/\mathbf{x} \Leftrightarrow z = (\cdots((x_1 \odot x_2) \odot x_3) \odot \cdots) \odot x_n,$$

where  $\odot$  is any binary operator with a suitable domain. Thus  $+/$  is the sum,  $\times/$  is the product, and  $\vee/$  is the logical sum of the components of the vector  $\mathbf{x}$ .

It is easy to combine APL operators and functions into useful array operations. For example, the vector dot product function is written

$$c \leftarrow +/\mathbf{B} \times \mathbf{C}$$

In this program, the binary function  $\times$  is applied between vectors  $\mathbf{B}$  and  $\mathbf{C}$  in component-by-component form and the result is summed by application of the “reduce-by-plus” operator  $+/$ . The result is assigned to the scalar variable  $c$ . Absent from this program are all statements related to addressing and iteration. They are specified to the hardware through the functional forms without forcing the programmer to spell out extraneous details.

In APL, matrix multiplication is written with the *inner product* operator. The matrix product of two matrices  $\mathbf{B}$  and  $\mathbf{C}$  is defined as a matrix whose  $ij$ th element is obtained by summing the result of taking an element-by-element product between row  $i$  of  $\mathbf{B}$  and column  $j$  of  $\mathbf{C}$ , expressible in APL as  $+/ \mathbf{B}[i;] \times \mathbf{C}[;j]$ . The inner product operator, denoted by a period, applies to two functions  $f$  and  $g$  (as in  $+. \times$ ) to produce a functions analogous to matrix product, but with  $f/\mathbf{B}[i;]g\mathbf{C}$  replacing  $+/ \mathbf{B}[i;] \times \mathbf{C}$ . Consequently, the inner product operator is a generalization of matrix product, producing it in the special case  $+. \times$ . Thus the APL program for matrix multiplication reduces to the binary array operator  $+. \times$ , a substantial simplification when compared to the one-page C version in figure 2.7.

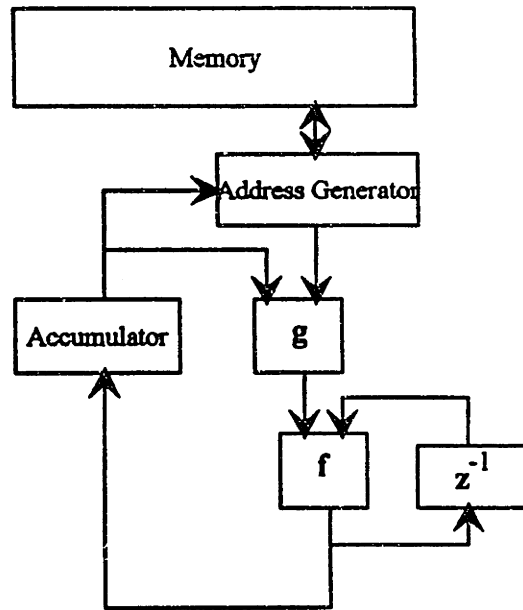


Figure 2.13: Simple hardware model suggested by APL functional forms.

Functional forms are important from a hardware perspective because computer languages provide intellectual models for computer design. Although the lack of functional forms in word-at-a-time languages makes this point difficult to illustrate for them, it is easy to show for the operators in APL. The simple block diagram in figure 2.13 shows one computing model suggested by the parallelism in APL. In this diagram, an arithmetic unit applies a function  $g$  between two input arrays in accordance with the implicit *element-by-element* functional form and the first function argument in *inner product*. The arithmetic unit also applies a function  $f$  in the feedback form necessary for execution of *reduction* and the second function argument  $g$  in *inner product*. An address generator device fetches array elements from a single shared memory in either the default row-major order or any permutation specified by APL operators, including the sequencing defined in the *inner product* functional form. This hardware model is just one possible implementation of APL. Massively parallel implementation of APL would also be possible.

A hardware-oriented language related to APL was the subject of the author's master's thesis [32]. The language, known as MFL, defined a small set of primitive operators which concisely expressed most APL primitives in a form designed for efficient hardware implementation for real-time signal processing applications. The thesis showed how the instruction set was also conducive to scientific computation. Details of the language [33] and hardware [34] are documented elsewhere. A key feature of the hardware design was an address generator device [35],[36] that could execute all of the array addressing patterns implied in the APL hardware model without microcode. It was possible to reparameterize a single microinstruction to do all of the address sequencing, thereby eliminating instruction overhead.

Another unique feature of the MFL system was an interactive workstation [37] that computed expected execution times for the user's algorithm as he wrote the code. By giving the cycle count for a hypothetical vector processor every time a program is run, the MFL programming environment assists the algorithm designer in optimizing programs from a high-level language perspective while the parallel algorithm is still under development.

MFL was not unique to one particular hardware model. A later paper [38] showed how MFL could be implemented on the Navy's standard MIMD signal processor, the AN/UYS-2 [39]. One advantage here is the ability to estimate performance as the parallel algorithm and high-level language implementation is developed rather than during hand-microcoded optimization. In this way, constraints and features of the parallel architecture can exert a detailed influence on algorithmic development. Moreover, if the array-oriented constructs of a parallel language such as MFL are easier to vectorize than a word-at-a-time language such as Fortran, the performance loss due to automatic compilation rather than hand optimization



could be negligible. The savings in time and expense is then sufficient to shift more emphasis to developing more efficient parallel algorithms, ultimately leading to more optimum use of concurrent computing resources.

One could also implement APL, MFL or any other set of parallel constructs in LISP [40]. Among well-known languages, LISP is unique in enabling users to design functional forms. Three common restrictions in word-at-a-time languages are: (1) requiring that a procedure be named and then referred to by name, (2) forbidding procedures to be returned as values of other procedures, (3) forbidding procedures to be components of such data structures as records or arrays. Without these constraints, it is easy for programmers to define new functional forms in terms of LISP primitives [41]. Considering the limitations of word-at-time languages, it is ironic that the major caution in using LISP pertains to its power and generality: a parallel hardware model would require fixed specification of a few functional forms defined in a specific LISP program in order to obtain an efficient design.

Another approach to obtaining parallelism in contemporary computer languages is Linda [42]. Linda consists of a few simple operators designed to support and simplify the construction of explicitly-parallel programs. These operators can be injected into existing programming languages to convert them to parallel languages. A Linda-based parallel language is a new language to the extent when the compiler recognizes the Linda operations and generates parallel code.

A review article by Saperstein and Blelloch surveys the current state-of-the-art on parallel languages [43].

Fortran 90 [44] is similar to Linda in the sense that it adds parallel constructs to the

standard Fortran 77 dialect in order to specify concurrency. Fortran 90 allows an array to be treated either as a set of scalars or an array. As a set of scalars, array elements are indexed explicitly in a DO construct. As an array they are not. For example, the following two programs add one to every element in an array

```
DO I=1,100
  A(I) = A(I)+1
ENDDO
```

is the familiar Fortran 77 expression, while

```
A=A+1
```

is the corresponding Fortran 90 expression. After thirty years, Fortran is starting to use some of the APL constructs. Sectioning of arrays is also done identically with APL. For example,  $B[i,:]$  extracts row  $i$  from  $B$ . Fortran 90 also allows array arguments to intrinsic functions, for example

```
A = SIN(B)
```

stores the sine of every element in  $B$  into array  $A$ . Fortran 90 also supplies a limited set of array intrinsic functions to implement some of the more common functions possible with APL operators. For example

```
A = SUM(B)
```

```
A = DOTPRODUCT(B,C)
```

correspond to  $+/$  and  $+\times$  in APL. Fortran 90 is important because it currently is the most efficient way for scientists to program the Connection Machine.

## 2.3 Massively Parallel Hardware: The Connection Machine CM-2

The Connection Machine CM-2 warrants a detailed description because it is the machine that was used to obtain the new physical results described in subsequent chapters of this thesis. As such, it is the first computer used in modern electronic structure calculations that is not based on the von Neumann architecture.

The key idea in the Connection Machine's programming model is the association of processors with data [45]. An application might associate one processor with each Fourier component in an electronic wave function, one processor with each real space point in a charge density, etc. There must then be some notion of interprocessor connectivity so that processors can communicate, ranging from nearest neighbor communication on a Cartesian grid to full interconnectivity. A single instruction is broadcast to every processor in the CM-2 array which then executes this instruction in parallel on different data.

The Connection Machine hardware consists of a parallel processing unit containing thousands of data processors, a front-end computer and an I/O system that supports mass storage, graphics devices and fast peripherals. The central element in the system is the parallel processing unit, which contains:

- from 4K to 64K 1-bit data processors
- a sequencer controlling the data processors

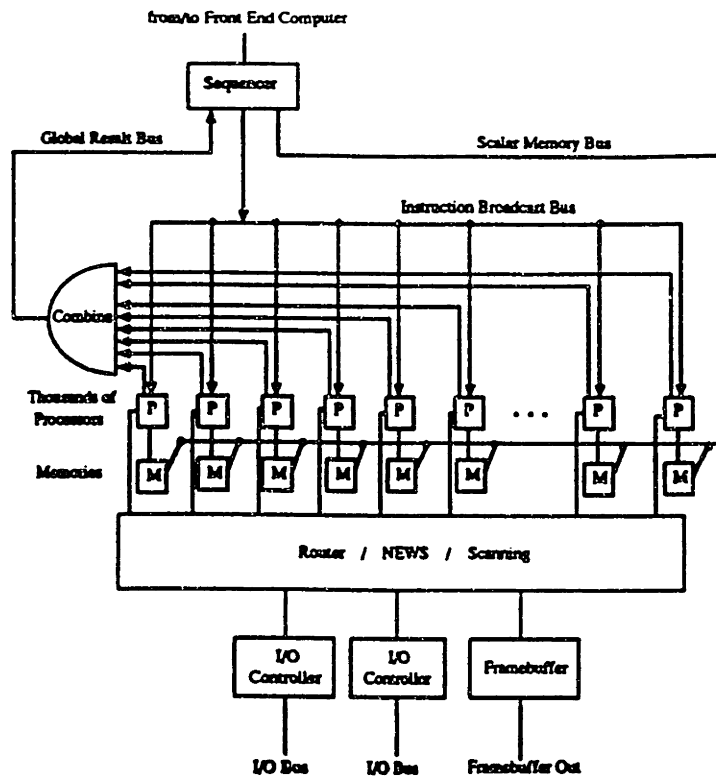


Figure 2.14: Connection machine CM-2 programmer's model

- an interprocessor communications network
- zero or more I/O controllers and/or graphics buffer modules

Figure 2.14 shows a simplified block diagram of the CM-2 parallel processing unit. Each data processor executes instructions globally broadcast by the front end computer, a Sun or VAX minicomputer. These commands are decoded into microinstructions by a microcoded sequencer. Each data processor then executes the microinstructions in parallel on different data under synchronous control from the microsequencer.

While all processors can access their respective memories simultaneously, the sequencer must access the processor array memory serially, one 32-bit word at a time over the scalar memory bus. This bus is the same memory/processor bottleneck inherent in the Illiac IV. It is crucial to avoid this type of communication if one expects reasonable performance in a large calculation. The data processors can also combine results in a reduction operation

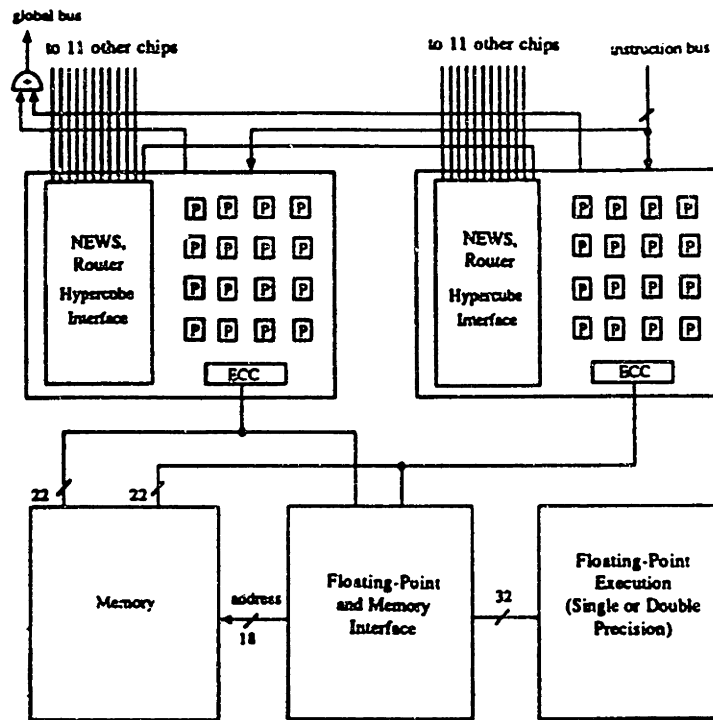


Figure 2.15: Connection machine CM-2 data processing node

such as sum (+/ in APL), delivering the reduced value on the global result bus. The data processors can exchange information among themselves in parallel through routing, NEWS and scanning mechanisms; these are in turn connected to the I/O interfaces.

Figure 2.15 shows a block diagram of a CM-2 data processing node consisting of

- 32 1-bit data processors with associated memory
- a floating-point unit
- communications interface.

Arithmetic is carried out bit serially in this processor, consuming at least three clock cycles per bit. A 32-bit add takes about 21 microseconds. Floating point execution also requires 32 nanoinstruction cycles, one for each of the 32 1-bit data processors connected to the floating point unit. The floating point coprocessor performs the floating point arithmetic

at the rate of one clock cycle per operand on pipelined data streams.

Interprocessor communication is accomplished in the CM-2 parallel processing unit by special-purpose hardware. Message passing happens in parallel; all processors can simultaneously fetch or send data to other processors through the most general of the CM-2's communications mechanisms, the router. Each CM-2 processor chip contains one router node to serve the 16 data processors on the chip. All router nodes are wired together in a boolean  $n$ -cube network topology. Each router node is connected to twelve other router nodes. Store-and-forward message passing is thus used: a message travels from one router node to another until the receiving processor is found.

This chapter has surveyed techniques for putting concurrency into computer technology. It shows to what extent electronic structure calculations have employed the state of the art in computer hardware and what types of more advanced computer systems are available to work with more complex material systems. We next describe specifically how the massively parallel CM-2 applies to electronic structure calculations.

## Chapter 3

# A Massively Parallel Implementation of the Car-Parrinello Algorithm

*Ab initio* total energy pseudopotential calculations can predict many physical properties of materials using only the atomic numbers of the constituent atoms as an input. The method has been successfully used to predict equilibrium lattice constants, bulk moduli, phonons, piezoelectric constants and phase transition pressures and temperatures. The method is based on expanding the electronic wavefunction in terms of plane wave basis functions for a set of ionic positions. For a given configuration, the total energy is computed. The plane wave coefficients and ionic positions are varied until the values minimizing the total energy of the system are obtained.

Even for systems containing only a few atoms in the unit cell, total energy pseudopotential calculations consume a substantial amount of computer time. The total energy is computed from a Kohn-Sham Hamiltonian matrix. Since this matrix depends on the charge density, the total energy must be computed self-consistently through an iterative process.

Each step toward diagonalization of the Kohn-Sham Hamiltonian leads to an improved solution for the wave function which in turn improves the Hamiltonian until self-consistency is obtained.

The molecular dynamics method developed by Car and Parrinello[46] transformed the philosophy of total energy pseudopotential calculations. Instead of determining the minimizing set of plane wave coefficients by explicitly diagonalizing the Kohn-Sham Hamiltonian, Car and Parrinello recognized that a more direct minimization of the Kohn-Sham energy could be more efficient and allowed both electronic and ionic degrees of freedom to appear on the same footing. To perform the minimization, they introduced an iterative optimization technique where the plane wave coefficients and ionic coordinates are treated as fictitious particles interacting through a classical Lagrangian. Integration of the equations of motion for these particles leads to the configuration that minimizes the total energy of the system. Recently, conjugate gradient methods have been developed[47] that consume in certain cases an order of magnitude less CPU time than the best fictitious electronic dynamics schemes. Still, the memory and throughput available through conventional vector supercomputers limits both methods to systems involving around 100 atoms.

Massively parallel computation provides another technique for working with larger systems. The challenge in exploiting the massively parallel architecture is to efficiently map the algorithm onto the parallel architecture without incurring excessive loss due to communication among the processors. In this case, one can scale the size  $P$  of the processor array with the characteristic size  $N$  of the system under study. As  $N$  is increased, a corresponding increase in  $P$  results in more throughput, reducing the processing time below the increase that would be incurred for fixed  $P$ , including  $P = 1$ , a serial machine. On a Connection



Machine CM-2, a parallel implementation of the Car-Parrinello algorithm makes the study of systems approaching a thousand atoms possible.

### 3.1 Algorithm Description

The Car-Parrinello algorithm is a quantum mechanical method of computing electronic wavefunctions within the density functional theoretic treatment of the many body problem. In practice the local density approximation is generally used. A detailed review of *ab initio* total energy techniques is given in Ref. [48]. The algorithm is summarized here in order to introduce the most important data structures and procedures as implemented in the parallel computation.

Each electronic wavefunction is expanded as a sum of plane waves

$$\psi_i(\mathbf{r}) = \sum_{\mathbf{G}} c_{i,\mathbf{k}+\mathbf{G}} \exp[i(\mathbf{k} + \mathbf{G}) \cdot \mathbf{r}]. \quad (3.1)$$

In this expansion,  $\mathbf{G}$  denotes the set of reciprocal lattice vectors defined by  $\mathbf{G} \cdot \mathbf{l} = 2\pi m$  where  $\mathbf{l}$  is a lattice vector of the crystal and  $m$  is an integer.  $\mathbf{k}$  denotes the point in the Brillouin zone where the calculation is performed. Total energy calculations generally approximate integrals over the crystal Brillouin zone with discrete sums over a few “special”  $\mathbf{k}$  points[49]. The electronic wavefunction  $\psi$  is usually calculated for a few independent values of  $\mathbf{k}$ . In very large systems, one  $\mathbf{k}$  point is often sufficient to sample the Brillouin zone.

In an ideal expansion, the sum over reciprocal lattice vectors in Eqn. (1) runs over an infinite set of coefficients  $c_{i,\mathbf{k}+\mathbf{G}}$  with corresponding kinetic energy  $\frac{\hbar^2}{2m}|\mathbf{k} + \mathbf{G}|^2$ . However, in a pseudopotential calculation the coefficients for the plane waves with small kinetic energy

are typically more important than those with large kinetic energy. The plane wave basis set can then be truncated. In total energy pseudopotential calculations one need only include the plane waves which have kinetic energies less than some particular cut-off energy. This parameter is a convenient quantification of the number of plane waves in the computation.

Substitution of the plane wave expansion for the wavefunction in the Kohn-Sham total energy functional and integration over  $\mathbf{r}$  gives the secular equation[50]

$$\sum_{\mathbf{G}'} \left[ \frac{\hbar^2}{2m} |\mathbf{k} + \mathbf{G}|^2 \delta_{\mathbf{G}\mathbf{G}'} + V_{ION}(\mathbf{k} + \mathbf{G}, \mathbf{k} + \mathbf{G}') + V_H(\mathbf{G} - \mathbf{G}') + V_{XC}(\mathbf{G} - \mathbf{G}') \right] c_{i,\mathbf{k}+\mathbf{G}'} = \varepsilon_i c_{i,\mathbf{k}+\mathbf{G}} \quad (3.2)$$

where  $V_H$  denotes the Hartree (Coulomb) interaction,  $V_{XC}$  denotes the exchange-correlation interaction and  $V_{ION}$  denotes the ionic pseudopotential.

While this equation can be solved through conventional matrix diagonalization, Car and Parrinello recognized that the electronic wavefunctions can be treated as variables in a fictitious dynamical system. A Lagrangian is defined for the electronic system according to

$$L = \sum_i \mu \langle \dot{\psi}_i | \dot{\psi}_i \rangle - E[\{\psi_i\}, \{\mathbf{R}_I\}, \alpha_n] \quad (3.3)$$

where  $\mu$  is a fictitious mass associated with the electronic wavefunctions,  $E$  is the Kohn-Sham energy functional,

$$\begin{aligned} E[\{\psi_i\}] &= 2 \sum_i \psi_i \left[ -\frac{\hbar^2}{2m} \nabla^2 \psi_i \right. \\ &+ \int V_{ION}(\mathbf{r}) \rho(\mathbf{r}) d^3\mathbf{r} + \frac{1}{2} \int \frac{\rho(\mathbf{r}) \rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d^3\mathbf{r} d^3\mathbf{r}' \\ &+ E_{xc}[\rho(\mathbf{r})] + E_{ION}(\{\mathbf{R}_I\}) \left. \right], \end{aligned} \quad (3.4)$$

$\rho$  is the charge density,  $\mathbf{R}_I$  is the position of ion  $I$ , and the parameters  $\{\alpha_n, n = 1, 2, 3\}$  define the dimensions of the unit cell. The kinetic energy term in the Lagrangian is due to

the fictitious dynamics of the electronic degrees of freedom.

The electron wavefunctions are subject to the constraints of orthonormality

$$\int \psi_i^*(\mathbf{r})\psi_j(\mathbf{r})d^3\mathbf{r} = \delta_{ij}. \quad (3.5)$$

These constraints are incorporated in the molecular dynamics Lagrangian through the method of Lagrange multipliers. The molecular dynamics Lagrangian becomes

$$L = \sum_i \mu \langle \dot{\psi}_i | \dot{\psi}_i \rangle - E[\psi_i, \mathbf{R}_I, \alpha_n] + \sum_{i,j} \Lambda_{ij} \left[ \int \psi_i^*(\mathbf{r})\psi_j(\mathbf{r})d^3\mathbf{r} - \delta_{ij} \right]. \quad (3.6)$$

The Lagrange multipliers  $\Lambda_{jj}$  ensure that the wavefunctions remain normalized and the Lagrange multipliers  $\{\Lambda_{ij}, i \neq j\}$  ensure that the wavefunctions remain orthogonal. It is more computationally efficient to maintain orthogonality only at the end of each discrete time step. In this case, the Lagrange multipliers for the constraints of normalization  $\Lambda_{ii}$  are replaced by the expectation values of the energies of the states,  $\lambda_i$ , where

$$\lambda_i = \langle \psi_i | H | \psi_i \rangle \quad (3.7)$$

resulting in the following equation of motion for the plane wave coefficients

$$\begin{aligned} \mu \ddot{c}_{i,\mathbf{k}+\mathbf{G}} = & -\left[\frac{\hbar^2}{2m}|\mathbf{k} + \mathbf{G}|^2 - \lambda_i\right]c_{i,\mathbf{k}+\mathbf{G}} - \\ & \sum_{\mathbf{G}'} V_H(\mathbf{G} - \mathbf{G}')c_{i,\mathbf{k}+\mathbf{G}'} - \\ & \sum_{\mathbf{G}'} V_{XC}(\mathbf{G} - \mathbf{G}')c_{i,\mathbf{k}+\mathbf{G}'} - \\ & \sum_{\mathbf{G}'} V_{ION}(\mathbf{k} + \mathbf{G}, \mathbf{k} + \mathbf{G}')c_{i,\mathbf{k}+\mathbf{G}'}. \end{aligned} \quad (3.8)$$

When a local pseudopotential is used, this equation can be written

$$\mu \ddot{c}_{i,\mathbf{k}+\mathbf{G}} = -\left[\frac{\hbar^2}{2m}|\mathbf{k} + \mathbf{G}|^2 - \lambda_i\right]c_{i,\mathbf{k}+\mathbf{G}} - \sum_{\mathbf{G}'} V_T(\mathbf{G} - \mathbf{G}')c_{i,\mathbf{k}+\mathbf{G}'} \quad (3.9)$$

where  $V_T(\mathbf{G})$  is the total potential given by

$$V_T(\mathbf{G}) = V_{ION}(\mathbf{G}) + V_H(\mathbf{G}) + V_{XC}(\mathbf{G}). \quad (3.10)$$

If one now introduces the definitions

$$\omega_{i,\mathbf{k}+\mathbf{G}}^2 = [\frac{\hbar^2}{2m}|\mathbf{k} + \mathbf{G}|^2 - \lambda_i]/\mu \quad (3.11)$$

and

$$B_{i,\mathbf{k}+\mathbf{G}} = [\sum_{\mathbf{G}' \neq \mathbf{G}} V_T(\mathbf{G} - \mathbf{G}')c_{i,\mathbf{k}+\mathbf{G}'}]/\mu, \quad (3.12)$$

the equation of motion becomes

$$\ddot{c}_{i,\mathbf{k}+\mathbf{G}} = -\omega_{i,\mathbf{k}+\mathbf{G}}^2 c_{i,\mathbf{k}+\mathbf{G}} - B_{i,\mathbf{k}+\mathbf{G}}. \quad (3.13)$$

This equation can be numerically integrated at short time steps. After each time step the potentials are recomputed based on the new charge density derived from the updated plane wave coefficients in the wavefunction.

As noted in Ref. [51], Eqn. (13) is an oscillator equation. During a sufficiently short time step, the plane wave coefficients should behave approximately as

$$c_{i,\mathbf{k}+\mathbf{G}}(t) = c_{i,\mathbf{k}+\mathbf{G}}(0) \cos(\omega_{i,\mathbf{k}+\mathbf{G}} t) \quad (3.14)$$

Thus the equations of motion can be integrated analytically up to the next time step, resulting in the following difference equation:

$$\begin{aligned} c_{i,\mathbf{k}+\mathbf{G}}(\Delta t) &= 2 \cos(\omega_{i,\mathbf{k}+\mathbf{G}} \Delta t) c_{i,\mathbf{k}+\mathbf{G}}(0) \\ &\quad - c_{i,\mathbf{k}+\mathbf{G}}(-\Delta t) - 2[1 - \cos(\omega_{i,\mathbf{k}+\mathbf{G}} \Delta t)] B_{i,\mathbf{k}+\mathbf{G}} / \omega_{i,\mathbf{k}+\mathbf{G}}^2. \end{aligned} \quad (3.15)$$

It is also possible to form a first order equation of motion for the electronic degrees of freedom[52]. In this case, the equation of motion is

$$\dot{c}_{i,k+G} = -\omega_{i,k+G}^2 c_{i,k+G} - B_{i,k+G}, \quad (3.16)$$

leading to the difference equation

$$c_{i,k+G}(\Delta t) = -\frac{B_{i,k+G}}{\omega_{i,k+G}^2} + [c_{i,k+G}(0) + \frac{B_{i,k+G}}{\omega_{i,k+G}^2}] \exp(-\omega_{i,k+G}^2 \Delta t). \quad (3.17)$$

This first order equation gives roughly the same asymptotic convergence rate as the corresponding second order equation. It is more efficient in the sense that only half the storage is required since only  $\psi(0)$  and not  $\psi(-\Delta t)$  are maintained.

This version of the Car-Parrinello algorithm was chosen for parallel implementation with one addition. To better describe scattering properties of the Si atoms, a nonlocal Kleinman-Bylander pseudopotential[53] was employed. This pseudopotential has the following form:

$$V_{ION} = V_{LOC} + \sum_{lm} \frac{|\phi_{lm}^0 \delta V_l| > < \delta V_l \phi_{lm}^0|}{< \phi_{lm}^0 | \delta V_l | \phi_{lm}^0 >} \quad (3.18)$$

where  $V_{LOC}$  is a local, norm-conserving pseudopotential[54],  $\phi_{lm}^0$  are the wavefunctions of the pseudoatom and  $\delta V_l$  is

$$\delta V_l = V_{l,NL} - V_{LOC} \quad (3.19)$$

where  $V_{l,NL}$  is the  $l$  angular momentum component of any non-local pseudopotential.

Table 3.1 summarizes the notation used in this implementation of the Car-Parrinello algorithm with the first-order equation of motion and the Kleinman-Bylander nonlocal pseudopotential. Dimensions in terms of parameters defined in Table 3.2 are also given. Of all data structures in the program, the wavefunction occupies the most memory. It

Table 3.1: Summary of arrays used in the Car-Parrinello implementation.

Variable	Dimension	Name
$L_i$	3	lattice vectors
$Z_s$	$N_S$	atomic number
$\mathbf{R}_a$	$3N_A$	ionic coordinates
$c_{i,\mathbf{k}+\mathbf{G}}$	$N_B N_k N_G$	reciprocal space wavefunction
$p_{\mathbf{k},\mathbf{G}}$	$N_k N_G$	mapping from packed to unpacked $\mathbf{G}$
$\rho(\mathbf{G})$	$N_X N_Y N_Z$	reciprocal space charge density
$\rho(\mathbf{r})$	$N_X N_Y N_Z$	real space charge density
$V_H(\mathbf{G})$	$N_X N_Y N_Z$	Hartree potential
$E_H$	1	Hartree energy
$V_{XC}(\mathbf{G})$	$N_X N_Y N_Z$	exchange-correlation potential
$E_{XC}$	1	exchange-correlation energy
$S(i, \mathbf{G})$	$N_S N_X N_Y N_Z$	structure factor ( $\sum_{ions}$ )
$V_{PS}(i, \mathbf{R})$	$N_S N_X N_Y N_Z$	pseudopotential interpolation
$V_{PS}(\mathbf{G})$	$N_X N_Y N_Z$	pseudopotential
$E_{PS}$	1	pseudopotential energy
$F_{a,i}^{I-I}$	$3N_A$	ion-ion force
$E_{I-I}$	1	ion-ion energy
$F_{a,i}^L$	$3N_A$	local electron-ion force
$R_{i,\mathbf{k}+\mathbf{G}}^L$	$3N_G N_k$	$= G_{i,k}/\ \mathbf{G}\ $
$f_{lm,\mathbf{k}+\mathbf{G}}$	$N_S N_{lm} N_k N_G$	$= \int_0^\infty dr r^2 j_l(qr) \phi_{lm}(r) \delta V_l(r)$
$S_{a,\mathbf{k}+\mathbf{G}}^A$	$N_A N_k N_G$	$e^{-2\pi i \mathbf{k} + \mathbf{G} \cdot \mathbf{r}_a}$
$g_{l,\alpha,n,k}$	$N_A N_l 3N_B N_k$	nonlocal pseudopotential sums
$F_{a,i}^{NL}$	$3N_A$	nonlocal electron-ion force
$\lambda_n$	$N_B$	band eigenvalues (energies)

Table 3.2: Array size abbreviations

Parameter	Description
$N_A$	atoms
$N_B$	bands
$N_G$	plane waves per band
$N_k$	special k points
$N_{lm}$	atomic angular momentum quantum numbers
$N_S$	atomic species
$N_X$	lattice points in direction 1
$N_Y$	lattice points in direction 2
$N_Z$	lattice points in direction 3

scales as the number of bands times the number of plane waves. The second largest array is a structure factor  $e^{-2\pi i \mathbf{G} \cdot \mathbf{r}_A}$  that is computed for each atom. It is used repeatedly in a computationally intensive part of the nonlocal pseudopotential calculation. The memory required for these two arrays provides a coarse estimate of the memory required for the entire computation. All other arrays are much smaller for systems with large numbers of ions.

Fig. 3.1 summarizes the algorithm. It begins with a trial ionic configuration and a trial wavefunction consisting of an initial set of plane wave coefficients. An initial charge density is computed from this wavefunction for use in the Hartree, exchange-correlation, local and nonlocal pseudopotentials. The total energy for this trial wavefunction is then computed, and the plane wave coefficients are updated by integrating their fictitious equations of motion according to the Car-Parrinello scheme. After every integration, the wavefunction is re-orthogonalized. Then the next iteration begins. After every five to ten iterations, ionic forces are computed and the ions are moved in order to relax the ions simultaneously with the wavefunction. This procedure continues until the total energy and the ionic forces do

not change substantially from one iteration to the next.

### 3.2 Parallel Implementation of Car-Parrinello Algorithm

Most aspects of the algorithm we used for our calculations are sufficiently general to apply to a wide class of parallel machines, though we have chosen the CM-2 for our implementation. In order to discuss the algorithmic issues involved in the parallel Car-Parrinello procedure, it helps to distinguish features of general importance for a wide class of parallel architectures from those of particular interest for the machine we used, the CM-2. In order to keep these issues separate, we will use a simple abstract model of a parallel architecture, a distributed memory parallel random access machine (DM-PRAM) [55]. This model is similar to the standard one discussed in elementary computer science texts on parallelism, but with distributed memory. In this model,  $P$  processors are connected by a network which can route messages in  $O(1)$  time. In the DM model, all memory is local to a specific processor, and only that processor can read/write data to its memory.

Many real network architectures, including hypercube, butterfly, and hypertree, are *universal* in that they can simulate any other network of  $P$  processors, including the simple PRAM network, with at most a  $\log P$  slowdown. Other real networks such as  $d$ -dimensional grids are not universal in this sense, because simulating the PRAM routing process requires  $O(P^{1/d})$  time. Nevertheless engineering factors can make machines based on such networks able to simulate the PRAM over a useful range of  $P$ . We use the term communication to refer to all operations involving the network.

The PRAM model has some drawbacks. In practice the assumption that network com-



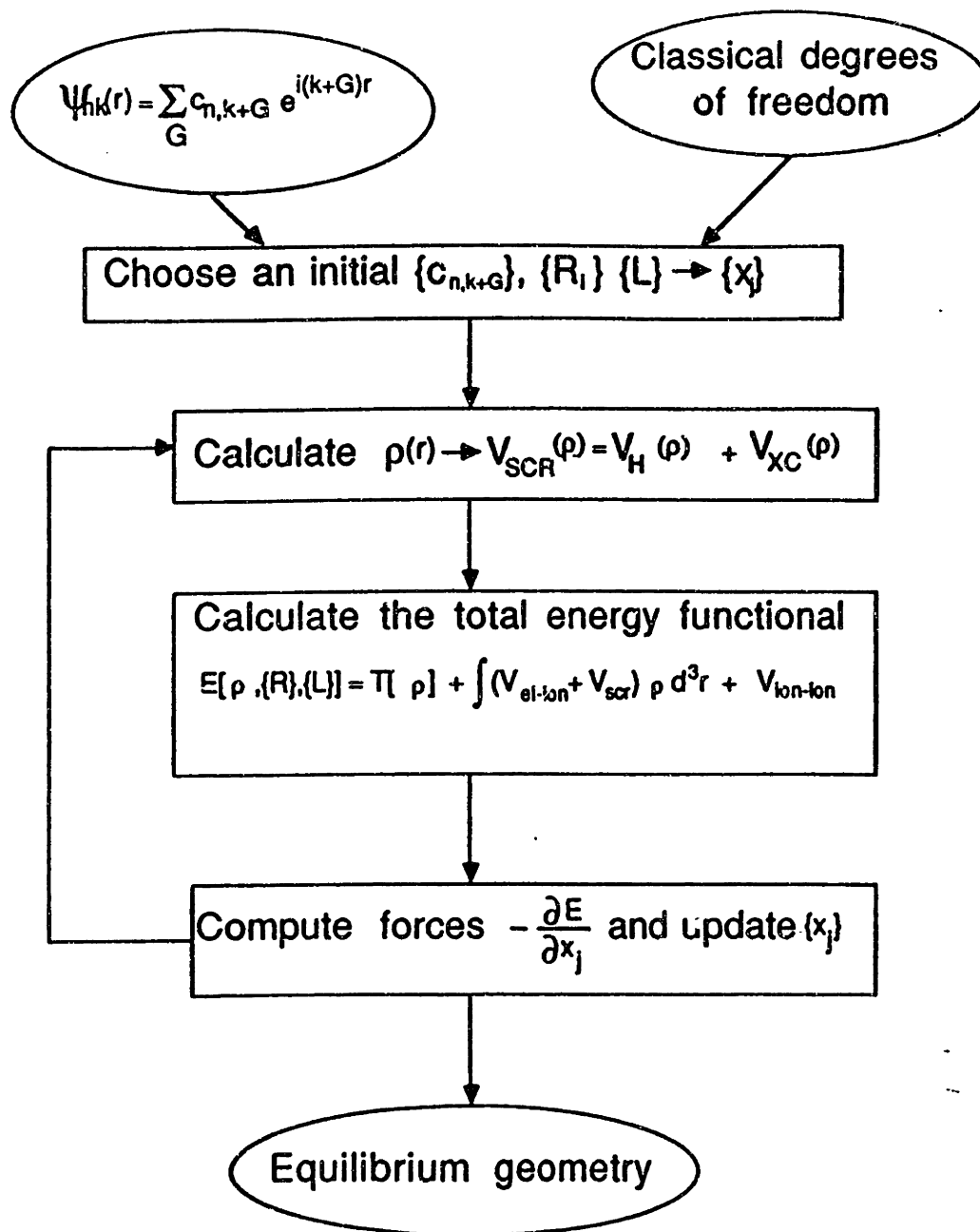


Figure 3.1: Car-Parrinello algorithm summary.

munication times scale in the same way as computations is less important than the large, if constant, factor multiplying the communication time step relative to computation. Efficient codes attempt to minimize the network communications as much as possible. The assumption that communications cost is equal no matter what the relative location of source and destination is much closer to reality and to programming practice. A final drawback is that the simple PRAM network does not take account of special purpose hardware for performing certain patterned operations such as broadcast and reduction. These can be synthesized in  $\mathcal{O}(\log P)$  time but with hardware become effectively constant-time.

Here we formalize some of the concepts relating to the scalability of an acceptable algorithm. Define  $T(N, P)$  to be the time required to solve a problem of size  $N$  on  $P$  processors. To be definite,  $N$  might be the number of atoms in the unit cell. Let  $t(N)$  be the time for solving the same problem using the best sequential algorithm, then the *efficiency*  $e$  is defined by  $e = t(N)/(PT(N, P))$  and must be less than 1, since a serial machine can emulate a  $P$ -processor parallel machine with a slowdown of order  $P$  at worst. The goal is to design a *scalable* parallel algorithm. In this context, scalability means the  $T(N, 1) = \mathcal{O}(t(N))$ ; and that  $T(N, P) = \mathcal{O}(T(N, 1)/P)$  for sufficiently large  $P$  and  $N$ . These conditions guarantee, in effect, that the algorithm is not worse in its  $N$ -scaling than the best serial algorithms, and that the algorithm is cost effective in its  $P$ -scaling. We shall relax these scalability requirements slightly and allow deviations containing polylogarithmic factors (in  $N$  or  $P$ ) because of their modest growth rates over ranges of the parameters of interest and of current computational feasibility.

Of course no formal considerations are a substitute for a test of whether a real parallel implementation can solve new problems. In view of this, no formal proof that our algorithm

is scalable will be attempted. Indeed most of our experience with the code makes clear that the constant multipliers, preasymptotic factors, memory constraints, etc. left out of the formal analysis are of critical importance for the present range of  $(N, P)$ . We will offer estimates of the scaling behavior of component parts of the code however as an indicator of their performance on the next generation of parallel machines, for the correspondingly larger problems which will be solved.

Our principal concern is to solve large problems in a tractable amount of time rather than to solve moderate problems in very short time. One of the most critical properties of larger parallel machines is that the memory scales with the number of processors. Table 3.3 shows that the memory requirements of the code scale roughly as  $N^2$ , with  $N$  denoting the number of atoms  $N_A$ . In actual practice, we followed a curve in the  $(N, P)$  plane corresponding to the smallest value of  $P$  with enough memory for doing a problem of size  $N$ . Therefore we only explore a small part of the 2-parameter function  $T(N, P)$ .

Our specific target machine was the Connection Machine CM-2, a massively parallel computer with up to  $2^{16} = 65536$  bit-serial processors and  $2^{11} = 2048$  64-bit floating point units. The current CM Fortran compiler views the machine in terms of the floating point units (FPU's), ignoring the bit-serial processors, and we will describe the machine also from the this point of view. Thus  $P$  means the number of FPU's. The  $2K$  FPU's are connected in an 11-dimensional hypercube. ( We use the common notation where  $K = 1024$ ,  $M = K^2$ , etc.) The channels forming the edges of the hypercube are bi-directional and all edges originating from a processing node can carry messages simultaneously. Each FPU has an associated memory of up to one million 32-bit words.

The architecture is a Single-Instruction, Multiple-Data (SIMD), meaning that instruc-

tions are broadcast from a sequencer controlling a partition of processing nodes and executed by the nodes one at a time. The nodes do not execute separate programs, but can, based on data in their memory, ignore a particular instruction. The node can also execute indirect address accesses to their memory. The CM-2 has an associated mass-storage device called the Datavault, with capacities up to 80 GB. We stored checkpoint files on Datavault during the long runs necessary for calculations on large material systems.

The software model is data parallelism. This programming model provides the programmer with the abstraction of a global address space and logically arbitrary numbers of processors. Programs are conceived as alternating sequences of (a) independent operations on elements of a data set in parallel, and (b) communications among different elements. We are presented with this model through the CM Fortran 1.0 compiler. The CM Fortran language implements the support for array operations of the standard Fortran 90.

The CM Fortran compiler provides directives allowing the user to control the mapping or layout of arrays onto the processors. Because communication is relatively expensive, important distinctions exist among different ordering choices. These directives control whether all positions along a specified array axis will be located in a single processor's memory, a `:SERIAL` axis, or whether they will be spread in contiguous chunks over all processors, a `:NEWS` axis. "Communication" operations applied along `:SERIAL` axes translate into in-processor memory moves, and so are much faster than normal communication involving the network. The `:NEWS` ordered axes are laid out on the processors to optimize the performance of local communication on multidimensional Cartesian grids. A third ordering, `:SEND`, allows an axis to be laid out across the processors similar to `:NEWS`, but ordered to optimize FFT's rather than local communication. For this reason `:SEND` ordering was chosen for the

charge density arrays and all arrays which need to be conformable with them.

Although our general design philosophy was to make as much of the code Fortran 90 compatible as possible, there were two places where performance issues dictated more machine-specific solutions. The first is the FFT. The code depends heavily on this primitive, and an efficient FFT was therefore essential. We used the optimized FFT in the Connection Machine Scientific Subroutine Library (CMSSL). The reason is that the library takes advantage of a natural mapping between the FFT algorithm and the physical hypercube network structure of the CM-2. This cannot be done nearly as efficiently from high level languages such as CM Fortran.

The second instance concerns packing and unpacking the wavefunction from one-dimensional slices into its full three-dimensional form. This operation is also performed many times in the code, but the pattern of the communication never changes. The communication hardware of the CM-2 is not exposed to the CM Fortran compiler, which relies on calls to a run-time software layer to manage communication. The run-time software treats all communication as dynamic. What was required is precomputed communication for the given static pattern and this required a machine-specific low level treatment. Fortunately there is a pair of CMSSL functions, `gather` and `scatter`, which perform this communication after a call to a setup routine (executed once at program initialization) which precompiles the computation and stores this information for the routines to use. For the pack/unpack in the charge density computation module, a factor of three gain in throughput was obtained by replacing the generic Fortran 90 `UNPACK` function with the CMSSL `scatter` function.

The parallel implementation begins with an analysis to determine the distribution of data across the parallel processing array. Fig. 3.2 shows the memory consumed by each

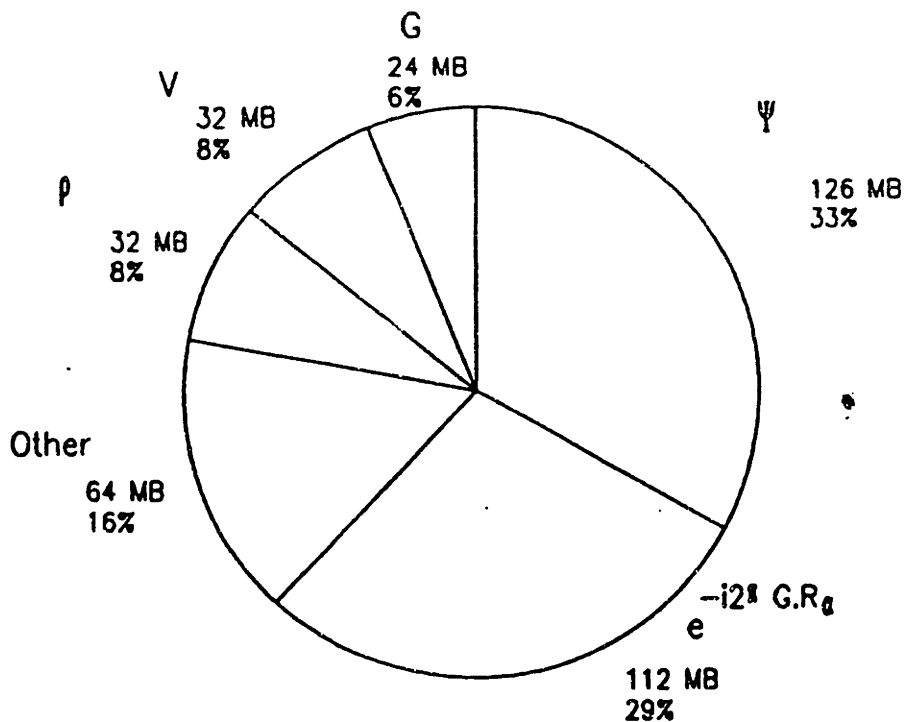


Figure 3.2: Memory utilization for the Si(111)- $7 \times 7$  calculation with 896 bands at a 8 Ry cutoff energy.

major array for a Si(111)- $7 \times 7$  calculation at an 8 Ry cutoff energy. This semiconductor surface is the focus of the physical results presented in the next two chapters. It is described in this chapter in terms of the performance requirements it imposes on the parallel computing hardware. All arrays except the wavefunction and the  $e^{-i2\pi \mathbf{G} \cdot \mathbf{R}_A}$  factors were stored in double precision. All computations are performed in double precision except Fourier transforms on the unpacked single precision wavefunction. The rationale for using single precision on the large arrays is to save memory. When the calculation has converged, band eigenvalues are accurate to about four decimal places. The eigenvectors have comparable or less precision. Storing the extra 32 bits in a double precision wavefunction would only store computational noise. More memory was saved by configuring the unit cell to exploit inversion symmetry. An inversion-symmetric system can be computed with a real wavefunction and a conjugate-symmetric structure factor array, reducing the memory requirements roughly in half.

Table 3.3: Array distributions

Array	Dimensions	Distribution
$c_{i,k+G}$	$N_B N_k N_G$	:SERIAL, :SERIAL, :NEWS
$S_{a,k+G}$	$N_A N_k N_G$	:SERIAL, :SERIAL, :NEWS
$g_{a,l,i,n,k}$	$N_A N_{lm} 3 N_B N_k$	:SERIAL, :SERIAL, :SERIAL, :SERIAL, :NEWS
$V(G)$	$N_X N_Y N_Z$	:SEND, :SEND, :SEND
$\rho(G)$	$N_X N_Y N_Z$	:SEND, :SEND, :SEND
$V_{PS}(i, R)$	$N_S N_X N_Y N_Z$	:SERIAL, :SEND, :SEND, :SEND
$\rho(r)$	$N_X N_Y N_Z$	:SEND, :SEND, :SEND
$S(i, G)$	$N_S N_X N_Y N_Z$	:SERIAL, :SEND, :SEND, :SEND
$T(G)$	$N_X N_Y N_Z$	:SEND, :SEND, :SEND

Fig. 3.2 shows that an 8 Ry calculation on 700 effective silicon atoms consumes about 380 million bytes of memory. More memory is allocated by the compiler for intermediate computations. Of the memory specified by the programmer, 62 per cent is allocated for the two large single precision arrays.

Table 3.3 shows the data distribution for each data structure in the figure. The :SERIAL and :NEWS designations in the table entries indicate that a given array axis is either contained entirely in local memory, or spread evenly across the processors, respectively. This notation is borrowed from the CM Fortran compiler. The need to choose the layout of data on a parallel machine is of course a universal feature of parallel algorithm design.

The most important distribution is probably the mapping of the plane wave coefficients  $c_{i,k+G}$ . The :SERIAL, :SERIAL, :NEWS designation for indices  $i, k, G$  indicates that plane waves are spread across the processors with each processor containing the same plane wave subset for all band indices as shown in Fig. 3.3. With this mapping operations on plane waves will be performed in parallel for each band and  $k$  point index. The iterations over

```

REAL*4 psi(896,36846)
CMF$ LAYOUT psi(:SERIAL,:NEWS)

```

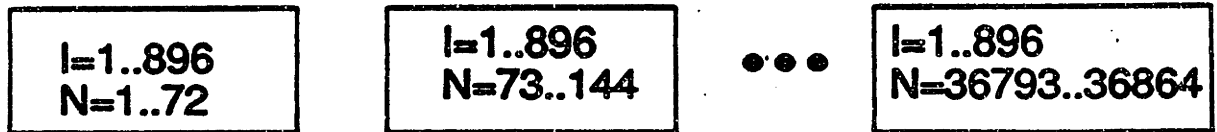


Figure 3.3: Distribution of the wavefunction across the processing array.

the :SERIAL band and k point indices in the wavefunction are then performed, with each processor performing a loop over these values for its subset of  $G$ -vector components. The  $T(G)$  table entry refers to large, temporary arrays sized to the reciprocal space grid in order to accelerate computations performed at every grid point in parallel.

An alternative mapping would be to assign a given band and k point  $(i, k)$  to each processor, with every plane wave coefficient  $G$  for a given  $(i, k)$  in a single processor. While this scheme would require less communication, there is insufficient memory in each processor to carry out the resulting computations in parallel. The other mapping was used because available memory was more of a limiting constraint than processing time in determining the size of the calculation that could be performed.

The other major decision involved the geometry of all arrays dimensioned as the  $128 \times 128 \times 64$  grid in the unit cell, including the charge densities  $\rho(G)$  and  $\rho(r)$ , and the potentials  $V(G)$ . It is a straightforward decision to distribute all three indices across the



```

COMPLEX*16 chden(128,128,64)
CMF$ LAYOUT chden(:SEND,:SEND,:SEND)

```

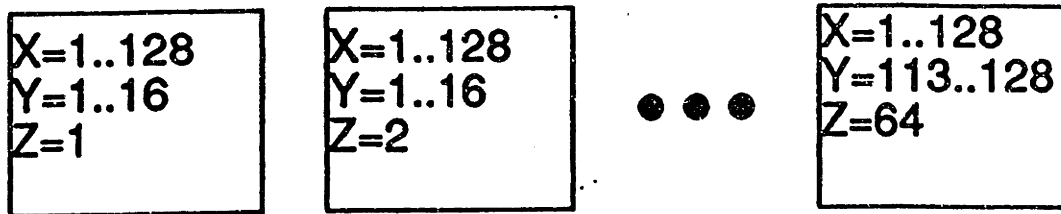


Figure 3.4: Distribution of the charge density across the processing array.

processor array as shown in Fig. 3.4. Operations on all three lattice coordinates proceed in parallel. Mappings for all other arrays involving FFT grid coordinates were chosen to conform to the mappings for the charge densities and potentials, thereby minimizing communication.

The other distributions was chosen out of expedience: it is necessary to allocate some data structures on the processor array rather than the CM front end because front end memory is limited. The mapping here is arbitrary since communication will be incurred in any case.

Once data distributions are determined, one can proceed to write parallel code. Our code is written entirely in Fortran 90 with the exception of calls to math library routines for performing the FFT, generating random numbers, and performing the global gather/scatter for optimized packing of the wavefunction. Fortran 90 versions of gather and scatter are also available.

Throughout the calculation, it is possible to compute array values for different  $\mathbf{k}$  points in parallel. The Car-Parrinello algorithm treats each  $\mathbf{k}$  point independently to the extent that different sets of energy eigenvalues are computed for each  $\mathbf{k}$  point. Where memory permits, it is straightforward to generalize the following description of the parallel implementation to simultaneously compute arrays at different  $\mathbf{k}$  points. This technique will accelerate computations over smaller unit cells involving multiple  $\mathbf{k}$  points.

With these data distributions, the only communication required occurs in one of three places: performing sums over  $G$ -vectors, performing FFT's, and packing/unpacking the wavefunction. For operations which do not include communication, it is easy to see that at least for large  $N$  and  $P$  the scaling will be  $T(N, P) = \mathcal{O}(t(N)/P)$ . This means the performance of such segments approaches a constant efficiency and scales with problem size and machine size. To understand the scaling then requires examination of the communication-dependent kernels.

A sum of  $N$  values can be performed on  $P$  processors in time  $T(N, P) = \mathcal{O}(k_0 \log P + N/P)$  by recourse to the following procedure. Each processor initially has  $N/P$  values and independently adds them in  $\mathcal{O}(N/P)$  time. Now each processor has one value and the values must be combined. To do so a balanced binary tree is embedded in the processor array. In  $\mathcal{O}(\log P)$  steps the global sum is computed because such a tree has  $\log P$  levels between the leaves (here the processors) and the root.

A global FFT of length  $N$  can be performed in  $T(N, P) = \mathcal{O}((N/P)[\log N/P + k_0 \log P])$  time. The easiest way to see this is to make use of the butterfly pattern of data combinations in the FFT[56]. The butterfly has  $\log N$  stages, the last  $\log P$  stages being performed by actual communication. Thus the first  $\log N - \log P = \log(N/P)$  stages must be performed

in-processor. Each stage (whether in-processor or not) requires a constant amount of work for each of  $N/P$  data values. Thus the FFT is completed in  $\mathcal{O}(N/P)[\log(N/P) + k_0 \log P]$  time, as claimed.

Packing the wavefunction data can be accomplished in  $T(N, P) = \mathcal{O}(N/P)$  time in our abstract model, since there are  $N/P$  words of data per processor, and the network can route one word per cycle.

All of these segments represent scalable algorithms, in the sense suggested earlier. The serial times for the three kernels are  $t(N) = \mathcal{O}(N)$  for computing a sum,  $t(N) = \mathcal{O}(N \log N)$  for the FFT, and  $t(N) = \mathcal{O}(N)$  for the packing. Thus up to logarithmic factors they perform as well to within constant factors as the best serial algorithms for the same tasks.

Next the parallel modules will be described. For the computationally intensive modules we give an estimated  $T(N, P)$  based on the analysis of the communication-dependent pieces given above.

**Wave function initialization.** This is accomplished by setting each plane wave coefficient  $c_{i, \mathbf{k} + \mathbf{G}}$  to a random number using a parallel random number generator. Wave function initialization requires  $\mathcal{O}(N_G N_B N_k / P)$  time on  $P$  processors. The initialization is completed by orthonormalizing the wavefunction using the Gram-Schmidt orthogonalization scheme described below.

**Charge density computation.** The wavefunction  $c_{i, l(\mathbf{k} + \mathbf{G})}$  contains all three-dimensional reciprocal lattice vectors  $\{ \mathbf{G} \}$  with kinetic energy below the cutoff energy packed into a one-dimensional array. The wavefunction is unpacked and distributed on a full three-dimensional real space grid before it is Fourier transformed to real space. The charge density is com-

puted by unpacking each reciprocal space band contained in the wavefunction, transforming to real space and summing the contributions over all bands

$$\rho(\mathbf{r}) = \sum_n \left| \sum_{\mathbf{G}} e^{i\mathbf{G} \cdot \mathbf{r}} c_{n,I(\mathbf{k}+\mathbf{G})} \right|^2 \quad (3.20)$$

The address indirection  $I(\mathbf{k} + \mathbf{G})$  determines the unpacking order. This vector maps the packed non-zero values of the plane wave coefficients onto the full three-dimensional FFT grid. Points not stored in the packed wavefunction are left set to zero. This process is implemented through the `scatter` function.

A factor of two acceleration was obtained by leaving the partial sum in Eqn. (20) in bit-reversed order. Again due to communication losses, shuffling the final result of an FFT from bit-reversed to normal order costs roughly as much as doing the FFT. There is no arithmetic difference in performing the summation in bit-reversed order, then unreversing  $\rho(\mathbf{r})$  at the end of the computation. The charge density computation time is dominated by the FFT's and sum calculations. Based on the estimates for these primitives given above, the time is  $\mathcal{O}(N_B N_k \{N_X N_Y N_Z / P [k_0 + k_1 \log P + \log(N/P)] + k_2 \log P\})$ .

**Hartree potential.** This module computes the Coulomb potential

$$V_H(\mathbf{G}) = \frac{\rho(\mathbf{G})}{|\mathbf{G}|^2} \quad (3.21)$$

and the Coulomb energy

$$E_H = \frac{1}{2} \Omega \sum_{\mathbf{G}} \frac{\rho^*(\mathbf{G}) \rho(\mathbf{G})}{|\mathbf{G}|^2} \quad (3.22)$$

where  $\Omega$  is the volume of the unit cell. By spreading the values of  $\mathbf{G}$  over the processor array, this computation proceeds efficiently since no communication is required except in the final summation in computing  $E_H$ .

**Exchange-correlation potential.** The Perdew-Zunger parameterized correlation energies [57] are computed in parallel at each grid point. The performance is similar to the Hartree case.

**Local pseudopotential.** This is computed according to

$$V_L(\mathbf{G}) = \sum_{i=1}^{N_S} S(i, \mathbf{G}) U_{PS}(\mathbf{G}) \quad (3.23)$$

The sum is over the  $N_S$  different atomic species in the system. For the Si(111) reconstruction, this sum is over one species only. Computation of  $U_{PS}(\mathbf{G})$  involves iteration over all values of  $\mathbf{G}$  with an interpolation in a large pseudopotential lookup table. The execution of this step was deliberately left inefficient because it consumes a very small fraction of the total processing time. Allocating memory for large temporary arrays in order to accelerate this function was not desirable since available memory resources rather than throughput ultimately limits the size of systems that can be modeled with the Car-Parrinello algorithm on the CM-2.

**Kleinman-Bylander nonlocal pseudopotential.** The contribution to the product of the Hamiltonian and the wavefunction  $\psi_i$  at wave vector  $\mathbf{k} + \mathbf{G}$  for the Kleinman-Bylander pseudopotential is given by [58]

$$\frac{2\omega(\mathbf{k})}{\Omega_0} \sum_{\mathbf{G}'} \sum_{\alpha} e^{-i2\pi\mathbf{R}_{\alpha} \cdot (\mathbf{G}-\mathbf{G}')} V_{\alpha}(\mathbf{k} + \mathbf{G}, \mathbf{k} + \mathbf{G}') c_{i, \mathbf{k}+\mathbf{G}'} \quad (3.24)$$

where  $\omega(\mathbf{k})$  is the  $\mathbf{k}$ -point weight,  $\Omega_0$  is the unit cell volume, and

$$\begin{aligned} V_{\alpha}(\mathbf{k} + \mathbf{G}, \mathbf{k} + \mathbf{G}') &= \frac{4\pi}{\eta_0^{\alpha}} f_0^{\alpha}(|\mathbf{k} + \mathbf{G}|) f_0^{\alpha}(|\mathbf{k} + \mathbf{G}'|) \\ &+ \frac{12\pi}{\eta_1^{\alpha}} \frac{\mathbf{k} + \mathbf{G}}{|\mathbf{k} + \mathbf{G}|} f_1^{\alpha}(|\mathbf{k} + \mathbf{G}|) \cdot \frac{\mathbf{k} + \mathbf{G}'}{|\mathbf{k} + \mathbf{G}'|} f_1^{\alpha}(|\mathbf{k} + \mathbf{G}'|) \end{aligned} \quad (3.25)$$

if  $\delta V_{l=2} = 0$ . In Eqn. (25),

$$\eta_l = \int_0^{\infty} r^2 dr \delta V_l(\phi_{lm}^0)^2 \quad (3.26)$$

and

$$f_l(q) = \int_0^\infty r^2 dr j_l(2\pi qr) \delta V_l(r) \phi_{lm}^0(r). \quad (3.27)$$

$j_l$  is the spherical Bessel function of order  $l$ . The spherical Bessel function  $j_l(2\pi qr)$  gives the amplitude of the  $l$  angular momentum component of the plane wave  $\exp[i2\pi qr]$  a distance  $r$  from the origin.

In this module, it is convenient to compute the partial sums

$$g_{l\alpha nk} = \sum_{\mathbf{G}'} e^{-i2\pi \mathbf{R}_\alpha \cdot (\mathbf{G} - \mathbf{G}')} f_l^\alpha(|\mathbf{k} + \mathbf{G}'|) c_{n, \mathbf{k} + \mathbf{G}'} \quad (3.28)$$

appearing in Eqns. (24) and (25). This module consumes a substantial fraction of the total processing time. It consists entirely of dot products over the plane wave sets per band. The dot product is computed for every band and for every atom in the system. The time for this segment is  $\mathcal{O}(N_k N_A N_B [N_G/P + k_0 \log P])$ .

**Wave function update.** This module begins by convolving the wavefunction with the total local potential

$$B_{i, \mathbf{k} + \mathbf{G}} = \sum_{\mathbf{G}'} V_T(\mathbf{G} - \mathbf{G}') c_{i, \mathbf{k} + \mathbf{G}'} \quad (3.29)$$

The convolution is implemented by Fourier transforming the wavefunction, multiplying it by the Fourier transform of the local potential, then inverse Fourier transforming the result. The throughput of this computation is doubled by exploiting the lower communication costs associated with leaving FFT results in bit-reversed order. To do so, the local potential is first shuffled into bit-reversed order. Then when the wavefunction is Fourier transformed, it is also left in bit-reversed order to align the data for multiplication by the local potential. When the bit-reversed product is then inverse Fourier transformed, communication time is less when the result is left in real order. Avoiding the shuffles from bit-reversed to normal

order saves a factor of two in total FFT processing time. A further factor of two savings is realized for inversion symmetric systems by packing two bands from the real wavefunction together into a single complex input for the convolution. The real and imaginary parts of the result correspond to two independent real convolutions. The time for this convolution is  $\mathcal{O}(N_k N_B N_X N_Y N_Z / P [k_0 \log P + \log(N/P)])$  on  $P$  processors.

Computation of the nonlocal Kleinman-Bylander pseudopotential is next completed according to Eqns. (24) and (25) in terms of the previously computed partial sums  $g_{l\alpha k}$  with the parenthesized sum previously computed. This step takes  $\mathcal{O}(N_k N_A N_B N_G / P)$  time.

Finally, the wavefunction plane wave coefficients are updated according to Eqn. (15). This step consists of numerous operations performed in parallel over every plane wave in each band. The update requires  $\mathcal{O}(N_k N_B N_G / P)$  time.

This module iterates over the bands, performing calculations consecutively on each band, with the exception of the convolution where two adjacent bands are packed together for more throughput.

**Wave function orthogonalization.** This is accomplished according to the Gram-Schmidt scheme:

$$\psi_i'' = \psi_i - \sum_{j < i} \langle \psi_j' | \psi_i \rangle \psi_j' \quad (3.30)$$

with

$$\psi_i' = \frac{\psi_i''}{\|\psi_i''\|}. \quad (3.31)$$

This step probably presents the least opportunity for parallelism of all of the major computational functions since orthogonalization of band  $i + 1$  cannot begin until orthogonalization of band  $i$  is completed. High throughput is only possible from this module when there is a

large number of plane waves per processor. This step consists of  $N_k N_B (N_B - 1)$  dot products over the  $G$ -vector components of each band. Thus the time is  $\mathcal{O}(N_k N_B^2 [N_G/P + k_0 \log P])$  on  $P$  processors.

Every few iterations when the electronic coordinates are sufficiently relaxed, several other modules are invoked compute forces on the ions. Efficiency is not as crucial here since the force computations are applied less often.

**Local force.** This is computed according to

$$F_{i,a} = \text{Im}(\sum_{\mathbf{G}} \rho(\mathbf{G}) V_{PS}(\mathbf{G}) \mathbf{G}_i e^{i\mathbf{G} \cdot \mathbf{R}_a}), \quad (3.32)$$

a relatively efficient operation involving products and sums over the reciprocal space grid.

**Nonlocal force.** The contribution to ionic forces from the nonlocal pseudopotential is given by terms of the form

$$\frac{\partial E_{NL}(l=0)}{\partial \mathbf{R}_{\alpha\mu}} = \frac{8\pi}{\eta_0^\alpha \Omega_0} \sum_{n,k} \omega(\mathbf{k}) \text{Re}[(g_{l=0,\alpha nk})^* \frac{\partial g_{l=0,\alpha nk}}{\partial \mathbf{R}_{\alpha\mu}}] \quad (3.33)$$

where

$$\frac{\partial g_{l=0,\alpha nk}}{\partial \mathbf{R}_{\alpha\mu}} = \sum_{\mathbf{G}} 2\pi \frac{\mathbf{k} + \mathbf{G}}{|\mathbf{k} + \mathbf{G}|} e^{-i2\pi \mathbf{R}_{\alpha\mu} \cdot (\mathbf{G})} f_0^\alpha(|\mathbf{k} + \mathbf{G}|) c_{n,\mathbf{k}+\mathbf{G}} \quad (3.34)$$

The terms for  $l = 1$  are similar. The time required for the dot products over the plane waves per band in this module is roughly equivalent to the time required for an entire total energy calculation during an iteration without ionic force computation.

**Ewald energy and forces.** The Ewald energy accounts for ion-ion interactions due to long-range Coulomb forces. When the ions have moved, the Ewald energy for the new configuration must be determined. The total Ewald energy is given by[59]:

$$E_{ION} = \frac{1}{2} \sum_{i,j} Z_i Z_j e^2 \left\{ \sum_{l'} \frac{\text{erfc}(\eta |\mathbf{R}_1 + \mathbf{l}' - \mathbf{R}_2|)}{|\mathbf{R}_1 + \mathbf{l}' - \mathbf{R}_2|} - \frac{2\eta}{\sqrt{p}} \delta_{ij} \right\}$$



$$+ \frac{4\pi}{\Omega} \sum_{\mathbf{G} \neq 0} \frac{1}{|\mathbf{G}|^2} e^{-\frac{|\mathbf{G}|^2}{4\eta^2}} \cos[(\mathbf{R}_1 - \mathbf{R}_2) \cdot \mathbf{G}] - \frac{\pi}{\eta^2 \Omega} \} \quad (3.35)$$

where  $Z_i$  and  $Z_j$  are the valences of ions  $i$  and  $j$  respectively and  $\text{erfc}$  is the complementary error function.

This computation is implemented in two different ways. It can be done “off-line” as a separate process with the results stored into a file, or it can be parallelized and included in the CM-2 Car-Parrinello program. In the latter case, more memory is needed to make the computation run efficiently on the processing array. The trade-off then is memory against the convenience of not having to compute off-line.

Table 3.4 shows the allocation of processing load among the major functions both algebraically and in terms of actual numbers for a  $7 \times 7$  calculation at an 8 Ry cutoff energy. The nonlocal pseudopotential, wavefunction update, orthonormalization and charge density consume almost all of the processing time. On iterations where forces are computed, the nonlocal force computation adds an additional 60 percent to the processing load. However, since this function is called only once every five to ten iterations, its fraction of total processing time is small. Noting that generally  $N_G$ ,  $N_B$ , and the product  $N_X N_Y N_Z$  all grow linearly with the number of atoms  $N_A$ , and neglecting logarithmic factors, we find that the overall algorithm scales as  $\mathcal{O}(N_A^3/P)$  on  $P$  processors. The algorithm is thus scalable and is well suited to handling larger numbers of atoms and running on larger parallel machines. In Section V we will show that the  $N_A^3$  scaling is beginning to be approached for 500 atoms and a  $P = 1024$  floating point processor CM-2.

Table 3.4: Approximate allocation of processing load among the major computational functions.

Module	Processing Load (FLOPS)	MFLOP's at 8 Ry
Hartree	$43N_X N_Y N_Z$	.04
Exch.-corr.	$165N_X N_Y N_Z$	.17
Local pseudop.	$5N_X N_Y N_Z \log_2(N_X N_Y N_Z)$	.11
NL pseudop.	$10N_A N_B N_k N_G$	127.62
Update $\psi$	$N_k N_B ((N_X N_Y N_Z) 5 \log_2(N_X N_Y N_Z) + 15 N_G N_A)$	290.82
Orthonorm. $\psi$	$2N_k N_G (N_B)^2$	57.27
Compute $\rho$	$5N_k N_E N_X N_Y N_Z \log_2(N_X N_Y N_Z)$	102.41

### 3.3 Performance

The performance of the 8 Ry computation on a 16K CM-2 processor array is shown in Fig. 3.5. Among the modules computed on every iteration, the Hartree energy runs the fastest. This computation of  $V_H(\mathbf{G}) = \rho(\mathbf{G})/|\mathbf{G}|^2$  requires virtually no communication since each value of  $V_H(\mathbf{G})$  is computed locally in the processor containing the corresponding value of  $\mathbf{G}$ . The large size of the  $\mathbf{G}$  lattice also contributes to the efficiency since one can easily amortize overhead for a function computing  $128 \times 128 \times 64$  lattice points. The exchange-correlation energy computation is efficient for similar reasons. However, neither one of these two fast modules performs a significant fraction of the total processing work load. The local pseudopotential does not run very fast, but it is not worth wasting memory to optimize since it does not consume a significant amount of processing time.

Fig. 3.6 shows the distribution of work load and processing time among the different modules in the computation. The nonlocal pseudopotential, wavefunction update, orthonormalization and charge density computation form the computational burden of the

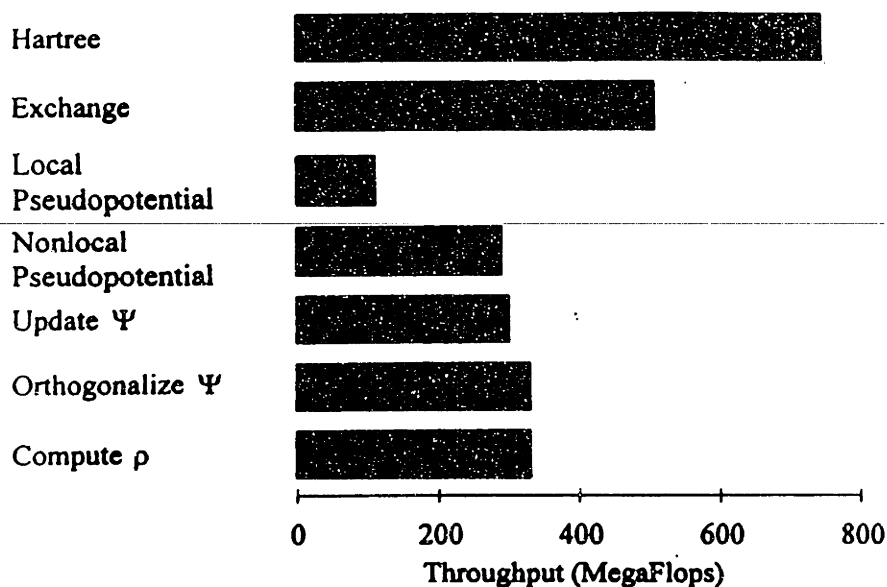


Figure 3.5: Algorithm performance for a 8 Ry Si(111)- $7 \times 7$  calculation on the CM-2.

Car-Parrinello algorithm. To obtain efficient performance, it was necessary to optimize each of these functions. Fig. 3.5 showed that each of these modules runs at roughly 300 MFlops for an 8 Ry Si(111)- $7 \times 7$  calculation on a 16K section of the CM-2. The similarity in throughput among the four modules is somewhat surprising since they perform different mixes of convolutions, Fourier transforms and dot products. As Fig. 3.5 shows, the distribution in processing time consumed by each major function corresponds closely to the distribution of processing load performed by the function. In this respect, the implementation is balanced with no outstanding bottlenecks.

To further investigate how performance scales with the size of a material system, performance was measured for processing loads associated with different sizes of bulk silicon lattices ranging from a cube of side 5.43 Å containing 8 atoms up to a cube of side 21.72 Å containing 512 atoms. For very large numbers of atoms, the processing load scales roughly as the cube of the number of atoms in the system. If the number of atoms in the system

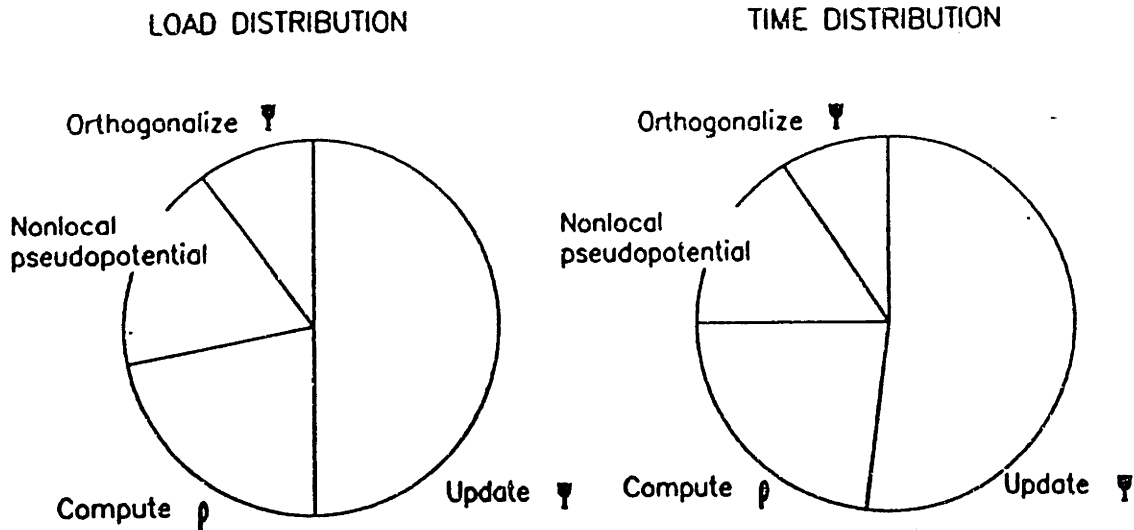


Figure 3.6: Allocation of processing load and processing time among the major computational functions.

is doubled, then one of the dimensions in the unit cell will also double as will the number of bands. The simultaneous doubling of atoms, bands and plane wave coefficients increases the load by a factor of eight.

The processing time for different numbers of atoms in a 12 Ry calculation was measured on the CM-2 at the Pittsburgh Supercomputing Center. The results fall into two domains. Before the problem is large enough to fill the memory of the parallel processing array, the total processing time scales as the cube of the number of atoms. In this domain, the system is communication-bound and parallelism is not effective in accelerating the computation. The arrow in Fig. 3.7 marks the point where a 12 Ry calculation saturates a single 8K quadrant of the Pittsburgh CM-2, corresponding to about 64 atoms. Beyond this problem size, one can efficiently allocate larger processing arrays for larger physical systems. Now, the processing time increases more slowly than the processing load because the processing efficiency increases. This behavior is confirmed in Fig. 3.8. The processing

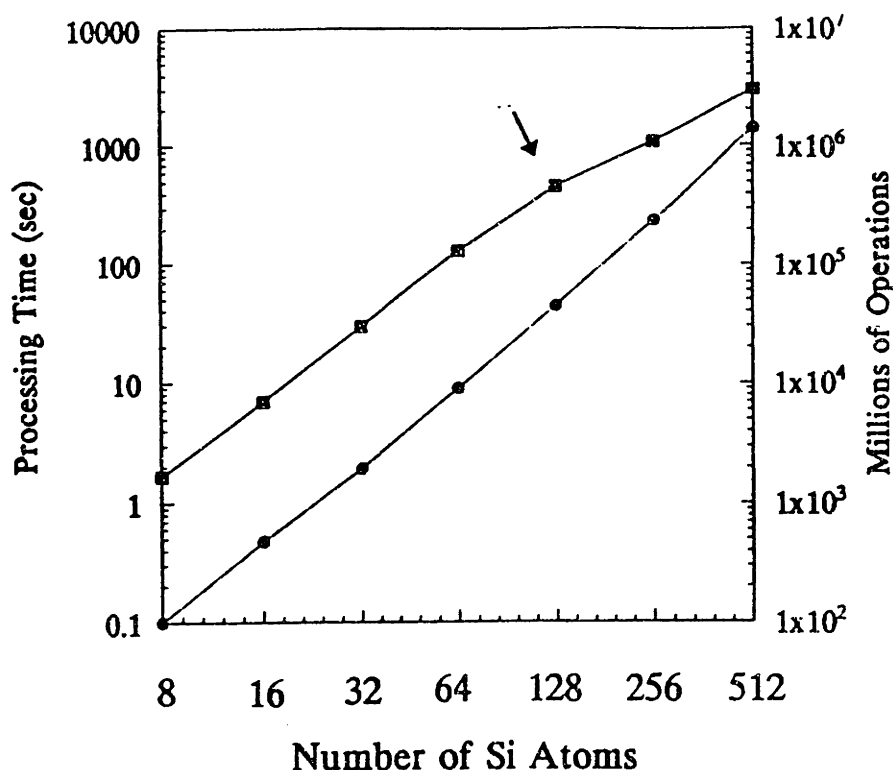


Figure 3.7: Processing time and processing load vs. number of bulk Si atoms at 12 Ry cutoff energy.

rate is approximately constant at 70 Megaflops for systems with 64 or less atoms. As the number of atoms exceeds this value, the throughput rate increases substantially. For 512 atoms on the entire 32K processing array at Pittsburgh, the throughput approaches 500 Megaflops. Of course, the required computational load also increases rapidly when one attempts to increase the number of atoms in the system. But the results depicted in Fig. 3.8 show that adding more processors in parallel will increase the throughput in order to handle larger systems.

The performance enhancement is more favorable when the cutoff energy is increased for a fixed number of atoms. Due to the soft pseudopotential for silicon and the use of nonlocal pseudopotentials, silicon calculations may be performed with an unusually low cutoff energy. A larger cutoff energy is necessary in working with transition elements and first-row elements. The computational load increases roughly as  $E_c^{\frac{3}{2}}$ , where  $E_c$  is the cutoff energy. For 256 Si atoms, roughly 500 billion floating point operations are necessary to

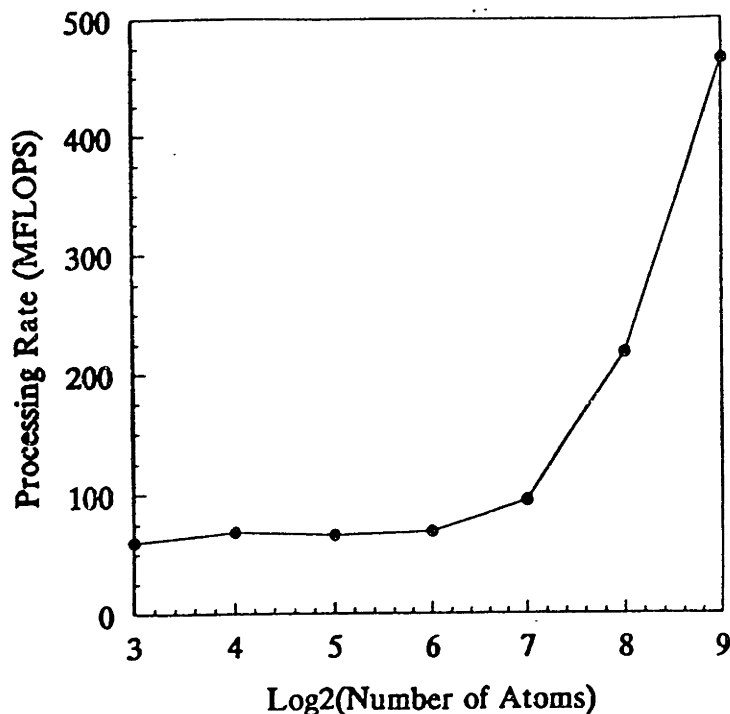


Figure 3.8: Processing rate vs.  $\log_2(N)$ , where  $N$  = number of bulk Si atoms.

perform each iteration in this cutoff energy range. Fig. 3.9 shows that the processing time increases much more slowly than the computational load as the cutoff energy increases. The drop in processing time as the cutoff energy increases from 20 to 24 Ry illustrates the effect of doubling the size of the processor array. At 20 Ry and below, the calculation fits on 16K processors. The processing time per iteration drops from 1200 to 750 seconds when the system no longer fits on 16K processors and the processor array size is doubled. Fig. 10 shows the corresponding throughput values. A 512-atom calculation at a plane wave energy saturating a 32K Connection Machine approaches 750 Megaflops. Moving to a larger processing array to handle systems with more plane waves in a given volume is an efficient use of parallelism. These results show that the largest throughput rates will be obtained for problems with a relatively high cutoff energy and a relatively low number of atoms.

The parallel implementation of the Car-Parrinello algorithm demonstrates that mas-

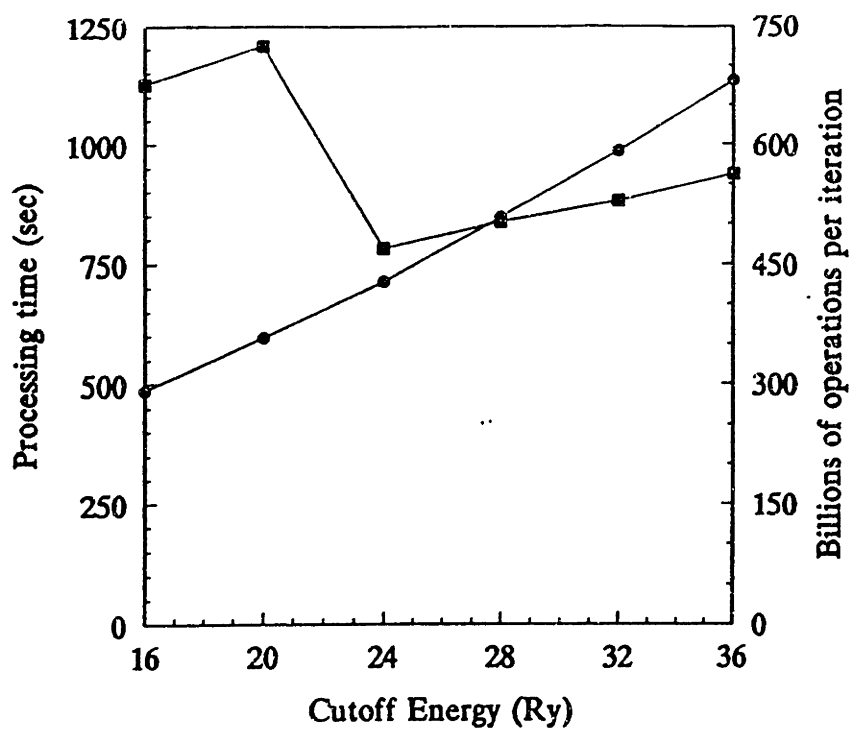


Figure 3.9: Processing time and processing load vs. cutoff energy for 256 bulk Si atoms.

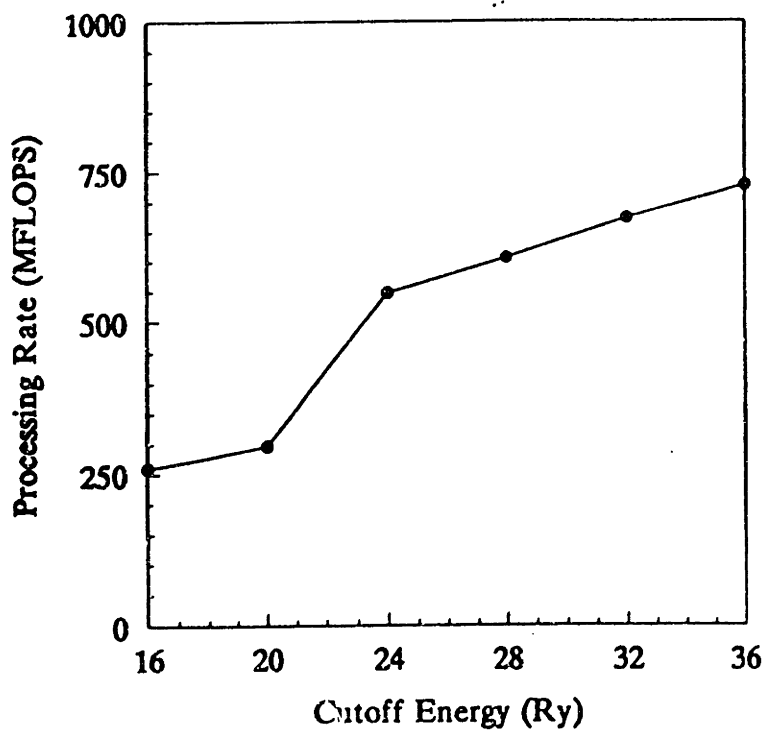


Figure 3.10: Processing rate vs. cutoff energy for 256 bulk Si atoms.

sively parallel computers enable one to perform *ab initio* total energy calculations on systems that are too large for treatment with conventional vector-pipelined supercomputers. As one attempts to work with larger numbers of atoms in a material system, adding more processors in the parallel computation increases the throughput rate. Massively parallel computers become increasingly efficient with larger computations, so that the processing time increases much more slowly than the increase in the computational work load. This result also demonstrates that building larger parallel processing arrays would further reduce the processing time. We found that the efficiency of massive parallelism is most favorable when there is a large number of plane waves relative to the number of bands. A version of this chapter is published in reference [60].



## Chapter 4

# Application to the Si(111)-(7 × 7) Surface Reconstruction

The size and speed of massively parallel supercomputers has raised the complexity of material systems that can be modeled using *ab initio* quantum-mechanical techniques to a new echelon. In this section, we use our parallel Car-Parinello to show what this statement really means by demonstrating the feasibility of performing *ab initio* calculations with supercells approaching one thousand atoms. Specifically, we study the Si(111)-(7×7) reconstruction using a supercell geometry with 700 effective atoms.

Semiconductor surface structures can be divided into two broad categories[61]. The first of these consist of structures which conserve the size and shape of the original two dimensional unit cell. These are referred to as unreconstructed structures. The second category corresponds to atomic displacements which modify the shape and size of the unit cell. These distortions, referred to as reconstructions, are found on all faces of elemental semiconductors. Another characteristic common to a large fraction of the known semiconductor surface structures is the fact that atomic displacements are not confined to the surface layer.

Several semiconductors display substantial multilayer atomic displacements [62]. This extra degree of complexity further separates most semiconductors from metals which generally are characterized by surface geometries described in terms of small uniform contractions or expansions of the top atomic plane alone, without change in the space group symmetry of the surface.

The  $(7\times 7)$  reconstruction of Si(111) is perhaps the most complex and widely studied surface of a solid. Since its discovery through low-energy electron diffraction more than thirty years ago[63], an enormous amount of effort has been expended to elucidate the properties of this important surface[64]-[76]. Based on this work, it is now generally accepted that the geometry of the  $(7\times 7)$  reconstruction is described by a dimer-adatom-stacking fault (DAS) model as proposed by Takayanagi *et. al.*[68]. The complexity of this geometry, however, has defied any complete and realistic theoretical treatment of its properties. The only progress that could be made theoretically was by isolating and modeling bits and pieces of the surface. The only attempts at a complete work has been using an empirical tight binding model to study the Si(111)- $(7\times 7)$  reconstruction in a supercell geometry with 196 atoms[75] and an *ab initio* calculation at very low cutoff energy[77].

The calculations presented in this chapter predict the relaxed atomic geometry of this system; allow construction of theoretical STM images as a function of bias voltages; and predict the energy difference between the  $(7\times 7)$  and  $(2\times 1)$  reconstructions.

## 4.1 DAS Model

The DAS model for the Si(111)-(7×7) surface reconstruction is shown in Fig. 4.1. The top view is shown in figure 4.1(a). A dashed line outlines the unit cell boundary. Atoms at increasing distance from the surface are indicated by circles of decreasing size. The largest black circles denote the twelve adatoms. The large shaded circles denote six rest atoms that lie one layer below the surface and are three-fold coordinated. There is also a single corner hole in each unit cell, marked with a small shaded circle. Together, these nineteen atoms account for the nineteen unbonded electrons on the reconstructed surface. Hollow circles mark the remaining atoms on the first bilayer below the surface. The dimer pairs surrounding each adatom triangle are an important feature of the first bilayer. Nine dimers are present in each unit cell. The small black circles mark the atoms in the second bilayer below the surface on the unfaulted half of the unit cell. The stacking sequence in the right half of the unit is the same as in bulk Si while the stacking sequence in the left half is faulted.

The side view (b) through the long diagonal emphasizes the depth of corner hole with its dangling bond one full bilayer below adatom surface. The deep corner hole and (7×7) periodicity of the unit cell necessitates a large supercell to accurately calculate the DAS model.

## 4.2 Method of Calculation

Our surface model consists of a supercell with a slab geometry containing vacuum on both sides. Periodic boundary conditions are applied in all directions. Figure 4.2 illustrates

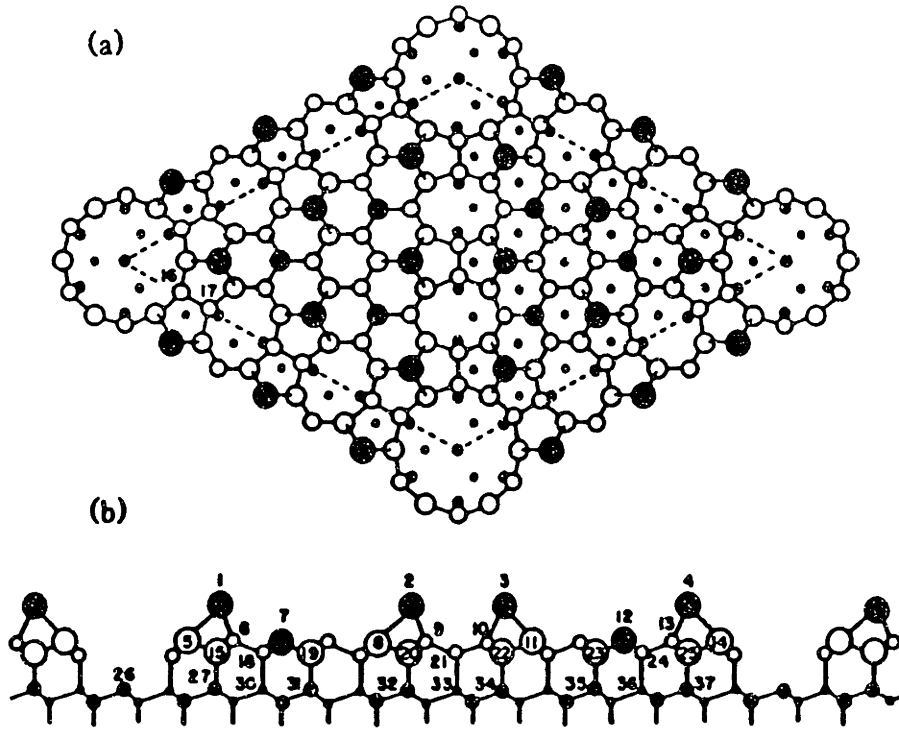


Figure 4.1: The DAS model for the Si(111)-(7 $\times$ 7) surface reconstruction.

the scope of the slab approximation. The slab contains the adatom layer and four surface layers shown in Fig.4.1(b) plus a mirror image reflection in the vertical direction. Thus the supercell consists of eight layers of atoms with adatom layers on both slab surfaces and a region of 10Å of vacuum. The 400 silicon atoms and the vacuum layer make this supercell equivalent to a 700-atom system. Unit cell dimensions are 26.6Å  $\times$  26.6Å  $\times$  22.5Å.

We used the *ab initio* molecular dynamics scheme for calculating total energies and performing simulated quenching[46],[51]. To compute the total energy, we used the Car-Parrinello algorithm described in the previous chapter. Due to the large size of our unit cell, only the  $\Gamma$  point of the Brillouin zone was required for k-point sampling.

All atoms were allowed to move freely except the innermost two layers, which were frozen in bulk positions. These atoms are marked by hollow circles in Fig. 4.2. Thus our model of the surface includes the relaxation of three surface layers in addition to the adatom layer. The atoms were assumed to be in their fully relaxed positions when the forces on the

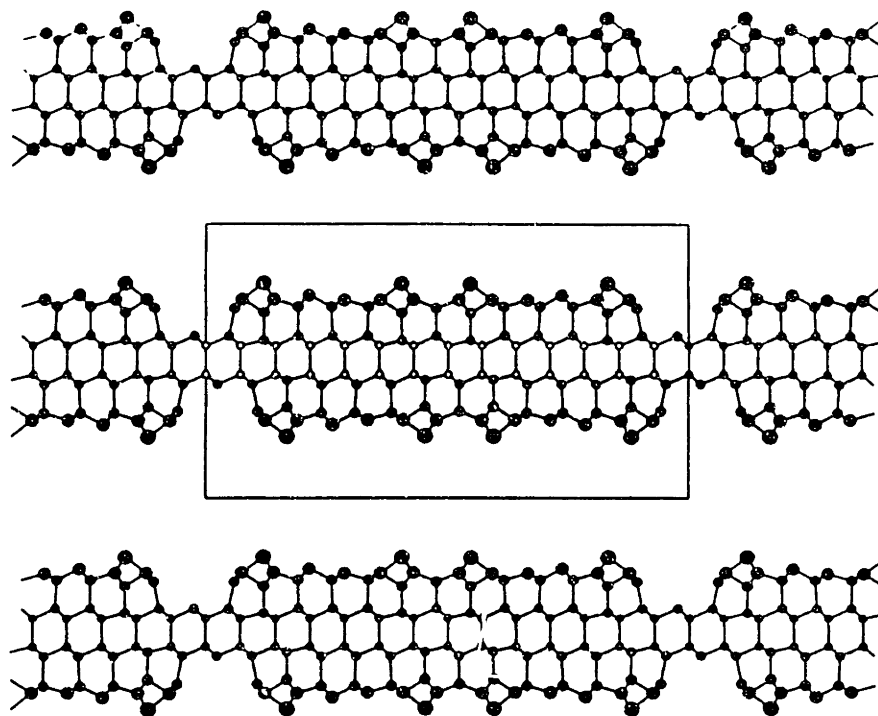


Figure 4.2: Vertical cross section through the plane of the long diagonal of the  $(7 \times 7)$  unit cell with periodic extensions illustrating the scope of the calculation.

ions were converged to  $0.15 \text{ eV/\AA}$ .

To check for convergence, we performed the calculation at 4, 5, 6 and 8 Ry. The 8 Ry calculation requires a basis set size of about 36,000 plane waves per unit cell. Electronic states were computed for 1,000 bands. The unit cell was sampled over a  $64 \times 128 \times 128$  lattice. Exclusive of temporary workspace, the calculation requires 400 million bytes of memory. Computation of the wave function and relaxed ionic positions requires  $\sim 10^{14}$  floating operations, equivalent to running a supercomputer at one billion floating point operations per second for one week. With a massively parallel Connection Machine CM-2 supercomputer, these performance requirements are currently feasible.

### 4.3 Surface Energy

The  $(7\times 7)$  surface reconstruction is experimentally observed to be the most energetically favorable Si(111) geometry. We compared the formation energies of the  $(7\times 7)$  reconstruction with the other experimentally observed Si(111) phase, the  $\pi$ -bonded  $(2\times 1)$ [78]. Cleavage of Si to create the (111) surface results in a  $(2\times 1)$  metastable structure. This surface must then be annealed to generate the stable  $(7\times 7)$  structure. The surface energy per  $(1\times 1)$  cell for both structures is determined from the slab energy by

$$E_{surf} = E_{slab} - N_{atom}E_{bulk} \quad (4.1)$$

where  $E_{bulk}$  is the energy of bulk silicon computed with an equivalent k-point scheme,  $N_{atom}$  is the number of atoms in the slab and  $E_{slab}$  is the slab energy. To compute the energy of the  $(2\times 1)$  surface, we used sixteen k points in the irreducible Brillouin zone that correspond exactly to the k point  $\Gamma$  in the  $(7\times 7)$  unit cell. The equivalent bulk computation required 49 k points. At an 8-Ry cutoff energy, we find that the  $(7\times 7)$  reconstruction is energetically favorable over the metastable  $(2\times 1)$  surface by 60 meV per  $(1\times 1)$  cell. Figure 4.3 plots the surface energy of the  $(2\times 1)$  and  $(7\times 7)$  reconstructions as a function of cutoff energy. The curves flatten significantly above 6 Ry, indicating that the 8 Ry calculation is close to the converged energy value.

### 4.4 Ionic Coordinates

The coordinates for all relaxed atoms in the slab are given in table 4.1. The coordinate system corresponds to the tight-binding results of Qian and Chadi[75]. All reduced coordinates  $(X, Y, Z)$  are with respect to the Cartesian system indicated in Fig. 1, where the

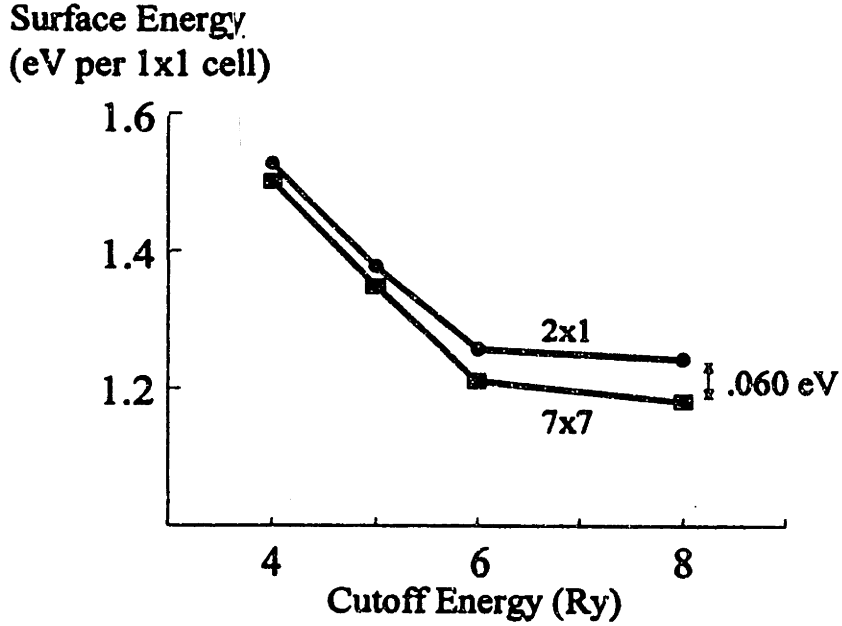


Figure 4.3: The surface energies of  $\pi$ -bonded (2 $\times$ 1) and (7 $\times$ 7) reconstructions vs. plane-wave cutoff energy computed for four-bilayer slabs at equivalent k-points.

x axis is along the cubic [110] direction, the y axis is along the [111] outward normal to the surface. The actual atomic coordinates  $(x, y, z)$  are related to  $(X, Y, Z)$  by the scaling relations  $x = aX, y = aY/\sqrt{3}, z = aZ/\sqrt{24}$ , where  $a \approx 3.85\text{\AA}$  is the (1 $\times$ 1) surface hexagonal lattice constant. We compare the results here with previous experimental and theoretical work.

The degree of relaxation from bulk positions was far less in the horizontal plane parallel to bulk layers than the degree of relaxation in the vertical direction. We found that most bond lengths projected in the surface plane as listed by Robinson *et. al.*[79] agreed within 0.02  $\text{\AA}$  with the semi-empirical calculations of Qian and Chadi[75]. The most significant difference was an average dimer bond length of 2.45  $\text{\AA}$ . Semi-empirical calculations predict a bond length of 2.42  $\text{\AA}$ . LEED and x-ray values are 2.45 and 2.50 respectively.

Vertical displacements are summarized by the average relaxation of each surface layer.

Table 4.1: Relaxed atomic positions for the adatom layer and the first three surface layers for the *ab initio*  $7\times 7$  calculation.

	Atom	$x$	$y$	$z$
Adatoms	1	1.500	1.500	1.662
	2	4.492	4.491	1.594
	3	6.002	6.002	1.555
	4	9.005	9.005	1.603
First-layer atoms	5	1.033	1.033	-0.057
	6	1.958	0.993	-0.058
	7	2.481	2.481	0.348
	8	4.021	4.022	-0.155
	9	4.974	4.050	-0.099
	10	5.980	5.057	-0.135
	11	6.478	6.478	-0.163
	12	8.019	8.019	0.345
	13	9.029	8.059	-0.104
	14	9.472	9.472	-0.111
Second-layer atoms	15	1.504	1.504	-1.512
	16	1.162	0.006	-1.057
	17	1.800	0.007	-1.042
	18	2.492	1.499	-0.867
	19	2.969	2.969	-0.863
	20	4.494	4.495	-1.561
	21	5.403	4.780	-1.046
	22	6.004	6.004	-1.567
	23	7.532	7.532	-0.871
	24	8.505	7.510	-0.874
	25	8.998	8.998	-1.537
Third-layer atoms	26	0.001	0.001	-3.986
	27	1.500	1.500	-4.416
	28	1.006	-0.002	-3.989
	29	1.990	0.000	-3.964
	30	2.499	1.499	-3.946
	31	2.997	2.997	-3.951
	32	4.499	4.499	-4.460
	33	5.496	4.512	-3.973
	34	5.999	5.999	-4.440
	35	7.497	7.497	-3.933
	36	8.505	7.496	-3.930
	37	9.001	9.001	-4.414



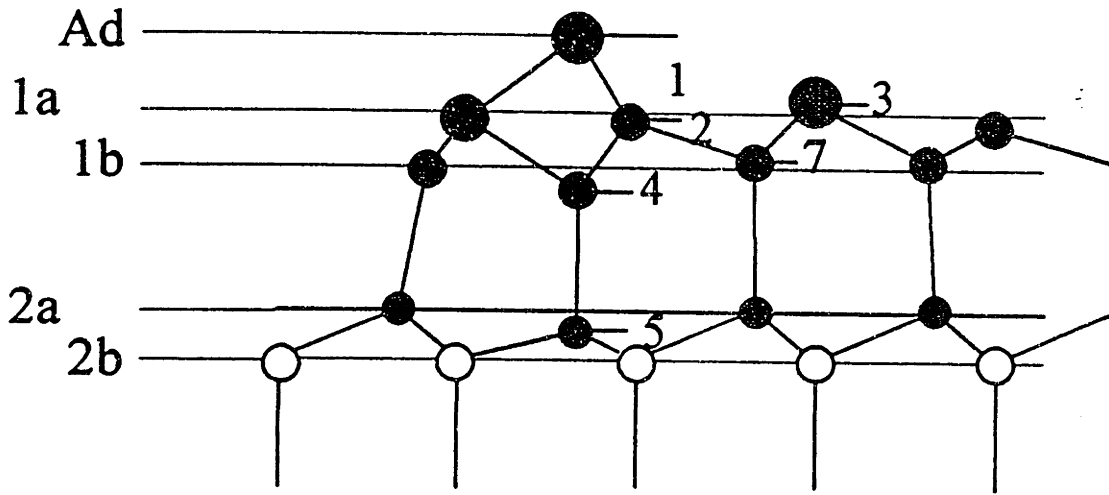


Figure 4.4: Labels for various vertical displacements of interest.

We found that the adatom and top surface layer relaxed outward from their bulk positions, while the second bilayer below was compressed. Figure 4.4 shows a side view of the unit cell labelled according to Robinson and Vlieg [80] for the surface layers and Ichimiya [81] for several interesting vertical displacements. Table 4.2 compares the layer relaxation with previously published x-ray[80], LEED[73],[74] and RHEED[81] experiments, a semi-empirical calculation[75] and an local density approximation (LDA)[76] calculation of the  $(2 \times 2)$  geometry. The slight compression of the two layers in the second bilayer below the surface is seen in all results. All experiments and models also agree that the top surface layers relax apart relative to their bulk values, and that the adatoms move away from the top surface layer. The most outstanding difference among the results is the spacing between the first and second bilayers. The x-ray and  $(2 \times 2)$  LDA calculations report outward relaxations of  $0.06 \text{ \AA}$  and  $0.02 \text{ \AA}$  relative to bulk spacings. LEED and RHEED report compressions of  $0.01 \text{ \AA}$  and  $0.09 \text{ \AA}$  relative to bulk. Tight binding reports no change. We found a compression of  $0.02 \text{ \AA}$ . A limitation on the confidence of this value is the limited number of bilayers in

Table 4.2: Average surface layer displacements.

Spacing	bulk value	present work	x-ray [80]	RHEED [81]	LEED [73]	(2×2) [76]	Semi-emp [75]
Ad-1a	0.78	1.29	1.57	1.23	1.21	1.14	1.25
1a-1b	0.78	0.83	0.85	0.83	0.84	0.80	0.88
1b-2a	2.35	2.33	2.41	2.34	2.26	2.37	2.35
2a-2b	0.78	0.73	0.75	0.68	0.72	0.76	0.66

our calculation. The (2×2) calculation used five bilayers instead of our four, which may account for some of the difference beyond differences in the (7×7) and (2×2) geometries. Experiments[82] report relaxation as deep as the eighth layer.

The adatom relaxation relative to the bulk value of the top surface layer is 0.06 Å more than the RHEED experiment and 0.15 Å more than the (2×2). However, this spacing still does not fall within the range of  $1.58 \pm .20$  reported from x-ray reflectivity data. In another (7×7) LDA calculation, Stich *et. al.* [83] found a slightly higher value of 1.33 Å, which is within error range of our result.

Table 4.3 shows more of the details of vertical displacements. The surface atoms bonded to the adatoms (2) are slightly depressed from their bulk values by 0.08 Å, which is consistent with all published results. During ionic relaxation, the rest atoms (3) rose 0.27 Å above their bulk positions. This value falls between 0.25 Å found by RHEED and the (2×2) LDA calculation's 0.30. The other published (7×7) LDA calculation found a lower value of 0.20 Å. The atoms below the adatoms (4) dropped .43 Å, an amount similar to other published results. The atoms below these atoms (5) dropped 0.34 Å, with 0.09 Å taken as compression of the vertical bond. Compression of these atoms is the fundamental source of the net compression of layer 2a.

A difficult and interesting question surrounding the Takayanagi reconstruction has been

Table 4.3: Average vertical ionic displacements.

Parameter	present work	Stich [83]	RHEED [81]	LEED [73]	Semi-emp [75]	(2x2) [76]
1	1.25	1.29	1.16	1.10	1.23	1.16
2	-0.08		-0.12	-0.11	-0.08	-0.13
3	0.27	0.20	0.25	0.13	0.29	0.30
4	-0.43		-0.45	-0.28	-0.46	-0.49
5	-0.34		-0.41	-0.19	-0.45	-0.40
6	-0.04		-0.08	-0.11	-0.08	
7	0.10		0.09	0.16	0.05	0.17

the magnitude and source of the asymmetry in STM heights of the adatom triangles in the faulted and unfaulted halves of the unit cell. Since the STM height difference varies for occupied and unoccupied states, the effect is thought to be predominantly electronic. However, the stacking fault should also influence the asymmetry by increasing the height of adatoms on the faulted half. Chou, Cohen and Louie[84] found relaxations of  $\sim 0.05$  Å by introducing a stacking fault into bulk Si.

The difference in vertical displacements between faulted and unfaulted halves is shown in table 4.4. The faulted adatom island lies  $0.039$  Å higher than its unfaulted counterpart. The semi-empirical result is  $0.031$  Å. The only experimental result comes from LEED,  $0.08$  Å. The asymmetry is slightly more pronounced for corner adatoms compared to center adatoms, a result differing qualitatively with the semi-empirical calculation where virtually all of the asymmetry was confined to corner adatoms. The rest atom heights were virtually the same on faulted and unfaulted halves, which means that the rest atoms on the unfaulted half are  $\sim .04$  Å higher relative to neighboring adatoms. A similar result was found in the semi-empirical calculation. Moving down into the unit cell, the asymmetry between faulted and unfaulted halves decreases to  $0.026$  Å for the first layer (*1a*) and  $0.017$  Å for the second (*1b*). Experimental LEED values are  $0.050$  for both layers. In this calculation the third layer (*2a*) is slightly lower by  $0.015$  Å for the faulted half due to the finite thickness of the

Table 4.4: Difference in vertical displacements for faulted and unfaulted halves.

Structure	present work	Semi-emp [75]	LEED [73]
Corner adatoms	0.047	0.055	0.080
Center adatoms	0.030	0.007	0.080
Rest atoms	0.003	-0.009	0.050
First layer	0.026	0.016	0.050
Second layer	0.017	-0.003	0.050
Third layer	-0.015	-0.013	0.000

supercell slab.

## 4.5 Overview of Electronic Surface States

Scanning transmission microscopy has been successfully used to probe the electronic surface states of the Si(111) in real space. An *ab initio* model of the surface enhances our understanding of STM data by confirming previous experiments and predicting features of the surface which are difficult to measure experimentally.

According to the method of Tersoff and Hamman[85], the current measured at the tip of an STM probe due to tunneling from surface states is given by a sum over the local-density of states between the bias voltage and the Fermi energy.

$$i_0(\mathbf{r}) = \sum_{E_n=E_0}^{E_F} |\psi_n(\mathbf{r})|^2 \delta(E_n - E_F) \quad (4.2)$$

Adjusting the probe height to maintain constant current then traces out contours of constant local charge density known as topographs. Figure 4.5 compares theoretical and experimental STM topographs of the occupied and unoccupied states at a 2V tip bias. The plots are based on the results of this calculation (top half, (a), (b)), and the experi-

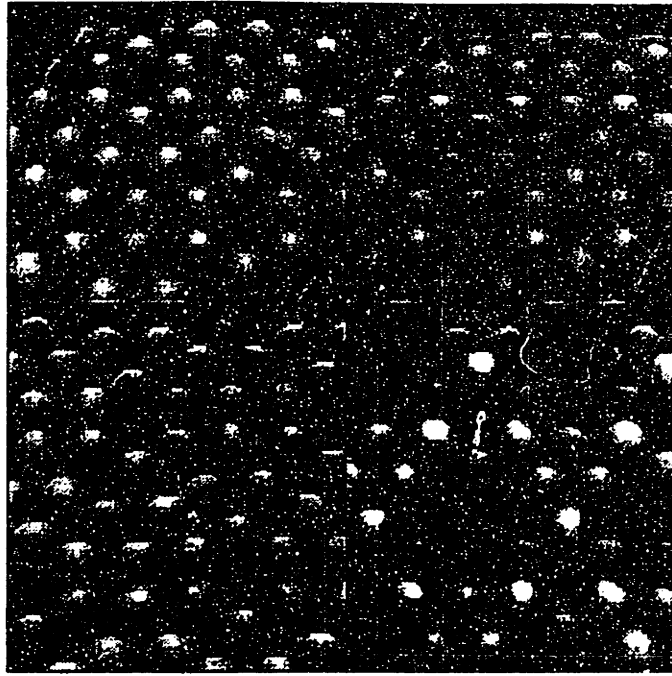


Figure 4.5: STM images of the Si(111)-(7 $\times$ 7) surface reconstruction.

mental results of Avouris and Wolkow[86] (bottom half, (c), (d))). Shown are theoretical unoccupied (a) and occupied (b) state images, computed using the method of Tersoff and Hamman[85], together with corresponding experimental unoccupied (c) and occupied (d) state images. In these semi-quantitative gray-scale plots, a white pixel of maximum intensity corresponds to a point with STM height greater than some maximum threshold, while a black pixel of minimum intensity corresponds to a point with STM height less than some minimum threshold. Points of height between the two thresholds are assigned intensity values linearly proportional to their height above the minimum threshold.

Comparing first the topographs of the unoccupied states, one clearly observes twelve adatoms in each unit cell surrounded by four deep corner holes. The adatoms on the faulted and unfaulted halves appear to be at the same height, resulting in a flat ring of six adatoms surrounding each corner hole.

For the topographs of the occupied electronic states, one observes the same twelve

adatoms per unit cell surrounded by four corner holes, with several additional features present. Rest atoms are now visible as light regions between adatoms, particularly on the unfaulted half of the unit cell. The rest atoms are especially visible on the theoretical topograph. The adatoms now appear at different heights, depending on their location in the unit cell. Adatoms on the faulted half appear higher than their counterparts on the unfaulted half. This effect is particularly noticeable by looking at alternating heights of the six adatoms surrounding each corner hole. Corner adatoms appear higher than center adatoms within the same adatom triangle.

The major differences between theory and experiment surround the visibility of rest atoms and the relative heights of the adatoms. Both differences occur in the images of the occupied states. The differences are due to the finite size of an experimental STM tip compared to the infinitesimal size of a theoretical one. Tersoff and Hamman showed that the finite size of a physical STM tip is taken into account by convolving a "perfect" STM topograph with a gaussian window of width equal  $(R + d)/K$ , where  $R$  is the tip radius,  $d$  is the tip spacing and  $K$  is the exponential surface charge decay length. In order to preserve the sharpness of the features in the theoretical topograph, no such convolution was performed. The images were then made to look qualitatively similar by adjusting the theoretical gray scale range to about 5 Å, roughly five times the experimental range for this particular image. The small tip size gives the theoretical image sufficient resolution to make the rest atoms more visible. The smaller gray scale explains why vertical height differences appear larger in the experimental topograph.

More quantitative data is found by plotting line contours through the long diagonal. Figure 4.6 plots the STM height of both occupied and unoccupied states through the long

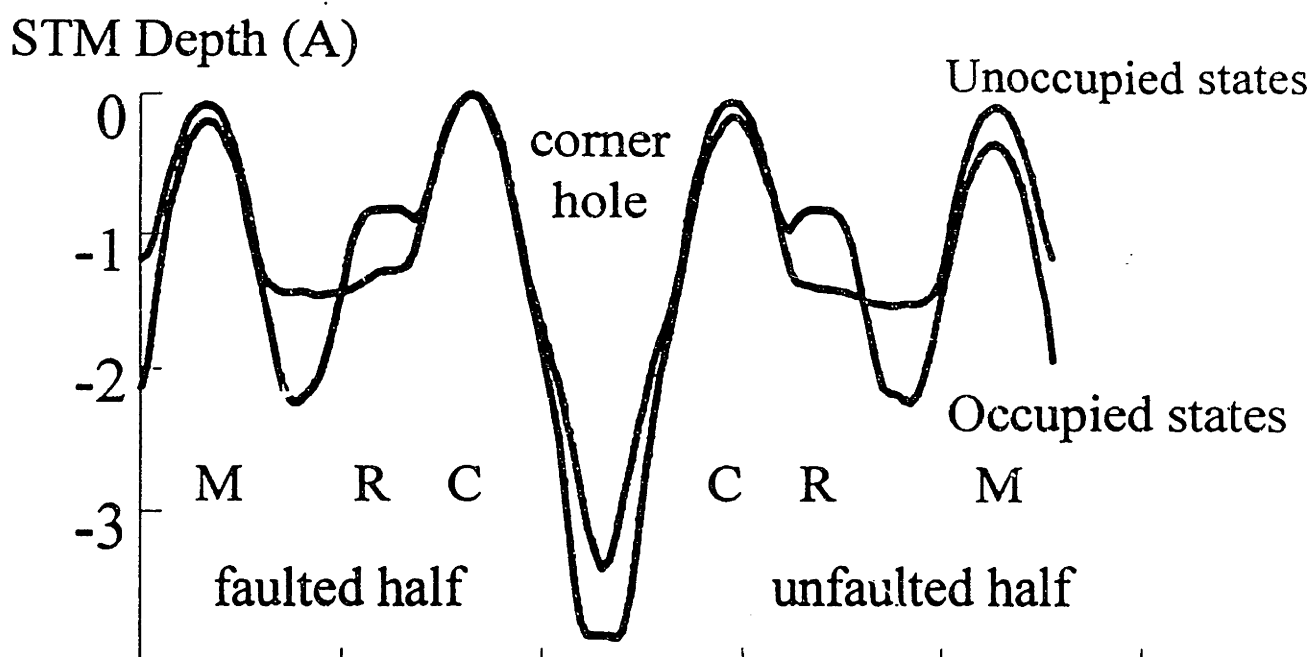


Figure 4.6: Theoretical STM depth vs position through the long diagonal.

diagonal of the unit cell. Tip bias is  $\pm 2$  V. The dashed line shows the unoccupied states. There are four large peaks corresponding to four adatoms along the long diagonal. The difference in height for corresponding peaks on the faulted and unfaulted halves of the unit cell is small, approximately  $0.04\text{\AA}$ , roughly the same amount as the structural height difference between adatoms. The solid line shows the occupied states. In addition to four large adatom peaks, two small peaks corresponding to rest atoms are also visible. The height difference between faulted and unfaulted corner adatoms is  $\sim 0.11\text{\AA}$  which agrees with experiments by Becker *et. al.* [70] and Tromp, Hamers and Demuth [87]. As shown in Fig. 4.7 the height difference increases sharply when the bias voltage drops below 1.5 V, an effect which has also been seen experimentally. The reason for this increase can be understood by more closely studying the energy dependence of the occupied electronic states.

While STM topographs give information about a range of states beginning or ending

# STM Height Difference (Å)

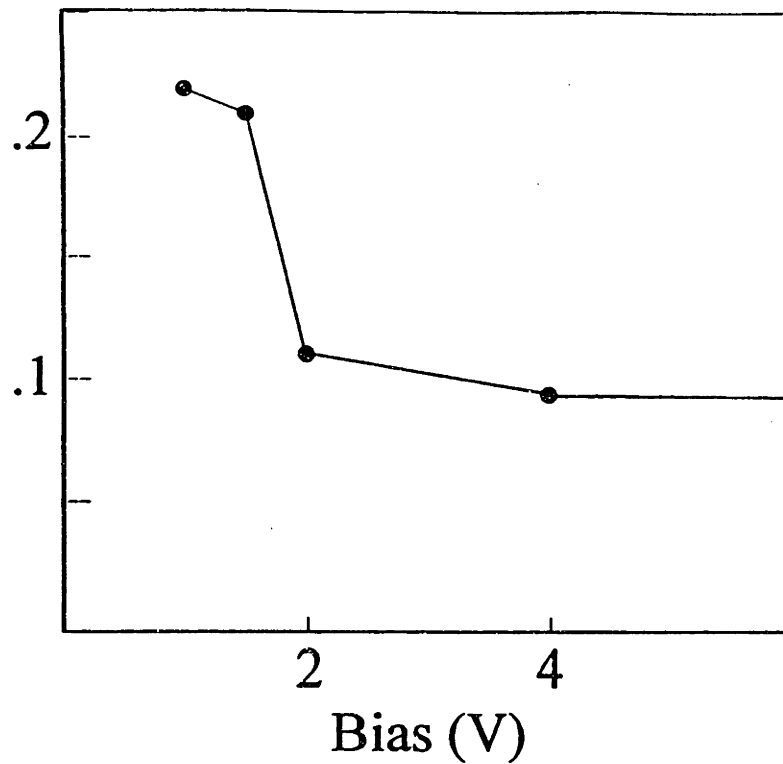


Figure 4.7: Difference in STM height for faulted and unfaulted corner adatoms vs. bias voltage.

at the Fermi energy, it is possible to subtract two topographs in order to find particular surface states. Hamers, Tromp and Demuth[88] experimentally identified three such states: an adatom state at -0.35 V, a rest atom state at -0.8 V and a backbond state at -1.7 V.

Figure 4.8 shows the local density of states between -0.2 V and the Fermi energy throughout a vertical plane along the long diagonal. The surface runs along the upper right edge of plot, where the faulted and unfaulted sides of the surface are labelled. The four shaded peaks near this edge are the adatom dangling bonds. The height of each peak is proportional to  $|\psi_{E_0 < E < E_F}(x, z)|^2$ , where  $(x, z)$  is the location of the point in the vertical plane along the long diagonal. The peaks are larger on the faulted half, indicating more charge on this side of the unit cell. A rest atom peak is also visible on the faulted half at a greater depth into the surface. Surface layers are labelled along the lower right edge. In this energy range near the Fermi energy, the largest peak in the crystal is not on an adatom or a rest atom but on the corner hole. This state is not reported experimentally in STM data.



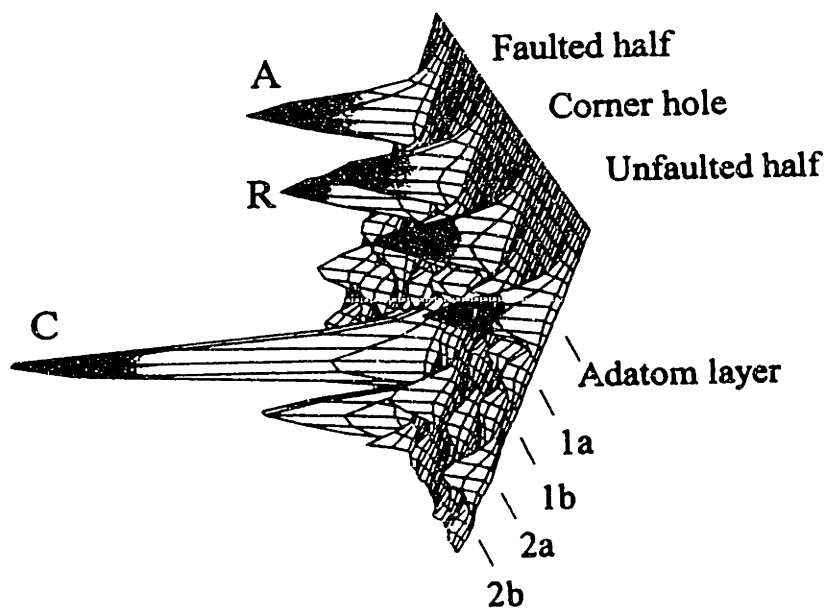


Figure 4.8: Occupied electronic states within 0.2 V of the Fermi energy through the long diagonal plane.

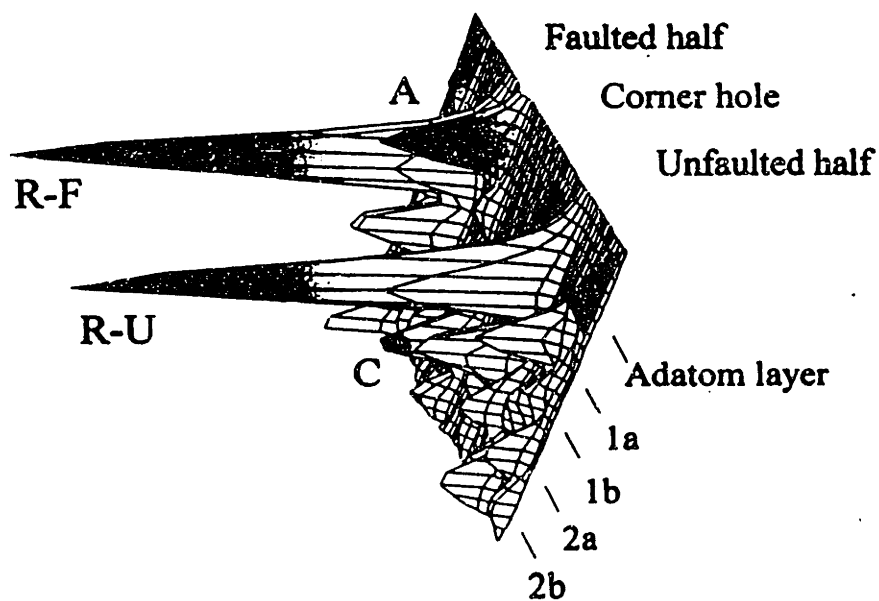


Figure 4.9: Occupied electronic states between 0.9 and 0.7 V below the Fermi energy.

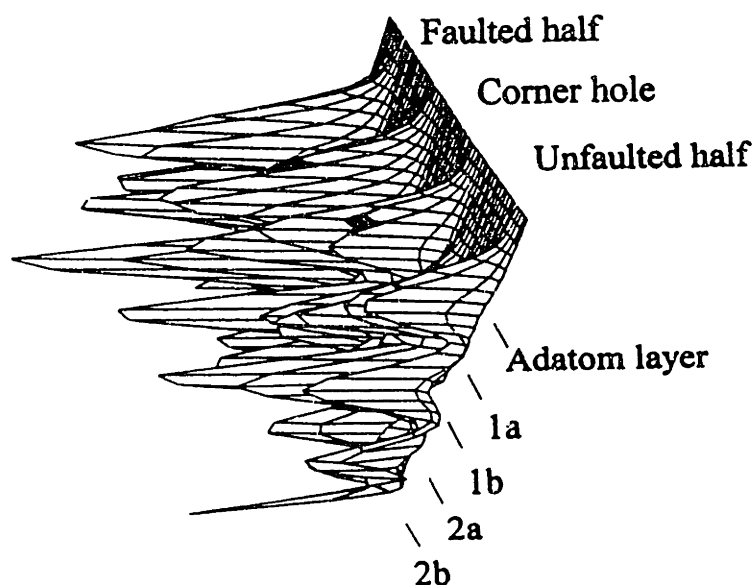


Figure 4.10: Occupied electronic states between 1.7 and 1.6 V below the Fermi energy.

Figure 4.9 shows the local density of states between -0.9 V and -0.7 V. This surface state differs qualitatively with the adatom dangling bond state. The large peaks near on the faulted (R-F) and unfaulted (R-U) rest atoms show that this state is localized on the rest atom dangling bonds. There is a smaller peak on the faulted corner adatom. Fig. 1(D) in Hamers, Tromp and Demuth [88] images the rest atoms as six dots in the unit cell for this energy range. The blurs on the faulted rest atom dots are evidence of the charge on the faulted corner adatom.

Figure 4.10 shows the backbond state near -1.7 V. Now the charge is distributed through the slab with no clearly visible dangling bonds. Small shaded areas near the surface show the remnants of adatom dangling bonds.

The differences between the occupied electronic surface states can be seen by comparing contour plots in Fig. 4.11. The plot shows the adatom state near the Fermi energy (A); rest atom dangling bonds at -0.8 V (B); and adatom backbonds at -1.7 V (C). The area and

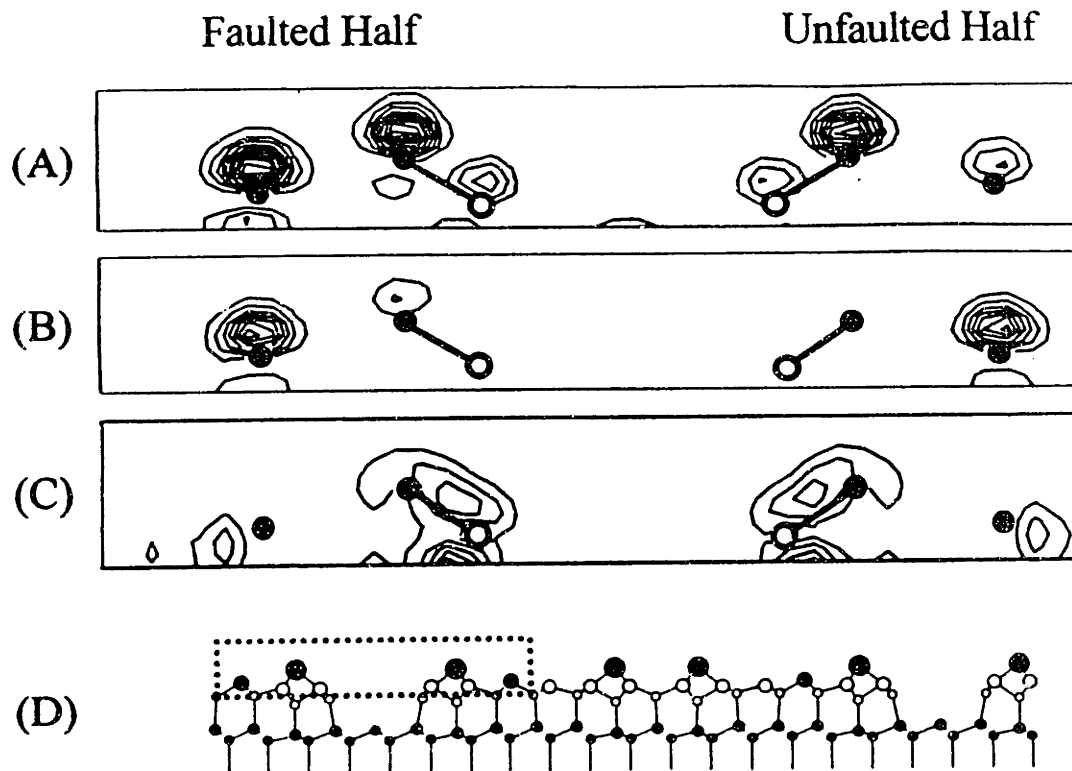


Figure 4.11: Surface contours near the corner hole comparing the three occupied surface states.

orientation of the plots is outlined in (D). These plots show the region of the surface along the long diagonal near the corner hole. For the state near the Fermi energy, large dangling bonds are visible on the corner adatoms. It can also be seen that the dangling bond on the faulted corner adatom is larger and extends further upward than its unfaulted counterpart. There is also a significantly large dangling bond on the faulted rest atom. However, it is more than  $1\text{\AA}$  deeper into the surface. The corner hole is especially shallow at this energy due to the large corner hole state, one contour of which is visible. This contour is nearly  $2.5\text{\AA}$  above the corner hole atom, about twice as high as the other dangling bonds. The state at  $-0.8\text{ V}$  shows two large dangling bonds on the rest atoms with a smaller dangling bond on a faulted corner adatom. The backbond state at  $-1.7\text{ V}$  shows most of the charge along the bond between corner adatoms and atoms in the layer directly below. No dangling bonds are found on the rest atom, and very little charge sits above the adatoms.

Based on the relative amounts of charge on each dangling bond and its proximity to the

Fermi energy, we might expect that the corner hole should be the most reactive sight, then the rest atoms and last the adatoms. In an elegant STM study of  $\text{NH}_3$  decomposition on  $\text{Si}(111)\text{-}7\times 7$ , Wolkow and Avouris [89] have shown that, among all the silicon atoms with dangling bonds, the Si rest atoms are more reactive than Si adatoms and the center adatoms are more reactive than the corner adatoms. Their study could not monitor the activity of the atom recessed at the bottom of the corner hole. The electronic spectra obtained with STM suggest that the rest atoms are reactive because they can transfer extra charge to the neighboring adatoms [90] and thus react with  $\text{NH}_3$ . The dangling bond of adatoms experimentally appears to be more delocalized with a low density of states at the adatoms site, leading to lower reactivity. Our theoretical results confirm this adatom feature. With respect to the corner hole, we must look to the infrared spectra obtained by Chabal [91]-[93] for H on  $\text{Si}(111)$ , since STM results are inconclusive. At low coverages of less than 0.2 monolayers, Chabal observed that the hydrogen bonds predominantly at the corner hole, a result that is consistent with the corner hole being the most reactive site. In the next chapter we will demonstrate how these results and others can be understood within the context of surface softness and electronegativity.

Changes with bias voltage in the experimentally observed unoccupied states are more subtle. At all bias voltages, Hamers, Tromp and Demuth[88] observed adatom structures. At 0.65 V above the Fermi energy, the experimental STM heights of corresponding dangling adatom bonds on the faulted and unfaulted halves of the unit cell was nearly the same. The image near 1.2 V exhibits a reverse asymmetry compared to most energies since the unfaulted half of the unit cell appears higher than the faulted half. At 1.6 V, the asymmetry disappears once again.

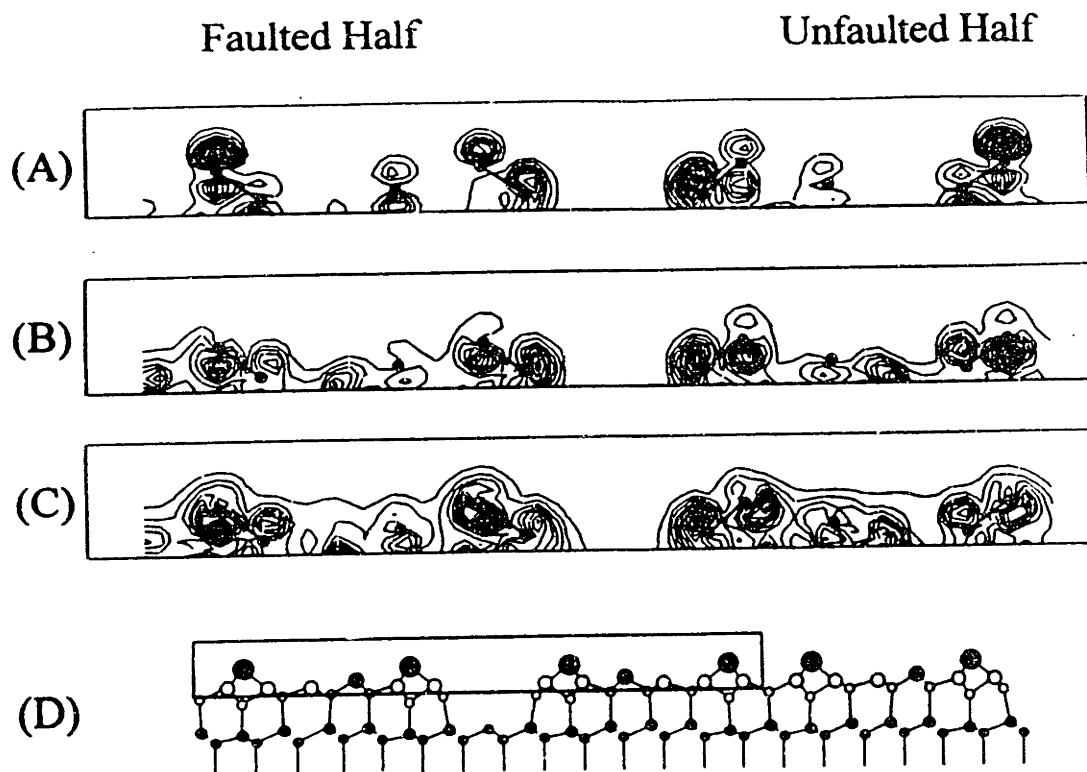


Figure 4.12: Contours comparing unoccupied surface states.

The theoretical picture is similar, as shown in Fig. 4.12. The contour plot in Fig. 4.12(A) shows the flat adatom dangling bond structure at an energy of  $+0.5$  V above the Fermi energy. The dangling bonds on corner and center adatoms are nearly identical on faulted and unfaulted halves. Also present are smaller dangling bonds on the rest atoms. Again, there is little difference between faulted and unfaulted halves. At  $+1.1$  V, our calculation also finds that the dangling bonds are larger on the unfaulted half of the unit cell as seen in Fig. 4.12(B). We also find that the charge on the rest atoms has nearly disappeared, and a large amount of charge appears on the atoms bonded directly to the adatoms. Figure 4.12(C) shows that at  $+1.7$  V the surface again flattens out, with charge distributed more uniformly through the surface and little asymmetry between the faulted and unfaulted sides.

This section has shown how massively parallel computation raises the complexity of materials systems that can be modeled from first principles to a new echelon. Study of the Takayanagi surface reconstruction of Si(111) shows that *ab initio* methods can successfully

calculate several fundamental aspects of systems involving  $\sim 700$  atoms. It is possible to quantify the energy difference between a metastable state and the ground state even when the systems are very large as was shown by comparison of the surface energies of the  $(7\times 7)$  and  $(2\times 1)$  reconstructions. Since *ab initio* methods give accurate forces on ions, the relaxed positions can be determined to a degree of accuracy limited mainly by the number of layers in the slab. Detailed studies of electronic states are also possible. In this chapter we showed that the results agree with available experimental data and predict new results in other cases. This material in this chapter is published in references [1] and [94].

## Chapter 5

# Surface Reactivity of the Si(111)-(7×7) Reconstruction: A Density Functional-Theoretic Analysis

Electronic properties of semiconductor surfaces are active areas of experimental and theoretical investigation[88]. It is known that the valence electrons of surface atoms that are not involved in covalent bonding, referred to as dangling bonds, constitute active sites for surface reactions[95]. Dangling bonds determine, to a large extent, the chemical properties of the surface. However, the details regarding the effect of the local environment on the surface reactivity are not well understood.

In this chapter, we discuss an *ab initio* theoretical investigation of the chemistry on the Si(111)-(7×7) surface reconstruction. This surface is a challenge for studying surface chemistry because a single atomic species reconstructs into seven different types of dangling bonds in the unit cell. While experiments testing the reactivity of these sites have been performed with a variety of chemisorbed gases[96], [97], no corresponding theoretical

picture of the complete surface has emerged due to the complexity of the unit cell. However, with parallel supercomputers, realistic theoretical calculations of the Si(111)-(7×7) reconstruction are finally possible. Thus, an *ab initio* rationalization of the chemistry of this system is now feasible through the use of Density Functional Theory methodologies (DFT)[48]. Recently, DFT has also provided an interesting technique for using electronic structure to analyze chemical reactivity[98]. In particular, local softness[99] has emerged as a useful tool in the analysis of local reactivity behavior in systems ranging from molecules to solids[100]-[104]. An interesting feature of local softness is that for surfaces, it could be estimated through scanning tunneling microscopy under conditions of low temperature and low bias voltage[105].

Our attempt to understand some aspects of the rich chemistry of the Si(111)-(7×7) reconstruction is based on a careful study of the behavior of regional softness on the surface. Thus, our rationale for analyzing chemical reactivity uses two well defined DFT concepts. The assignation of donor or acceptor roles in charge transfer processes depends on chemical potential (electronegativity) differences between the surface and the attacking species. Once the role of the surface in the charge transfer process has been determined, local softness distinguishes the abilities of different sites (dangling bonds) on the surface to perform charge transfer.

The chapter is organized as follows. Section one describes the electronic surface states of the Si(111)-(7×7) reconstruction in a chemical reactivity context. The basic definitions behind local softness and its application to surface reactivity are discussed in section two. The chemical reactivity of the Si(111)-(7×7) reconstruction is analyzed in section three. Conclusions are presented in section four.



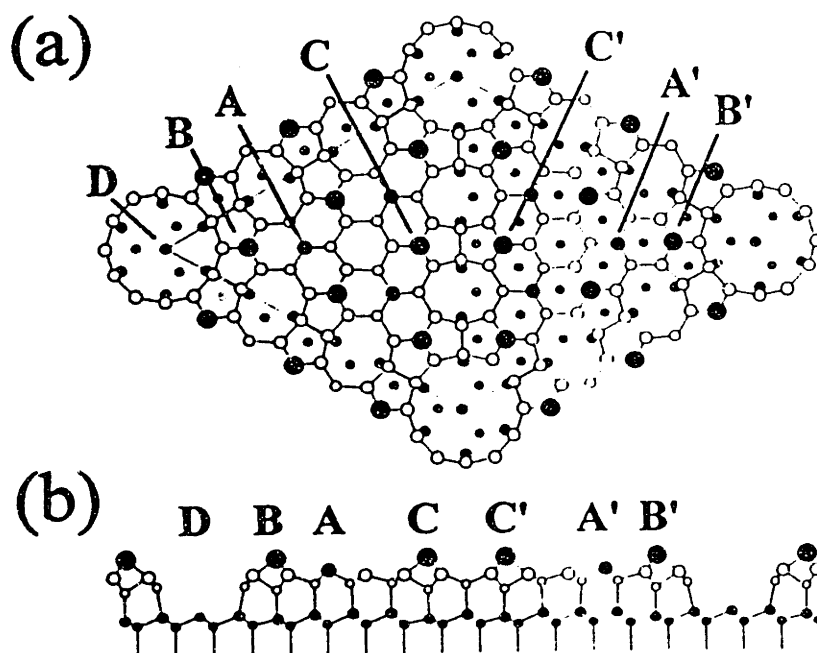


Figure 5.1: Si(111)-(7 $\times$ 7) surface dangling bond sites.

## 5.1 Local Properties of Electronic Surface States

The Si(111)-(7 $\times$ 7) reconstruction has a rich and complex chemistry because it contains a diverse set of dangling bonds which are in many respects similar; thus, it is difficult and challenging to rationalize the dramatic differences in reactivity among the different sites on the surface. This reconstruction is one of the most studied in the literature, however, only with the use of massively parallel computers are *ab initio* studies of its electronic structure tractable. In particular, we used the *ab initio* molecular dynamics scheme for calculating the electronic states and computing the relaxed positions of the ions. Details of the calculation, including parallel implementation, were described in the Chapter 3.

The relaxed positions of atoms in the 7 $\times$ 7 unit cell are plotted in Figure 5.1. A top view of the surface with the unit cell outlined with dotted lines is shown in Figure 5.1(a). The principal features were described in Chapter 4. In this figure, all unique dangling bonds

are labeled with capital letters corresponding to Wolkow and Avouris[89]. The faulted and unfaulted rest atoms, denoted by shaded circles labeled  $A$  and  $A'$  respectively, sit on the top surface bilayer. There are a total of six rest atoms, three on each side of the stacking fault. The large black circles labeled  $B, B', C$  and  $C'$  denote adatoms sitting on the top of the first surface bilayer. The six adatoms on each side of the unit cell form triangles. Adatoms at the faulted and unfaulted triangle corners  $B$  and  $B'$  are chemically distinct from adatoms at the triangle centers  $C$  and  $C'$ . The relative heights of the surface atoms are plotted in Figure 5.1(b), showing a side view through the long diagonal of the cell. It can be seen that the rest atoms with the dangling bonds are buckled slightly upward from the positions of other first-layer atoms. In the side view, the depth of the large hole at each corner of the cell is also apparent. Inside this corner hole sits the nineteenth dangling bond site, labeled  $D$ .

In this section we describe the behavior of the local density of states of the Si(111)-(7×7) reconstruction as a function of energy for the different atomic sites on the surface. Figure 5.2 surveys surface states with a series of contour plots at 0.2 eV intervals between -2 and +2 eV. The figures compare the same vertical slice through the long diagonal of the corner hole with the faulted side on the right. The state at -1.9 eV is clearly a backbond state, as charge is evident between adatoms and first-layer atoms. The adatom states show evidence of pentavalent character. A fifth bond between adatoms and the atoms directly below is visible. The charge distributions near the corner hole and the rest atom also have a backbond character, but no charge extends directly above or below these atoms. Only the adatom states have a slight dangling bond character. At -1.7 and -1.5 eV the surface states retain their backbond character. At -1.5 eV there is a bond between first-layer atoms and atoms directly below adatoms.

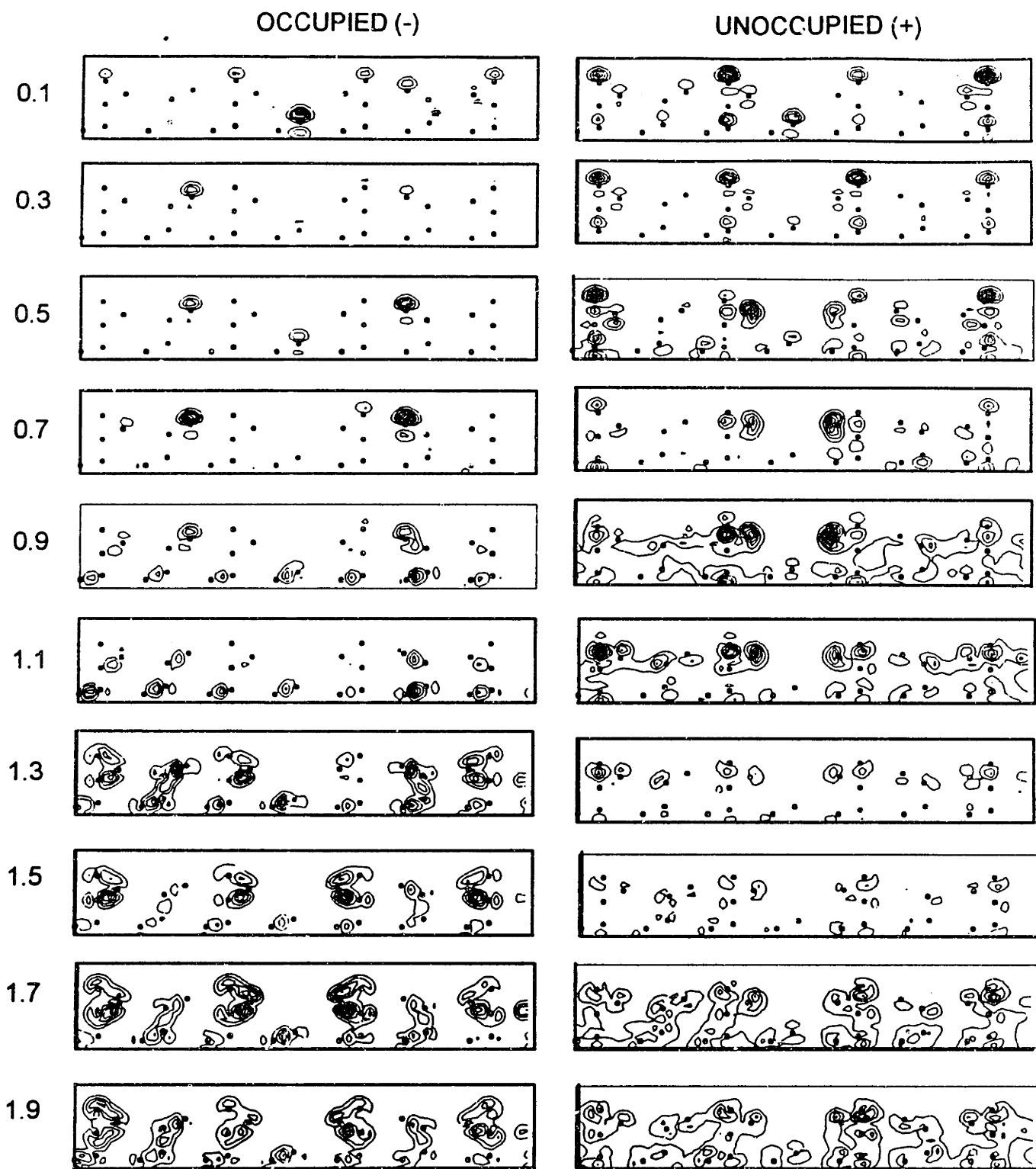


Figure 5.2: Contour plots of the local density of states for several values of energy in a plane perpendicular to the surface along the longest diagonal.

The charge density decreases but retains a backbond character until an energy of -0.9 eV, where rest atom dangling bonds appear. At this energy, the only backbond left on the surface is the corner hole. The rest atom dangling bonds are symmetric between faulted and unfaulted halves of the surface. A slight dangling bond is also visible on the corner faulted adatom. Contrast between rest atom and adatom dangling bonds is maximum at this voltage. Experimental STM images of these rest atoms states were reported by Hamers, Demuth and Tromp[72],[88]. The rest atom dangling bonds increase in intensity as the energy approaches 0.5 eV below the Fermi level. At this energy, the rest atom bond density is high, with no difference between faulted and unfaulted halves. The corner hole dangling bond is also large. These bonds have  $sp^3$  character, with backlobes below the surface. The depth of the corner hole decreases substantially at this energy. The dangling bond on the faulted corner adatom is also visible. To lesser extent, one can see dangling bonds on the other adatoms. The order of intensity of adatom dangling bonds is faulted corner > faulted center > unfaulted corner > unfaulted center.

At -0.3 eV, the dangling bonds on the adatoms reach their maximum density. However, the adatom dangling bond densities are still less than the rest atoms dangling bond densities, proof of charge transfer to the rest atoms. The corner hole peak is smaller. The occupied states -0.1 eV below  $E_F$  have strong dangling bond character. Adatom dangling bond densities are comparable to the faulted rest atom. The unfaulted rest atom is smaller. The largest dangling bond is at the corner hole.

The most notable difference between occupied and unoccupied states is the absence of dangling bonds on the rest atoms. At 0.1 V above  $E_F$ , adatom dangling bonds are present with little dangling bond character at the rest atoms. The corner hole dangling bond is

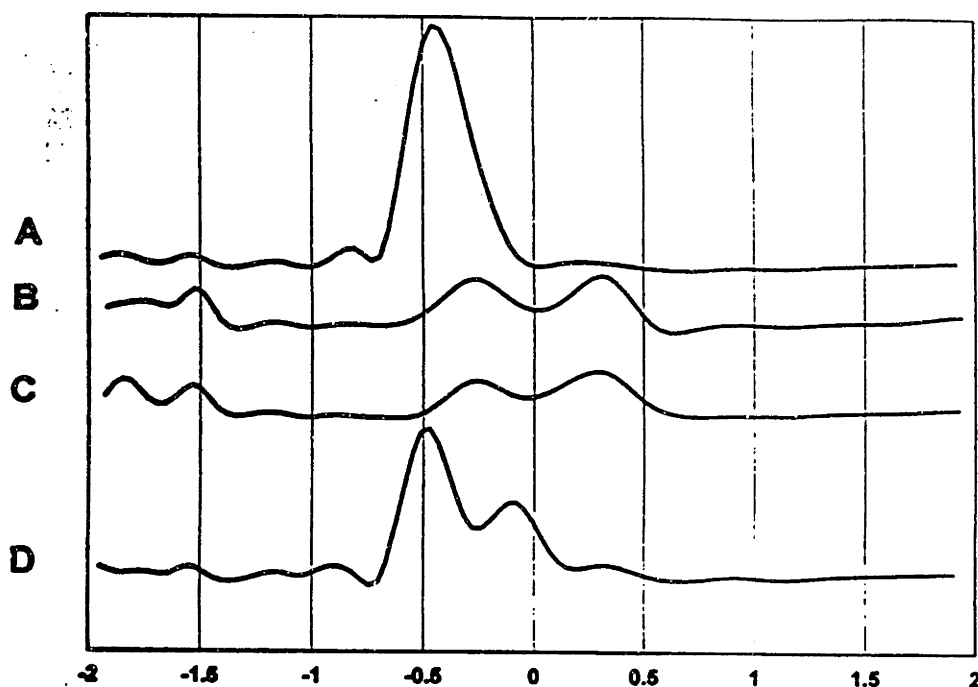


Figure 5.3: Density of states at different dangling bonds averaged over faulted and unfaulted halves of the unit cell.

still visible. The adatom dangling bond densities peak at 0.3 eV, indicating the unoccupied nature of the adatom surface states.

At 0.7 eV, the unoccupied backbond states begin. There is a substantial amount of charge on first-layer atoms near the corner hole. The corner hole is now deeper in the absence of its dangling bond as was the case for occupied states. At 1.3 eV, the pentavalent bond is visible, as are backbonds. At 1.7 and 1.9 eV, the charge distribution across the surface actually flattens out, something that Hamers *et al.* observed.

Figure 5.3 plots the local density of states as a function of energy near the Fermi level. Unfaulted and faulted dangling bonds are averaged together here because differences between these surface states are more subtle as subsequent figures will show. The curves are computed by averaging the charge density over a  $3\text{\AA} \times 3\text{\AA} \times 3\text{\AA}$  volume at each dangling bond site. Computations were performed at the  $\Gamma$  point in the Brillouin zone with Gaussian

smoothing applied. Curve A shows average density spectra near rest atom sites. The characteristic features are strong occupied bands at  $\sim 0.5$  eV below  $E_F$ . The energy and intensity of this band suggest that the corresponding rest atom dangling bond state is fully occupied. Curve B shows densities averaged near corner adatoms. It shows an occupied band at  $\sim 0.3$  eV below  $E_F$  and an unoccupied band at  $\sim 0.4$  eV above  $E_F$ . These bands correspond to the occupied and unoccupied parts of the corner-adatom dangling bond states. It is clear from the size of the occupied peak and the presence of a large unoccupied peak that the adatom bands are not fully occupied. The band at  $\sim 2$  eV below  $E_F$  is an adatom back-bond state. The peak is small due to delocalized nature of this state. The very small band at  $\sim 1.5$  eV above  $E_F$  is also primarily an adatom backbond state. The center-adatom spectrum (curve C) looks essentially similar to the corner adatoms on the scale of this plot. Curve D shows two large occupied dangling bond peaks in the corner hole directly below  $E_F$ , one at  $\sim 0.1$  eV and the other at  $\sim 0.5$  eV. Cherif[106] recently observed experimentally that the corner hole state is similar to the rest atom state. We see from Figure 5.3 that the corner hole density of states peaks at  $-0.5$  eV in a manner similar to the rest atoms. However, the corner hole DOS has an additional peak directly below  $E_F$ . This difference will affect the reactivity of this surface site.

Figures 5.4 and 5.5 compare the much smaller differences between dangling bonds. Figure 5.4 compares corner and center adatoms. It can be seen that the corner adatoms have larger peaks below  $E_F$ , while the center adatoms have slightly larger unoccupied peaks. These qualitative differences have been observed in STM images. Figure 5.5(a) compares faulted and unfaulted corner adatoms. The faulted adatom has a larger peak below  $E_F$ , while the unfaulted corner adatom has a slightly larger peak slightly above  $E_F$ . The faulted and unfaulted center adatoms, are compared in Figure 5.5(b). The local density of states on

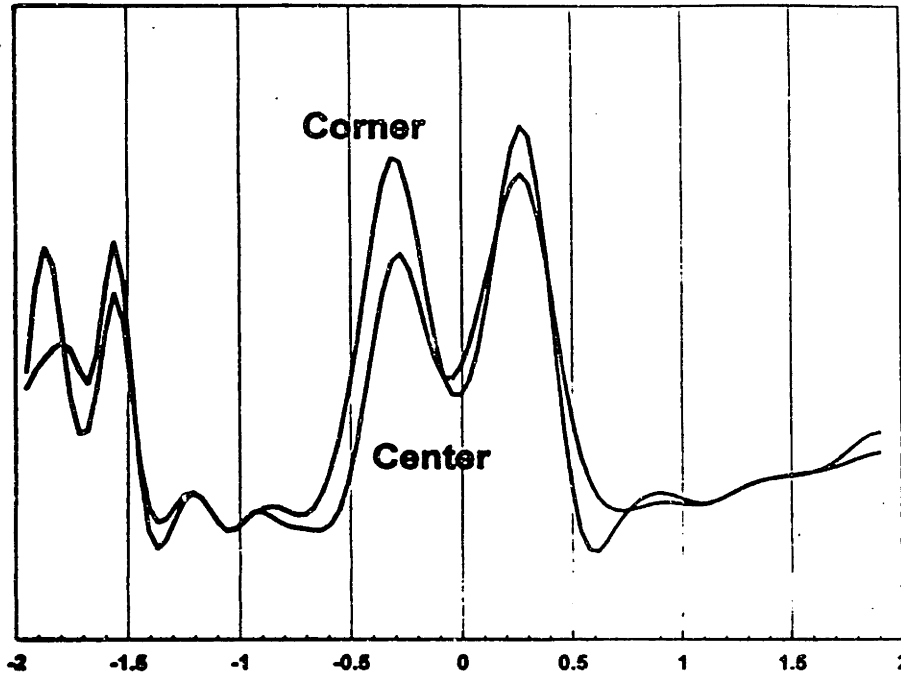


Figure 5.4: Local density of states for corner and center adatoms averaged over faulted and unfaulted halves of the unit cell.

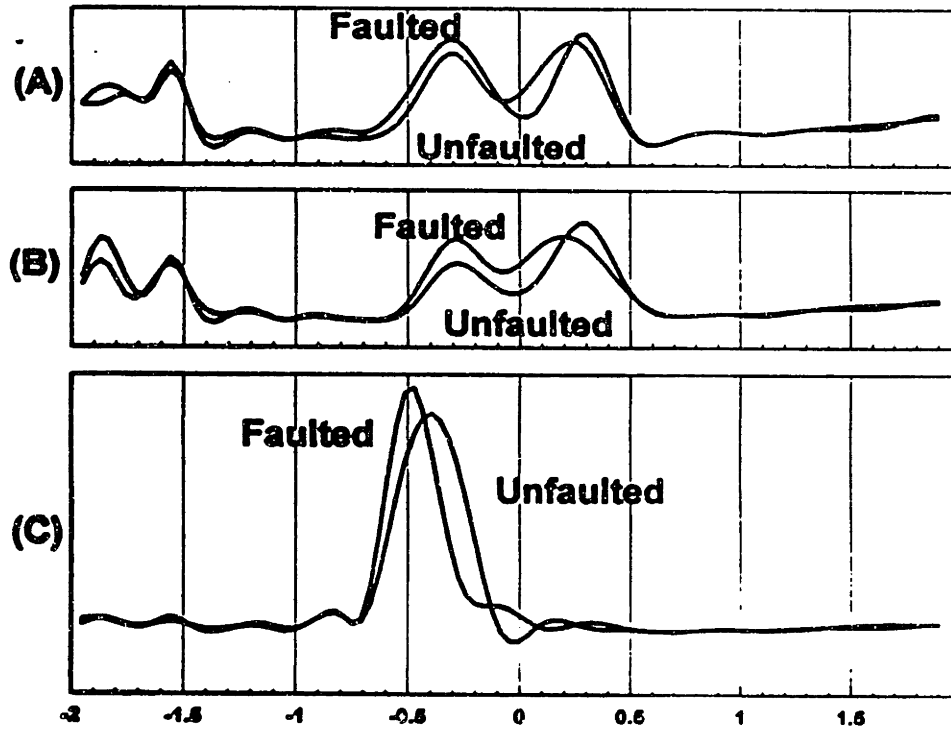


Figure 5.5: Differences between faulted and unfaulted states: (a) faulted and unfaulted corner adatoms; (b) unfaulted and faulted center adatoms; (c) faulted and unfaulted rest atoms

the unfaulted center adatom occupied state is substantially less than the density of states on the faulted corner adatom. The faulted center adatom also has a higher unoccupied density of states close to  $E_F$ . The filling of the rest atom dangling bond state and the small occupation of the adatom state suggests an adatom to rest atom charge transfer process[90]. The relative local densities for center and corner adatoms show that most of this charge is contributed by the center adatoms. Each center adatom has two rest atom neighbors while corner adatoms have only one. Figure 5.5(c) compares faulted and unfaulted rest atom states. The unfaulted state is at slightly higher energy, while the faulted state is slightly more occupied.

Differences in the electronic structure surrounding each site on the surface, such as the ones described above, should have important consequences in the chemical reactivity of the surface. In the following sections we analyze those differences in the context of a reactivity parameter which contains regional differences: the local softness.

## 5.2 Local Softness and Surface Reactivity

Local softness was introduced within the context of the finite temperature extension of DFT. In this formalism, the grand potential is a functional of the density

$$\Omega[\rho(\mathbf{r})] = \int (v(\mathbf{r}) - \mu)\rho(\mathbf{r})d\mathbf{r} + F[\rho(\mathbf{r})] \quad (5.1)$$

where  $\mu$  is the chemical potential,  $v(\mathbf{r})$  is the external potential and  $F[\rho(\mathbf{r})]$  is a universal functional of the density, that incorporates the kinetic energy, the electron-electron interaction and an entropy term. In this representation of DFT, the natural variables are temperature  $T$ , volume  $V$  and chemical potential.



According to Yang and Parr[99], local softness is defined as

$$s(\mathbf{r}) = \left( \frac{\partial \rho(\mathbf{r})}{\partial \mu} \right)_{v(\mathbf{r})}. \quad (5.2)$$

Three alternative expressions for  $s(\mathbf{r})$  have given this quantity a remarkable place in the spectrum of local reactivity indices defined in the framework of DFT. The first one was obtained by Yang and Parr,

$$s(\mathbf{r}) = \frac{1}{kT} (\langle \rho(\mathbf{r})N \rangle - \langle \rho(\mathbf{r}) \rangle \langle N \rangle) \quad (5.3)$$

where  $N$  is the number of electrons, and  $\langle \rangle$  implies an average over the grand canonical ensemble. This expression indicates that large local fluctuations in  $\rho(\mathbf{r})$  go together with large local softness. The relevance of this fact in catalysis, as pointed out by Yang and Parr, is related with the idea of Falicov and Somorjai[107] that catalytic activity in transition metals is associated with low-energy electronic fluctuations. The second relation establishes that local softness and the derivative of the density with respect to the number of electrons, the Fukui function[108],  $f(\mathbf{r})$  has the same local information

$$s(\mathbf{r}) = \left( \frac{\partial \rho(\mathbf{r})}{\partial N} \right)_{v(\mathbf{r})} = S f(\mathbf{r}). \quad (5.4)$$

The constant scale factor  $S$  is the global softness which is the integral of  $s(\mathbf{r})$ [98]. As has been pointed out, if the frozen-core approximation is used,  $f(\mathbf{r})$  reduces to the density of the highest occupied orbital or to the density of the lowest unoccupied orbital (frontier orbitals) depending on the charge transfer process considered to perform the derivation with respect to  $N$ ; consequently, Eq. (4) indicates that  $s(\mathbf{r})$  contains the frontier orbital behavior as a limiting case. The third equation,

$$s(\mathbf{r}) = \left( \frac{\delta N}{\delta v(\mathbf{r})} \right)_{\mu, T} \quad (5.5)$$

gives an interpretation of  $s(\mathbf{r})$  which is interesting for our purposes of obtaining an index that distinguishes regional capabilities for charge transfer effects on surfaces: *local softness measures the ability of the system to donate or accept electrons at a particular point in space as the external potential is changed at that point.* This equation was first obtained by Harbola, Chattarraj and Parr[109], and was used recently[103], [105] for justifying a relation between  $s(\mathbf{r})$  and Scanning Tunneling Microscopy (STM) images. The relation with STM images establishes that under conditions of low bias voltage and low temperature, local softness is an experimentally measurable quantity for surfaces. There is also an interesting result that identifies the difference in local softness between reactants  $A$  and  $B$ ,  $\Delta s(\mathbf{r}) = s_A(\mathbf{r}) - s_B(\mathbf{r})$ , as a driving force for charge transfer. In this approach, an expression for the change in the number of electrons in terms of  $\Delta s(\mathbf{r})$  is obtained:

$$d \langle N \rangle = \frac{1}{2} \int \Delta s(\mathbf{r}) \Delta[\mu - v^n(\mathbf{r})] d\mathbf{r} \quad (5.6)$$

where  $\Delta v^n(\mathbf{r})$  represents relaxation of the nuclei, and  $\Delta\mu$  is the change in the chemical potential of the combined system  $AB$ . Then it follows that for a given relaxation of the nuclei and a given change in the chemical potential, a large charge transfer arises from a large softness difference. The physical meaning of local softness extracted from Eqs. (5) and (6) shows that local softness should be a good measurement of regional charge transfer abilities.

Now, let us focus the attention on regional softness. As we want to distinguish the capabilities of different dangling bonds on the surface to perform charge transfer, we define regional softness  $s_i$  as

$$s_i = \int_{\Omega_i} s(\mathbf{r}) d\mathbf{r}. \quad (5.7)$$

This integral is done on a volume  $\Omega_i$  surrounding the dangling bond  $i$ . This quantity

measures the differences in the environment of each dangling bond on the surface. As an extension of the physical meaning of the local property, we assign to  $s_i$  the meaning of a measure of the ability of the dangling bond  $i$  on the surface to perform charge transfer. Following Politzer[110], we can use the name proposed by Huheey[111] for this kind of property: *charge capacity*. Thus, in the present approach to the reactivity of the Si(111)-(7×7) reconstruction, differences in charge transfer capabilities among different dangling bonds will be determined through  $s_i$ .

The usefulness of hardness and softness in solid state systems has been pointed out in several applications of the concepts. In order to consider a semiconductor surface, we must extend the  $s(\mathbf{r})$  definition of Yang and Parr for metals to systems with a gap. From now on the discussion is restricted to zero temperature but it may be extended to finite temperature. The charge density  $\rho(\mathbf{r})$  for a solid state system can be expressed as

$$\rho(\mathbf{r}, \mu) = \int_0^\mu dE g(E, \mathbf{r}) \quad (5.8)$$

where  $g(E, \mathbf{r})$  is the local density of states

$$g(E, \mathbf{r}) = \sum_i |\psi_i(\mathbf{r})|^2 \delta(E - E_i). \quad (5.9)$$

The chemical potential  $\mu$  is equal to the fermi level  $E_F$ . The wave function  $\psi_i(\mathbf{r})$  is associated to the eigenvalue  $E_i$ . According to the definition of  $s(\mathbf{r})$  (Eq. 2) one has to deal with the derivative of the density with respect to the chemical potential. By using Eqs. (2) and (8), local softness may be written in the form

$$\left( \frac{\partial \rho(\mathbf{r})}{\partial \mu} \right)_{T, v(\mathbf{r})} = \lim_{\delta\mu \rightarrow 0} \frac{1}{\delta\mu} \int_\mu^{\mu+\delta\mu} dE g(E, \mathbf{r}). \quad (5.10)$$

In general,  $\delta\mu$  could be positive or negative, thus, there are two different derivatives. We must analyze the physical meaning of the two possible signs in  $\delta\mu$ . Local softness was

defined in the context of the finite temperature extension of DFT. In this formalism, the grand potential is a functional of the density and the independent variables are  $\mu, v(\mathbf{r})$  and temperature. At constant  $v(\mathbf{r})$  and  $T$ , the physical situation may be visualized as depicted in Figure 5.6. The system is in thermal contact with a heat reservoir at temperature  $T$ , and the system and the reservoir freely interchange electrons. The external potential of the system as determined by the atomic positions for a surface is assumed fixed. In this picture, the chemical potential of the system changes according to the modifications of the same variable in the reservoir at constant temperature. During this process the system interchanges electrons with the reservoir. For some changes in  $\mu$  the electrons flow to the reservoir and for others the flow goes to the system. The former process is related to the interaction of the system with electron acceptors (electrophiles) and the later to the interaction with electron donors (nucleophiles). At this point it is important to distinguish two different cases. For gapless systems (i.e. metals) the number of electrons as a function of the chemical potential  $N(\mu)$  is a continuous increasing function with continuous first derivative in the vicinity of the fermi level. In systems with a gap, (i.e. semiconductors)  $N(\mu)$  is constant in the vicinity of the fermi level, and is an increasing function for values of  $\mu$  which are in the conduction or in the valence regions. The differences between metals and semiconductors are illustrated in Figure 5.7. For semiconductors,  $N(\mu)$  has a discontinuous first derivative indicating that the gap is identified as a region in the  $(N, \mu)$  space in which global softness  $S$  is equal to zero:

$$\left(\frac{\partial N}{\partial \mu}\right)_{v(\mathbf{r})} = 0. \quad (5.11)$$

The discussion in this section shows that positive values of  $\delta\mu$  are related to processes where the system increases its number of electrons while negative values are related to a

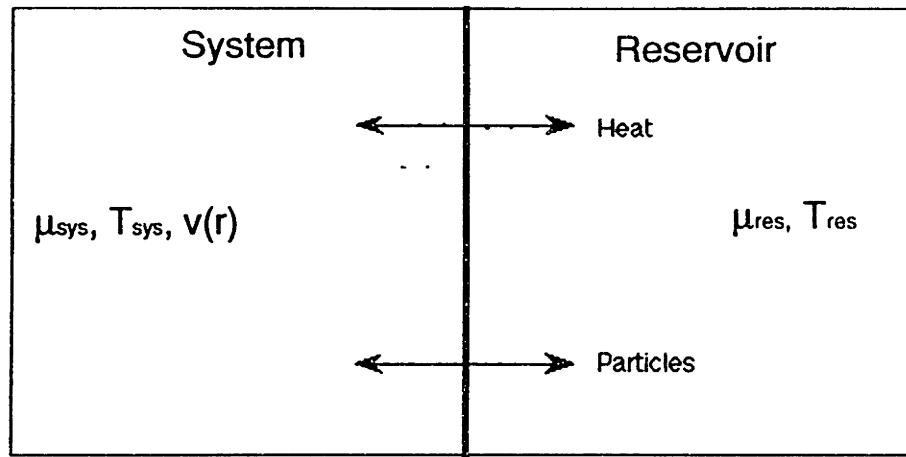


Figure 5.6: Schematic illustration of the interaction of the electronic system with a charge reservoir.

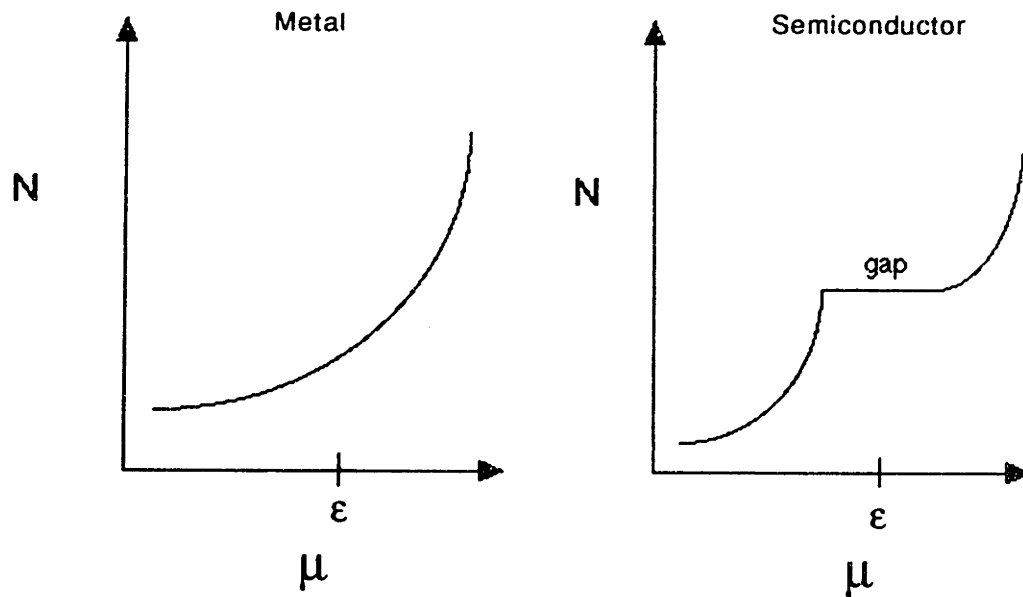


Figure 5.7: Schematic illustration of the chemical potential as a function of the number of electrons.

reduction in the number of electrons. The two limits associated with Eq. (10) are therefore different for a semiconductor because for positive  $\delta\mu$  the change in the density is due to bands which are on the edge of the valence band, while for negative  $\delta\mu$  the density changes involve bands at the edge of the conduction band. An interesting conclusion emerges from Eq. 10 if we recognize the integral as the contribution to charge density from states which are in a vicinity  $\delta\mu$  of the Fermi level. Then, due to the weighting factor  $\frac{1}{\delta\mu}$  the states that contribute the most to  $s(\mathbf{r})$  are those which are closer to the Fermi level. By recalling the discussion surrounding Eq. (4), one can conclude that Eq. (10) contains the frontier orbital theory for solid state systems, giving a strong support for the use of local softness as a chemical reactivity index in solids and in surfaces.

### 5.3 Chemical Reactivity of Si(111)-(7×7) - Local Softness and Charge Capacity

In this work, the local softness is estimated through a finite differences version of Eq. (8),

$$s_{\Delta\mu}(\mathbf{r}) = \left( \frac{\partial \rho(\mathbf{r})}{\partial \mu} \right)_{T, v(\mathbf{r})} \approx \frac{1}{\Delta\mu} \int_{\mu}^{\mu+\Delta\mu} dE g(E, \mathbf{r}) \quad (5.12)$$

The values for charge capacity are obtained by the integration of  $s(\mathbf{r}, \Delta\mu)$  over volumes of  $3 \times 3 \times 3 \text{ \AA}$  centered on each dangling bond. These values depend on the interval  $\Delta\mu$  chosen for the finite differences derivative. According to the formal definition of  $s(\mathbf{r})$ , Eq. (8), one should select  $\Delta\mu$  as small as possible. However, one must be careful to include the basic differences between the occupied and unoccupied states close to the Fermi level. To compromise between the formal definition and physical content of  $\int s(\mathbf{r}, \Delta\mu) d^3\mathbf{r}$  we average from  $\Delta\mu = 0$  to  $\Delta\mu = \pm 0.5 \text{ eV}$ . The final values obtained are summarized in Table 5.1.

It was mentioned above that charge capacity is a measure of charge-donor or charge-

Table 5.1: Charge capacity for different atoms on the surface.

Atom	Donor Capacity	Acceptor Capacity
corner hole atom	1.00	0.29
faulted corner adatom	0.00	0.47
unfaulted corner adatom	0.07	0.96
faulted rest atom	0.92	0.00
unfaulted rest atom	0.75	0.24
faulted center adatom	0.20	1.00
unfaulted center adatom	0.15	0.78

Table 5.2: Charge capacity order for different sites on the Si(111)-(7×7) reconstruction.

surface donor capacity	corner hole > rest atoms > adatoms
surface acceptor capacity	adatoms > corner hole > rest atoms

Table 5.3: Differences in charge capacity between faulted and unfaulted halves

As a donor	faulted rest atom > unfaulted rest atom
As an acceptor	faulted center adatom > unfaulted center adatom unfaulted corner adatom > faulted corner adatom

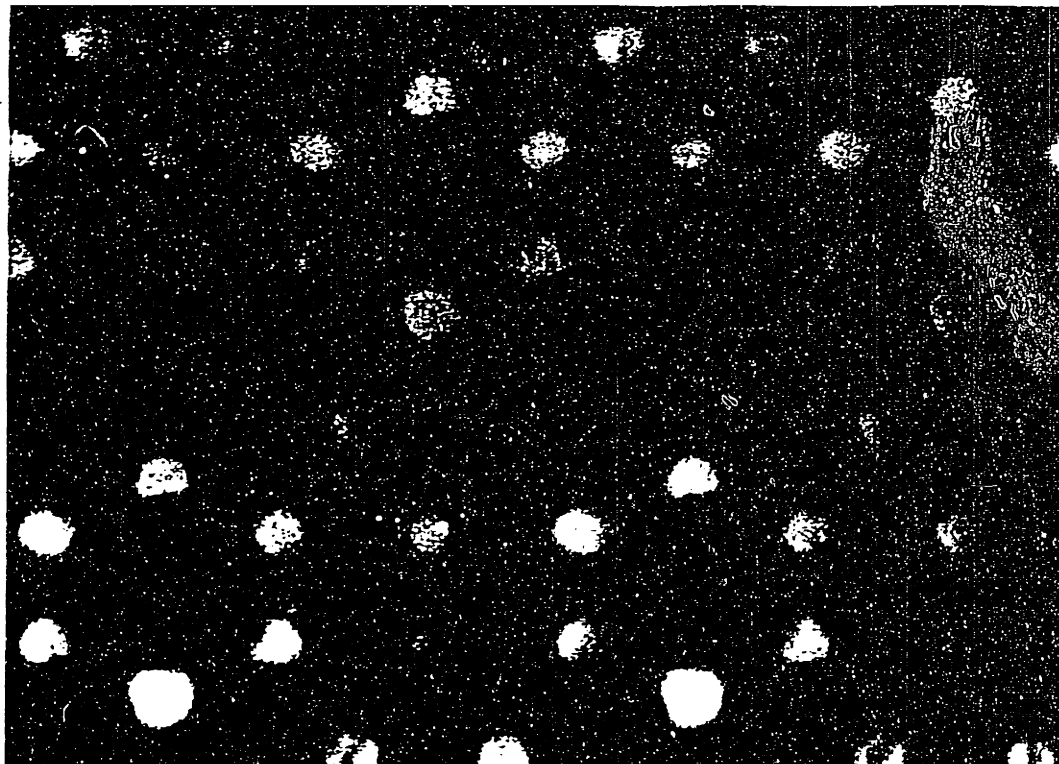


Figure 5.8: Local softness for a plane parallel to the surface showing hard and soft channels.

acceptor abilities. The analysis of the values of Table 5.1 shows that the donor and acceptor capacities follow the order displayed in Table 5.2. Table 5.3 summarizes the differences in charge capacity between faulted and unfaulted halves of the unit cell.

Local softness calculated on a plane parallel to the surface is displayed in Figure 5.8. As is expected, its behavior is dominated by the electronic structure of the adatom layer. One can see that there are hard channels associated with the sites in which there are rest atoms in the second layer, and corner hole atoms in the deepest layer of the reconstruction. On the other hand the soft channels are situated on top of the adatoms.

*General Reactivity Patterns.* According to Table 5.2, electrophilic attacking groups interact with the surface in the following order: corner hole > rest atom > adatoms. The corner hole and rest atom reactivities are strong, while the adatom reactivity is relatively weak. The nucleophilic groups interact with the surface mainly through the adatoms, to a lesser extent with corner holes, and most weakly with the rest atoms. Among the



Table 5.4: Chemical reactivity of the Si(111)-(7×7) reconstruction

Reactant	Method	Reactivity
H	Spectroscopy[91]	The most active site is the corner hole atom
Pd,Ag,Li	STM[95],[112][113][114]	The faulted half is preferred and reactants are believed to bond primarily to rest atoms and center adatoms
NH <sub>3</sub> ,H <sub>2</sub> O,PH <sub>3</sub>	STM[95]	rest atoms > center adatoms > corner adatoms

seven different dangling bonds the corner hole is unique in exhibiting a strongly active site for electrophilic reactants as well as exhibiting some reactivity for nucleophilic reactants. Adatoms are more selective towards nucleophiles. While rest atoms are strongly reactive toward nucleophilic species, the rest atoms are weakly reactive toward electrophilic species.

We now consider the effects of the stacking fault in the reactivity of the Si(111)-(7×7) reconstruction. With respect to donor capacity, there is a clear distinction between reactivity at faulted and unfaulted halves as shown in Table 5.3. Electrophiles should prefer to interact with the faulted half of the surface, and in particular with the faulted rest atoms. With respect to acceptor capacity, there does not appear as strong a selectivity for one half or the other. Nevertheless if a donor (nucleophile) prefers to react with the faulted half, the interaction will be primarily with the faulted center adatom, whereas if the unfaulted half is selected the interaction is generally with the unfaulted corner adatom.

*Reactions Preserving Reconstruction.* To illustrate the utility of the present approach we will apply it to analyze some experimental reactivity patterns of the Si(111)-(7×7) reconstruction. In particular, Table 5.4 displays the reactivity patterns for some attacking

compounds which maintain the reconstruction. The experimental information was obtained using infrared spectroscopic techniques by Chabal *et al.*[91]-[93] and by scanning tunneling microscopy (STM) by Avouris and colleagues[95]. To classify the reactants as donor or acceptors we used an electronegativity criterion. If the difference in electronegativity between the surface and the reactant is negative then the reactant is the acceptor and the surface the donor. If the difference is positive the roles are inverted. Table 5.5 shows these differences for a series of compounds. Electronegativities for the reactants were taken from Pearson's estimations[115],[116]. The surface electronegativity is the work function, 4.8 eV.

*Interaction with Hydrogen.* According to Table 5.5, H can be classified as an acceptor species with respect to the surface. Hydrogen is one of the most electronegative of the neutral reactants listed in this table. From Table 5.1 it is clear that the corner hole atoms are the most reactive sites in this case and this is in complete agreement with the experiments of Chabal *et al.*[91]-[93] summarized in Table 5.4.

*Interaction with Pd, Ag, and Li.* According to Table 5.5, these metal atoms are electron donors with respect to the surface[112]-[114]. Thus, our theoretical calculations suggest that they should interact primarily with adatoms and specifically with faulted center adatoms. This is again consistent with the experimental evidence summarized in Table 5.4.

*Interaction with  $NH_3$ ,  $H_2O$ , and  $PH_3$ .* These molecules dissociate on the surface into the anions  $OH^-$ ,  $(NH_2)^-$ , and  $(PH_2)^-$  and the cation  $H^+$ . From Table 5.5 all of the dissociation products except the proton are donors. The proton is clearly a strong acceptor. By assuming that the dissociation process takes place in the initial step of the reaction without inducing large changes in the chemical potential of the surface, we determine the order of preferential reactivity as corner hole atoms, then rest atoms and last the adatoms. Table 5.4 shows that

Table 5.5: Global softness and relative electronegativity  $\eta$  of reactants

Reactant	global softness (eV)	electronegativity
Li	0.42	-1.8
Ag	0.32	-0.4
Pd	0.26	-.03
PH <sub>3</sub>	0.17	-0.7
H	0.16	2.4
NH <sub>3</sub>	0.12	-2.2
H <sub>2</sub> O	0.11	-1.7
<i>Anionic Radicals</i>		
NH <sub>2</sub>	0.19	1.3
OH	0.18	1.7
PH <sub>2</sub>	0.23	0.7
<i>Hard Acceptors</i>		
BF <sub>3</sub>	0.10	1.4
K <sup>+</sup>	0.07	13.2
N <sub>2</sub>	0.11	1.9
<i>Soft Donors</i>		
Na	0.42	-1.9
Al	0.36	-1.6
Ga	0.34	-1.6
<i>Hard Donors</i>		
CH <sub>4</sub>	0.10	-2.3
(CH <sub>3</sub> ) <sub>2</sub> O	0.13	-2.8
c-C <sub>3</sub> H <sub>6</sub>	0.13	-2.0
<i>Soft Acceptors</i>		
I <sub>2</sub>	0.29	1.2
Ir <sup>+</sup>	0.26	8.2
Pt	0.29	0.8

Table 5.6: Predicted reactivity of the Si(111)-(7×7) reconstruction.

Reactant	Reactivity
Hard Acceptors: $\text{BF}_3, \text{K}^+, \text{N}_2$	corner hole > rest atoms > adatoms faulted rest atom > unfaulted rest atom
Soft Donors: Na, Al, Ga	adatoms > corner hole > rest atoms faulted center adatoms > unfaulted center adatoms unfaulted corner adatoms > faulted corner adatoms
Hard Donors: $\text{CH}_4, (\text{CH}_3)_2\text{O}, \text{c-C}_3\text{H}_6$	corner hole > rest atoms > adatoms surface interaction could active or break some bonds in the reactant.
Soft Acceptors: $\text{I}_2, \text{Ir}^+, \text{Pt}$	should react weakly with the order: corner hole > rest atoms > adatoms

the predicted reactivity order of rest atoms and adatoms agrees with experimental STM data[95]. Our calculations show that corner hole site is the most reactive. Due to the depth of the corner hole below the adatom and rest atom layers our predictions regarding the reactivity of the corner hole with respect to  $\text{NH}_3$ ,  $\text{PH}_3$ , and  $\text{H}_2\text{O}$ . have not yet been tested experimentally.

By using Pearson's electronegativity tables and the work function of the surface, one is able to classify a great variety of atoms and molecules as donors or acceptors with respect to the Si(111)-(7×7) reconstruction. Selected examples are presented in Table 5.6 including predictions of possible reaction patterns. The predictions assume non-dissociative interactions and that surface reconstruction is maintained.

*The local version of the Hard and Soft Acids and Bases (HSAB).* Now, we analyze the behavior of local softness for the Si(111)-(7×7) reconstruction in the context of the local version of the HSAB principle. This principle establishes that given a system with

different reactive sites, its hard regions prefer to interact with hard species whereas its soft areas prefer soft attacking groups to react. Recently, the validity of this principle has been confirmed in the case of the  $\text{Si}_4$  cluster[103]. In the present case, for the reactions we analyzed above, hydrogen and the three molecules ( $\text{NH}_3$ ,  $\text{H}_2\text{O}$ , and  $\text{PH}_3$ ) have low values of softness, indicating that they are hard while the metals (Li, Pd, and Ag) have high softness values indicating that they are soft. Since charge capacities quantify the “global softness” for each region of the surface involved, we can test the validity of the local HSAB principle for this case. Considering hydrogen first, it is a hard acceptor according to Table 5.5 but it mainly reacts with the softest site on the surface (corner hole atom) for this kind of interaction, in contradiction to the local HSAB principle. Metals which are soft donors react with soft sites (adatoms) on the surface in agreement with the local HSAB principle. The three molecules are hard donors but their corresponding dissociation products are slightly softer and positive acceptors. Thus, if the actual reactants are the dissociation products they are reacting mainly with donor sites (rest atoms) that are softer than adatoms. Again, the behavior is contrary to the local HSAB principle.

A possible explanation comes from the topography of the surface. At relatively large distances from the surface, the local softness is dominated by the relatively close adatom layer since rest atoms and the corner hole are buried below the surface at a much greater distance from the approaching reactant. Applying the local HSAB principle at long distances from the surface identifies the subsurface rest atoms and corner hole as hard channels for  $\text{H}$ ,  $\text{OH}^-$ ,  $(\text{NH}_2)^-$  and  $(\text{PH}_2)^-$ . The soft channels for the interaction of metals with the surface are then the adatoms, in agreement with the local HSAB principle. In this picture, the local HSAB principle orients the reactants at long distances from the surface. In the context of the charge transfers effects and the HSAB principle  $\text{H}_2\text{O}$ ,  $\text{NH}_3$  and  $\text{PH}_3$  interact

with the surface as though trapped in a wrong channel, resulting in dissociation in the early stages of the reaction. Consequently, the dissociation could be attributed to some charge transfer toward antibonding orbitals.

## 5.4 Conclusions

Surface chemisorption depends on a variety of factors, especially electronic effects dictating the existence and magnitude of activation barriers, as well as dissipative channels for the energy of incident reactants, steric constraints, and surface diffusion. The close similarity of different dangling bonds sites on the complex Si(111)-(7×7) surface reconstruction means that most of the common chemisorption factors are virtually identical among the different sites. The present work shows that the general qualitative behavior of the reactions with the Si(111)-(7×7) reconstruction is explained by differences based on two parameters describing the electronic surface states: the global electronegativity, and the local softness.

The definition of local softness introduced by Yang and Parr for metals was extended to systems with a gap to obtain regional softnesses for the Si(111)-(7×7) reconstruction. Two different classes of regional softness were calculated, one related to the nucleophilic (donor) capacity, and the other related to the electrophilic (acceptor) capacity of the surface. Accordingly, an order was assigned for the nucleophilic and electrophilic nature of the seven dangling bonds of the surface. From this analysis of regional softnesses, a general qualitative behavior for the reactivity of this surface reconstruction emerges.

The analysis of some particular cases illustrates the utility of the present approach:

- By using the electrophilic order of the dangling bonds obtained from regional softness,

one can understand the strong selectivity of the corner hole atom for atomic hydrogen.

- Interesting reactive differences of the surface depend on the stacking fault. These differences are clearly present in the regional softness. In particular differences in regional softness determine the preference of soft donor metals for interaction with the faulted center adatoms. Perhaps, this is the origin of the predilection of metallic particles to condense on the faulted half of the unit cell.
- Understanding the reactivity of molecules such as  $\text{H}_2\text{O}$ ,  $\text{NH}_3$ , and  $\text{PH}_3$ , which are species that dissociate during the interaction, imply the use of the dissociation products for performing the analysis. Thus, one should assume that the dissociation occurs in a first step of the reaction, and that during this first step the Fermi level (work function) and the regional softnesses of the surface do not change strongly.

Some particular cases are mentioned as predictions subject to the restrictions of maintaining surface reconstruction, and to a non-dissociative chemisorption. A detailed analysis of the local HSAB principle indicates that, for this case, this principle is orienting the reactants long distances defining hard and soft channels. Finally, it is important to notice that local softness and consequently regional softness could be obtained experimentally from STM experiments. Then, the methodology for analyzing surface reactivity described in this work involve two experimental system properties: Fermi level and local softness. A version of this chapter is published in reference [117].





## Chapter 6

# A New Quantum Monte Carlo Method for Fermions

The results described in preceeding chapters were all based on calculations within the local density approximation. While this approximation is sufficient for calculating many interesting properties of electronic systems, it is sometimes desirable to attempt more exact computations. This chapter contains a brief description of a new approach for calculating the ground states of many-Fermion systems [118].

This work is motivated by a large class of physical systems with many Fermion degrees of freedom for which perturbation theory is inadequate. In addition to small electronic systems, examples include the ground states of nuclei, nuclear matter, the structure of hadrons, and the zero temperature equation of state of hadronic matter. Many significant physics questions only require calculation of ground state expectation values of few-body operators, such as the total energy, density, form factor, or correlation functions. Since the resulting integrals over the ground state wavefunction may be evaluated effectively using Monte Carlo techniques, we therefore seek an appropriate Markov process which samples

the ground state wavefunction with controlled statistical and systematic errors. For Boson systems, for which the ground state wavefunction may be defined positive everywhere, two essentially equivalent sampling techniques are well known and widely used. The path integral Monte Carlo method [119] and Green's function Monte Carlo method [120] project the ground state from an arbitrary state  $|\phi\rangle$  by evaluating  $(e^{-\epsilon H})^N|\phi\rangle$  and  $(1/(H - E))^N|\phi\rangle$  respectively. For Fermion systems, antisymmetry introduces nearly cancelling positive and negative contributions to physical observables which render straightforward generalizations of either method impractical. Thus, at present, the only alternative for Fermions is to reformulate the problem in terms of a functional integral over some scalar field of an effective action which is bilinear in the Fermion field operators and to integrate out the Fermion fields analytically. In problems such as lattice gauge theory, in which the resulting integrand is positive, although some calculations are feasible, the resulting Fermion determinant is extremely cumbersome computationally and one loses all direct information concerning the behavior of the Fermion ground state. For other problems, such as nuclei and liquid helium 3, there is no known way to introduce an auxiliary field to obtain a satisfactory effective action. Thus, more general and powerful methods are required for Fermion systems.

This work explores more general Markov processes for Fermion systems. The essential idea is to relax the condition, implicit in previous path integral and Green's function approaches, that the random walk should sample the solution to a linear Schrödinger equation. Since sampling the Schrödinger equation unavoidably introduces explicit antisymmetry in the stochastic variables, minus signs and the corresponding near cancellation of large positive and negative contributions are introduced at the very first step and can never be controlled subsequently. Hence, instead we construct a more general Markov process for

two positive functions which satisfy a non-linear coupled Schrödinger equations and whose difference yields the antisymmetric Fermion wavefunction. We emphasize at the outset that we have explored only the simplest generalization we could find that avoids the pathologies of conventional approaches, so the basic idea may well be valid even if the specific form of the non-linear equations and technical details of the present stochastic method are inadequate for large systems. Finally we acknowledge the crucial role that Malvin Kalos [121] had in showing us Eq.(6.9), which led to our thinking about the problem in this way and to the development of the final practical method in Eq.(6.21), as well as the influence of the related work of Arnow *et al.* [122] and Elser [123].

## 6.1 Conventional Path Integral Monte Carlo

To establish the rationale for this new approach, it is useful to recall the basic elements of the conventional path integral Monte Carlo method. We begin with a single particle in a potential, and then generalize to systems of Bosons and systems of Fermions.

A simple way to sample the ground state  $|\psi\rangle$  for a particle in a potential  $V(x)$  is to calculate

$$|\psi\rangle \approx e^{-T(\hat{H}-E)}|\phi\rangle, \quad (6.1)$$

where  $\phi(x)$  is a positive function (and thus not orthogonal to the nodeless ground state),  $T$  is sufficiently large that  $e^{-T(E_1-E_0)} \ll 1$  (so that all excited state admixtures in  $\phi(x)$  are projected out to any desired precision), and  $E$  is a constant (the ground state energy) defined to keep the normalization of  $|\psi\rangle$  finite as  $T \rightarrow \infty$ . If we divide  $T$  into  $n$  steps of length  $\epsilon = T/n$  and approximate  $e^{-T(\hat{H}-E)}$  by the usual Euclidean Feynman path integral [124],

the wave function may be written

$$\begin{aligned}
\langle x_n | \psi \rangle &= \prod_{m=0}^{n-1} \int dx_m \langle x_{m+1} | e^{-\epsilon(\hat{H}-E)} | x_m \rangle \langle x_0 | \psi \rangle \\
&= \prod_{m=0}^{n-1} \left[ \int dx_m P(x_{m+1}, x_m) W(x_m) \right] \phi(x_0),
\end{aligned} \tag{6.2}$$

where  $P(x_{m+1}, x_m) = \sqrt{\frac{M}{2\pi\epsilon}} e^{-(M/2\epsilon)(x_{m+1}-x_m)^2}$  and  $W(x_m) = e^{-\epsilon(V(x_m)-E)}$ .

Each of the integrals over  $x_m$  may be performed sequentially using the fundamental Monte Carlo relation

$$\int dx f(x) P(x) = \frac{1}{N} \sum_{\substack{i=1 \\ x^{(i)} \in P(x)}}^N f(x^{(i)}) \pm \frac{1}{\sqrt{N}} [(\langle f^2 \rangle_P - \langle f \rangle_P^2)]^{1/2}, \tag{6.3}$$

where the integrand is decomposed into the product of a probability distribution  $P(x)$  and a residual function  $f(x)$ . First, an ensemble of  $N$  points  $\{x_0^{(i)}\}$  is selected, distributed according to  $\phi(x_0)$ . Each point is replicated or deleted so that the probability of finding the particular value  $x_0$  in the ensemble is  $W(x_0)$ . For example, if  $[W(x_0)]$  denotes the greatest integer in  $W(x_0)$ , the value  $x_0$  is included in the ensemble  $[W(x_0)]$  times and with probability  $W(x_0) - [W(x_0)]$  is included one additional time. For each  $x_0^{(i)}$ ,  $P(x_1, x_0^{(i)})$  is sampled as a probability distribution for  $x_1$  and the resulting values of  $x$  are replicated or deleted according to the weight  $W(x_1)$  yielding an ensemble  $\{x_1^i\}$ . The  $(m+1)$ st step in the Markov chain is defined by sampling  $P(x_{m+1}, x_m^{(i)})$  for each element of  $\{x_m^{(i)}\}$  and replicating or deleting according to the weight  $W(x_{m+1})$  to obtain  $\{x_{m+1}^{(i)}\}$ .

The form of the resulting branched random walk for the population  $\{x_m^{(i)}\}$  is sketched in figure 6.1. In this example, an initial ensemble of five points is sampled from  $\phi(x)$  which is more delocalized than  $\psi(x)$ . At each step, the points first undergo gaussian diffusion under  $P(x_{m+1}, x_m)$ , which represents the effect of the kinetic energy. The effect of the potential

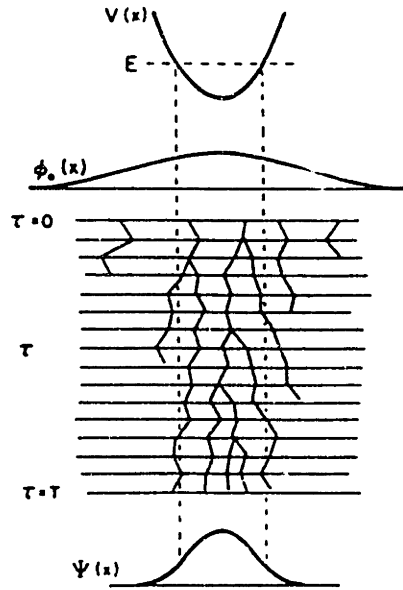


Figure 6.1: Sketch of the branched random walk which samples the ground state distribution for a particle in the potential  $V(x)$ .

is then included by the replication or deletion, with points being removed outside the classical turning points and created in the classically allowed region. Note that the energy  $E$  appearing in the weight  $W(x_n) = e^{-\epsilon(V(x_n)-E)}$  is adjusted like a chemical potential to keep the average population constant. By the final time, the combination of diffusion and replication yields the distribution representative of  $\psi(x)$  as shown.

The generalization of this process for a system of identical Bosons is trivial. The coordinate  $x$  for the single particle is simply replaced by a vector  $\mathbf{x}$  denoting the coordinates for all  $N$  particles  $\mathbf{x}_m \equiv \{x_{1,m}, x_{2,m}, \dots, x_{N,m}\}$ , where in more than one spatial dimension, each of the  $x_{j,m}$ 's is itself a vector coordinate for particle  $j$ . The diffusion term is simply a product of Gaussians

$$P(\mathbf{x}_{m+1}, \mathbf{x}_m) = \left( \frac{M}{2\pi\epsilon} \right)^{N/2} \prod_j e^{-(M/2\epsilon)(x_{j,(m+1)} - x_{j,m})^2} \quad (6.4)$$

and for a two-body interaction  $v(x_j - x_k)$  the weight is

$$V(\mathbf{x}_m) = \exp \left( -\epsilon \left( \sum_{j < k}^N v(x_{j,m} - x_{k,m}) - E \right) \right). \quad (6.5)$$

For future reference, note that the potential involves a sum over each of the particle coordinates for a particular member of the ensemble, but no sum over ensemble members. Although there are many technical improvements to increase the efficiency of practical calculations, such as importance sampling based on an approximate trial function, this basic idea allows one to calculate virtually any desired ground state properties of liquid  $^4\text{He}$  or finite drops of  $^4\text{He}$ . The two errors in the calculation arising from the finite  $T$  in Eq( 6.1) and the finite sampling statistics can both be reduced below any predetermined accuracy level by continuing the Markov process long enough. The error arising from the finite size  $\epsilon$  in the path integral is controlled by choosing  $\epsilon$  sufficiently small and extrapolating to  $\epsilon = 0$ .

For Fermions, however, the conventional method is fundamentally deficient. The essential difficulty is the fact that  $e^{-T\hat{H}}|\phi\rangle$  filters out the lowest eigenstate of  $H$ , irrespective of symmetry, so that at large times the component of the lowest symmetric state is exponentially enhanced relative to the antisymmetric state by the factor  $e^{T(E_A - E_S)}$ . Hence, at some point in the calculation, it is necessary to project onto the antisymmetric space. If this projection can be done exactly, as in the case of exact integration of Fermion fields for a bilinear action mentioned in the Introduction, there is no problem. However, if it is only done stochastically, there is competition between the exponential growth of symmetric noise, which increases as  $e^{T(E_A - E_S)}$ , and the stochastic projection which only decreases as  $1/\sqrt{N}$ .

Formally, a path integral Monte Carlo method for Fermions may be obtained by writing

the Feynman path integral with antisymmetric coordinate states

$$(1/\sqrt{N!}) \sum_P (-)^P |x_{P1} \dots x_{PN} \rangle$$

at each step, with the result that Eq.(6.5) is unchanged and Eq.(6.4) is replaced by

$$P(\mathbf{x}_{m+1}, \mathbf{x}_m) = \left(\frac{M}{2\pi\epsilon}\right)^{N/2} \sum_P (-)^P \prod_j e^{-M/2\epsilon}(x_{j,(m+1)} - x_{j,m})^2} \quad (6.6)$$

The essential features are evident in the simple case of two particle interacting via a two-body potential. Letting  $x = x_1 - x_2$  denote the relative coordinate and  $\mu$  denote the reduced mass and ignoring the irrelevant cm wavefunction, the two-Fermion problem reduces to finding the first odd state in a potential  $v(x)$ . The Markov process in this case is defined by

$$P(x_{m+1}, x_m) = \frac{1}{2} \sqrt{\frac{\mu}{2\pi\epsilon}} [e^{-(\mu/2\epsilon)(x_{m+1} - x_m)^2} - e^{-(\mu/2\epsilon)(x_{m+1} + x_m)^2}] \quad (6.7)$$

and

$$W(x_m) = e^{-\epsilon(v(x_m) - E)}. \quad (6.8)$$

In contrast to the Boson case, the minus sign now requires that we associate with each coordinate sampling the wavefunction a sign, so that effectively the random walk is now characterized by two species of random walkers: + walkers and - walkers. Typical configurations encountered for such a random walk for the first odd state of a one-dimensional harmonic oscillator are sketched in figure 6.2. At the first step, sampling an odd wavefunction yields  $N/2$  + walkers to the right of the origin and  $N/2$  - walkers to the left. At each subsequent step a walker at  $x$  has a 50% probability of undergoing Gaussian diffusion about  $x$  with no sign change and a 50% probability of changing sign and diffusing around  $-x$ . The result, as shown in the lower segments of figure 6.2, is that both the + walkers and the - walkers separately approach the symmetric ground state while their difference, which approaches the desired antisymmetric ground state, is exponentially suppressed. Aside from

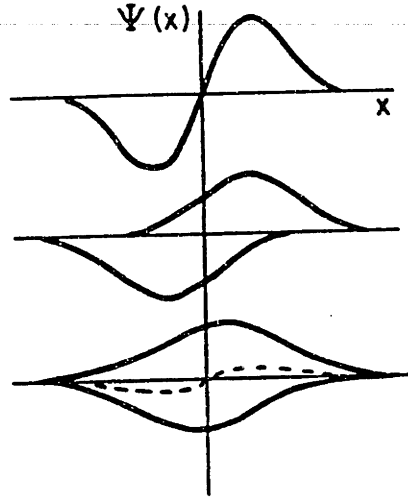


Figure 6.2: Sketch of sequence of configurations obtained for the first odd state of a harmonic oscillator.

the special trick of working in an ordered subspace  $x_1 \leq x_2 \leq \dots \leq x_n$  in one dimension, which cannot be generalized to higher dimensions, all known variants of the conventional path integral Monte Carlo method suffer the fatal flaw shown in figure 6.2.

## 6.2 Generalized Markov Process

Since we have just shown that the essential problem is displayed by the first odd state of a symmetric potential, representing the relative wave function for two Fermions, we will discuss the new method for this special case. The basic ideas are equally applicable to one or more dimensions, and for notational simplicity the equations will generally be written in one dimension.

Let  $\psi^+(x)$  denote the distribution of + walkers and  $\psi^-(x)$  the distribution of - walkers. The essential issue is to introduce a non-linear term which keeps  $\psi^+$  and  $\psi^-$  localized in



separate domains. First, let us consider the following coupled non-linear equations [121] for  $\psi^+(x)$  and  $\psi^-(x)$  which are similar in structure to the familiar Hartree equations for single-particle wavefunctions:

$$\begin{aligned}(T + V(x) + K\psi^-(x))\psi^+(x) &= E\psi^+(x) \\ (T + V(x) + K\psi^+(x))\psi^-(x) &= E\psi^-(x)\end{aligned}\tag{6.9}$$

Subtracting these equations shows that  $\psi^+(x) - \psi^-(x)$  satisfies the desired linear Schrödinger equation

$$(T + V(x))(\psi^+(x) - \psi^-(x)) = E(\psi^+(x) - \psi^-(x)).\tag{6.10}$$

The analogy with Hartree equations suggests that there should exist solutions of the nature we want. Think of the term  $K\psi^-(x)$  as the Hartree potential seen by the orbital  $\psi^+(x)$  generated by the orbital  $\psi^-(x)$  through a zero-range repulsive interaction. The only difference in the Hartree theory is that  $|\psi^-(x)|^2$  rather than  $\psi^-(x)$  appears, but since  $\psi^-(x)$  is a positive function, the qualitative behavior should be the same. Furthermore, since  $\psi^+$  and  $\psi^-$  interchange roles under particle exchange, it is reasonable to seek solutions in which the two single particle energies are degenerate.

Consider now our experience with symmetry breaking in Hartree or Hartree-Fock solutions. We know that when the self-consistent equations are solved iteratively, starting from some initial guess, the solutions converge to some fixed point (or get stuck in some limit cycle which we ignore for our present discussion). A whole range of initial guesses in some basin of attraction will converge to one fixed point whereas a set of initial guesses in another basin of attraction may converge to a different fixed point with different symmetry properties. For example, in some regions of the periodic table or with particular force parameters one may only find spherical solutions for nuclei; in other cases, there may only

be deformed solutions; and in still other cases, one may find either spherical or deformed solutions depending upon the shape of the initial starting guess.

Returning now to Eq.(6.9), in the case in which  $K$  is zero,  $\psi^+$  and  $\psi^-$  satisfy the same Schrödinger equation and thus converge to the symmetric ground state in the symmetric potential  $V(x)$ . When  $K$  is very small, the equations continue to have stable symmetric self-consistent solutions. As  $K$  is increased, however, the repulsive interaction between  $\psi^+$  and  $\psi^-$  continues to increase until one eventually reaches a critical value of  $K$  above which the symmetric solution becomes unstable and the fixed point is a solution with broken symmetry in which  $\psi^+$  is localized in one region and  $\psi^-$  is localized in the opposite region. This is the regime of interest to us. The parameter  $K$  represents a repulsive interaction between unlike walkers which keeps them spatially separated, and the construction of the equations ensures that  $\psi^+(x) - \psi^-(x)$  is the desired solution to the Schrödinger equation. Note that in this try, the nodal structure of the Fermion wavefunction is determined by the phase separation into separate domains of the  $+$  and  $-$  walkers. That this separation should be possible is reasonable because the exact many-Fermion wavefunction minimizes its energy by producing nodal surfaces at the optimal locations. Starting from such a solution, it is plausible that when repulsion is introduced between regions of opposite signs, this same nodal geometry would continue to be stable over a range of  $K$ .

To verify this behavior inferred from the Hartree analogy, equations (6.9) were solved numerically for a harmonic oscillator potential in one and two dimensions. Note that in this simple case, because  $\psi^+(\mathbf{x}) = \psi^+(-\mathbf{x})$ , the equations reduce to a single non-linear equation for  $\psi^+(\mathbf{x})$ . The results for the one-dimensional harmonic oscillator with mass, spring constant, and  $\hbar = 1$  are graphed in figure 6.3 and display all the expected properties.

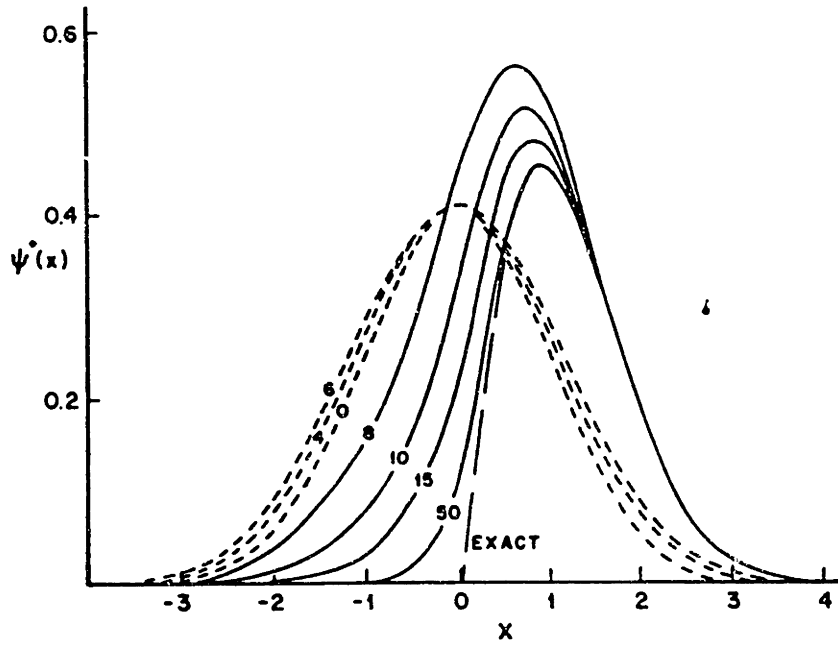


Figure 6.3: Self-consistent solutions  $\psi^-(x)$  to the coupled equations for the first odd state of a harmonic oscillator.

Below  $K \sim 8$ , the solutions are symmetric and above this value they are strongly localized at positive  $x$ . For each broken symmetry solution,  $\psi^+(x) - \psi^-(-x)$  is the exact odd solution to the harmonic oscillator with the correct energy. Observe that if the integral of  $\psi^+(x) - \psi^-(x)$  with any odd function were evaluated stochastically by first sampling  $\psi^+(x)$  and then subtracting the result of sampling  $\psi^-(x)$ , the region of cancellation arising from overlap of  $\psi^+$  and  $\psi^-$  would be well controlled by the use of any reasonable large  $K$ , and the calculation would be free of the catastrophic cancellation sketched in figure 6.2 for the conventional theory. The results for the two-dimensional harmonic oscillator are analogous, with  $\psi^+(x)$  being strongly localized in one half plane for  $K$  larger than the critical value.

The next question is how to adopt equation (6.9) for stochastic evaluation as in section 6.1. Given the fact that the method of choice for solution of many Hartree or Hartree-Fock problems is evolution in imaginary time, at first sight it appears obvious that one should just use the standard path integral Monte Carlo method for  $\psi^+(x)$  including in the

potential term  $(V(x) + K\psi^-(x))$  and similarly for  $\psi^-(x)$ . Unfortunately, this is impossible since at any step in the Markov chain one never knows the functions  $\psi^+(x)$  or  $\psi^-(x)$ . Instead one only has ensembles of points  $\{x^+\}$  and  $\{x^-\}$  distributed according to these functions. Having these ensembles allows us to calculate integrals of  $\psi^+(x)$  and  $\psi^-(x)$  with any function  $f(x)$  but not to construct the functions themselves.

Given that we can calculate integrals over  $\psi^+(x)$  and  $\psi^-(x)$ , suppose we replace equation(6.9) by the non-local equations

$$\begin{aligned}(T + V(x))\psi^+(x) + \int dydz K(x, y, z)\psi^-(y)\psi^+(z) &= E\psi^+(x) \\ (T + V(x))\psi^-(x) + \int dydz K(x, y, z)\psi^+(y)\psi^-(z) &= E\psi^-(x)\end{aligned}\quad (6.11)$$

What requirements must be placed on the kernel  $K(x, y, z)$ ? Subtraction of equation (6.11b) from (6.11a) shows that in order for  $\psi^+ - \psi^-(x)$  to satisfy the original Schrödinger equation,  $K$  must be symmetric in the first two arguments:

$$K(x, y, z) = K(y, x, z). \quad (6.12)$$

In addition, although it may not be strictly necessary, it appears desirable to require that the single-particle Hamiltonians for  $\psi^+(x)$  and  $\psi^-(x)$  be Hermitian so that

$$K(x, y, z) = K(z, y, x) = K(x, z, y). \quad (6.13)$$

Therefore  $K(x, y, z)$  must be totally symmetric. Hence, for any reasonable finite range symmetric kernel one could straightforwardly evaluate the non-local potential appearing in the equation for  $\phi^+(x)$ ,

$$\begin{aligned}U(x, z) &= \int dy K(x, y, z)\psi^-(y) \\ &= \frac{1}{N} \sum_{y_i \in \psi^-(y)} K(x, y_i, z),\end{aligned}\quad (6.14)$$

and write the infinitesimal evolution operator for  $\psi^+$

$$\langle x_{m+1} | e^{-\epsilon(\hat{T} + V(\hat{x}) + U(\hat{x}, \hat{x}'))} | x_m \rangle .$$

However, at this point we lose the positivity of the infinitesimal evolution operator, since the off diagonal matrix elements of  $e^{-\epsilon U(\hat{x}, \hat{x}')}$  may be negative. Hence, as in the conventional path integral Monte Carlo method, + and - walkers intermingle at each step, and there is no way in principle to maintain the phase separation crucial to this new method.

Hence, barring some new insight, it is necessary to introduce an additional approximation. In order to have a local potential for  $\psi^+(x)$  computed from an integral over  $\psi^-(x)$ , we will relax the requirement that  $\psi^+(x) - \psi^-(x)$  exactly satisfy the Schrödinger equation. Instead, we will allow a small error term in the Schrödinger equation governed by an explicit small parameter. We will correct for this small parameter to leading order in perturbation theory and, if necessary, extrapolate the final result in this parameter as we do for the step size  $\epsilon$  appearing in this path integral. In fact, the analogy with the introduction of finite  $\epsilon$  in the path integral is quite suggestive. In order to treat the non-commutativity of  $\hat{T}$  and  $\hat{V}$  and obtain a useful expression for the infinitesimal evolution operator, it was essential to replace  $e^{-\epsilon(\hat{T} + \hat{V})}$  by  $e^{-\epsilon\hat{T}}e^{-\epsilon\hat{V}}(1 + O(\epsilon^2))$  or  $e^{-(\epsilon/2)\hat{V}}e^{-\epsilon\hat{T}}e^{-(\epsilon/2)\hat{V}}(1 + O(\epsilon^3))$ . Similarly, in order to proceed with the Fermionic problem, it is essential to introduce a small controlled approximation at this point.

Therefore, consider the effect of replacing the nonlinear term  $K\psi^-(x)$  in the potential in equation (6.9) by the convolution of  $\psi^-$  with a short-range function. Let  $v(x)$  be a short-range even function with the zeroth and second moments

$$\begin{aligned} \int dx v(x) &= 1, \\ \int dx x^2 v(x) &= a^2 \end{aligned} \tag{6.15}$$

define the convolution with any function  $\phi$  as

$$\begin{aligned}\tilde{\phi}(x) &\equiv \int dy \phi(x+y)v(y) \\ &= \phi(x) + \frac{a^2}{2}\phi''(x) + O(a^4),\end{aligned}\tag{6.16}$$

and let  $\psi^+(x)$  and  $\psi^-(x)$  satisfy the coupled equation

$$(T + V(x) + K\tilde{\psi}^\mp(x))\psi^\pm(x) = E\psi^\pm(x).\tag{6.17}$$

Subtraction of the equations for  $\psi^+$  and  $\psi^-$  and using the expansion 6.16 then yield

$$\begin{aligned}(T + V(x) - E)(\psi^+(x) - \psi^-(x)) &= -K(\tilde{\psi}^-(x)\psi^+(x) - \tilde{\psi}^+(x)\psi^-(x)) \\ &= -K\frac{a^2}{2}(\psi^{--}(x)\psi^+(x) - \psi^{++}(x)\psi^-(x)) + O(a^4).\end{aligned}\tag{6.18}$$

The result shows that the solution of 6.17 yields an antisymmetric solution to the desired Schrödinger equation which is valid to order  $a^2$ , where  $a^2$  can be decreased by decreasing the range of the convolution potential.

At this stage, it is useful to consider the structure of the stochastic solution to the coupled equation 6.17 and its relation to an associated polymer problem. As each  $+$  walker evolves in its random walk, it effectively sees two different interaction potentials. At each time step, each  $+$  walker (which in the present example represents the coordinates of a pair of physical particles) is acted upon by the attractive physical two-body potential  $V$  in the conventional way. In addition, however, each walker of a given species ( $+$  or  $-$ ) sees a repulsive potential  $Kv$  between itself and all other walkers of the opposite species. In coordinate space, one may visualize a domain surrounded by nodal surfaces and filled with  $+$  walkers distributed according to the magnitude of  $\psi^+$ . This domain is surrounded by

domains of  $\psi^-$ , and each walker in these domains generates a small repulsive region of range  $\underline{a}$ , the cumulative effect of which is to create a repulsive barrier around the  $\psi^+$  domain which keeps all the  $+$  walkers confined. Physically, it is clear that this range  $\underline{a}$  must be larger than the mean spacing between walkers in the periphery of the domain in order to confine the walkers and much smaller than the size of the domain itself in order to keep the violation of the Schrödinger equation acceptable small. Looking at the path integral for the  $+$  and  $-$  walkers from a different perspective, the world lines correspond to the partition function for a two-species polymer which has an attractive interaction between all the particles within a given ensemble element and a repulsive interaction between all ensemble elements of unlike species. This analogy potentially may be useful in understanding when the range  $\underline{a}$  and strength  $K$  are capable of giving rise to phase separation and thus domain formation.

Having seen that equation (6.17) has the essential features required for the stochastic solution of the Fermion problem, it is useful to introduce a small modification to remove the leading error of order  $\underline{a}^2$  in equation (6.18). Using the relation

$$\psi^{\pm''}(x) = \frac{2m}{\hbar^2} \{V(x) + K\psi^{\mp}(x) - E\}\psi^{\pm}(x) + O(\underline{a}^2) \quad (6.19)$$

from equation (6.17) to replace the second derivatives in equation (6.18) yields

$$\left(T + V(x) + \frac{K^2 \underline{a}^2 m}{\hbar^2} \psi^+ \psi^-\right) (\psi^+ - \psi^-) = E(\psi^+ - \psi^-) + O(\underline{a}^4). \quad (6.20)$$

Hence, the leading correction of order  $\underline{a}^2$  may be cancelled completely by introducing the additional term  $-(K^2 \underline{a}^2 m / \hbar^2) \tilde{\psi}^+(x) \tilde{\psi}^-(x)$  in the coupled equations defining  $\psi^+$  and  $\psi^-$ .

Thus, the final non-linear equations for solution of the Fermion problem are

$$\left(T + V(x) + K\tilde{\psi}^{\mp}(x) - \frac{K^2 \underline{a}^2 m}{\hbar^2} \tilde{\psi}^+ \tilde{\psi}^-\right) \psi^{\pm}(x) = E\psi^{\pm}(x) \quad (6.21)$$

for which

$$(T + V(x))(\psi^+(x) - \psi^-(x)) = E(\psi^+(x) - \psi^-(x)) + O(\underline{a}^4). \quad (6.22)$$

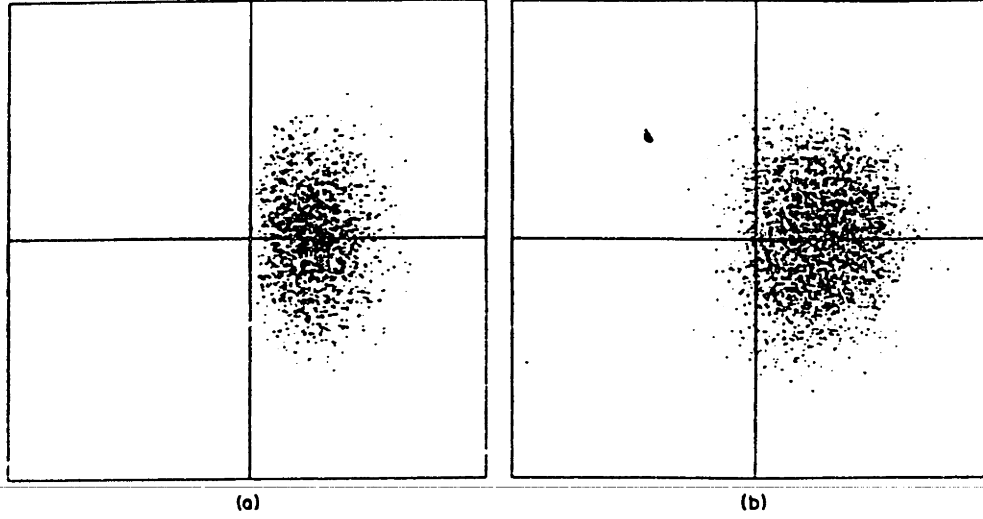


Figure 6.4: Distribution of points sampling the positive region of the first odd eigenstate of the two-dimensional harmonic oscillator.

The dominant non-linear term is still  $K\tilde{\psi}$  which creates a repulsive barrier around the domain boundary. The small attractive correction  $-(K^2 a^2 m / \hbar^2) \tilde{\psi}^+ \tilde{\psi}^-$  enters like a small attractive three-body force and serves as the leading order perturbative correction to repair the damage introduced by the finite range  $\underline{a}$ .

As an example to verify the equation (6.21) indeed works as advertised, we have solved for the first odd state of a two-dimensional harmonic oscillator. Note that for this two-dimensional problem,  $x$  is replaced everywhere by  $\mathbf{x}$ , denoting the relative coordinates in the  $x - y$  plane, and  $\psi^-(\mathbf{x}) = \psi^+(-\mathbf{x})$ . This is the simplest Fermion problem which undergoes the catastrophic buildup of symmetric noise discussed in section 6.1 and thus poses a non-trivial test of the new method. The oscillator Hamiltonian was defined with spring constant, mass and  $\hbar = 1$ , yielding energy 2 and size parameter 1; the step size was  $\epsilon = 0.2$  leading to an error in the total energy of order  $2 \times (\epsilon^2/8) = 1 \times 10^{-4}$ ;  $K$  was 50, the rms radius  $\underline{a}$  was 0.5, and the ensemble had average population  $N = 500$ .



The equilibrium distribution of points  $\{\mathbf{x}_i\}$  sampling  $\psi^+(\mathbf{x})$  averaged over 30 independent samples of 500 points is shown in figure 6.4. Note that because there is no potential acting on the polar angle  $\theta$ , and this angle therefore undergoes Gaussian diffusion with time, we have plotted the ensemble at each time in the frame of the principal axes. Comparison with the distribution of points for the positive portion of the exact wavefunction in the second portion of the picture shows that the nonlinear equations did just what they were supposed to in generating the correct nodal line and spatial distribution.

Calculation of  $E$  in equation (6.17) by ensemble normalization with 800 samples of approximately 500 points, perturbative correction for the  $a^2$  term in equation (6.21), and extrapolation to remove the linear  $\epsilon$  dependence yielded  $1.995 \pm 0.016$ . This energy is consistent with the exact result  $E = 2$  and demonstrates the statistical accuracy attainable with this new method.

### 6.3 Discussion

We hope that the random walk for two positive functions defined by equation (6.21) whose difference satisfies the Schrödinger equation has the potential to overcome the fundamental limitations of conventional techniques for sampling general many-Fermion ground states. Since we have only demonstrated its feasibility for an extremely simple two-particle example, it is useful to consider the obstacles to applying it to larger systems of physical interest and the possible means of overcoming these obstacles.

The biggest practical difference between the present calculation and conventional path integral Monte Carlo calculations is the fact that evaluation of the potential for each en-

semble member requires summations over all the other elements of the ensemble. If the ensemble is too large, it will be impractical to perform enough steps in the path integral to obtain a useful approximation. Thus, the key issue is how rapidly the size of the ensemble must grow with dimensionality to keep the domain separation stable. The worst fear is that as the dimensionality increases, the surface surrounding a given domain becomes progressively more sparsely populated, giving rise the breakthrough of the  $+$  walkers in the  $-$  domain where there are insufficient walkers to provide an adequate barrier.

To some extent, the sparseness of the population may be compensated by increasing the range  $\underline{a}$  of the convolution, since this range controls the size of the repulsive region defined by each walker. Although in the present formulation, increasing the range introduces errors in the Fermion ground state of order  $a^4$ , it may be possible to write more sophisticated correction terms which increase the accuracy to higher order in  $\underline{a}$ .

In addition, since the convolution potential is short range, it is possible to group the walkers into spatial bins and to restrict the ensemble sums to those bins which are within the convolution range.

The biggest conceptual question is whether the nonlinear equation (6.21) sustains broken symmetry solutions with domain structure corresponding to the nodal surfaces of the true ground state Fermion wavefunction. At this point, we can only appeal to the heuristic arguments presented earlier that since the geometry of the nodal surfaces for the Fermion ground state minimizes the nodal surface energy, this same geometry should be favorable for minimizing the additional repulsion associated with  $K$ .

Since these questions about the ensemble size and domain structure cannot be resolved

analytically, we explored them numerically for two- and three-dimensional Fermi gases. We found that a relatively large ensemble size was required to maintain the nodal sub-manifold. Indeed, determining the location of this nodal sub-manifold is the crucial part of the many-Fermion calculation, since the wave function can be immediately determined by a Bosonic stochastic calculation within the space enclosed by the nodal sub-manifold once it is known.

One approach we considered was to choose an initial guess for the nodal sub-manifold and then to iteratively improve this guess by moving the sub-manifold in response to the flux of positive and negative walkers across the nodal boundary. The iteration converges to the true nodal boundary when equal fluxes of positive and negative walkers across the boundary are obtained. While we never designed an algorithmic implementation for this approach, Hamman [125] has been successful in implementing a method for iteratively determining nodal boundaries based on removing discontinuities in the slopes of  $\psi^+$  and  $\psi^-$  at the nodal barrier. With this improvement, the method of using  $\psi^+$  and  $\psi^-$  still shows promise.

The notion of guessing the overall structure of a solution as was suggested here for the nodal sub-manifolds was used by the author to develop a fast, accurate digital radio direction finding system based on wide baseline interferometry. In this case, the unknown parameter is the number of wavelengths of a radio signal between two antennas spaced multiple wavelengths apart. This parameter depends on the (unknown) elevation of the radio source relative to the interferometer. Once the integer part of the number of wavelengths is correctly guessed, it is relatively easy to use phase difference measurements to obtain the fractional part and a conventional correlation detector to estimate the location of the radio source. The details are beyond the scope of this thesis, but can be found in U.S. Patent 5,099,248 [126].



## Chapter 7

# Conclusion

This thesis has presented the first *ab initio* investigation of a material system with complexity on the scale of one thousand atoms. We used the unprecedented level of performance possible with massively parallel computation to make the first *ab initio* study of the Takayanagi  $7\times 7$  reconstruction of the Si(111) surfaces, a famous problem for its complexity, both in terms of the rich surface physics and in terms of how it has eluded realistic theoretical treatment.

Our first result was a careful determination of the surface energy. This calculation was essential for initial verification of the accuracy of the computation. It is known experimentally that the (111) surface of silicon cleaves into a metastable  $(2\times 1)$  phase. It is only through annealing at 600 degrees Kelvin that the surface reconstructs into the complicated  $(7\times 7)$  phase. Our value of 60 meV per surface atom for the energy difference is the most accurate to date.

Ionic positions were determined to an accuracy of  $.05\text{\AA}$ . Taking into account the various symmetries of the unit cell, there are roughly one hundred unique coordinates describing

locations of the atoms in the first four layers of the surface. The most important result involved the unknown difference in heights of atoms on the faulted and unfaulted halves of the unit cell. It is known from STM that the unoccupied states of Si(111) produce a flat STM image, while the occupied states produce an asymmetric image with the faulted side of the unit cell raised in height by about  $.2 \text{ \AA}$ . The separate contributions due to structural and electronic differences between halves of the unit cell cannot be determined through STM. We found a structural height difference of  $.04 \text{ \AA}$  between adatoms on faulted and unfaulted sides of the unit cell, roughly half the value estimated from LEED experiments. This result leads us to believe that the parametric fit to LEED data overestimated the structural asymmetry.

We found that our computations of isocharge contours of the electronic wave function according to the method of Tersoff and Hammann accurately reproduced STM results, including a height difference between faulted and unfaulted halves of the unit cell of  $0.2 \text{ \AA}$  for the occupied electronic states at a tip bias of 2.0 Volts. This result proves that the asymmetry between faulted and unfaulted halves of the unit cell is predominantly due to electronic charge transfer, rather than structural differences between the two sides of the unit cell.

We proceeded to compare properties of our electronic wave function with differential STM measurements used to measure qualitative properties of individual surface states of Si(111). Our results agree with published images of occupied and unoccupied surface states for the adatom and rest atom layers. We found many details related to the dangling bond in the sub-surface corner hole of the unit cell that we offer as challenges for experimental detection. We believe that the experimental inaccessibility of the large amount of energetic

charge in the corner hole shows the current limits in resolution possible with STM. This result has technological implications related to our ability to observe and fabricate three-dimensional devices on a nanometer length scale.

We also made the first detailed comparison between theory and experiment regarding the chemical reactivity of electronic surface states, an exciting area since the diversity of bonding sites on the Si(111) surface enhances our understanding of surface chemistry. The different reactive sites on this surface are identical to first order in terms of classical electronegativity analysis and yet exhibit distinctly different chemical behavior. A second-order theory of chemical reactivity in terms of local softness and charge capacity at each reactive site explains the preferential chemisorption of the surface for a variety of reactants. Our results explain reported single-radical chemisorption experiments and refine the interpretation of preferential chemisorption of clusters of metallic atoms at different surface sites. We conclude with predictions of surface chemisorption for unreported reactants. These predictions are made using the charge capacity of the surface and the softness and electronegativity of the reactants without recourse to expensive first-principles calculations for the surface-reactant system.

This thesis shows that massively parallel computation allows us to investigate material systems of unprecedented scope. Using the Local Density Approximation, we investigated the most complex and widely studied semiconductor surface, the Si(111)-(7×7) reconstruction. This problem was previously too large for realistic theoretical treatment. Our LDA calculation reproduced well-established experimental results, refined our understanding of more subtle aspects of the surface behavior and predicted how surface chemisorption should proceed for different reactants. The results demonstrate that both theory and experiment

are now capable of accurately describing detailed electronic properties of material systems at nanometer length scales. LDA and chemical softness computations are now established as valuable tools for analysis of nanometer-scale structures, an area of increasing technological importance.



# Bibliography

- [1] K. D. Brommer, M. Needels, B. E. Larson and J. D. Joannopoulos, *Phys. Rev. Lett.* **68**, 1355 (1992).
- [2] J. Travis, *Science* **255**, 1354 (1992).
- [3] G. Srivastave, *Physics World*, March 1992.
- [4] Thinking Machines, Connection Machine CM-5 Documentation, 1992.
- [5] T. Arias, M. Payne and J. Joannopoulos, *Phys. Rev. Lett.* **69**, (1992).
- [6] V. Zakharov, *IEEE Trans. Computers* **C-33**, 45 (1984).
- [7] J. P. Hayes, *Digital System Design and Microprocessors*, McGraw-Hill, New York, NY, 1984.
- [8] L. F. Menabrea, *Bibliothèque Universelle de Genève, Série 3, Tome 41*, 352 (1842).
- [9] P. Denyer and D. Renshaw, *VLSI Signal Processing: A Bit-Serial Approach*, Addison-Wesley, Reading, MA, 1985.
- [10] H. Lorin, *Parallelism in Hardware and Software: Real and Apparent Concurrency*, Prentice-Hall, Englewood, NJ, 1972.
- [11] T. Jones, *IEEE Computer* **22**, 36 (1989).
- [12] Cray-1 hardware reference manual no. 2240004, Rev. E, 1980.
- [13] M. J. Flynn, *IEEE Trans. Computers* **C-21**, 948 (1972).
- [14] R. Duncan, *IEEE Computer* **23**, 5, February 1990.
- [15] S. H. Unger, *Proc. IRE* **46**, 1744 (1958).
- [16] G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick and R. A. Stokes, *IEEE Trans. Computers* **C-17**, 746 (1968).
- [17] B. A. Bowen and W. R. Brown, *VLSI Systems Design for Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [18] Falk, *IEEE Spectrum*, October, 1976.

- [19] Raytheon Company Submarine Signal Division report *Multipl Array Processing Study* prepared for Naval Electronic Systems Command, Project NE-31-320-06, (1976).
- [20] C. H. Knapp and G. C. Carter, IEEE Trans. Acoustics, Speech and Signal Processing ASSP-24, 320 (1976).
- [21] L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
- [22] J. Bronder, V. V. W. Moseley, A. Wenk, Proc. Govt. Microcircuit Applications Conference, Las Vegas, NV, p. 168, 1984.
- [23] W. K. Gass *et al*, Proc. IEEE 75, 1246 (1987).
- [24] K. D. Brommer, unpublished technical memorandum, Analytyx, Hudson, NH (1988).
- [25] F. F. Yassa, J. R. Jascia, R. I. Hartley and S. E. Noujaim, Proc. IEEE 75, 1272 (1987).
- [26] Report of the IEEE Scientific Supercomputer Subcommittee, IEEE Computer 21, 70, December 1988.
- [27] B. J. Robinet, in *High-Level Language Computer Architecture*, Y. Chu, editor, Academic Press, London, 1985.
- [28] Analogic Corp., Computer Design, 213, December 1984.
- [29] J. Backus, Communications of the ACM 21, 613 (1978).
- [30] K. E. Iverson, *A Programming Language*, Wiley, New York, 1962.
- [31] K. E. Iverson, ACM Trans. Programming Languages and Systems 1, 161 (1979).
- [32] K. D. Brommer, S.M. Thesis, MIT, 1988.
- [33] K. D. Brommer and A. J. Deerfield, "Macro Function Set Formalization", Final Report No NADC-85071-50 on Contract NADC-N62269-83-C-0441 to Naval Air Development Center, 1985.
- [34] K. D. Brommer, D. J. Dechant, A. J. Deerfield and R. Layton, "Macro Function Validation Model Development," Final Report, Contract NADC-N62269-83-C-0440
- [35] K. D. Brommer, A. Deerfield, S. Dymek, P. Martin and S. Scheid "Intelligent Memory for Use in Design and Implementation of Digital Processing Systems Addendum," Final Report NADC-82168-50 on Contract N62269-82-C-0492.
- [36] K. D. Brommer, A. J. Deerfield and R. Fedorak, Government Microcircuit Applications Conference, Orlando, FL 1985.
- [37] K. D. Brommer, M. Kantrowitz and A. J. Deerfield, Government Microcircuit Applications Conference, Orlando, FL 1987.
- [38] K. D. Brommer and T. Kline, AIAA Computers in Aerospace Conference, Monterey, CA 1989.

- [39] AT&T Bell Laboratories, Enhanced Modular Signal Processor Arithmetic Processor Programmers Handbook, 1987.
- [40] H. Abelson, G. J. Sussman and J. Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, 1985.
- [41] H. Abelson and G. J. Sussman, "Lisp : A Language for Stratified Design", MIT AI Lab memo 986, 1987.
- [42] S. Ahuja, N. Carriero and D. Gelertner, IEEE Computer **19**, 26, August 1986.
- [43] J. M. Sipelstein and G. E. Blelloch, Proc. IEEE **79**, 504 (1991).
- [44] Thinking Machines, Connection Machine Technical Summary, 1990.
- [45] B. M. Boghosian, Computers in Physics, 14, January 1990.
- [46] R. Car and M. Parrinello, Phys. Rev. Lett. **55**, 2471 (1985).
- [47] M.P. Teter, M.C. Payne and D.C. Allan, Physical Review **B 40**, 12255 (1989).
- [48] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias and J.D. Joannopoulos, Rev. Mod. Phys. **64**, 1045 (1992).
- [49] J.D. Joannopoulos and M.L. Cohen, J. Phys. C **6**, 1572 (1973).
- [50] J. Ihm, A. Zunger and M.L. Cohen, J. Phys C **12**, 4409 (1979).
- [51] M. C. Payne, J. D. Joannopoulos, D. C. Allan, M. P. Teter, and D. H. Vanderbilt, Phys. Rev. Lett. **56**, 2656 (1986).
- [52] A. Williams and J. Soler, Bull. Am. Phys. Soc. **32**, 562 (1987).
- [53] L. Kleinman and D. M. Bylander, Phys. Rev. Lett. **48**, 1425 (1982).
- [54] D.R. Hamann, M. Schlüter and C. Chiang, Phys. Rev. Lett. **43**, 1494 (1979).
- [55] M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill, New York, 1987.
- [56] A.V. Oppenheim and R. Shafer, *Digital Signal Processing*, Prentice Hall, 1974.
- [57] J. Perdew and A. Zunger, Phys. Rev. **B 23**, 5048 (1984).
- [58] D.C. Allan, unpublished, 1987.
- [59] M.T. Yin and M.L. Cohen, Phys. Rev. **B 26**, 5668 (1982).
- [60] K. D. Brommer, B. E. Larson, M. Needels and J. D. Joannopoulos, Computers in Physics, May 1993.
- [61] A. Zangwill, *Physics at Surfaces*, Cambridge University Press, Cambridge, 1988.
- [62] D. Haneman, Rep. Prog. Phys. **50**, 1045 (1987).

- [63] R. E. Schlier and H. E. Farnsworth, J. Chem. Phys. **30**, 917 (1959).
- [64] G. Binning, H. Rohrer, Ch. Gerber, and E. Weibel, Phys. Rev. Lett. **50**, 120 (1983).
- [65] A. Bennett, L. C. Feldman, Y. Kuk, E. G. McRae, and J. P. Rose, Phys. Rev. B **28**, 3656 (1983).
- [66] E. G. McRae and C. W. Caldwell, Phys. Rev. Lett. **46**, 1632 (1981)
- [67] E. G. McRae, Phys. Rev. B **28**, 2305 (1983).
- [68] K. Takayanagi, Y. Tanishiro, M. Takahashi, and S. Takahashi, J. Vac. Sci. Technol. A **3**, 1502 (1985).
- [69] R. S. Becker, J. A. Golovchenko, E. G. McRae and B. S. Swartzentruber, Phys. Rev. Lett. **55**, 2028 (1985)
- [70] R. S. Becker, J. A. Golovchenko, D.R. Hamann and B. S. Swartzentruber, Phys. Rev. Lett. **55**, 2032 (1985).
- [71] R. M. Tromp and E. J. van Loenen, Surf. Sci. **155**, 441 (1985).
- [72] R. J. Hamers, R. M. Tromp and J. E. Demuth, Phys. Rev. Lett. **56**, 1972 (1986).
- [73] H. Huang, S. Y. Tong, W. E. Packard, and M. B. Webb, Phys. Lett. A **130**, 166 (1988).
- [74] S. Y. Tong, H. Huang, C. M. Wei, W. E. Packard, F. K. Men, G. Glander, and M. B. Webb, J. Vac. Sci. Technol. A **6**, 615 (1988).
- [75] G. X. Qian and D. J. Chadi, Phys. Rev. B **35**, 1288 (1987).
- [76] R. D. Meade and D. Vanderbilt, Phys. Rev. B **40**, 3905 (1989).
- [77] M. Fujita, H. Nagayoshi and A. Yoshimori, Surf. Sci **242**, 229 (1991).
- [78] K.C. Pandey, Phys. Rev. Lett. **47**, 1913 (1981).
- [79] I. K. Robinson, W. K. Waskiewicz, P. H. Fuoss and L. J. Norton, Phys. Rev. B **37**, 4325 (1987).
- [80] I.K. Robinson and E. Vlieg, Surf. Sci. (1991).
- [81] A. Ichimiya, Surf. Sci. **192**, L893 (1987).
- [82] F. Grey, R. L. Johnson, J. Skov Pederson, R. Feidenhans'l and M. Nielsen, *The Structure of Surfaces II*, edited by J. F. van der Veen and M. A. van Hove (Springer-Verlag, Berlin, 1987), p. 292.
- [83] I. Štich, M.C. Payne, R.D. King-Smith, J.S. Lin, and L.J. Clarke, Phys. Rev. Lett. **68**, 1351 (1992).
- [84] M.Y. Chou, M.L. Cohen and S.G. Louie, Phys. Rev. B **32**, 7979 (1985).

- [85] J. Tersoff and D. R. Hamann, Phys. Rev. **B 31**, 805 (1985).
- [86] Ph. Avouris and R. Wolkow, Phys. Rev **B 39**, 5091 (1989).
- [87] R. M. Tromp, R. J. Hamers and J. E. Demuth, Phys. Rev **B 15**, 1388 (1986).
- [88] R. J. Hamers, R. M. Tromp and J. E. Demuth, Surf. Sci. **181**, 346 (1987).
- [89] R. Wolkow and Ph. Avouris, Phys. Rev. Lett. **60**, 1049 (1988).
- [90] J. Northrup, Phys. Rev. Lett. **56**, 998 (1986).
- [91] Y. Chabal, Phys. Rev. Lett. **50**, 1850 (1983).
- [92] Y. Chabal, Phys. Rev. **B 28**, 4472 (1983).
- [93] Y. Chabal in *Internal Reflection Spectroscopy: Theory and Application*, Mirabella, ed., Marcel Dekker, New York, 1992.
- [94] K. D. Brommer, B. E. Larson, M. Needels and J. D. Joannopoulos, Japanese Journal of Applied Physics, March 1993.
- [95] Ph. Avouris, J. Phys. Chem. **94**, 2246 (1990) and references therein.
- [96] Ph. Avouris and I. Lyo in *Chemistry and Physics of Solid Surfaces VIII* eds. R Vanselow and R. Howe, Springer Verlag, Berlin, 1990.
- [97] J. J. Boland, Surf Sci **244**, 1 (1991).
- [98] R. G. Parr and W. Yang, *Density Functional Theory of Atoms and Molecules*, Oxford Univeristy Press, New York, 1989.
- [99] W. Yang and R. G. Parr, Proc. Nat. Acad. Sci. USA **82**, 6723 (1985).
- [100] Ch. Lee, W. Yang and R. G. Parr, J. Mol. Struct. (THEOCHEM) **163**, 305 (1992).
- [101] W. Yang and W. J. Mortier, J. Am. Chem. Soc. **108**, 5708 (1986).
- [102] F. Méndez, M. Galván, A. Garritz, A. Vela and J. L. Gázquez, J. Mol. Struc. (THEOCHEM) **277**, 81 (1992).
- [103] M. Galván, A. Dal Pino and J. D. Joannopoulos, Phys. Rev. Lett. **70**, 21 (1993).
- [104] A. Dal Pino, M. Galván, T. A. Arias and J. D. Joannopoulos, J. Chem. Phys. **98**, 1606 (1993).
- [105] M. Galván, A. Dal Pino, J. Wang and J. D. Joannopoulos, J. Phys. Chem. **97**, 783 (1993).
- [106] Cherif Surf. Sci. (1992).
- [107] L. M. Falicov and G. M. Sommorjai, Proc. Nat. Acad. Sci **82**, 2207 (1985).
- [108] K. Fukui, Science **218**, 747 (1987).

- [109] M. K. Harbola, P. K. Chattaraj and R. G. Parr, *Isr. J. Chem.* **31**, 395 (1991).
- [110] P. Politzer, *J. Chem Phys.* **86**, 1072 (1987).
- [111] J. E. Huheey, *J. Phys. Chem.* **69**, 3284 (1965).
- [112] St. Tosch and H. Neddermeyer, *Phys. Rev. Lett.* **61**, 349 (1988).
- [113] U. K. Koehler, *J. Chem. Phys.* **89**, 1709 (1988).
- [114] T. Hashizume, *Proc. STM* 1989.
- [115] R. G. Pearson, *Inorg. Chem.* **27**, 734 (1988).
- [116] R. G. Pearson, *Inorg. Chem.* **54**, 1423 (1989).
- [117] K. D. Brommer, M. Galván and J. D. Joannopoulos, submitted to *Surf. Sci.* 1993.
- [118] J. W. Negele and K. D. Brommer, *Annals of Physics* **192**, 119 (1989).
- [119] C. M. Ceperley and M. H. Kalos, *Monte Carlo Methods in Statistical Mechanics*, K. Binder, Ed. Springer-Verlag, New York, 1979.
- [120] J. W. Negele and H. Orland, *Quantum Many-Particle Systems*, Chapter 8, Addison-Wesley, Reading, MA, 1988.
- [121] J. Carlson and M. H. Kalos, *Phys. Rev. C* **32**, 1735 (1985).
- [122] D. M. Arnow, M. H. Kalos, M. A. Lee and K. E. Schmidt, *J. Chem. Phys.* **77**, 11 (1982).
- [123] V. Elser, *Phys. Rev. A* **34**, 2293 (1986).
- [124] R. P. Feynman and A. R. Hibbs, *Quantum Mechanics and Path Integrals*, McGraw-Hill, New York, 1965.
- [125] D. R. Hamann, private communication.
- [126] K. D. Brommer, U.S. Patent 5,099,248. (1992).

## **About the Author**

Karl Brommer was born in Pine Grove, a small town in the mountains of eastern Pennsylvania. His bachelor's degree is from Cornell in engineering physics and his master's is from MIT in physics. For the last twelve years he has been doing research toward a doctorate at MIT. During this time he designed computers for five years in the radar laboratory at the Raytheon Equipment Division. For the last five years, Lockheed Sanders Surveillance Systems Division employed him to design diverse aspects of sophisticated electronic equipment. His current position is Senior Principal Electrical Engineer. He has published a few dozen papers related to computers, signal processing and the physics of electronic and photonic systems. He holds several patents related to signal processing and microwave technology. He is a former Lieutenant in the U.S. Naval Reserve. Along with his wife Connie and his children Tracey and Dieter, he lives on the New Hampshire seacoast.