Financial Simulations on a Massively Parallel Connection Machine

by

James M. Hutchinson & Stavros A. Zenios

OR 213-90 April 1990

Financial Simulations on a Massively Parallel Connection Machine

James M. Hutchinson [1]
Stavros A. Zenios [2]

April, 1990.

[1] Thinking Machines Corporation, 245 First St., Cambridge, MA 02142.
[2] Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104. Currently with the Operations Research Center of the Sloan School, MIT, and Thinking Machines Corporation, Cambridge, MA.

Contents

Abstract

The valuation of interest rate contingencies relies on simulation methods that integrate models of the term structure of interest rates with estimations of the contingent cash flows. Such simulations are quite complex and computationally intensive.

In this paper we describe the development of massively parallel procedures for financial modeling simulations on systems like the Connection Machine. Several primitives — generic among multiple applications — are first presented, and the discussion culminates in an *option adjusted analysis* model for mortgage-backed securities. The primitives achieve sustained computing rates in the range 30 — 120 MFLOPS on a 4K processor Connection Machine, model CM-2a. The response time for a complete option adjusted analysis (1 - 2 seconds) makes the model usable for real time analysis and trading applications.

1 Introduction

It is difficult to find an area of the fixed income market that does not rely on analytic techniques for instrument pricing. When the instruments under study embody features of options, and are contingent on the prevailing interest rate environment, the pricing methodology is often based on statistical simulations. Interest rate sensitive instruments appear in several forms in European and American style options, callable and putable bonds, mortgage backed securities and their derivatives, and products from the insurance industry such as guaranteed insurance products and single premium deferred annuities.

For several of those instruments the cash-flows are not only contingent on the prevailing interest rate, but also depend on the path of interest rates during the lifetime of the instrument. Path dependencies create huge sample spaces from which the simulation model has to draw a representative sample. Consider, for example, a binomial lattice model of interest rates — as explained later in Section 2.2.2 — over a period of 30 years in monthly intervals. Short term interest rates can be at any one of 360 possible states at the end of the period, and the number of paths that lead to these states is 2^{360}. Even with the use of variance reduction techniques, a sample of several hundred paths — typically around 1000 — has to be analyzed in order to obtain pricing estimates within acceptable error bounds. It is not surprising then that the valuation of interest-rate-contingent and path-dependent instruments leads to compute intensive simulations.

The computational complexities of these models led users to investigate parallel computers as suitable compute-servers for such applications. The rapidly changing environment where the analyses are performed makes the quest for faster response times even more pressing. The interest of Wall Street analysts in parallel computers has been the topic of recent articles in popular magazines [4]. It appears, however, that no significant progress has been made in utilizing this technology and we are not aware of any publications that discuss parallel computing applications in financial analysis.

In this paper we report on the use of massively parallel architectures — like the Connection Machine CM-2 — for the simulation of interest-rate-contingent and path-dependent cash flows. Simulating multiple paths of interest rates is indeed an *embarrassingly parallel* procedure. Exploiting

[4] See the article "Supercomputers: Era Dawns on Wall Street" in the November 1989 issue of *Wall Street Computer Review* and the special report "Supercomputers Break Through" in the May 1988 issue of *Datamation*.

parallelism in performing the path-dependent calculations appears, on first examination, to be impossible. However, with suitable reformulation of the models and use of the parallel prefix primitives of the Connection Machine we are able to exploit parallelism both in executing multiple simulations and in performing the path-dependent calculations along each path. The result is a library of primitives that run efficiently on the CM-2 and achieve sustained computing rates of 30 — 120 MFLOPS on a 4K processor system CM-2a. The library is used in building an *option adjusted analysis* model for mortgage backed securities. This system can price mortgage backed securities in real time. It is, therefore, now feasible not only to price complex instruments in real time, but also compare alternative instruments under a host of interest rate scenarios and other exogenous factor that affect the price of such instruments. For example analysing the large variety of pools of mortgage pass through securities that exist today in the U.S. (say 3000) would require over 14 hours on a CRAY supercomputer and a more than three weeks on an Apollo workstation. This analysis can be carried out on a 4K CM-2a in approximately 1.5 hours, and in less than 5 minutes on a fully configured CM-2 with 64K processing elements.

In Section 2 we discuss the methodology for valuating interest rate contingencies. Section 3 provides a brief overview of the Connection Machine CM-2. Section 4 provides details on the parallelization of key modules of the valuation methodology, and Section 5 reports timing results and comparisons with an identical model running on a CRAY X-MP vector supercomputer. Concluding remarks are the topic of Section 6.

2 Valuation of Interest Rate Contingent Cash-flows

In this section we review the methodology commonly adopted in the valuation of interest rate contingent cash flows as given, for example, in the generalized pricing framework of Jacob, Lord and Tilley [1986]. Applications in the mortgage-backed securities market are reported in Brazil [1988], Hayre and Lauterbach [1988] or Pinkus et al. [1987] and the valuation of single premium deferred annuities is discussed in Asay et al. [1989].

The general framework of the valuation analysis has three phases:

Phase I: Generate arbitrage free interest rate scenarios that are consistent with the prevailing term structure of interest rates.

Phase II: Generate cash flows along each interest rate scenario.

Phase III: Use the cash flows and the short-term interest rates along each path to compute expected net present value of the cash flows, or to compute an *option adjusted spread* over the Treasury yield curve.

In the approach we take here — known as *option adjusted analysis* — it is assumed that the interest rate contingency is priced by the market, and hence the purpose of the analysis is to calculate the option adjusted spread (*oas* for short). This quantity indicates the incremental spread of the contingency over the Treasury yield curve, that is implied by the market price — a precise definition of *oas* is given in the following section, equation (1). The significance of this measure in the context of investment decisions is documented in Hayre and Lauterbach [1988] for mortgage backed securities, and in Asay et al. [1989] for insurance products.

2.1 Option Adjusted Analysis

The overall design of an option adjusted analysis (OAA, for short) model is illustrated in Figure 1. For each iteration of the simulation the model accepts as input a series of short term forward rates, estimated cash-flows at every point in time, and the market price of the security. Let

S , be the sample of interest rate scenarios (paths) with cardinality $\mid S \mid$,

r_t^s , be the short term forward rate at time period $t \in \{1, 2, 3, \ldots, T\}$ under scenario $s \in S$,

cf_t^s , be the cash-flow at time period t under scenario s.

The option adjusted spread is the incremental spread over the short-term rates that equates the expected present value of the cash-flows under all scenarios with the market price. If P denotes the market price, the option adjusted spread is obtained by solving for *oas* the equation

$$P = \frac{1}{\mid S \mid} \sum_{s \in S} \left\{ \sum_{t=1}^{T} cf_t^s \prod_{\tau=1}^{t} \frac{1}{(1 + r_\tau^s + oas)} \right\} \qquad (1)$$

We assume in the sequel that a fair price for the security (P) is determined by the market, and we want to estimate the value of *oas* that will satisfy equation (1). Hence, we need to solve a nonlinear equation in the

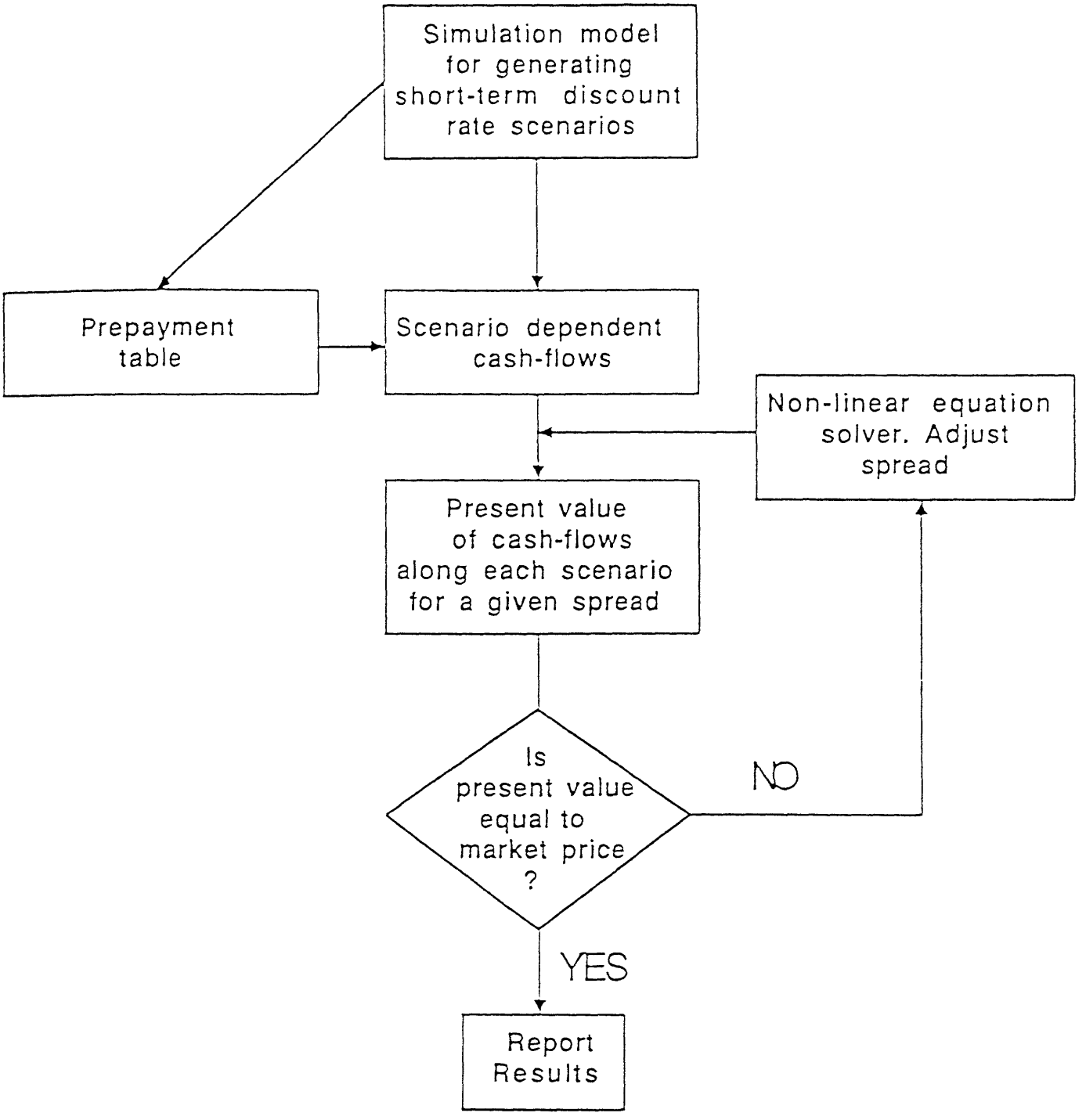

Figure 1: The Option Adjusted Analysis Model.

unknown oas. This nonlinear equation is solved using an iterative method, see Kahaner et al. [1989], that generates a sequence $\{oas_k\}$ until the difference between the right and left hand sides of equation (1) is less than a user specified tolerance (e.g., .1 basis point). For every trial point oas_k of the nonlinear equation solver we have to evaluate the expression on the right hand side of equation (1). The massively parallel evaluation of this expression is the topic of Section 4.1. We turn now to the problem of generating the sample of interest rate paths S.

2.2 Generating Scenarios of Interest Rate Paths

Models for the evolution of the term structure of interest rates are of central interest in financial modeling applications. Two commonly used procedures — for which we have developed massively parallel methods — are based on (1) the Monte Carlo simulation of a diffusion process, and (2) on the construction of binomial lattices.

2.2.1 Monte Carlo Simulation

Interest rates are assumed to follow a lognormal distribution with mean reversion modifications that keep the interest rate paths within historically acceptable bounds, (Cox, Ingersoll and Ross [1985]). Drift factors are used to calibrate the model for consistency with the term structure of interest rates.

The one-year forward rate f_t at time $t \in \{1, 2, 3, \ldots, T\}$, is generated from the rate at period $t-1$ as follows:

$$\log \frac{f_t}{f_{t-1}} = z * \sigma_t + R(f_t) + \mu_t \tag{2}$$

where:

z is a normally distributed random variable, $z \sim N(0,1)$,

σ_t is the volatility of interest rates per unit time, at instance t,

$R(f_t)$ is a mean-reversion term that forces the simulated rates to stay within historically acceptable limits defined by

$$R(f_t) = \begin{cases} 0 & if \;\; l \leq f_t \leq u \\ -\gamma_0 (f_t - u)^2 & if \;\; f_t > u \\ \gamma_1 [(l - f_t)/f_t]^2 & if \;\; f_t < l \end{cases} \tag{3}$$

where γ_0 and γ_1 are constants estimated from empirical data.

μ_t are drift factors estimated so that the model rates are consistent with the term structure. They are estimated by requiring that the present value of on-the-run treasuries, computed using the model rates, is in agreement with the current market prices.

A large number of interest rate paths can be generated by repeated application of equation (2).

2.2.2 Sampling from a Binomial Lattice

An alternative approach to Monte Carlo simulation is to assume that term structure movements can be approximated by a discrete binomial process, represented by a lattice as shown in Figure 2. Discrete points in time are marked on the horizontal axis, and nodes of the lattice represent possible states of interest rates at every point in time. The term structure can move to one of two possible states between successive points in time — conveniently called the "up" and "down" states. The lattice is connected in the sense that an "up, down" and a "down, up" path starting from the same state will lead to the same state after two periods. After t time periods from the origin the lattice has t possible states. Each one of this states can be reached through 2^t possible paths. For example, a binomial lattice of interest rates over a 30 year time horizon in monthly intervals has 360 possible states at the end of the planning horizon, and a formidable 2^{360} possible paths of interest rates that lead to these states.

Short term forward rates at the nodes of the lattice can be computed based on market data, in such a way that the arbitrage free property is satisfied. Alternative techniques for fitting a binomial lattice to market data are beyond our present task. Readers are referred to Ho and Lee [1986] or Sharpe [1985] for comprehensive discussions, and to Bookstaber, Jacob and Langsam [1986] for critique. In our work we use the single factor model of Black, Derman and Toy [1987].

Once the binomial lattice has been fit to the current term structure — in itself a difficult and compute intensive process — we can represent the short term rate at time period t and at state ω by the relation

$$r_{t\omega} = r_{t0} k_t^{\omega} \tag{4}$$

The quantities r_{t0}, k_t for $t = \{1, 2, 3, \ldots, T\}$ represent the 0-th (i.e., ground) state, and the volatility of short term rates at period t; they are parameters

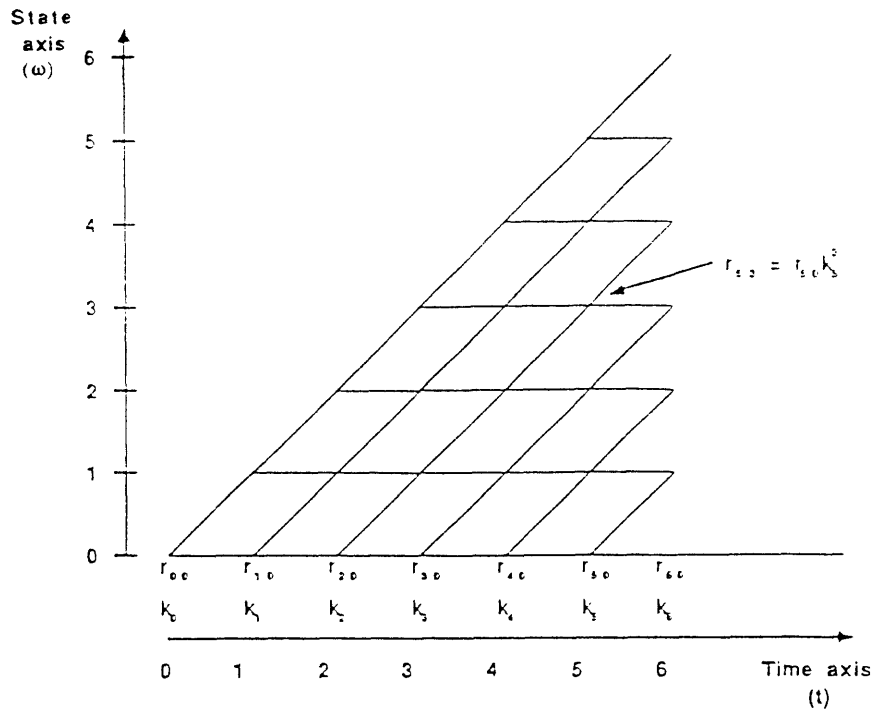

Figure 2: Binomial Lattice of Interest Rates.

estimated by the binomial lattice model of choice. A large number of interest rate scenarios is computed by sampling paths from the binomial lattice using equation (4). A massively parallel procedure for this task is presented in Section 4.3.

2.3 Cash-flow Calculations for Mortgage Backed Securities

We now complete the presentation of the valuation framework by reviewing cash-flow generation methods for a particular class of interest rate contingencies: *mortgage backed securities* (MBS, for short). For a comprehensive treatment of this topic refer to Fabozzi [1987] or Bartlett [1988].

The cash-flow model determines the cash flow at discrete points in time, (i.e., $t \in \{1, 2, 3, \ldots, T\}$) for each interest rate scenario. The first step is to determine the monthly prepayment rate. We use data provided in the form of tables of monthly survivorship (the complement of prepayment). Each table has as its dimensions the range of possible interest rates — e.g., 6% to 16% — and the age of the security. If the one-year forward rate from the simulation model is f_t in month t, we simply look up the monthly survivorship in the $t-th$ column of the table, and at the row that corresponds to the interest rate closest to f_t. These prepayment tables are large datasets: the range of interest rates is divided into small increments (e.g., in increments of .01 basis points), and the maximum number of months under consideration for the security is 360. Hence, each table has 360,000

entries.

Once the monthly survivorship is read from the table it is straightforward to calculate the cashflows: see for example Fabozzi [1985]. The interest rate on the underlying mortgages, i, and the servicing rate, s, are known. Also known is the remaining term, n, and the current mortgage balance mb_0. For each month we calculate the monthly cashflow that would occur without any prepayment of principal or servicing fees, and then add prepayment of principal and net the servicing fee. We give details below.

Let the monthly cashflow, interest and principal, expected in month t without prepayment or servicing fees be mp_t, and the remaining balance at the end of month t be mb_t. Then:

$$mp_t = mb_{t-1}\frac{i(1+i)^{n-t+1}}{(1+i)^{n-t+1}-1} \tag{5}$$

The dollar amount of interest paid by the homeowner, in month t, is

$$I_t = mb_{t-1} * i \tag{6}$$

The net interest ni_t received by security holders in month t is this interest less the servicing fee, or

$$ni_t = mb_{t-1} * (i - s) \tag{7}$$

The scheduled principal payment, sp_t, is the monthly payment less the mortgage interest, or

$$sp_t = mp_t - I_t \tag{8}$$

The amount of principal prepaid is determined by the monthly survivorship factor, call it x_t. Then, the prepayment in month t of principal, pr_t, is calculated as follows:

$$pr_t = (1 - x_t)(mb_{t-1} - sp_t) \tag{9}$$

The net cash flow to investors in this pass through security in month t, cf_t, is the sum of net interest, scheduled principal payments, and principal prepayments:

$$cf_t = ni_t + sp_t + pr_t \tag{10}$$

Finally, the mortgage balance is computed recursively by the equation

$$mb_t = mb_{t-1} - (sp_t + pr_t) \tag{11}$$

For each iteration of the simulation, we determine the sequence of survivorship factors $\{x_t\}$ from the prepayment table, and then compute the cash-flow by month for the security under analysis. These cash-flows are then used in equation (1).

3 The Connection Machine CM–2 System

3.1 Hardware Overview

The Connection Machine, see Hillis [1985], is a fine grain massively parallel supercomputer. It uses a single instruction stream, multiple data stream (SIMD) methodology: each processor executes the same instruction broadcast by a microcontroller on its unique piece of data, or optionally sits out of the computation. The CM–2 has from 4096 to 65536 bit-serial processing elements, each with local memory of either 8 or 32 Kbytes. Each processor also has access to 32-bit or 64-bit floating point acceleration hardware.

The interconnection scheme for processors in the CM–2 is an N-dimensional hypercube. The hardware supports two distinct types of communication patterns: local grid-based patterns, and general patterns. Local grid-based patterns are supported through direct use of the hypercube wires, while general patterns are supported through the *router*, which implements arbitrary point to point communication.

The CM–2 is controlled by a serial front-end computer. Programs are compiled, debugged, and executed on the front-end computer, passing CM–2 instructions to the microcontroller as appropriate. Data can also be passed either way along this path.

3.2 Programming Model Overview

The programming model for the Connection Machine is called data parallel computing, which means that the same computations are performed on many data elements simultaneously. Each data element is associated with a processor of its own. Applications are not restricted, however, to data sets matching the physical size of the machine. The Connection Machine system software supports the abstraction of an arbitrary number of *virtual processors* (VPs), allowing users to easily handle data sets with potentially millions of elements. VPs are implemented by partitioning the memory and time-sharing the cycles of each physical processor. A collection of virtual processors used to handle a data set is called a *VP set*, and the number of

virtual processors that each physical processor must emulate is called the *VP ratio*. Note that because of pipelining and other optimizations, the expected linear slowdown from implementing VPs is actually a worst case scenario.

Applications that have no dependencies between data elements are often called *embarrassingly parallel*, and are easy to implement efficiently on any parallel computer. Most applications, however, exhibit dependencies between data elements. This gives rise to the need for communications between the processing elements. Furthermore, there is often a natural physical shape to arrange the data, such that dependent elements are laid out close to one another. Weather simulations, for instance, are naturally laid out in a three dimensional grid representing the volume of atmosphere being simulated.

The Connection Machine software allows users to specify the shape of a data set by associating a *geometry* with the VP set of the data. Geometries can be any N-dimensional Cartesian grid. Because an N-dimensional hypercube can be projected onto any lower dimensional Cartesian grid, the software can lay out the grid on the machine so that there is either a hypercube wire between each neighboring grid point, or they are held in the same physical processor (i.e., reside on VPs that are emulated by the same physical processor), and therefore local communication is fast and efficient. This kind of communication is called NEWS communication, and processors in the grid can be uniquely identified by the N-tuple of Cartesian coordinates called its NEWS address. General router-based communications is performed with a single unique identifier for each VP called its send address.

An important class of Connection Machine instructions that combine computations and communications are the parallel prefix operations, (Blelloch [1988]). We discuss the *scan* and the *spread* primitives here, although other variants exist. These primitives apply associative binary arithmetic operators to a variable in a binary combining tree along one axis of the geometry that holds the variable. For example, a plus-scan along the first axis of a 1-dimensional VP set for the variable x[0],x[1],...,x[n] produces the result y[0],y[1],...,y[n], where y[i]=x[0]+x[1]+...+x[i] (i.e., the cumulative sum to that point in the axis). User options allow the result to omit a processor's own value for x (e.g., y[i]=x[0]+x[1]+...+x[i-1]) and/or for the operation to proceed in the reverse order (e.g., y[i]=x[i]+x[i+1]+...+x[n]). A spread is a scan without directionality. The result of an add-spread is defined by y[i]=x[0]+x[1]+...+x[n].

The primitives have natural extensions to multi-dimensional geometries.

For example, a plus-scan operation on the set of variables x[0][0], ... x[m][n] along the first axis of a 2-dimensional geometry, will produce the result y[i][j] = x[i][0]+x[i][1]+ ... + x[i][j], for all i=1,2,3,...,m, and j=1,2,3,...,n. Finally, note that with a binary combining tree implementation, the execution time of such operations is proportional to the logarithm of the number of VPs, and thus scales nicely to large data sets.

The programming languages supported for the Connection Machine are C, Fortran, and Lisp. The Parallel Instruction Set (PARIS) subroutine library contains all of the instructions that manipulate processors or data on the CM, and can be called directly from serial programs running on the front end for explicit low level control of the machine. In addition, higher level versions of each of the languages have been built on top of PARIS: C*, CM Fortran, and *Lisp.

4 Massively Parallel Computing for Option Adjusted Analysis

The simulation procedure parallelizes nicely if each processor carries out all computations for a single interest rate path. Multiple processors can then execute in parallel multiple simulations. Communication across processors is only required in computing statistics across all simulations. This form of parallelism has been proved very successful on shared memory and distributed memory architectures with a limited number of processors, see Zenios [1989].

On the CM-2, however, we want to exploit the massive parallelism in performing the calculations for each path, in addition to simulating multiple paths simultaneously. Otherwise, a large number of processing elements will remain unused and the performance of the program will fall far short of the typical performance of the hardware. The path dependencies that are inherent in the OAA methodology appear, on first examination, to preclude the exploitation of parallelism within each path. However, we have found formulations of the method that do allow this form of parallelism. The following sections illustrate the efficient parallel implementation of the primary components of the OAA model.

The key to our implementation is the configuration of the CM-2 processing elements into a 2-dimensional NEWS grid of size 1024 x 512. Each one of the 1024 rows of virtual processors of the 0-axis carry out the calculations for a single path. The first 360 virtual processors in each row execute the

path dependent calculations (the remaining 152 are idle). In the sequel we will index virtual processors in the 0-axis by $s = 1, 2, 3, \ldots, 1024$ indicating simulation paths, and virtual processors in the 1-axis are indexed by $t = 1, 2, 3, \ldots, 360$ indicating time. Calculations along the 1-axis (time) are explained in detail, and it is to be understood the identical calculations are carried out simultaneously by all the VPs along the 0-axis (simulation).

4.1 Present Value Calculation

The present value calculation that appears in equation (1) can be written in the form:

$$PV = \sum_{t=1}^{360} cf_t^s \prod_{\tau=1}^{t} \frac{1}{(1 + r_\tau^s + oas)} \tag{12}$$

A *scan-multiply* operation on the ratio $\frac{1}{(1+r_\tau^t+oas)}$ produces at virtual processor with NEWS address (s,t) the value $\rho_t^s = \prod_{\tau=1}^{t} \frac{1}{(1+r_\tau^t+oas)}$. Each virtual processor proceeds to multiply the local values of cf_s^t with ρ_t^s and a *scan-add* on the product completes the calculation.

4.2 Monte Carlo Simulation

The generation of mean-reverting lognormally distributed series is not an associative operation, due to the mean-reversion term (3). Hence, an implementation that uses the scan primitives efficiently is not possible. It is possible, for example, to generate the lognormally distributed series first using the scans, and then proceed to apply the mean-reverting equation. However, this approach implies that whenever $R(f_t) \neq 0$ (see equation (3)) the generated series for periods t to 360 has to be discarded, the mean-reversion term $R(f_t)$ added, and a lognormally distributed series generated anew using the scan primitives. This approach may lead to as many as 360 repeated uses of the scan primitives and was observed, in a preliminary implementation, to be inefficient.

The Monte Carlo simulation phase was implemented by marching forward along the time-axis in steps of 1 and generating in parallel the states of 1024 interest rate paths at each point in time. Of course this implementation requires the use of only 1024 virtual processors. Hence, instead of operating on the NEWS grid of dimension 1024 $\times$ 512 we implement this primitive on a NEWS grid of dimension 1024 $\times$ 4 at a VP ratio of 1. Now

it is possible to associate each column of VPs of the 1024 × 4 grid with a column of the 1024 × 512 grid. Doing so we facilitate the transfer of data from the 1024 × 4 grid where they are generated to the 1024 × 512 grid where they are used for subsequent analysis without the need to use router communications.

As will be shown in the section on computational results the generation of mean-reverting interest rate paths is the most expensive part of the simulation. Futhermore, its performance does not scale with the use of larger number of processors since it is already executed at a VP ratio of 1.

4.3 Sampling from a Binomial Lattice

In order to generate sample paths from the binomial lattice we need to determine the state of each path at each point in time. Once the state ω of the s-th path in the binomial lattice is specified at virtual processor with NEWS address (s,t) the short term rate can be computed by a simple application of equation (4). The problem in constructing a continuous path is that the state of the lattice at instance t must be attainable by either an "up" or "down" step from the state at instance $t-1$. Such a sequence of states is produced on the CM-2 as follows: A random bit $m_t \in \{0,1\}$ is first generated at each VP. A *scan-add* operation along the time axis on these bits generates an index (ω_t), indicating the state of the VP (i.e., its distance from the ground state r_{t0}). Clearly, the distance from the ground state at instance t differs by at most one unit from the distance at instance $t-1$. Once the distance ω_t is determined equation (4) can be evaluated simultaneously by all VPs. Figure 3 illustrates the sampling of two paths from a binomial lattice using this procedure.

An alternative way to the sampling of binomial lattices has been developed by Shtilman and Zenios [1990]. Their method eliminates the need to construct random path samples from the lattice. Instead, it generates a prespecified set of paths. These paths are then used to provide estimates for the path dependent discount functions that are within a user specified error from the exact value, with a given probability. These paths are constructed by sampling <u>all</u> possible paths from the first $\mathcal{T}$ time instances of the lattice and completing each path from $\mathcal{T}+1$ to T by any arbitrary sequence of change of state.

This sampling scheme is implemented as follows. The CM-2 is configured as a two-dimensional NEWS grid of dimensions $2^{\mathcal{T}} \times 512$. The binary representation of row index s of this grid specifies the change of states (i.e.,

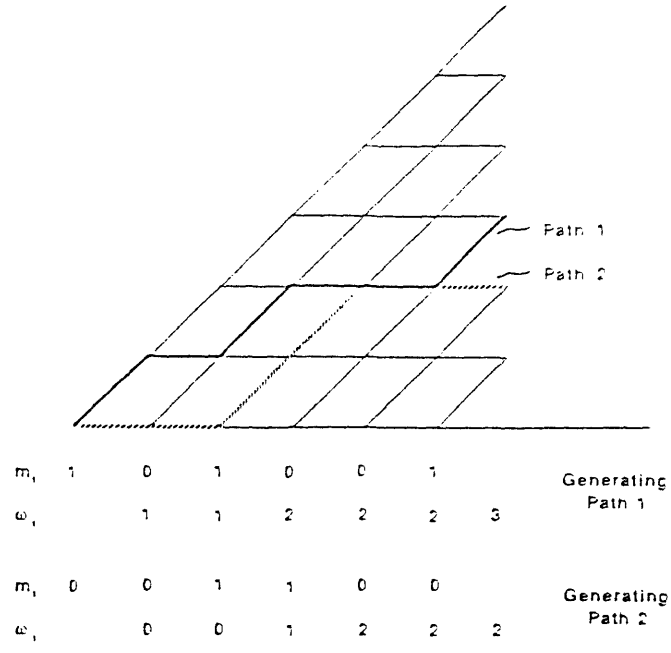

Figure 3: Sampling paths from a binomial lattice on the CM-2. $m_t = \{0,1\}$ indicates a transition to the "down" or "up" states at instance t. ω_t indicates the distance from the ground state r_{t0}

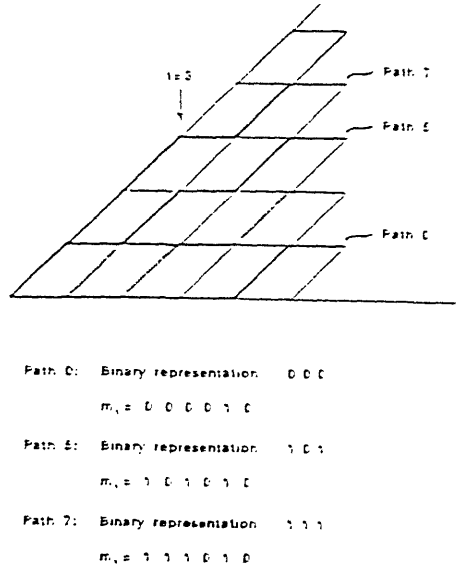

Figure 4: Sampling all paths from the first three time periods of a binomial lattice on the CM-2. $m_t = \{0,1\}$ indicates a transition to the "down" or "up" states at instance t. ω_t indicates the distance from the ground state r_{t0}

n_t) for the $s-th$ path from instance 0 to $\mathcal{T}$. The change of states from instance $\mathcal{T}+1$ to 360 is completed by the (arbitrary) sequence $\{01010101...\}$. Once the binary sequence is completed we use once more a *scan-add* operation to evaluate the state ω_t of each VP and equation (4) is used to evaluate the short term rates. Figure 4 illustrates the sampling of all paths from the first three time periods of a binomial lattice.

4.4 Lookup of Prepayment Tables

The estimation of prepayment characteristics of a pool of mortgages is a central issue in the use of OAA methods. Several models have been proposed in the literature, and are used in practice with varying levels of success. See, for example, Richard and Roll [1989] for a discussion of the key prepayment factors for mortgage-backed securities and a related model. The massively parallel implementation of prepayment models is the topic of current research and will be reported elsewhere, Hutchinson and Zenios [1990].

We consider here the case where the prepayment characteristics of a mortgage-backed security have been generated by a separate model and are stored in a table of *monthly survivorship factors*. This is not an unreasonable modeling practice: prepayment characteristics of mortgage securities are re-evaluated rather infrequently, e.g., on a monthly basis, whereas the OAA model is usually run on a daily basis, or, potentially, in real time.

Our problem is then one of looking up the correct monthly survivorship factor for the given level of interest rates and age of the mortgage. The monthly survivorship data is given in a two-dimensional table. The 1-axis indicates the age of the mortgage (1 – 360 months) and the 0-axis indicates the interest rate in a given range (e.g., 6 – 16%) in small increments (e.g., 0.01). This table is stored in a two-dimensional NEWS grid. The dimension of the 1-axis is 512, with only the first 360 columns being active. The dimension of the 0-axis depends on the range and increment of interest rates. In the example given above the 0-axis has dimension 1024 with only the first 1000 rows being active. It is only by coincidence that the dimension of the 0-axis of the NEWS grid used for the simulation is identical to the size of the grid required by the survivorship table. Hence, these two grids are associated with different VP sets. We shall refer to these sets as the *simulation-* and *data-vp-sets* respectively.

Looking up the survivorship factors given a path of interest rates is now a simple data transfer from the data- to the simulation-vp-set. Virtual processor with coordinates (s,t) in the simulation-vp-set will find the row

index i of the interval in the data-vp-set that corresponds to interest rate r_{st}. Let R_0 be the lowest interest rate in the survivorship table, and δ be the increment between successive rows. Then the row index is computed by

$$i = \lfloor \frac{r_{ts} - R_0}{\delta} \rfloor \tag{13}$$

The $(s,t)-th$ VP from the simulation-vp-set will get from the $(i,t)-th$ VP of the data-vp-set the survivorship factor x_t using router communications. This factor is then used in the cash-flow calculations of the following section.

4.5 Cash-flow Calculations for Mortgage Backed Securities

The parallel evaluation of the cash-flow equations of Section 2.3 presents some difficulties due to the recursive equation (11). It is possible to develop an alternative formulation of the cash-flow equations that lends itself to the application of the scan parallel primitives.

Let

$$c_t = \frac{i(1+i)^{n-t+1}}{(1+i)^{n-t+1} - 1} \tag{14}$$

and

$$\rho_t = (1 - x_t)(1 - c_t + i) \tag{15}$$

After some algebra using equations (5) — (10) we may rewrite the recursive mortgage balance equation (11) as

$$mb_t = mb_{t-1} * \rho_t \tag{16}$$

This in turn can be rewritten as

$$mb_t = mb_0 \prod_{\tau=1}^{t} \rho_\tau \tag{17}$$

Equation (17) can now be evaluated using a *scan-multiply* operation that will produce at the $t-th$ VP in each row of the NEWS grid the value $\prod_{\tau=1}^{t} \rho_\tau$, and a multiplication of the result by the constant mb_0 will produce the correct mortgage balance exclusive of prepayments. Scheduled payment, net interest, prepayment and total cash flow at each point in time can then be evaluated simultaneously for all $t = 1,2,3,\ldots,T$ by the sequence of equations

Function	CM time (seconds)	Elapsed time (seconds)	MFLOPS
Present value	0.15	0.15	93
Monte Carlo simulation	0.50	0.53	58
Random sampling of binomial lattice	0.24	0.25	59
Prepayment table lookup	0.27	0.28	NA
Cash-flow calculations	0.23	0.23	117

Table 1: Execution times for 1024 simulations on the Connection Machine CM-2a (seconds).

$$sp_t = mb_{t-1}(c_t - i) \quad (18)$$
$$ni_t = mb_{t-1}(i - s) \quad (19)$$
$$pr_t = (1 - x_t)(mb_{t-1} - sp_t) \quad (20)$$
$$cf_t = sp_t + ni_t + pr_t \quad (21)$$

5 Computational Results

The procedures of the previous section were implemented in C/Paris for the Connection Machine as part of a library of financial routines, and this library was used to build an OAA model. The library was built on top of Paris release 5.2. The results we report here are obtained using single precision (i.e., 23 bits in the mantissa) arithmetic, although double precision (i.e., 52 bits in the mantissa) is also available. A partial listing of the entries of the library is given in the Appendix.

Individual modules of the library were first tested both for efficiency in terms of solution times, and in terms of the sustained computing rate. All experiments were carried out on a 4K processor model CM-2a with a Sun-4/280 front end, 32-bit floating point accelerators and 8 Kbytes of memory per processor. In our implementation we use a 1024×512 NEWS grid for a total of 2^{19} virtual processors. Hence, the implementation is executed on the 4K CM-2a at a VP ratio of 128. Some improvements in performance will be realized if the programs were run on a bigger machine. Table 1 summarizes the execution time for the key components of the library in executing a total of 1024 simulations, together with the sustained MFLOPS rate.

Mode	4K CM-2	8K CM-2	16K CM-2	CRAY X-MP
Monte Carlo Simulation	1.77	1.07	0.78	17.0
Binomial Lattice Sampling	1.49	0.71	0.37	6.0

Table 2: Performance of the option adjusted analysis model on the CM-2 and CRAY X-MP (seconds).

A complete option adjusted analysis was carried out for a single mortgage backed security (GNMA) and was compared to an identical model implemented in Fortran 77 on a CRAY X-MP/48 vector supercomputer, Zenios [1989]. The program typically takes 4-6 iterations of the nonlinear equation solver to compute the *oas* to an accuracy of 0.1 basis points. The solution time for both the CM-2 implementation and the CRAY X-MP code are reported in Table 2. The CRAY code was compiled using the f77 vectorizing compiler. It is possible that some gains in performance can be realized with the CRAY code, with suitable modifications of the implementation to take advantage of the vector architecture beyond what is already achieved by the compiler.

6 Conclusions

Compute intensive financial simulations are well suited for execution on massively parallel architectures. While the execution of multiple simulations is easily parallelizable, the efficient execution of path dependent calculations is possible only with suitable modifications of the algorithms. It is then possible to implement the path dependent calculations by combining efficiently communications with computations via the parallel prefix primitives.

The performance of the massively parallel option adjusted analysis model on one of the smaller Connection Machine systems compares favorably with a CRAY X-MP vector supercomputer. As a result of very short response times it is possible to use such models in real time and to carry out a wide range of sensitivity analyses. We have implemented an interactive user interface that allows users to compute option adjusted spreads, price, duration and convexity, projected price paths or average cash-flow statistics. The response time, including intializations, calculations and graph generations are only a few seconds (2-10) for most functions. Even the generation of a complete path of projected prices is generated well within half minute of wall clock time.

A The Connection Machine Financial Library

We provide in the following table a list of the financial modeling primitives on the Connection Machine. Reported times are for the execution of calculations on a 1024 × 360 NEWS grid, on a 4K CM-2a with 32-bit floating point co-processors and a Sun4-280 front-end.

No.	Function	Real Time (seconds)	CM Time (seconds)	MFLOPS
1	prval	0.15	0.15	93
2	psa-prepayment	0.06	0.06	31
3	table-lookup	0.28	0.27	NA
4	rand-normal	0.25	0.25	94
5	rand-lognormal	0.42	0.40	73
6	rand-meanrev	0.53	0.50	58
7	sample-binomial-lattice	0.25	0.24	59
8	mortgage-cashflow	0.23	0.23	117
9	vector-mean	.00054	.00041	9

Acknowledgements: The work of S.A. Zenios was funded in part by NSF grant CCR-8811135 and AFOSR grant 89-0145. This research was completed while this author was visiting the Operations Research Center at the Sloan School of MIT, and Thinking Machines Corporation, Cambridge, MA. The support and useful discussions with J. Mesirov are greatfully acknowledged.

References

[1] M.R. Asay, P.J. Bouyoucos, and A.M. Marciano. *An Economic Approach to Valuation of Single Premium Deferred Annuities.* Working paper, Goldman, Sachs and Co., April 1989.

[2] W.W. Bartlett. *Mortgage-Backed Securities: products, analysis and trading.* New York Institute of Finance, 1988.

[3] F. Black, E. Derman, and W. Toy. *A One-factor Model of Interest Rates and its Application to Treasury Bond Options.* Discussion paper No. 1, Goldman, Sachs and Co., June 1988.

[4] G. E. Blelloch. *Scan Primitives and Parallel Vector Models.* PhD thesis, Artificial Intelligence Laboratory, Massachusetts Insitute of Technology, Cambridge, MA, November 1988.

[5] R. Bookstaber, D.P. Jacob, and J.A. Langsam. *Pitfalls in Debt Option Models.* Working paper, Morgan Stanley, Fixed Income Analytical Research, Feb. 1986.

[6] A.J. Brazil. Citicorp's mortgage valuation model: option-adjusted spreads and option-based duration. *Journal of Real Estate Finance and Economics*, 1:151–162, 1988.

[7] J. Cox, J. Ingersoll, and S. Ross. A theory of the term structure of interest rates. *Econometrica*, 53:385–407, 1985.

[8] C. Moler D. Kahaner and S. Nash. *Numerical Methods and Software.* Prentice Hall, New Jersey, 1989.

[9] F.J. Fabozzi, editor. *Mortgage Backed Securities: New Strategies, Applications and Research.* Probus Publishing Company, 1987.

[10] L. Hayer and K. Lauterbach. *Stochastic Valuation of Debt Securities.* Working paper, Financial Strategies Group, Prudential-Bache Capital Funding, 1988.

[11] W. D. Hillis. *The Connection Machine.* The MIT Press, Cambridge, Massachusetts, 1985.

[12] T. S.Y. Ho and S-B. Lee. Term structure movements and pricing interest rate contingent claims. *The Journal of Finance*, XLI(5):1011–1029, 1986.

[13] J.M. Hutchinson and S.A. Zenios. *Massively Parallel Computing for Mortgage Prepayment Models.* Working Paper, Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia, 1990.

[14] D.P. Jacob, G. Lord, and J.A. Tilley. A generalized framework for pricing contingent cash flows. *Financial Management*, August 1987.

[15] S.M. Pinkus, S.M. Hunter, and R. Roll. *An introduction to the mortgage market and mortgage analysis.* Technical Report, Goldman Sachs Mortgage Securities, 1987.

[16] S.F. Richard and R. Roll. Prepayments on fixed-rate mortgage-backed securities. *The Journal of Portfolio Management*, 73–82, Spring 1989.

[17] W.F. Sharpe. *Investments.* Prentice Hall, 1985.

[18] M. S. Shtilman and S.A. Zenios. *Constructing Optimal Samples from a Binomial Lattice.* Working paper, Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia, April 1990.

[19] S. A. Zenios. *Parallel Monte Carlo Simulation of Mortgage Backed Securities.* Report 89–10–06, Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia, October 1989.