**Observations Concerning Multidimensional Ultracomputers**

by

*Clyde Kruskal*
*Larry Rudolph*

Ultracomputer Research Laboratory
Courant Institute of Mathematical Sciences
715 Broadway, 10th Floor
New York, NY 10003

Ultracomputer Note #6
January, 1980

*ABSTRACT*

Schwartz introduced the idea of an ultracomputer and reviewed various basic algorithms for such an ensemble of processors containing ‘‘shuffle’’ interconnections. Later, Harrison and Kalos introduced the idea of an N-dimensional ultracomputer (N-D UC), which is well suited for implementing many algorithms. The main result of this note is that an N-D UC can simulate the right/left and shuffle/unshuffle connections of its 1-dimensional counterpart in N and 2N steps, respectively. Thus, for each N, an N-D UC is asymptotically as fast as a 1-D UC of the same size.

## 1. Introduction

Schwartz introduced the idea of an ultracomputer and reviewed various basic algorithms for such an ensemble of processors containing ‘‘shuffle’’ interconnections. Later, Harrison and Kalos introduced the idea of an N-dimensional ultracomputer (N-D UC), which is well suited for implementing many algorithms. The main result of this note is that an N-D can simulate the right/left ans shuffle/unshuffle connections of its 1-dimensional counterpart in N and 2N steps, respectively. Thus, for each N, and N-D UC is asympototically as fast as a 1-D UC of the same size.

In Section II we describe N-D UCs, in Section III we prove our main result along with some related theorems, and in Section IV we consider a couple of modified N-D UC models which yield slightly better results.

## 2. The Model

In this section we give the definitions necessary for reading this note.

The *size of a parallel processor* is the number of processing elements (PEs) it contains. We denote this processor size by P and number the processing elements $PE_0$, ..., $PE_{P-1}$.

**Def.** Given a parallel processor of size $P = 2^q$, let $b_{q-1}\, b_{q-2} \ldots b_0$ be the binary representation of B. Then in a *1-D ultracomputer* each $PE_B$ is directly connected to (its four neighbors):

**Right** $PE_{(B+1 \bmod P)}$;

**Left** $PE_{(B-1 \bmod P)}$;

**Shuffle** $PE_{(b_{q-2} b_{q-3} \cdots b_0 b_{q-1})}$ (i.e. a left rotate);

**Unshuffle** $PE_{(b_0 b_{q-1} b_{q-2} \ldots b_1)}$ (i.e. a right rotate).

Intuitively, an N-D UC is an N-dimensional rectilinear grid of PEs where every row (in each dimension) is a 1-D UC.

**Def.** A $d_N \times d_{N-1} \times \ldots \times d_1$ *N-D UC* is a set of $\prod_{i=1}^{N} d_i$ PEs, each a different member of the set of tuples $d_N \times d_{N-1} \times \ldots \times d_1$ such that $PE_{(e_N, \ldots, e_k, \ldots, e_1)}$ is connected to $PE_{(e_N, \ldots, e'_k, \ldots, e_1)}, 1 \le k \le N$, iff $e_k$ is connected to $e'_k$ in the size $d_k$ 1-D UC.

For example, a $d_2 \times d_1$ 2-D UC is a $d_2 \times d_1$ rectangle of PEs where each row and each column is independently a 1-D UC of size $d_2$ and $d_1$ respectively.

Throughout this note we let $d_i$, $1 \le i \le N$, be the size of the i-th dimension, $U_i$ be the set of 1-D UCs in the i-th dimension, and $e_i$ be the number of $PE_{(e_N, \ldots, e_{N-1}, \ldots, e_1)}$ within $U_i$.

A one-to-one correspondence is needed between each PE in a $d_N \times d_{N-1} \times \ldots \times d_1$ N-D UC and each PE in a 1-D UC of the same size. The following correspondence is called the *1-D counterpart* of the N-D UC.

We number the PEs from 0 to P-1 using the standard row-major indexing of an N-dimensional array, i.e. $PE_{(e_N, e_{N-1}, \ldots, e_1)}$ corresponds to the PE with address

$$e = \sum_{m=1}^{N} \left( e_m \prod_{i=1}^{m-1} d_i \right)$$

in a 1-D UC. We often write the address of $PE_{(e_N, \ldots, e_1)}$ in its 1-D counterpart as $e_N e_{N-1} \ldots e_1$ and refer to this PE as $PE_e$ or $PE_{e_N\, e_{N-1} \ldots e_1}$. Since each individual dimension is a 1-D UC each $d_i$ is a power of two. Thus, the least significant $\log d_1$ bits correspond to $e_1$; the next $\log d_2$ bits correspond to $e_2$;...; and the most significant $\log d_N$ bits correspond to $e_N$. Therefore, dealing with each $U_i$ merely requires manipulation of the proper $\log d_i$ contiguous bits. All of the results in this paper rely on the natural correspondence. Note that the correspondence depends on the ordering of the N dimensions and, in fact, there are N! equally valid 1-D counterparts for each N-D UC.

A potential drawback of N-D UCs is the number of connections to each PE (4N, as compared to 4 for a 1-D UC). However, this number is manageable for 2 and 3 dimensions, and moreover there are compensating advantages.

N-D UCs have embedded N-D mesh connected computers and are often useful for those problems which can be decomposed into a multidimensional grid, where the computation of a grid point often requires efficient access to data at neighboring points. For example, in fluid dynamics problems each processor can store the data for a small region of space and the N-D UC connections provide rapid access to neighboring regions.

In various applications, subproblems such as the FFT can be solved quickly by considering each row of processors to be an individual ultracomputer. This concept can be generalized by allowing an N-D UC to simulate many differently shaped ultracomputers of smaller dimension. Thus, we are able to choose subspaces in many different ways.

This partitioning is also adaptable to multiprogramming. Various programs can execute simultaneously (but separately) on these smaller dimensional UCs as need dictates. Moreover, if a few PEs are disabled, an appropriate partitioning would allow many of the (sub N)-D UCs to function.

Finally, the physical layout of a 2-D or 3-D UC may be more modular than its 1-D counterpart since each dimension can be wired separately.

## 3. Results

In this section we show our main result, that any N-D UC can simulate the 1-D UC right and left connections in N steps and the shuffle and unshuffle connections in 2N steps. Given our correspondence between PEs in a 1-D UC and N-D UC we show these results are almost optimal and also that 1-D UCs need $\Omega$ (log P) steps to simulate an N-D UC. Finally, we show that N-D UCs are quite versatile, and allow many PEs to function in a desirable fashion, even when some are disabled.

Since the unshuffle and left connections are merely the reverse of the shuffle and right, we will give proofs only for the shuffle and right connections.

**Def.** During every step of a permutation of a set of items in a parallel processor each item has two PE numbers of special interest: the PE number in which it currently resides called its *current number* and the PE number in which it will finally reside called its *destination number*. We refer to the PE number in which an item originates as its *initial number*.

In the following propositions we assume that an N-D UC has dimension sizes $d_1, d_2, \ldots, d_N$ and show how it can simulate the connections of a 1-D UC of size $\prod_{i=1}^{N} d_i$.

**Theorem** An N-D UC can realize the right/left connections of its 1-D counterparts in N steps.

**Proof** The following accomplishes a right send: (See figure 1.) Every PE sends right on $U_1$. Next, every PE whose number contains zeros in its low order log $d_1$ bits (i.e., $e_1$=0) sends right on $U_2$. Then, every PE whose number contains zeros in its low order log $d_1$+log $d_2$ bits sends right on $U_3$. Simulation continues in this fashion for a total of N steps and is easily seen to accomplish the desired right send. Figure 1. Steps to Simulate a Right Send.

| | $U_4$ | $U_3$ | $U_2$ | $U_1$ |
|---|---|---|---|---|
| Initial Number: | — | — | 0101 | 1111 |
| Current Number after 1st right send | — | — | 0101 | 0000 |
| Current Number after 2nd right send (and destination number): | — | — | 0110 | 0000 |

Note that if, as in many algorithms, the odd-even connections (i.e. $PE_{2i-1}$ with $PE_{2i}$) are not needed, then all right/left connections of the 1-D counterpart are just $U_1$ right/left connections.

**Def.** Given $2^q$ PEs, let $PE_k^{(i)}$ be the $PE_j$ where the binary representation of j matches that of k except in the i-th low binary representation of j matches that of k except in the i-th low order bit (which is

complemented). Then in a *cube connected computer* each PE is connected to $PE^{(i)}$, $0 \le i < q$. Thus, each PE has q connections.

**Proposition** A cube connected computer of $2^q$ PEs can realize the shuffle and unshuffle connections in q steps.

**Proof** Recall that a shuffle is a left rotate of the bits of the PE number. Since there are $2^q$ PEs, the initial PE number of an item can differ by at most q bits from its destination PE number. Our method is to iteratively correct (if necessary) each of the q bits.

**Method** Each PE executes the following:

```
For j = 0 to q-1
    ∀ items i in the PE
        IF DEST_j (i) ≠ CURR_j (i)
            THEN send i to PE^(j)
```

where: $DEST_j(i)$ is the j-th low order bit (LOB) of the destination number of item i.

and $CURR_j(i)$ is the j-th low order bit of the current number of item i.

For each item all q bits are correct at the end of the simulation, so the algorithm obviously carries out the simulation correctly in q loop iterations. However, we must show that at each iteration of the loop, each PE executes no more than one data communication step (i.e., one send and one receive).

After the j-th step, $1 \le j < q$, the current number of each data item has its low order j bits equal to the low order j bits of its destination number and its high order q-j bits equal to the high order q-j bits of its initial number. Since the destination number is a left rotation of the initial number, the j-th bit of the initial number (i.e. j+1 bit of the destination number) will be missing from the current number (see Figure 2). Half of the PEs (those with HOB = LOB) will contain exactly two items ($b_j$=0,1), and the remaining half of the PEs will contain no items. Moreover, on step j+1, each PE containing two items will send the one with $b_j \ne b_{j+1}$.

**Figure 2.** Simulation of a 1-D shuffle on a CCC.

| | |
|---|---|
| Initial number | $b_{q-1}\ b_{q-2}\ \cdots b_{j+1}\ b_j\ \cdots\ b_1\ b_0$ |
| Current number (after step j) | $b_{q-1}\ b_{q-2}\ \cdots b_{j+1}\ b_{j-1}\ \cdots b_0\ b_{q-1}$ |
| Destination number | $b_{q-2}\ b_{q-3}\ \cdots b_j\ b_{j-1}\ \cdots\ b_0\ b_{q-1}$ |

On the first iteration, half of the PEs retain their items since their low order bits are already correct. Each of these PEs must also receive an item from exactly one of the remaining PEs in order to satisfy the requirement that half the PEs contains two items. Thus, half of the PEs send an item and the other half receive an item.

Then, as noted above, for iterations 2 through q-1, the PEs with $b_q = b_1$ send and therefore receive exactly one item. On the last iteration, the PEs with two items send one item to a unique PE with no items. Q.E.D.

**Theorem** An N-D UC can realize the shuffle and unshuffle connections of its 1-D counterpart in 2N

steps.

**Proof** Recall that the number of $PE_{(e_N, e_{N-1}, \ldots, e_1)}$ in the 1-D counterpart of an N-D UC is $e_N e_{N-1} \ldots e_1$, and that the result of a 1-D UC shuffle is this number rotated left one bit. We can realize this 1-D shuffle as follows: First shuffle in each dimension separately, i.e. first all $U_1$s shuffle in parallel, then all $U_2$s, etc. (Note the order of the $U_i$s is irrelevant.) At this point the item in $PE_{(e_N, e_{N-1}, \ldots, e_1)}$ will have been moved to $PE e'_n, e'_{N-1}, \cdots, e'_1$ where each $e'_i$ is $e_i$ rotated left one bit. The items are almost in their correct destination PEs: The low order bit of each $e'_i$ is the high order bit of $e_i$ instead of being the high order bit of $e_{i-1}$ (where $e_0 = e_N$). Since the high order bit of $e_{i-1}$ is the low order bit of $e'_{i-1}$ we need only perform a left rotate on the low order bits of the N $e'_i$s. Consider each group of $2^N$ PEs whose numbers agree in all places except these N bits. Each forms a size $2^q$ cube connected computer: the right and left connections of each $U_i$ allow for the bit complements. By the preceding proposition, the desired permutation, the shuffle, can be realized in N steps.

Thus, the entire 1-D counterpart shuffle can be realized in 2N steps.

**Figure 3**. Simulation of a 1-D Shuffle on a 3-D UC

| | $U_3$ | $U_2$ | $U_1$ |
|---|---|---|---|
| Initial Number: | 0011001 | 001010 | 10110 |
| Current Number after Shuffle in each $U_i$: | 0110010 | 010100 | 01101 |
| Current Number after (LOB of) $U_1$ is Fixed: | 0110010 | 010100 | 01100 |
| Current Number after $U_2$ is Fixed (and destination number): | 0110010 | 010101 | 01100 |

**Corollary** For each N, an N-D UC can simulate its 1-D counterpart with no asymptotic loss of efficiency.

The next two propositions show that, given the way in which N-D UCs are mapped onto 1-D UCs the previous simulations are almost optimal.

**Proposition** An N-D UC cannot simulate a right or left communication of its 1-D counterpart in fewer than N steps.

**Proof** Consider sending an item right from $PE_{P-1}$ to $PE_0$ on an N-D UC. All the bits in the item's initial number are one, but in the destination number they are all zero. In one step, the N-D UC neighbor communication can only be used to change the bits associated with one of its dimensions (i.e. one of its $U_i$s). Since there are N dimensions the simulation must take at least N steps. Q.E.D.

**Proposition** An N-D UC cannot simulate a shuffle/unshuffle connection of its 1-D counterpart in fewer than 2N steps if N is even, or in fewer than 2N-1 steps if N is odd.

**Proof** Consider a PE with the following number: The HOB in $U_i$ is i mod 2, the LOB in $U_i$ is i+1 mod 2, and the remaining bits of $U_i$ keep reversing parity (except for one bit position when log $e_i$ is odd). The I-D counterpart shuffle of an item with such an initial number is, as usual, a left rotate of the bits. As noted in the preceding proof, an N-D UC can only affect the bits in a single dimension in one data communication step. Thus, in general, the N-D UC must perform N independent shuffles to affect all the bit reverses necessary. However, in the number constructed above, one bit will be wrong in each dimension even after these preliminary operations, except for the bit in $d_1$ when N is odd. Thus, when N is even N

additional data communication steps are needed, and when N is odd N-1 steps are needed. Q.E.D.

N-D UCs can be viewed not only as 1-D UCs, but also as intermediate dimensional UCs of a given total size. This use is realized by splitting the N dimensions on an N-D UC into subgroups, and using each group to simulate its 1-D UC counterpart.

**Proposition** For each N, a size P N-D UC can simulate $\frac{1}{M!}\sum_{i=Q}^{M-1}(-1)^i\,(M-i)^N$ size P M-D UCs, $M \le N$, with no asymptotic loss of efficiency.

**Proof** Partition the N dimensions into M nonempty groups. Let each group independently simulate a 1-D UC. Since there are M independent dimensions of 1-D UCs, by definition the entire connection scheme is an M-D UC.

The number of differently configured M-D UCs simulated is just the number of ways of placing the N original dimensions into M nonempty groups [Feller, pp. 60]:

$$\frac{1}{M!}\sum_{i=Q}^{M-1}(-1)^i(M\ -\ i)^N$$

This versatility result also allows various configurations to continue to function even if some PEs are disabled.

**Theorem** Given a size P N-D UC, with $d_1 \ge d_2 \ge \ldots \ge d_N$, $M \le N$, and $B \le \prod_{i=1}^{N-M} d_i$ disabled PEs, then $\prod_{i=1}^{N-M} d_i$ - B M-D UCs can be simulated simultaneously, with no asymptotic loss of efficiency, utilizing P - B $\prod_{i=N-M+1}^{N} d_i$ PEs.

**Proof** By ignoring the connections in the first N-M dimensions, $\prod_{i=1}^{N-M} d_i$ M-D UCs, of dimension $d_{N-M+1} \times \ldots \times d_N$, can be simulated by a fully functional N-D UC. If there are B disabled PEs then at most B of these M-D UCs cannot be used, leaving $\prod_{i=1}^{N-M} d_i$ - B functioning. Each M-D UC utilizes $\prod_{i=N-m+1}^{N} d_i$ PEs. Q.E.D.

In the preceding pages we have just seen that N-D UCs are as powerful as 1-D UCs. Next we go on to demonstrate that the converse is not true, at least for our natural correspondence.

**Theorem** For every given N > 1, any size P 1-D UC requires $\Omega(\log P)$ data communication steps to simulate some N-D UC counterpart.

**Proof** Assume, without loss of generality, that P is an even power of two (i.e. $P = 2^{2j}$). Consider the $\sqrt{P} \times \sqrt{P}$ 2-D UC and the connection between $PE_{(0,0)}$ and $PE_{(1,0)}$. Under our natural correspondence, these PEs correspond to $PE_0$ and $PE_{\sqrt{P}}$ in the 1-D UC. Both PE numbers (0 and $\sqrt{P}$) have binary representations that are completely zero except for a one in the $\sqrt{P}$ bit position of $PE_{(1,0)}$'s number. The fastest method to make this bit position a one, but to leave it surrounded by zeros, is to add one (right send) and then left rotate (shuffle) $\sqrt{P}$ times. Thus, the result holds for N = 2. For every N > 2, this connection also

exists in some N-D UC (i.e. whenever $d_1 \times d_2 \times ... \times d_i = \sqrt{P}$, where $1 \le i \le N$), and the result follows. Q.E.D.

## 4. Extended Multidimensional Ultracomputer Models

In this section we extend the previous N-D UC model, by adding extra connections, to increase efficiency in simulating smaller dimensioned UCs. We require that each PE has at most 4N connections so all connections must be added without compromising this requirement. This can be accomplished by exploiting the redundant shuffle and unshuffle connections in those PEs at the end of a single dimension (e.g., $PE_0$ and $PE_{d_i - 1}$ within $U_i$).

**Model 1** Connections are added to the standard N-D UC to allow the 1-D counterpart right/left communications to be accomplished in one step. A connection is added from $PE_B$ to $PE_{B+1}$, where the binary representation of B has all ones in the low order log ($d_i' d_{i-1}' ... ' d_1$) bits but not in the next log ($d_{i+1}$) low order bits. Thus, an extra connection is added only to some of the PEs which are at the end of a dimension (these have two spare connections in the original model), so that the bound on the number of connections per PE is maintained. When the odd-even PE pairs interact in this model the number of communication steps is reduced from N to 1. Note that N! different 1-D counterparts can be simulated by rearranging the N dimensions but one such configuration must be fixed in advance.

**Model 2** We extend the previous model, again by adding connections, so that for $M \le N$ any of the M dimensions can simulate the right/left connections of its 1-D counterpart in one step. In the previous model we were only concerned with the standard counterpart, here 1-D counterparts to any of the possible M-D UCs ($M \le N$) must meet the requirement.

Recall the simulation of the 1-D counterpart right connection in the original model: Let $d_{i_1}, d_{i_2}, ..., d_{i_M}$ be the M dimensions of a (sub N) - D UC. First each PE sends right on $U_{i_1}$. If a PE has $e_{i_1} = 0$ it sends right again, however this time on $U_{i_2}$. Each PE continues sending right in this fashion until, for some $1 \le k \le M$, $e_{i_k} \ne 0$. Obviously, a send right from a PE with $e_{i_j} = d_{i_j} - 1$ transfers data to a PE with $e_{i_j} = 0$. Thus, in order to handle all possible (sub N) -D UCs, a PE at the tail end of dimension k (i.e. $e_k = d_k - 1$) is connected to the PE which is at the other end of dimension k and also is the right neighbor in some other dimension j (i.e. $PE_{(...e_j...d_k - 1)}$ is connected to $PE_{(...e_{j...0...})}$ where $j \ne k$ and $e_j \ne d_j - 1$). Of course, this is generalized for PEs that are on the tail end of more than one dimension.

As the number of dimension grows, the number of connections on some PEs grows exponentially. Only for $N \le 3$ do all PEs have no more than 4N connections but these are the multidimensional UCs of most practical interest.

## 5. Conclusion

The N-D UC architecture looks promising. We have shown that it can simulate a 1-D UC efficiently. We also believe that N-D UCs can be used to handle certain problems, such as compressible flows, more efficiently than 1-D UCs.

As we have shown, all ultracomputer algorithms are efficiently implementable on an N-D UC. But using the connections of an N-D UC to simulate the 1-D UC connections is only one way of using it: many problems, such as summing and permutation, can be handled by more efficient algorithms tailored to the N-D UC connections.

One issue not addressed in the preceding is asynchronous communication. Up to this point, we have assumed the data communication of the 1-D counterpart to consist of synchronous shuffles, unshuffles, rights, and lefts. When this is not the case, i.e. when the connections in a simulated 1-D ultracomputer are used to accomplish a mixture of shuffles, unshuffles, rights, and lefts in a single cycle simulation is not quite so smooth. An obvious solution is to lengthen each simulation step so that first the shuffle communications are executed, then the unshuffles, then the rights and finally the lefts. Since this can be done, even an 'inhomogeneous' communication step of the kind we have just been considering can be simulated in at most 6N steps.

## Acknowledgements

We thank Allan Gottlieb and Jack Schwartz for reading early drafts and making helpful suggestions. We also thank Lenora Greene for typing the first draft and Connie Engle and Jane Glassey for typing the later draft.

## 6. References

[UC] Jacob Schwartz, "Ultracomputers", to appear in *TOPLAS.*

[Feller] William Feller, *An Introduction to Probability Theory and Its Applications*, v. 1, 3 Ed., J. Wiley & Sons, Inc., New York, 1968.