Buffered Banks in Multiprocessor Systems

Kay A. Robbins Steven Robbins

The University of Texas at San Antonio

Abstract A memory design based on logical banks is analyzed for shared memory multiprocessor systems. In this design, each physical bank is replaced by a logical bank consisting of a fast register and subbanks of slower memory. The subbanks are buffered by input and output queues which substantially reduce the effective cycle time when the reference rate is below saturation. The principal contribution of this work is the development of a simple analytical model which leads to scaling relationships among the efficiency, the bank cycle time, the number of processors, the size of the buffers, and the granularity of the banks. These scaling relationships imply that if the interconnection network has sufficient bandwidth to support efficient access using high-speed memory, then lower-speed memory can be substituted with little additional interconnection cost. The scaling relationships are shown to hold for a full datapath vector simulation based on the Cray Y-MP architecture. The model is used to develop design criteria for a system which supports 192 independent reference streams, and the performance of this system is evaluated by simulation over a range of loading conditions.

keywords Buffered memories, logical memory banks, memory conflicts, vector processors, Cray Y-MP.

I. Introduction

The gap between memory speed and processor request rate is increasing rapidly in high performance systems. This gap is due to a decrease in processor cycle time, the use of superscalar and other multiple issue mechanisms, the increase in the number of processors in shared memory systems, and the demands of gigabit per second network communication. In addition, designers have sought to replace expensive SRAM memories with cheaper, slower DRAMS in order support dramatically increased main memory sizes at a reasonable cost.

In the face of these demands, several manufacturers have introduced more complex circuitry on their DRAM chips in order to reduce the effective memory access time. Mitsubishi, for example, has introduced a proprietary cached DRAM. This chip has a small SRAM which reduces the memory access time if the reference is contained in the SRAM [9]. Another cached DRAM has been developed by Rambus [6]. Other approaches include synchronous DRAM technology [14] and enhanced DRAM technology [4]. Pipelined DRAM memories have also been proposed [11]. The effect of such memory hierarchies in high-performance memory systems has not been extensively studied.

In an ordinary interleaved memory, the memory cycle time is the minimum time required between successive references to a memory module. The cycle time regulates how quickly a processor can fill the memory pipeline. Conflicts due to bad reference patterns can cause the processor to block. The latency is the time it takes for a read request to navigate the memory pipeline and return a value to the processor.

In hierarchical memory systems, such as those which contain caches at the bank level, the memory cycle time and the latency are no longer constant. Caching can reduce both the effective cycle time and the latency. This paper explores buffering as an alternative or as a supplement to caching at the chip level. The proposed design is based on a buffering scheme called *logical bank buffering* in which physical banks are subdivided and buffered as described in Section II. The principal contribution of this paper is the development of a simple model and the derivation of scaling relationships among the efficiency, the bank cycle time, the number of processors, the size of the buffers, and the granularity of the banks. The goal of the logical bank design is to provide a mechanism for using large, slower memories with a moderate number of high performance processors while maintaining current operating efficiency.

A second contribution of this work is the full data-path simulation with register feedback for a realistic interconnection network. High performance machines, such as the Cray Y-MP, have a separate interconnection network for read return values. When simple memory banks are replaced by a memory hierarchy, references arrive at the return network at unpredictable times. Under moderate loading, the resulting contention does not appear to be a problem. Several approaches for equalizing performance of reads and writes under heavy loading are examined.

Buffering has been proposed by a number of authors as a possible solution to the problem of memory conflicts. A simulation study by Briggs [2] showed that buffering at the processor level in pipelined multiprocessors can improve memory bandwidth provided that the average request rate does not exceed the memory service time. Smith and Taylor [20] explored the effects of interconnection network buffering in a realistic simulation model. The simulations in this paper were based on a similar, but simpler interconnection network. The buffering in this paper is at the memory modules rather than within the interconnection network, and the emphasis of the simulations is on the verification of the scaling relationships.

Other proposals to reduce memory conflicts have been made. Skewing and related techniques [7,8,12,13] have been shown to be effective in reducing intraprocessor conflicts. A recent simulation study by Sohi [21] explores skewing and input and output buffering for single reference streams consisting of vectors of length 1024 with fixed strides. Skewing techniques are not as effective for the situation considered in this paper where conflicts between processors are the main cause of performance degradation. Skewing can be used in conjunction with logical banks to reduce intraprocessor contention.

This study uses memory efficiency and throughput as its primary measures of memory performance. The *efficiency* of a memory system is defined as the ratio:

$$E = \frac{\mathit{Successful\ memory\ requests}}{\mathit{Total\ memory\ requests}}$$

The total number of memory requests includes those requests which are denied because of a conflict such as a bank conflict. It is assumed that when such a conflict occurs, the processor attempts the reference on the next cycle. This memory efficiency is measured from the viewpoint of the processor. It indicates the degree to which processors will be able to successfully issue memory references. The efficiency is essentially P_A, the probability of acceptance, as calculated by Briggs and Davidson [3] in their models of L-M memories without buffering.

Following Sohi [21], the *throughput* is defined as the ratio:

$$T = \frac{\mathit{Time\ for\ vector\ references\ in\ a\ conflict\text{-}free\ system}}{\mathit{Actual\ time\ for\ vector\ references}}$$

Sohi argues that this ratio is the appropriate throughput measure when comparing memory designs in a vector processing environment. The throughput is the fraction of the optimal rate at which entire vectors are delivered through the system. The vector element read latency is defined as the time between the first attempt to access a vector element and the availability of that element at the vector register.

The efficiency depends on the number of processors, the number of banks, the bank cycle time, and the load. Unbuffered designs for multiprocessor machines give a quadratic relationship between memory speed and number of banks for fixed performance [1]. If the memory cycle time is doubled relative to the processor speed, the interconnection costs must be quadrupled to maintain the same memory performance. The proposed design is a two-tier system. The results show that if the interconnection network bandwidth is sufficient to support the processors using high-speed memory, then lower-speed memory with buffers can be substituted for little additional interconnection cost or performance degradation.

Section II describes the logical bank design. An analytical model for writes is developed in Section III and is shown to be in reasonable agreement with random reference simulations in Section IV. The relationship between the efficiency and system parameters such as the bank cycle time, number of banks and number of processors is then analyzed. Several design criteria are developed which are applied in later sections to vector systems.

Section V introduces a simulation model which uses synthetically generated references for a vector multiprocessor system similar to the Cray Y-MP. The model incorporates a full data-path simulation including return conflicts and register feedback as recommended in the simulation study by Smith and Taylor [20]. The vector results are compared with model predictions under moderate processor loading in Section VI, and it is shown that even a small number of buffer slots can result in significant gains in performance. In Section VII the design criteria are applied to a 64-processor system (192 independent reference streams) under heavy processor loading for a range of stride distributions. Performance for writes is excellent, but there is degradation for reads. Several approaches for reducing this degradation are examined including subbank output buffering, additional lines, optimal arbitration, port handshaking, port-line buffers, and increased return bandwidth. It is found that only the last alternative completely eliminates the degradation. A final discussion and conclusions are presented in Section VIII.

II. Logical Banks

A logical bank [16] consists of a fast register, the *logical bank register* (LBR), and a number of subbanks each with a queue of pending requests as shown in Figure 1. The memory within the logical bank is divided into equal subbanks and addressed using the standard interleaving techniques so that consecutive addresses go to consecutive subbanks. A reference to the logical bank can be gated into the LBR in T_{la} cycles if the register is free. T_{la} is the *logical bank access time.* If there is room in the queue for the specified subbank, the reference is then routed to the queue. Otherwise, the LBR remains busy until a slot is available. Only one reference to a logical bank can occur during an interval T_l. T_l is called the *logical bank cycle time.* It is the minimum time interval between successful references to a logical bank. The interval may be longer if the LBR is waiting for a queue slot. If reference streams attempt to access the same logical bank while the LBR is busy, a *logical bank conflict* occurs and all but one reference is delayed. In the model and all of the simulations discussed later, it is assumed that $T_l = T_{la}$.

The parameters which define a multiprocessor system with shared memory organized into logical banks are shown in Table 1. The default values are used in later simulations except where otherwise indicated. For single-port processors the number of reference streams, n, is also equal to the number of processors. In the vector simulations, the processors are allowed to have multiple ports so there can be more reference streams than processors. Reference streams are assumed to be either read or write. Read streams are more difficult to handle because values must be returned to the processor. The return fan-in network for reads requires additional hardware for arbitration because read values do not arrive at the fan-in network at a predictable time. The return values must include tag bits indicating the destination. This hardware is also required if cached DRAMs are used since the effective access time is no longer constant in that case either. In fact, cached DRAMS can be used in conjunction with logical banks to reduce the effective physical subbank cycle time, T_c, with little additional hardware.

Logical banks were introduced by Seznec and Jegou to support the Data Synchronized Pipeline Architecture (DSPA) [19]. Their design includes a reordering unit so that data flows out of the logical bank in chronological order. In the scheme proposed in this paper, a reordering unit is not required at the logical bank, because reordering occurs at the processor.

Buffering is distinct from caching in that there is no miss penalty and no overhead for cache management. The proposed circuitry would take up a very small chip area if it were incorporated on a chip. Alternatively, it could be built as an interface between off-the-shelf memory chips and the system interconnection network. It is particularly appropriate in situations in which average utilization is below maximum capacity, but where there are periods of maximal loading. In addition to reducing average access time, logical banks can smooth the type of bursty memory traffic which is typical of highly vectorized programs [17].

III. A Model for Random Writes

A model for the efficiency of logical bank memories is now derived. In later sections the throughput and latency are related to the efficiency. It is assumed that a reference stream can initiate at most one reference per clock cycle and that when a reference attempt fails, it is retried by that reference stream on the following cycle. As long as the LBR is available, the processor sees a memory consisting of logical banks with a cycle time of T_l, the logical bank cycle time. The efficiency in this case is given by E_l. This efficiency is determined by the interreference stream conflicts at the logical bank level. When the queues are full, the memory behaves almost as though there were no logical banks. The effective memory cycle time in this case is $T_p = T_l + T_d + T_c$ where T_d is the minimum delay incurred in transferring a reference from the queue to the subbank and T_c is the physical memory cycle time. The efficiency in this case is denoted by E_p.

A simple probabilistic argument shows that if the probability of a successful reference is E, the expected number of attempts per successful reference is:

$$\sum_{k=1}^{\infty} k(1-E)^{k-1}E = \frac{1}{E}$$

$\frac{1}{E}$ is the average number of cycles that it takes a reference stream to initiate a reference from the viewpoint of the processor. In contrast, the average reference time from the viewpoint of the physical memory is directly related to the bank cycle time and other delays.

Let P be the probability that the logical bank register (LBR) is available when a reference is first initiated. A successful reference will take $\frac{1}{E_l}$ attempts with conditional probability P and $\frac{1}{E_p}$ attempts with probability $1 - P$. The effective efficiency is then a weighted average of the two cases depending on the probability that there are slots available in the appropriate subbank queue. The average number of cycles for a successful reference can then be estimated by:

$$\frac{1}{E} = \frac{P}{E_l} + \frac{(1-P)}{E_p}$$

where E is the combined or effective efficiency. This relationship can be written as:

$$E = \frac{E_l E_p}{PE_p + (1-P)E_l}$$

This expression for the effective efficiency will be called the *logical bank model* in the remainder of the paper.

The probability, P, that the LBR is not full can be estimated by considering each logical bank as a system of k independent queues under the M/D/1/B queuing discipline. This queuing model has an exponential arrival rate, deterministic service time, one server, and a finite queue. For fixed queue size, the distribution of the number of references in the queue depends on the parameter $\rho = \lambda T_p$ where λ is the average arrival rate and T_p is the effective queue service time. λ can be estimated as $\lambda \sim \frac{nq}{b}$ where q is the probability that a free stream initiates a reference, n is the number of independent reference streams, and b is the number of physical subbanks (kl). A simple simulation is used to compute a table of probabilities for a given value of ρ and queue size m.

Once the probabilities that the individual queues are free have been determined, the value of P for the entire logical bank can be estimated as follows. If there are k subbanks per logical bank, the LBR will be busy if any of the k queues has $m+1$ slots filled, the extra slot being from the LBR itself. Thus, if f is the probability that a queue of size $m + 1$ is not full, then the probability that the LBR is free is f^k. This method is used to calculate P for the graphs given later.

Estimates for E_p and E_l will now be derived. The situation in which there are no logical banks has been analyzed for random accesses by Bailey [1] for systems with n single-port processors and b memory banks. Let T be the memory bank cycle time. Each processor can be modeled as a Markov chain on $T+1$ states s_0, s_1, ..., s_T, where s_i represents the state in which the processor is waiting for a bank which will be busy for i more cycles and s_0 denotes the state in which a processor accesses a free bank. Bailey derived a steady state expression for the efficiency in such as system as:

$$E_B(q,T,n,b) = \frac{2q}{2q-1+\sqrt{1+2q^2nT(T+1)/b}}$$

Here q represents the probability that a free processor will attempt a reference on the current clock cycle. For relevant values of T, n, and b, the efficiency is dominated by the expression in the square root:

$$E_B \sim \frac{1}{T}\sqrt{\frac{2b}{n}}$$

The efficiency is inversely proportional to T and drops off fairly rapidly. If the bank cycle time is doubled, the number of banks must be increased by a factor of four to maintain the same efficiency. The Bailey model can be used to estimate E_p by using $T_p = T_c + T_d + T_l$ for the bank cycle time:

$$E_p = E_B(q, T_p, n, b)$$

The logical bank efficiency, E_l, can also be estimated using the Bailey model as $E_l = E_B(q, T_l, n, l)$ where l is the number of logical banks in the system. T_l, the logical bank cycle time, is assumed to be one for much of the discussion in this paper. Due to the assumptions made in the derivation of the Bailey model, it performs poorly when there are a small number of processors or when the bank cycle time is near one. Unfortunately the effective efficiency is very sensitive to the value of E_l when P is close to one, so another model will be developed for E_l when $T_l = 1$. This model will be called the *direct model* and is derived below using Markov chains in a manner similar to that used by Bailey.

Assume that each reference stream is in one of three states: the free state (1), a state in which it is making a successful reference (2), or the state in which it is attempting a reference which is unsuccessful (3). Let:

q	=	probability that a given free stream will attempt a reference
α	=	probability that the stream is in the free state
β	=	probability that the stream is making a successful reference
γ	=	probability that the stream is making an unsuccessful reference
δ	=	probability that a reference attempt will be successful

The following probability conservation equation holds:

$$\alpha + \beta + \gamma = 1$$

The matrix, Γ, of state transition probabilities is given in Table 2. $\Gamma_{i,j}$, the entry in the i-th row and the j-th column, represents the conditional probability that the next state is i given that the current state is j.

A reference attempt will be successful if no higher priority stream is making a reference to the same bank. On the average half of the remaining streams will have a higher priority than a given stream. Since only one higher priority stream can make a successful reference to a bank, the probability that one of the $(n-1)/2$ higher priority streams is making a successful reference to one of l banks may be estimated as:

$$\frac{\beta(n-1)}{2l}$$

and so δ is given by:

$$\delta = 1 - \frac{\beta(n-1)}{2l} = 1 - \beta\epsilon$$

where:

$$\epsilon = \frac{n-1}{2l}$$

Let $\bar{p} = (\alpha, \beta, \gamma)$ be the vector of a priori probabilities of the three states. The steady state probabilities, which can be obtained by the relationship $\bar{p} = \Gamma\bar{p}$, are then given by:

$$\begin{aligned}
\alpha &= (1-q)(\alpha+\beta) \\
\beta &= q\delta(\alpha+\beta) + \delta\gamma \\
\gamma &= (1-\delta)(q\alpha + q\beta + \gamma)
\end{aligned}$$

These equations plus the conservation equation can be used to obtain an expression for δ:

$$E_l = \delta = \frac{1 - 2q - q\epsilon + \sqrt{(1-2q-q\epsilon)^2 + 4q(1-q)}}{2(1-q)}$$

The direct model is in better agreement with the simulations when $T_l = 1$, and it will be used to estimate E_l for the remainder of the paper.

IV. Predictions of the Model

Buffering can produce fairly dramatic improvements in efficiency provided that the memory system is not close to saturation. In this section, the logical bank model is compared with model simulations for random references. The excellent agreement of this comparison validates the model and suggests relationships between the design parameters which are necessary for achieving a particular level of efficiency. In the following sections a vector simulation model is compared with the random-reference model, and the relationships suggested by the analytical model are tested.

Consider a multiprocessor system which has 24 independent reference streams and a shared memory consisting of 256 banks. (These parameters represent an eight-processor Cray Y-MP with three ports per processor and a maximal memory configuration.) The performance of this system is now compared with that of an augmented system in which each physical bank is replaced by a logical bank consisting of a single subbank with a queue size of two. This case corresponds to adding buffering at the physical bank level without adding any additional logical bank structure other than the buffers and the LBR. The reference streams are assumed to generate writes only and $T_l = T_d = 1$.

In Figure 2 the efficiency for random reference streams is plotted versus subbank cycle time. Following Bailey [1] a reference rate of $q = .4$ is selected to give a base operating efficiency in the unbuffered case of .67. The logical bank model agrees well with results from scalar simulations for operating efficiencies above .6. A significant improvement in performance is observed with buffering. When the bank cycle time is 18, the simulation shows an efficiency of .22 without buffering and an efficiency of .66 with buffering. (The memory efficiency of a real Cray Y-MP is higher than the predicted .67, because selected buffering mechanisms are incorporated at various stages in the Cray Y-MP interconnection network as described by Smith and Taylor [20].)

In Figure 3 the efficiency is plotted versus the number of reference streams. Again there is excellent agreement between the model predictions and those of random reference simulations. The base efficiency of .67 can be maintained with as many as 96 reference streams when buffering is introduced at the bank level. The logical bank model overestimates the efficiency near saturation because the M/D/1/B queuing model assumes that references are thrown away when the queues are full. The processor attempts to initiate a reference on the next cycle with a certain probability $q < 1$. In the real system and in the simulation the reference is retained and tried again on the next cycle.

A simple analysis is now presented which shows that queue sizes which are quite small can give substantial improvements in performance. Table 3 shows the probability that the number of items in the queue is less than the queue size, m, for different values of m and different system loadings. If $\rho = .5$, the probability that a queue will have fewer than three slots (two queue slots plus the LBR) filled is .9731. This value is indicative that small queues will suffice. The estimate may not be completely accurate near saturation, because references which are not fulfilled are thrown away in the model. Hence, the infinite queue case is now considered.

In the infinite queue model, references are never blocked, but are always queued. The expected queue size for each subbank in the infinite queue case is [10]:

$$Ex(\text{Queue Size}) = \frac{\rho^2}{2(1-\rho)}$$

For $\rho < .5$, the expected queue size is less than .25 for each subbank. Table 4 shows that the probability that a queue contains no more than x items in the queue for the infinite queue case. The probability that two or fewer slots are filled is .947 for $\rho = .5$. This probability is on the borderline of reasonable performance. The probability of having four or fewer elements in the queue is .9957. A subbank input queue size of four should be adequate to handle most references ($P \approx 1$), and the effective efficiency will be E_l.

For fixed load q, E_l is constant for constant:

$$\epsilon = \frac{n-1}{2l} \approx \frac{n}{2l}$$

Furthermore, $E_l \to 1$ as $\epsilon \to 0$, and $E_l > .9$ when $\epsilon < .1$ since $0 \leq q \leq 1$. This result is obtained by noting that E_l is a decreasing function of both q and ϵ. The condition $\epsilon < .1$ means there should be at least five times as many logical banks as there are reference streams for efficient performance.

The buffered and unbuffered cases can be compared in the case where there is one subbank per logical bank so that $b = l$. Consider the effect of buffering on efficiency when $\rho = qT_p n/b$ is held constant at .5 as the subbank cycle time and the number of banks are both increased. In this paper it is assumed that $T_l = T_d = 1$, so $T_p \approx T_c$. ϵ will be decreasing since ϵ only depends on the number of processors and the number of banks. When the queue size is four, the probability of a logical bank hit is .9957 so the efficiency is approximately E_l. Since E_l is independent of T_c and is an increasing function of $l = b$, the logical bank model predicts that the efficiency will actually increase slowly as the bank cycle time and number of banks are increased with ρ held fixed at .5. Thus, for a fixed reference rate, q, a doubling of T_c, can be compensated for by doubling the number of banks or by halving the number of processors (reference streams). This is in contrast to a system without logical banks where $T_c\sqrt{\frac{n}{b}}$ must remain fixed to maintain the same efficiency. In systems without logical banks one would have to quadruple the number of banks in order to compensate for doubling the bank cycle time. When the above argument is applied to the same system with a queue size of two, the probability of a logical bank hit is at least .947. The efficiency is now a weighted average of the relatively constant logical bank efficiency, E_l, and the unbuffered efficiency, E_p. (The latter efficiency drops off rapidly with bank cycle time.)

To confirm these relationships in the models with and without logical banks, the load q is fixed at .4, and the number of reference streams is fixed at 24. In Figure 4 the efficiency is plotted versus the subbank cycle time when the number of banks is varied so that ρ is held constant at .5. The logical bank model maintains an almost constant efficiency as the bank cycle time is increased as predicted for queue size of four. The system with queue size two shows a slight fall-off. The efficiency is initially lower than the asymptotic value because when the bank cycle time is small and ρ is held at .5, there are so few banks that logical bank conflicts become significant. When the Bailey model is run for the same parameter values, the efficiency drops dramatically as predicted by the model. Similar scaling relationships can be derived when the bank cycle time is fixed and the number of banks and the number of reference streams are varied.

One can use the relationship between ρ and E to determine design parameters required to achieve a specified level of performance. The Cray Y-MP has eight processors and three ports per processor (24 independent reference streams), a bank cycle time of five, and 256 physical banks. If a maximum reference rate of $q = 1.0$ is assumed, then $\rho = .468$ and $E_l = \frac{1}{1+\epsilon}$. With a queue size of four, the probability of a logical bank hit is nearly one. The efficiency can be simply estimated from the previous expression for ϵ. When there are 256 logical banks (one subbank per logical bank), $\epsilon = .09$ and $E_l = .92$. The logical bank model predicts that buffering with four queue slots will result in a high efficiency. The unbuffered efficiency is predicted by the Bailey model for these parameters to be .56. These model predictions are tested in Section VII for a vector simulation.

The results of the logical bank model can be summarized as follows. For a fully loaded system ($q = 1.0$) consisting of n reference streams, l logical banks, a logical bank cycle time of one, and subbank input queue size of four, the efficiency is greater than .90 provided that:

$$\rho \approx \frac{T_p n}{b} < .5$$

and

$$\epsilon \approx \frac{n}{2l} < .1$$

For a queue size of two, ρ should be chosen to be less than .2.

Notice that the first relationship depends on the total number of subbanks, b, while the second relationship depends on the number of logical banks, l. The per processor interconnection costs depend on l. As long as there are enough logical banks to adequately field requests from the processors, an increase in bank cycle time can be compensated for by an increase in the number of subbanks without a significant increase in the interconnection costs. There is a point, however, at which the data bus arbitration scheme will not be able to handle read return traffic. This point is discussed more fully in Section VII.

An increase of T_l above one has the effect of lowering the overall efficiency, but the curves have the same shape. Design parameters can be determined in this case by using the Bailey model to estimate E_l when $q = 1$.

$$E_l = \frac{2}{1 + \sqrt{1 + 2nT_l(T_l + 1)/l}}$$

The efficiency now depends on the parameter $\eta = nT_l(T_l + 1)/l$. An efficiency of .90 can be obtained provided that $\eta < .25$ and $\rho < .5$.

V. Vector Simulation Model

In order to test the performance of the logical bank organization and the predictions of the logical bank model, a simulation study based on the Cray Y-MP architecture interconnection network was developed. The Cray Y-MP architecture was selected because its highly pipelined interconnection network can provide an effective $T_l = 1$. This is accomplished by having references issue immediately to the interconnection network and block later if conflicts should arise. A complete data path simulation of this system with processor register feedback was performed with reference streams which were generated randomly under realistic assumptions. The simulation includes ports, lines, sections, and subsections as described below.

The vector simulation model assumes there are n_p processors each with p ports. Each processor can initiate up to p memory operations on a cycle. These ports are assumed to generate independent reference streams ($n_p p = n$). Each port is designated either as a read stream or a write stream.

The interconnection model is a simplified version of the network described by Smith and Taylor [20]. Each processor has four lines which are direct connections to particular sections of memory. The ports from a particular processor access memory through a crossbar connection to the processor's four lines. The section number is determined by the lowest two bits of the address, so consecutive references are directed to different sections of memory. Each section is divided into eight subsections and the individual subsections are further subdivided in banks. In the case of the Cray Y-MP which has 256 banks, each subsection contains eight banks. Following the notation of Smith and Taylor, this interconnection network is denoted by:

$$8 \ \mathit{processors} \rightarrow 4 \times 4 \rightarrow 8 \times 8 \rightarrow 1 \times 8 \rightarrow 256 \ \mathit{memory\ banks.}$$

In simulations of systems with n_p processors and l logical banks, the number of subsections is fixed at eight and the number of banks per subsection is increased. The interconnection can then be described by:

$$n_p \ \mathit{processors} \rightarrow 4 \times 4 \rightarrow n_p \times 8 \rightarrow 1 \times l/8 \rightarrow l \ \mathit{memory\ banks.}$$

In the Cray Y-MP, a processor can access a particular subsection once every T_c cycles where T_c is the physical bank cycle time. This means that when a processor accesses a memory bank, the processor is blocked from issuing additional references to the entire subsection containing this bank for the full bank cycle time. Such a conflict is called a *subsection conflict* and, like the section conflict, is strictly an intraprocessor conflict. References from different processors to the same subsection can proceed without conflict provided that they are addressed to banks which are not already in use.

The simulation for the logical banks is based on the conflict scheme described above. When a particular memory location is referenced, the line, subsection, logical bank, and subbank numbers are calculated. If the line is free, it is reserved for T_r cycles and the subsection is checked. If the subsection is free, it is reserved for T_s cycles, and the logical bank is checked. If the logical bank register (LBR) is free, it is reserved for T_l cycles and the reference is initiated. The reference generates a hold and fails to issue if a conflict occurs at any level.

Once a reference has occupied the LBR for T_l cycles, it can be moved to the appropriate subbank queue if that queue is not full. The reference must spend T_d cycles in the queue before it can be processed by the physical memory. It is assumed that the reference must occupy the subbank for at least T_c cycles before the subbank can accept another reference. If the operation is a write, the subbank is free to accept another reference after T_c cycles. Reads are complicated by the return trip to the processor as now described.

A vector read reference is not considered to be completed until all of the element values have arrived at the processor. Read data values must be routed from the physical memory bank to the appropriate processor vector register. Additional conflicts may occur because more than one value may become available on a particular cycle. Each logical bank has a single output latch. If the latch is free, the value is moved from the subbank to the latch and the subbank is freed. If the latch is busy, the subbank must wait until the latch is free before accepting another value. If an output queue is included for each subbank as shown in Figure 1, the value is moved from the subbank to the output queue and blocking of the subbank due to return conflicts is less likely to occur.

All of the data values latched for a particular processor line compete for processing on the return interconnection network. The real system has separate forward and return interconnection networks. To simplify the simulation, the return interconnection network is modeled as a pipeline which can accept one value per section per cycle. The pipeline length is assumed to be ten which accounts for the length of both the forward and return pipelines. When the last value for a vector read has emerged from the pipeline, the read is considered to be complete.

The simulation also incorporates the feedback loop between the vector registers and memory. Each processor has a certain number of vector registers (eight was assumed for the runs in this paper). When a vector operation is initiated in the simulation, a free processor vector register is randomly selected and reserved for the duration of the operation.

If no register is available, the operation holds until a register becomes available. The register reserved for the operation is not freed until all of the elements of the vector have arrived at the processor. In contrast a vector write is considered to be completed when the last element operation has been issued. The vector register is freed at that time although the actual memory value may not be inserted until sometime later because of buffering.

Priority in the simulation is rotated among the processors in a circular fashion so that no processor is favored. This scheme is similar to the priority scheme used on the Cray X-MP. The Cray Y-MP uses a fixed subsection priority scheme which does not lend itself to modification when the number of processors is varied. The priority scheme should have little effect on the results of the simulation.

The simulation generates a representative reference stream for vectorized code as described below. All memory operations are assumed to be vector operations with an associated stride and length. Gather/scatter operations are not considered in this simulation. The stride is a fixed interval between successive references within a single vector operation. A stride of one is assumed to be the most probable with other strides up to a maximum stride being equally probable. A default probability of stride one vectors of .75 is used unless otherwise indicated. The effect of type of load on performance is examined in Section VII.

The maximum length of the vector operations is determined by the length of the vector registers in the processor. When an operation on a very long vector is required, the compiler splits it into several vector operations, all but one of which uses the maximum vector register length. All possible vector lengths are assumed to be equally probable, except the maximum length is assumed to occur more frequently.

The system load is determined by the operation initiation rate. When a port is free there is a certain probability, p_f, that on the current cycle a memory operation will be initiated. The value of p_f may be different for read and write ports and is a measure of the system load. A relationship between p_f and the scalar reference rate q is now derived in order to compare the vector case to the scalar case already discussed. Let V_L be the maximum allowed vector length and p_l be the probability of a maximum length reference. The average length of a vector reference is then:

$$V_A = p_l V_L + \frac{(1-p_l)V_L}{2} = \frac{(1+p_l)V_L}{2}$$

and:

$$q = \frac{V_A}{\frac{1}{p_f} + V_A - 1}$$

If $V_L = 64$, $p_l = .75$ and $q = .4$, then $V_A = 56$ and $p_f = .0118$. The value $q = .4$ was used by Bailey [1] in his calculations of efficiency for the unbuffered case. A study of interreference times for vector references for the Perfect Club Benchmarks run on a Cray Y-MP [18] shows typical interreference times on the order of 10 to 200 cycles, so a value of $p_f = .0118$ appears to be reasonable.

The parameters used in the simulation are chosen to be close to the current Cray Y-MP values and are summarized in Table 5. The scalar simulations performed in Section IV were performed by setting $V_L = 1$, $r = 0$, $T_d = 0$,i $T_c = 1$, $f_l = 9$, and $T_r = T_s = 0$. In the unbuffered case, T_l is the physical bank cycle time as seen by the interconnection network.

The vector data path simulator was written in C and run on a network of SUN workstations. Most of the vector simulations done for this paper were run for one hundred thousand cycles, although some runs were as long as ten million cycles. Each run was divided in blocks of cycles and the statistics were computed over each block in addition to over the entire run. The statistics for the longer runs did not vary significantly from those of the shorter runs. This lack of variation is not unexpected since each port on each processor can initiate a reference on each cycle so there are a large number of independent reference streams over which the statistics are averaged.

VI. Results for Moderate Loading

Figure 5 shows a comparison of the efficiencies as a function of bank cycle time for the vector simulation and the logical bank model when the system has eight processors and three ports per processor. The number of logical banks is fixed at 256 with one subbank per logical bank and the queue size is four slots per subbank. Buffering delays the drop in performance with increasing bank cycle time. For example, when all vectors have stride one ($p_s = 1$) and the bank cycle time is 20, the efficiency is .92 with logical bank buffering and

.22 without buffering. The agreement between the simulation and the logical bank model is good for subbank cycle times less than 10.

The agreement between model and simulation is better for loads which have a random component. The case where three fourths of the strides are one and the remainder are randomly distributed (p_s = .75) is also shown in Figure 5. In this case, the efficiency is .68 for a subbank cycle time of 20. The overall efficiency is slightly below the model, but the fall-off occurs in roughly the same place as predicted by the model. The logical bank model, which corresponds to random reference streams, falls in between the simulations for the two different stride distributions. The agreement between the logical bank model and the vector simulation is quite good considering that the vector simulation includes lines, subsections, and register feedback. The dip in efficiency at bank cycle times which are integral multiples of four is a real phenomenon which is preserved over very long simulation runs.

In Figure 6 the efficiency is plotted versus the number of reference streams when the subbank cycle time is five. The remaining parameters are the same as in Figure 5. The number of reference streams could be quadrupled from 24 (eight processors) to 96 (32 processors) while still maintaining an efficiency of .67.

The analysis of Section IV for random references predicts that the efficiency will be high and will increase slightly as the number of banks and the bank cycle time are increased keeping $\rho = qT_cn/b$ constant at a value $\leq .5$. This scaling relationship holds in the case of vector references as well. In Figure 7 the number of reference streams is fixed at 24 (eight processors) and the number of subbanks per logical bank is one. The number of banks is varied linearly with the bank cycle time in order to keep ρ constant at .5. The condition $\epsilon < .1$ is satisfied when the number of logical banks is greater than 120 which is the case provided that $T_c > 3$ for ρ held constant at .5. For comparison, a simulation with the same set of parameters was run without buffering and the efficiency dropped dramatically when the subbank cycle time was increased. The runs are shown for two stride distributions (p_s = .75 and p_s = .25).

The previous vector runs were performed with no read ports. In an unbuffered memory there is no difference in memory efficiency between reads and writes. Because buffering introduces unpredictable delays, more than one result can become available on a particular cycle for the interconnection network for a particular section. When conflicts of this type arise, some of the references are delayed which in turn causes subbanks to block. Return conflicts can thus cause an effective increase in bank cycle time and a corresponding drop in efficiency. The addition of a single output queue slot at each subbank eliminates the difference in performance between reads and writes for moderate loading. However, as the load is increased (either through increasing the initiation rate or the percentage of stride one vectors), the port return bandwidth may be insufficient to handle the load. This problem is addressed in the next section.

The analytical model derived in Section III predicts the efficiency of memory writes. Other possible indicators of performance include throughput and latency. The throughput defined in Section I is the fraction of the optimal rate at which entire vectors are delivered through the system [21]. The vector element read latency is defined as the time between the first attempt to access a vector element and the availability of that element at the vector register.

Figure 8 compares the efficiency, throughput, and read latency when the load is varied. A bank cycle time of 20 was chosen because it is near the knee of the curves in Figure 5. The parameters are the same as in that figure except that the bank cycle time is fixed and the initiation rate is varied. *Probability of OP* (p_f) refers to the probability that a vector operation will be initiated by a free port. The value $p_f = .0118$ corresponds to the value in Figure 5. The throughput is slightly higher than the efficiency in the unbuffered case and slightly lower than the efficiency in the buffered case. The buffered throughput is still considerably better than the unbuffered throughput.

In the unbuffered case, the read latency is just a constant plus the number of attempts it takes to issue the request. As mentioned in Section III, the number of attempts is just the reciprocal of the efficiency. In fact, in the unbuffered case the read latency curve in Figure 8 can be predicted to better than .3 percent from the efficiency curve. When $p_f = .01$, the unbuffered read element latency is 36 and the buffered read element latency is 51. If the efficiency is at least moderately good, this indicates that the last element of a vector read is delayed about 15 cycles over what it would be without buffering. With buffering, the read latency is affected by return conflicts, and so is dependent on the return scheme used. Different return schemes are discussed in the next section.

Since writes do not require a return path, the write element latency is directly related to the number of attempts to issue the element write operation. The write latency curves (not shown) can be predicted from the efficiency curve to within a few percent for both the buffered and the unbuffered case.

VII. Results for Maximal Loading

The extreme case where each port attempts a memory reference on each cycle is now considered. First an analysis is done for writes. The logical bank model is used to pick design parameters for a 64-processor system. The performance for reads is then analyzed and improvements in the return interconnection network are considered to equalize the performance of reads and writes.

The logical bank model can be used as a guide in picking design parameters for a 64-processor Cray Y-MP which minimizes the per processor interconnection cost, while achieving an efficiency of at least .90. Assuming a load of $q = p_f = 1.0$, the condition $\epsilon < .1$ gives $l > 960$. Assuming a bank cycle time of five, the condition $\rho < .5$ gives $b > 1920$. The number of logical and subbanks should be powers of two. Thus for 64 processors, the configuration which operates with minimum per processor interconnection cost and high efficiency has 1024 logical banks and 2048 physical subbanks. If ρ is approximately .5, a queue size of four is required to have a .99 probability of a free slot available in the queue according to Table 4. If the queue size is two, four subbanks per logical bank are required to bring ρ down to .3.

Figure 9 shows the performance of these designs as a function of the percentage of stride one vectors. The case of two subbanks per logical bank with a queue size of four is indistinguishable from the case of four subbanks per logical bank with a queue size of two as predicted by the logical bank model. The case of 2048 banks with a queue size of two is shown for comparison. The logical bank model is based on the assumption of random references and does not account for the presence of bad strides. Buffering has been shown to reduce the effect of bad strides under moderate loading [7], and the designs here do not preclude the use of address mapping to alleviate intraprocessor conflicts [5,8]

Reads are more difficult to handle because hot spots can develop on the return lines as shown in Figure 10. The performance difference between reads and writes as a function of stride is quite dramatic. Even more surprising is the fact that the performance for reads actually drops as the percentage of stride one vectors in the load is increased. The drop occurs because the arbitration method used for the return in the simulations is a simple round robin priority scheme on the logical banks. When a reference for a particular port is delayed, it causes all of the banks waiting for that port to be delayed. When the system is operating at a sustained maximal initiation rate, the ports can never catch up.

Various solutions for solving the hotspot problem have been examined including additional output buffering at the physical subbanks, additional lines, optimal arbitration, port handshaking, port-line queues, and additional return ports. It was found that additional lines alone do not solve the problem, while queue depths of 30 or more at each subbank are required to make a significant difference in efficiency.

In optimal arbitration, the logical banks are examined in round robin succession, but if the port to which a reference is made is already in use, that reference does not block the line and another reference can be issued. Port handshaking is a control mechanism by which a particular port is blocked from initiating a reference if there was a return conflict for that port on the previous cycle. Both methods improve performance, but they do not bring read performance up to write performance when most of the vector strides are one.

Since the drop in read performance appeared to be due to insufficient return port bandwidth, two alternative approaches were developed to increase the bandwidth. The first approach involved adding port-line queues. In this design modification, each port contains a queue for each line so that each line can deposit a result at a port on each cycle. The port services one queue per cycle in round robin succession. The second approach involved doubling the number of return ports. One possible design is to have a return port for the odd-numbered vector elements and another return port for the even-numbered vector elements. An alternative design is to designate one return port for each pair of return lines. The second alternative would simplify the port-line interconnection switches but would complicate the vector register bus structure internal to the processors.

Figure 11 compares the port-line buffering to double-return ports for a variety of strides at maximal loading. Output buffering at the subbanks has been eliminated. Port-line buffering improves efficiency but does not increase the port bandwidth. The doubling of the number of return ports brings the read efficiency in line with the write efficiency.

VIII. Discussion

The main result of this paper can be summarized as follows. If a shared memory system has sufficient bandwidth to achieve high efficiency with fast memory, the replacement of the physical banks by logical banks will allow the same efficiency to be achieved using considerably slower memory without significantly affecting the interconnection costs. This type of buffering is particularly useful in vector multiprocessors because vector memory operations are naturally pipelined, and increases in memory latency can be partially amortized over an entire vector operation.

The logical bank model and the detailed vector simulations given in this paper show that the number of logical banks scales with the number of processors and that the bank cycle time scales with the number of physical banks. Consequently slower memory can be used if the logical banks are divided into more subbanks. This is in contrast to the unbuffered case where $b/T_c^2 n$ must be constant for constant efficiency. The change from an inverse quadratic to an inverse linear dependence between bank cycle time and the number of banks is particularly important.

A drawback of memory systems with variable bank cycle time is that the values become ready for return at unpredictable times. The approach of Seznec and Jegou to reorder values at the bank level does not solve the problem of values arriving at the processor in the order issued. The problem can be addressed in the Cray Y-MP architecture by the addition of a tag to each return value. The Cray Y-MP architecture allows three independent vector memory operations to proceed simultaneously. These vector memory operations can be *chained* to the vector registers, and vector registers can in turn be chained to functional units. Chaining allows the results produced by one vector operation to be used as input to a succeeding operation before the first instruction has completed. The component results from the first instruction can be used by the second instruction as they become available. Pipeline setup can occur before any component of the previous operation is available [15]. In the current architecture values arrive in order so each vector register keeps track of the last value to have arrived. When the values arrive out of order each vector element must have a bit indicating whether that value has arrived. Some additional hardware can be incorporated to chain forward when the next element has arrived. The relative order among different registers is assured by the existing reservation and issue mechanisms.

It is well known [1] that the memory performance of shared memory vector processors is strongly dependent on the type of load which is generated. Since the choice of load distribution affects the results, it is desirable to test the design against realistic loading conditions. Address-trace collection methods [22] are useful for generating statistical information about the load, but the information collected from these types of investigations is difficult to use directly in testing new designs because the efficiency depends not only on the actual addresses, but on the exact time the references were issued. Because of these difficulties, the approach taken in this paper has been to develop guidelines which are applicable over a range of load distributions. The larger the number of reference streams, the less serious the impact of the details of one reference stream on the overall efficiency.

Buffering greatly reduces the dependence of memory efficiency on the type of load for writes as illustrated by the runs in the two previous sections. Most of the dependence on load occurred because of return conflicts for reads. A number of alternative designs were evaluated in an effort to reduce the performance degradation due to return conflicts. It was found that doubling the number of return ports eliminated the difference between reads and writes over a range of loads.

ACKNOWLEDGMENT

This work was supported by Cray Research Inc. Computational support was provided by the University of Texas Center for High Performance Computing and the National Science Foundation ILI Program, Grant USE-0950407.

References

[1] D. H. Bailey, "Vector computer memory bank contention," *IEEE Trans. on Computers*, vol. C-36, pp. 293–298, 1987.

[2] F. A. Briggs, "Effects of buffered memory requests in multiprocessor systems," *Proc ACM-SIGMETRICS Conf. on Simulation, Measurements, and Modeling of Computer Systems*, pp. 434–442, 1979.

[3] F. A. Briggs and E. S. Davidson, "Organization of semiconductor memories for parallel-pipelined processors," *IEEE Trans. on Computers*, vol. C-26, pp. 162–169, 1977.

[4] D. Bondurant, "Enhanced dynamic RAM," *IEEE Spectrum*, vol. 29, no. 10, pp. 49, 1992.

[5] T. Cheung and J. E. Smith, "A simulation study of the Cray X-MP memory system," *IEEE Trans. on Computers*, vol. C-35, pp. 613–622, 1986.

[6] M. P. Farmwald and D. Mooring, "A fast path to one memory," *IEEE Spectrum*, vol. 29, no. 10, pp. 50–51, 1992.

[7] D. T. Harper and J. R. Jump, "Vector access performance in parallel memories using a skewed storage scheme," *IEEE Trans. on Computers*, vol. C-36, pp. 1440–1449, 1987.

[8] D. T. Harper, "Address transformations to increase memory performance," *Proc 1989 Intl. Conf. on Parallel Processing*, pp. 237–241, 1989.

[9] C. A. Hart, "Dynamic RAM as secondary cache," *IEEE Spectrum*, vol. 29, no. 10, pp. 48, 1992.

[10] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, 1991.

[11] N. Kushiyama, Y. Watanabe, T. Ohsawa, K. Muraoka, Y. Nagahama, and T. Furuyama, "A 12-MHz data cycle 4-Mb DRAM with pipeline operation," *IEEE J. Solid-State Circuits*, vol. 26, no. 4, pp. 479–482, 1991.

[12] D. H. Lawrie and C. R. Vora, "The prime memory system for array access," *IEEE Trans. on Computers*, vol. C-31, pp. 435–442, 1982.

[13] D. Lee, "Scrambled storage for parallel memory systems," *Proc. 15th Intl. Conf. on Computer Architecture*, pp. 232–239, 1988.

[14] B. Prince, R. Norwood, J. Hartigan, and W. C. Vogley, "Synchronous dynamic RAM," *IEEE Spectrum*, vol. 29, no. 10., pp. 44–48, 1992.

[15] K. A. Robbins and S. Robbins, *The Cray X-MP/Model 24: A Case Study in Pipelined Architecture and Vector Processing*, Berlin Heidelberg: Springer Lecture Notes in Computer Science, vol. 374, 1989.

[16] K. A. Robbins and S. Robbins, "Bus conflicts for logical memory banks on a Cray Y-MP type processor system," *1991 Intl. Conf. Parallel Processing*, pp. 21–24, 1991.

[17] K. A. Robbins and S. Robbins, "Dynamic behavior of memory reference streams for the Perfect Club benchmarks," *Proc. 1992 Intl. Conf. on Parallel Processing*, pp. 48–52, 1992.

[18] K. A. Robbins and S. Robbins, "Characterization of memory loads for vectorized programs." preprint.

[19] A. Seznec and Y. Jegou, "Optimizing memory throughput in a tightly coupled multiprocessor," *Proc. 1987 Intl. Conf. on Parallel Processing*, pp. 344–346, 1987.

[20] J. E. Smith and W. R. Taylor, "Accurate modeling of interconnection networks in vector supercomputers," *1991 Intl. Conf. on Supercomputing*, pp. 264–273, 1991.

[21] G. S. Sohi, "High-bandwidth interleaved memories for vector processors — a simulation study," *IEEE Trans. on Computer Systems*, vol. 42, pp. 34–44, 1993.

[22] C. B. Stunkel, B. Janssens, and W. K. Fuchs, "Address tracing for parallel machines," *Computer*, pp. 31–38, 1991.

Kay A. Robbins graduated from the Massachusetts Institute of Technology with an S.B. in mathematics in 1971. She earned a Ph.D. from MIT in mathematics in 1975. After graduation she moved to the University of Texas at San Antonio where she now holds the rank of Professor of Computer Science. She has done research in the areas of mathematical modeling, dynamical systems, parallel architectures, and distributed computing. She has received funding support from the Office of Naval research, the National Science Foundation, and Cray Research.

Steven Robbins graduated from the Massachusetts Institute of Technology with S.B. and S.M. degrees in mathematics in 1969. In 1973 he earned a Ph.D. from MIT in mathematics. After graduation, he worked as an instructor at MIT and later moved to the University of Texas at San Antonio where he currently holds the rank of Associate Professor of Computer Science. He has done research in the fields of quantum field theory, parallel architectures, and distributed computing. He has received funding support from the National Science Foundation and Cray Research.

Parameter	Definition	Default
n	Number of independent reference streams	24
r	Number of independent read reference streams	0
k	Number of subbanks per logical bank	1
l	Number of logical banks in the system	256
b	Total number of subbanks = kl	256
m	Input queue slots per subbank	2
m_o	Output queue slots per subbank	0
f_l	Fan-in length	10
T_c	Physical subbank cycle time	5
T_d	Minimum delay incurred in transfer from queue to subbank	1
T_p	Effective subbank queue service time	7
T_l	Logical bank cycle and access time	1
q	Reference stream average initiation rate	.4

This part is not part of the caption.

Table 1: Parameters which define logical banks.

i	$j = 1$	$j = 2$	$j = 3$
1	$1 - q$	$1 - q$	0
2	$q\delta$	$q\delta$	δ
3	$q(1 - \delta)$	$q(1 - \delta)$	$(1 - \delta)$

Table 2: The transition probability matrix, $\Gamma_{i,j}$, in the direct model for the logical bank efficiency (E_l).

ρ	m=1	m=2	m=3	m=4	m=5	m=6	m=7	m=8
0.1	0.9099	0.9955	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000
0.2	0.8333	0.9820	0.9985	0.9999	1.0000	1.0000	1.0000	1.0000
0.3	0.7694	0.9609	0.9946	0.9993	0.9999	1.0000	1.0000	1.0000
0.4	0.7148	0.9346	0.9863	0.9973	0.9994	0.9999	1.0000	1.0000
0.5	0.6672	0.9043	0.9731	0.9923	0.9978	0.9994	0.9998	0.9999
0.6	0.6253	0.8707	0.9535	0.9826	0.9933	0.9974	0.9990	0.9996
0.7	0.5880	0.8358	0.9279	0.9661	0.9834	0.9917	0.9959	0.9979
0.8	0.5553	0.8005	0.8971	0.9415	0.9648	0.9783	0.9863	0.9912
0.9	0.5262	0.7655	0.8619	0.9089	0.9357	0.9528	0.9646	0.9729
1.0	0.5000	0.7313	0.8240	0.8699	0.8968	0.9144	0.9269	0.9362

Table 3: Probability that a request will reach a queue which is not full for a queue size of m and various values of ρ for the $M/D/1/B$ queuing discipline.

ρ	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$	$x = 6$	$x = 7$
0.1	0.9000	0.9947	0.9998	1.0000	1.0000	1.0000	1.0000	1.0000
0.2	0.8000	0.9771	0.9980	0.9999	1.0000	1.0000	1.0000	1.0000
0.3	0.7000	0.9449	0.9920	0.9990	0.9999	1.0000	1.0000	1.0000
0.4	0.6000	0.8951	0.9773	0.9954	0.9991	0.9998	1.0000	1.0000
0.5	0.5000	0.8244	0.9470	0.9847	0.9957	0.9988	0.9996	0.9999
0.6	0.4000	0.7288	0.8907	0.9574	0.9835	0.9936	0.9975	0.9990
0.7	0.3000	0.6041	0.7937	0.8947	0.9464	0.9727	0.9861	0.9929
0.8	0.2000	0.4451	0.6345	0.7621	0.8454	0.8995	0.9347	0.9575
0.9	0.1000	0.2460	0.3836	0.4987	0.5925	0.6687	0.7307	0.7811
1.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 4: Probability that there are no more than x items in the queue assuming an infinite queue and a system load specified by ρ.

Parameter	Definition	Default
n_p	Number of processors	8
p	Number of ports per processor	3
p_f	Free port initiation rate	.0118
p_l	Probability of a maximum length vector	.75
p_s	Probability of a stride-one vector	.75
V_L	Maximum vector length	64
n_V	Number of vector registers per processor	8

Table 5: Parameters for the vector simulation.

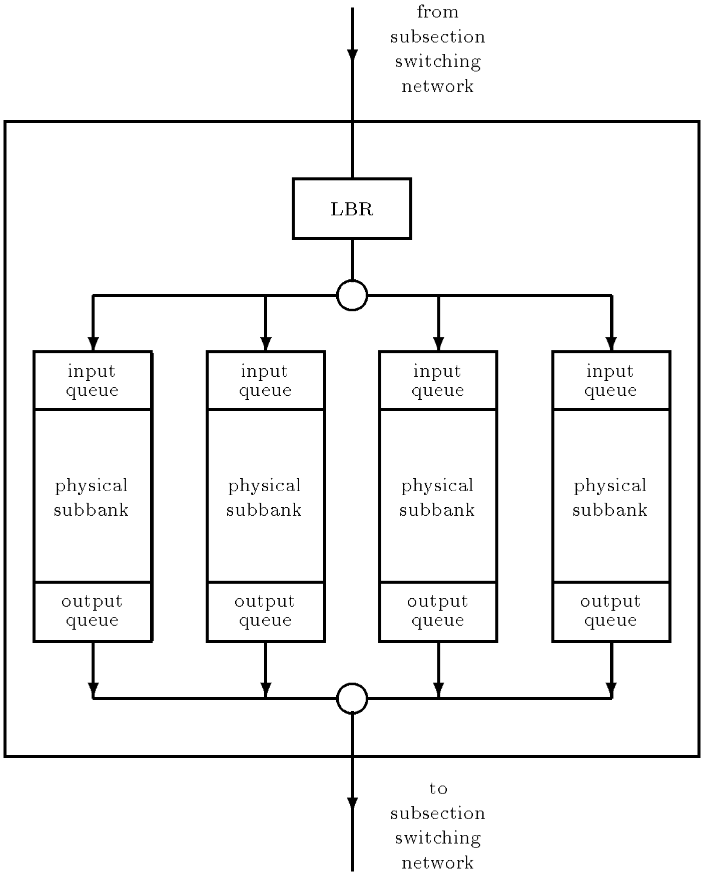

Figure 1: Structure of a logical bank with four subbanks.

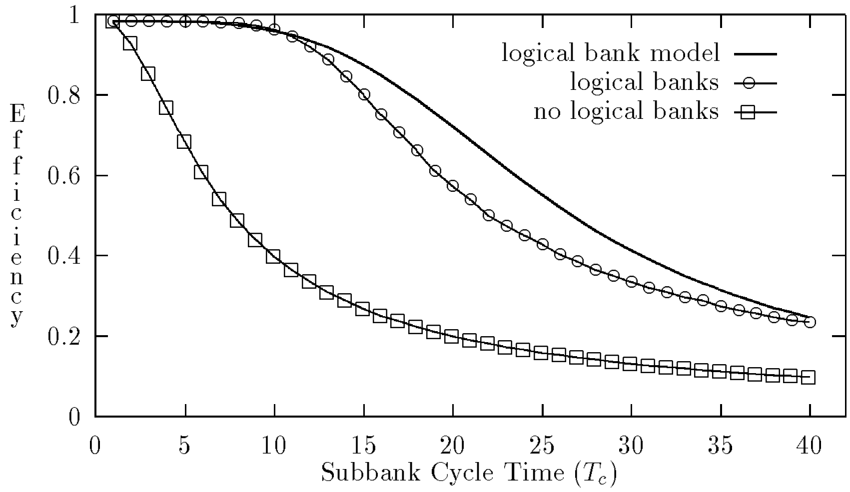

Figure 2: The logical bank model efficiency as a function of subbank cycle time is compared to random reference simulations for $m = 2$. The case of no logical banks ($m = 0$, $T_l = T_d = 0$) is shown for comparison.

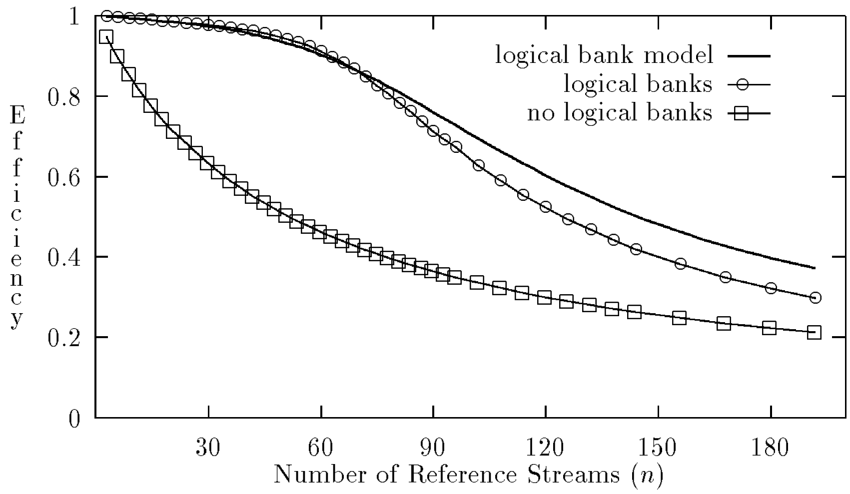

Figure 3: The logical bank model efficiency as a function of the number of processors is compared to random reference simulations for the same parameters as in Figure 2. $T_c = 5$.

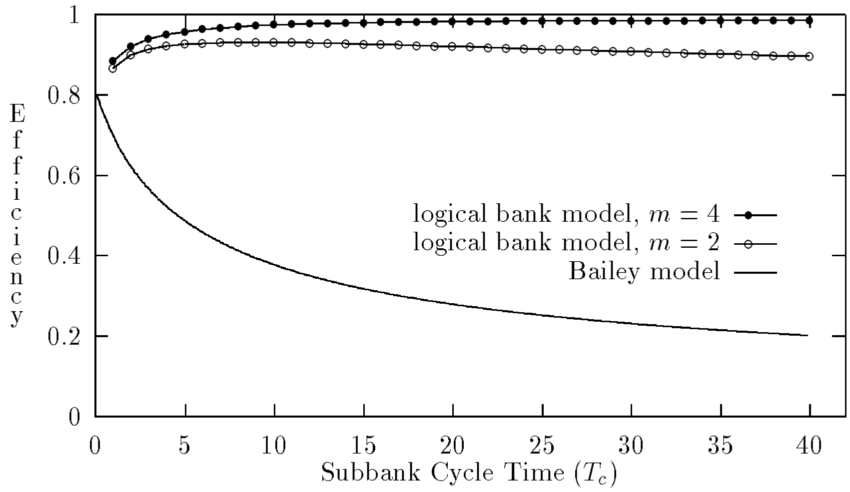

Figure 4: Performance of logical banks when the bank cycle time and the number of banks are varied keeping ρ fixed at .5. The efficiency of the Bailey model is shown for comparison.

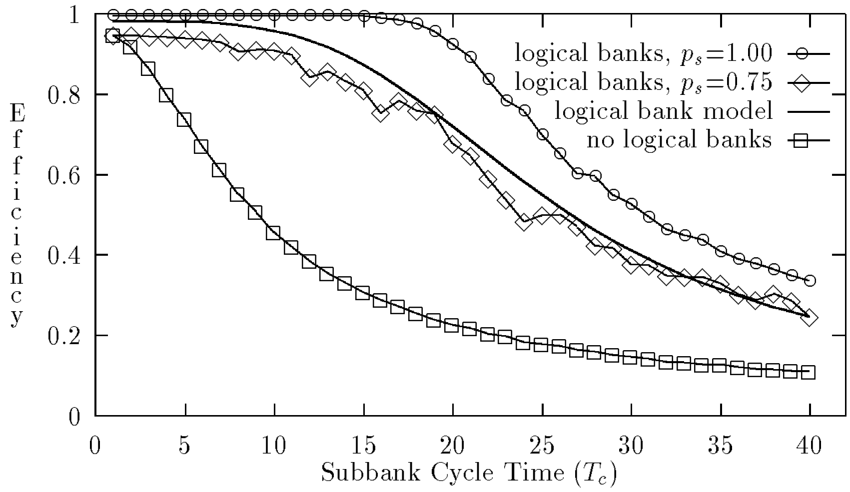

Figure 5: The efficiency as a function of bank cycle time for vector simulations with different stride distributions. The predictions of the logical bank model are shown for comparison. The case of no logical banks is run for $p_s = .75$.

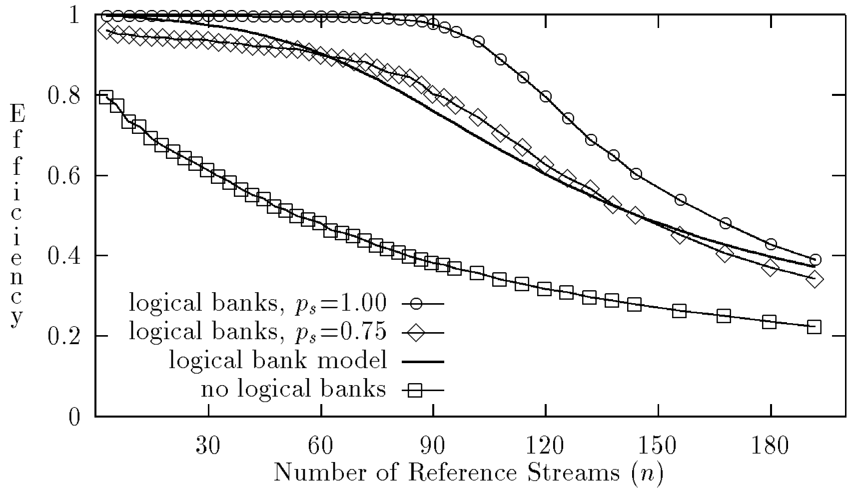

Figure 6: The efficiency as a function of number of reference streams for vector simulations with different stride distributions. The predictions of the logical bank model are shown for comparison. The case of no logical banks is run for $p_s = .75$.

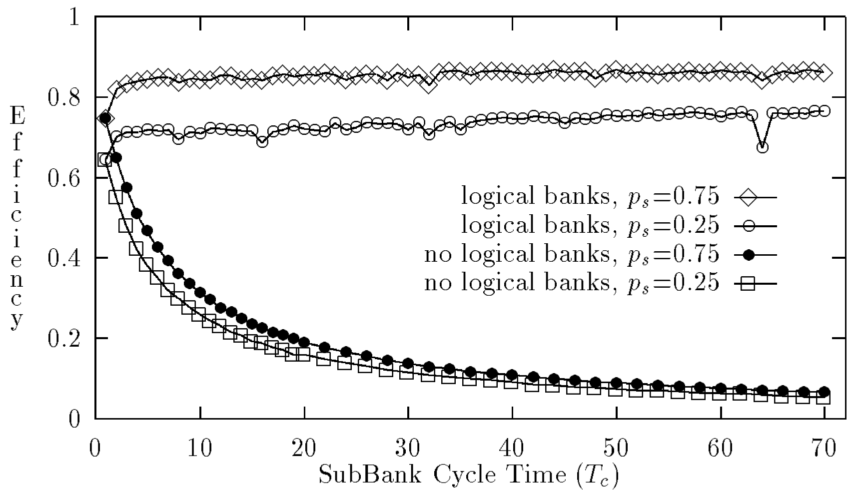

Figure 7: Performance of logical banks in the vector simulation when the bank cycle time and the number of logical banks are varied keeping ρ fixed at .5. Performance is shown for loads consisting of two different stride distributions.

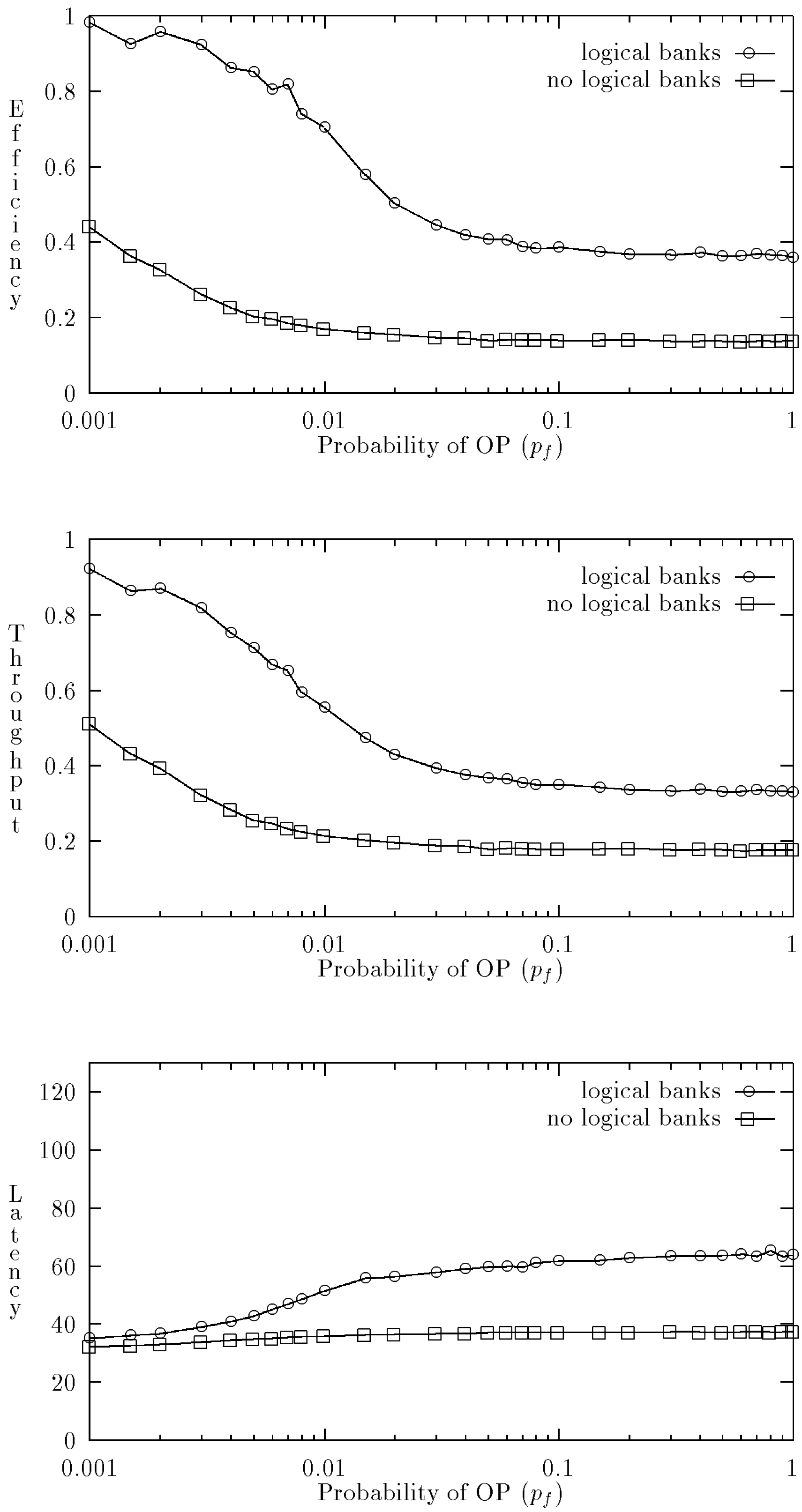

Figure 8: Efficiency, throughput, and read latency for back cycle time of 20 with a varying load and two read ports per processor ($r = 2$).

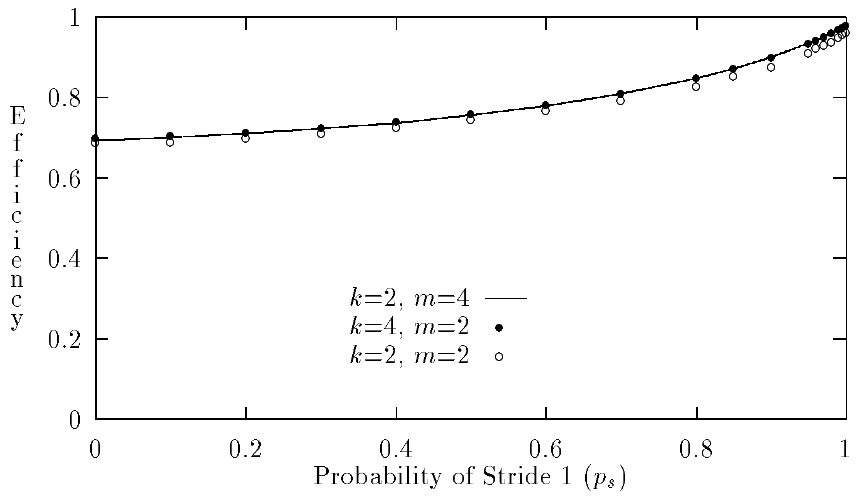

Figure 9: Effect of load distribution on efficiency for different amounts of buffering for writes under maximal loading. Other parameter values are: $l = 1024$, $n_p = 64$, $p = 3$, and $n = 192$.

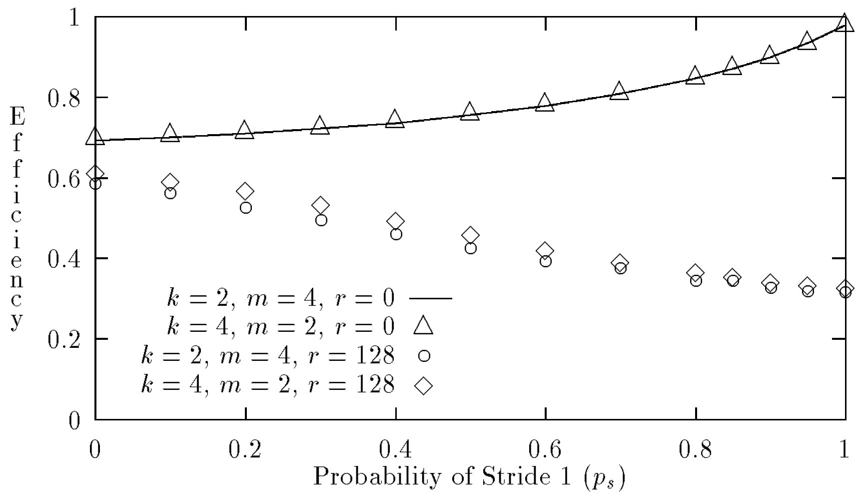

Figure 10: The effect of read ports on efficiency under maximal loading. The parameter values are: $l = 1024$, $n_p = 64$, $p = 3$, and $n = 196$.

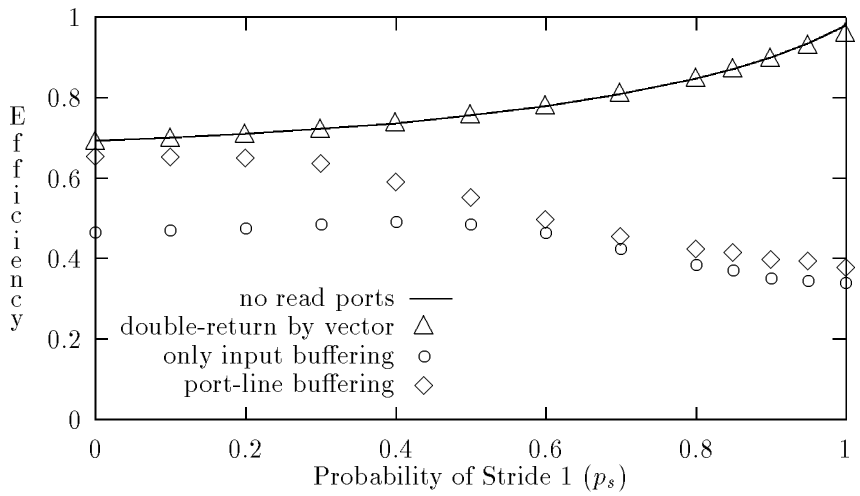

Figure 11: Comparison of double return ports and port buffering. The parameter values are: $l = 1024$, $n_p = 64$, $p = 3$, $n = 196$, $r = 128$, and $k = 2$.