Spherical Wave Front Generation in Lattice Computers

John Case and Dayanand S. Rajan*
Computer Science Department
103 Smith Hall
University of Delaware
Newark, DE 19716

Anil M. Shende
Department of Mathematics
Haverford College
Haverford, PA 19041

CR Classification Numbers: I.3.5;G.1.0;C.1.2;F.1.1;F.2.2;I.6.1;C.2;J.2;J.7

*Research supported in part by NSF Grant CCR 8713846 at the University of Delaware.
The email address for communication regarding this paper is 'case@cis.udel.edu'.

1 Introduction

In [CRS90a, CRS91a] euclidean n-space is *discretely* represented as a lattice[1] $\mathcal{D}$ and a *bounded* region R of euclidean n-space is represented by $(R \cap \mathcal{D})$, the corresponding *finite* subset of the lattice. In [CRS90a, CRS91a] the mesh computer $\mathcal{M}_{\mathcal{D}}(D)$ associated with subsets D of $\mathcal{D}$ is defined as having its synchronized, identical processors centered on the lattice points of D and with two way, immediate, identical communication links placed between a pair of these processors iff the lattice points corresponding to the pair of processors are at the $\mathcal{D}$-minimal euclidean distance apart.[2] When the context is clear we will drop the subscript from $\mathcal{M}_{\mathcal{D}}(D)$.

Beyond mere representing bounded regions of euclidean n-space by lattice mesh computers, we are generally interested in performing (approximately) linear in real-time, discrete, analogical simulations of the motion of virtual, actual, or geometric objects in those bounded regions of space. For physical phenomena, actually or virtually involving the motion of objects in space, approximately linear in real-time simulations provide an alternative to equation solving in scientific computing. This alternative is also likely superior for real-time control problems. For further discussion of our methodology, architecture considerations, and algorithms see [CRS90b, She91, CRS91b, RS91] and Section 7 below.

Two of our algorithms are the central theme of this paper in which we are concerned with simulating, in lattice mesh computers $\mathcal{M}(D)$, the "physical" propagation of constant speed, euclidean-spherical wave fronts in euclidean n-space. For example, in a homogeneous 3-dimensional medium, the propagation of sound or light from a point source follows such a wave front as does the 2-dimensional surface propagation of water waves induced by dropping a pebble in a still pond. We would like such lattice computer simulations to take place perfectly linear in the real-time for the corresponding physical wave to propagate. Herein we refer to such lattice computer simulations as *wave front broadcasts*.

Our principal results can be summarized as follows.[3]

1. *Perfect* wave front broadcasts (See Section 3 below) are impossible in all but trivial cases of lattice computers $\mathcal{M}(D)$,

2. *nearly perfect* wave front broadcasts (See Section 4 below) are impossible in all (except trivial) *unbounded* lattice computers $\mathcal{M}(D)$,

---

[1] $\mathbf{R}^m$ is (by definition) the set of m-tuples of reals. An *n-dimensional lattice* is (by definition [CS88, GL87]) a set of points $\mathcal{D}$ from $\mathbf{R}^{n'}$, for some $n' \geq n$, such that (i) $\mathcal{D}$ contains at most and at least n points which are linearly independent over the real numbers [Her64], (ii) $\mathcal{D}$ is closed under both vector addition of its points and scalar multiplication [Her64] *by integers*, and (iii) the set of sums of the squares of the coordinates of the non-zero points of $\mathcal{D}$ is bounded below by a non-zero constant. Examples include $\mathbf{Z}^n \stackrel{\text{def}}{=} \{(k_1, \ldots, k_n) \mid k_1, \ldots, k_n \text{ are integers}\}$ and $\mathbf{A}_n \stackrel{\text{def}}{=} \{(k_1, \ldots, k_{n+1}) \mid k_1, \ldots, k_{n+1} \text{ are integers summing to zero}\}$. The *regular* n-dimensional lattices of [CRS90a, CRS91a] are those n-dimensional lattices generated by a set $\mathcal{B}$ of minimal length vectors such that there is a constant angle between any two vectors of $\mathcal{B}$. $\mathbf{Z}^n$ and $\mathbf{A}_n$ are, in fact, n-dimensional *regular* lattices.

[2] In [CRS90a, CRS91a] it is essentially shown that $\mathbf{A}_n$ is the unique up to angle preserving isomorphism *regular* lattice $\mathcal{D}$ so that $\mathcal{M}_{\mathcal{D}}(\mathcal{D})$ has *maximum* near neighbor connectivity. The near neighbor connectivity of $\mathcal{M}(\mathbf{A}_n)$ is $n(n+1)$, that of $\mathcal{M}(\mathbf{Z}^n)$ is but $2n$. [CRS90a, CRS91a] also lists several other advantages of mesh computers based on $\mathbf{A}_n$ over those based on $\mathbf{Z}^n$. Almost all computer meshes built or discussed in the literature are square meshes, i.e., meshes whose underlying sets are finite subsets of $\mathbf{Z}^2$. CAM [TM87] features such a partially simulated mesh with the addition of simulated toroidal connections.

[3] While the methodology we employ herein bears a resemblance to that of cellular automata approaches [Fey82, Min82, FT82, Tof84, TM87, Sny47, Ung58, Mar84, SW86, Mac86, Tof77b, Tof77a, Vic84, Wol83, Zus69, Hil55, Svo86], our philosophy of algorithmic locality is not subject to the same apparent computational limitations.

3. in any reasonable (See Section 2), *bounded* lattice computer $\mathcal{M}(D)$, where D is contained in any one of a *wide* class of lattices, nearly perfect wave front broadcasts *are* possible (See Section 5 below), and

4. in at least the reasonable, bounded lattice computers $\mathcal{M}(D)$, *where D is contained in either of the lattices* $\mathbf{A}_n$ or $\mathbf{Z}^n$, more *physically natural* algorithms than those in Section 5 witnessing Item 3 just above are also available. (See Section 6 below).

It also seems clear to us that our wavefront algorithms could be extended to provide uniform expansion/contraction of geometric figures about a point, but we haven't worked that out in this paper.

2 Notation, Definitions and Assumptions

The major purpose of this section is to specify (Definition 2.5) the class $\mathbf{M}$ of reasonable lattice computers.

Suppose $\mathcal{D}$ is an n-dimensional lattice. Suppose $D \subseteq \mathcal{D}$. $\mathcal{M}(\mathcal{D})$ will sometimes be called the *idealized n-dimensional mesh based on $\mathcal{D}$*.[4] For a point P in $\mathcal{D}$, we write $\mathcal{M}(P)$ for $\mathcal{M}(\{P\})$. $N(\mathcal{D})$ denotes the euclidean minimal distance in $\mathcal{D}$. For the lattice computer $\mathcal{M}(D)$, D is called its *underlying set.*

Definition 2.1 *A lattice $\mathcal{D}$ is said to be* minimally generated *iff there is a basis B for $\mathcal{D}$ so that each vector in B has length equal to $N(\mathcal{D})$.*

Theorem 2.1 *An idealized mesh based on a lattice $\mathcal{D}$ is connected iff $\mathcal{D}$ is a minimally generated lattice.*

□

So we will consider only minimally generated lattices.

Definition 2.2 *Suppose $\mathcal{D}$ is a minimally generated lattice. Suppose P and O are distinct points in $\mathcal{D}$.*

1. *The set of* basic directions in $\mathcal{D}$*, denoted* $\mathrm{Dirs}(\mathcal{D})$*, is the set of minimal length vectors in $\mathcal{D}$.*

2. *The* nearest neighbors of P*, denoted* $\mathrm{Nbrs}_{\mathcal{D}}(P)$*, is the set of points Q, in $\mathcal{D}$, so that* $(Q - P) \in \mathrm{Dirs}(\mathcal{D})$*.*

3. *The* Graphical Path Length of P with respect to O*, denoted* $\mathrm{GraphPL}_{\mathcal{D}}(P, O)$*, is defined as:*

$$\mathrm{GraphPL}_{\mathcal{D}}(P, O) = \begin{cases} 0, & \textit{if } P = O; \\ 1 + \min\{\mathrm{GraphPL}_{\mathcal{D}}(Q, O) | Q \in \mathrm{Nbrs}_{\mathcal{D}}(P)\}, & \textit{otherwise.} \end{cases}$$

4. *N is an* Inner Neighbor of P with respect to O in $\mathcal{D}$*, iff* $N \in \mathrm{Nbrs}_{\mathcal{D}}(P)$*, and*

$$\mathrm{GraphPL}_{\mathcal{D}}(N, O) = \mathrm{GraphPL}_{\mathcal{D}}(P, O) - 1.$$

$\mathit{Inner}_{\mathcal{D}}(P, O)$ is the set of all points N, such that N is an inner neighbor of P with respect to O in $\mathcal{D}$.

[4] The construction of $\mathcal{M}(\mathcal{D})$ described in the previous Section as taking place in n-dimension euclidean space could, without loss of generality, also have been described as taking place in 3-space, where we *index* the processors by points in $\mathcal{D}$ and make equal euclidean length (possibly bent) connections between appropriate processors.

5. *N is an* Outer Neighbor of P with respect to O in $\mathcal{D}$, *iff* $N \in \mathrm{Nbrs}_{\mathcal{D}}(P)$, and

$$\mathrm{GraphPL}_{\mathcal{D}}(N, O) = \mathrm{GraphPL}_{\mathcal{D}}(P, O) + 1.$$

   $Outer_{\mathcal{D}}(P, O)$ is the set of all points N, such that N is an outer neighbor of P with respect to O in $\mathcal{D}$.

6. *A sequence of points $(P =)P_0, P_1, \ldots P_m(= Q)$, $P_i \in \mathcal{D}, 0 \leq i \leq m$, is a* shortest path from P to Q in $\mathcal{D}$, *iff* $\forall i, 0 < i \leq m, P_{i-1} \in Inner_{\mathcal{D}}(P_i, P)$.
   N.B. $\mathrm{GraphPL}_{\mathcal{D}}(P, Q) = m = \mathrm{GraphPL}_{\mathcal{D}}(Q, P)$.

7. *D is* convex with respect to $\mathcal{D}$ *iff every shortest path, in $\mathcal{D}$, between any two points in D is contained in D.*

8. *For any point P in D we say that D is P-*convex with respect to $\mathcal{D}$ *iff for every point Q in D,*

$$Inner_{\mathcal{D}}(Q, P) \subseteq D.$$

The following lemma characterizes subsets D of $\mathcal{D}$ which are convex with respect to $\mathcal{D}$ for minimally generated lattices $\mathcal{D}$.

Lemma 2.1 *Suppose $\mathcal{D}$ is a minimally generated lattice, and $D \subseteq \mathcal{D}$. Then D is convex with respect to $\mathcal{D}$ iff, for each point P in D, D is P-convex with respect to $\mathcal{D}$.*

$\square$

Definition 2.3 *Let D be a subset of a minimally generated lattice $\mathcal{D}$.*

1. *$d(P, Q)$ denotes the* euclidean *distance between points P and Q.*

2. *D is* non-trivial *iff there are three distinct non-collinear points, P, Q, and O, in D so that $d(P, O) \neq d(Q, O)$.*

3. *The* diameter of D, *denoted* $\mathrm{Diam}(D)$, *is the maximum, over pairs, (P, Q), of points in D, of $\lceil d(P, Q) \rceil$. (N.B.* $\mathrm{Diam}(D)$ *is always an integer.)*

4. *The* norm *of P with respect to O, denoted* $\mathrm{Norm}(P, O)$, *is the square of the euclidean distance between P and O. The* difference in norms *between two points P and Q in $\mathcal{D}$ with respect to O, denoted* $\mathrm{NormDiff}(P, Q, O)$, *is given by*
   $\mathrm{NormDiff}(P, Q, O) = |\mathrm{Norm}(P, O) - \mathrm{Norm}(Q, O)|$.

In the interests of both simplicity of the individual processors in a lattice computer and the avoidance of error propagation, we would like to limit our calculations to integers. As will be seen below, coordinates of points in the underlying set of a lattice computer, and vectors and vector inner-product in the lattice containing the underlying set are ubiquitous in our algorithms.

Definition 2.4 *An n-dimensional lattice $\mathcal{D}$ is said to be* integrally representable *iff there is an $n' \geq n$ so that $\mathcal{D} \subseteq \mathbf{Z}^{n'}$. $\vec{0}$ denotes the origin of $\mathbf{Z}^{n'}$ and is always contained in $\mathcal{D}$.*

$\mathbf{A}_n$ and $\mathbf{Z}^n$ are clearly example minimally-generated *and* integrally-representable n-dimensional lattices.

Now we define the class of lattice computers $\mathbf{M}$ that we are interested in.

Definition 2.5 **M** *consists of all those lattice computers* $\mathcal{M}_{\mathcal{D}}(D)$ *based on* $D \subseteq \mathcal{D}$ *where*

1. $\mathcal{D}$ *is an* n*-dimensional minimally generated, integrally representable lattice and* $n > 1$,
2. D *is non-trivial, and*
3. D *is convex with respect to* $\mathcal{D}$,

and, furthermore, $\mathcal{M}(D) = \mathcal{M}_{\mathcal{D}}(D)$ *satisfies Assumption 2.1 just below.*

Assumption 2.1

1. *All communication channels in* $\mathcal{M}(D)$ *are identical in their abilities to carry data between processors. We assume that all communication channels are capable of carrying the same amount of data in the same amount of time.*
2. *There is a common, finite set of basic instructions that every processor in* $\mathcal{M}(D)$ *can execute. These are the only basic instructions that any processor in* $\mathcal{M}(D)$ *can execute.*
3. *The actual time to execute the same basic instruction by any processor in* $\mathcal{M}(D)$ *is a constant. By* instruction cycle of a processor *we mean the minimum possible time interval within which any single instruction can be fully executed by any processor. Hence, in each processor, each basic instruction takes the same instruction cycle amount of time to execute.*
4. *All processors in* $\mathcal{M}(D)$ *execute basic instructions synchronously, i.e., instruction cycles of all processors share a same, actual time interval. This discretizes time, for the lattice computer as a whole, into intervals of equal length.* `SimClock` *is the instruction counter for each* $\mathcal{M}(P)$.
5. *We assume that we have an instruction of the form*

$$\textbf{Schedule } \langle\langle message\rangle, \langle neighbor\rangle\rangle \textbf{ for } \texttt{SimClock} = \langle time\rangle$$

This instruction is for scheduling *the sending of messages to neighbors and is executed correctly provided the value of* $\langle time\rangle$ *is greater than the number of instructions executed prior to executing this instruction. Upon execution of this scheduling instruction the pair* $\langle\langle message\rangle, \langle neighbor\rangle\rangle$ *is included in the set* $SendList(\langle time\rangle)$ *of such pairs. Here is how each processor uses* its *set* $SendList(\langle time\rangle)$. *Completely within the* $\langle time\rangle$*-th instruction cycle, for* each $\langle\langle message\rangle, \langle neighbor\rangle\rangle \in SendList(\langle time\rangle)$, *the processor sends* $\langle message\rangle$ *to* $\langle neighbor\rangle$ *(if it exists in* $\mathcal{M}(D)$*), and* $\langle neighbor\rangle$ *(if it exists in* $\mathcal{M}(D)$*) immediately* receives $\langle message\rangle$. *It is understood that a neighbor of the processor cannot* read *such a message received from that processor* until *in the* $\langle time\rangle + 1$*-st instruction cycle. It is further understood that the processor need execute no explicit instructions in addition to the scheduling instruction for the resultant, eventual sending to take place.*

3 Perfect Wave Front Broadcast

In this paper, when we speak of an algorithm executing in a lattice computer, we will mean that each processor in the lattice computer executes a copy of this algorithm; all processors in the lattice computer start executing the algorithm at the same time. In any execution of an algorithm in $\mathcal{M}_{\mathcal{D}}(D)$, one processor, called the `Initiator`, is *initiated* by input external to the lattice computer. All other processors can get input only as messages from their neighbors in the lattice computer. For any processor $\mathcal{M}_{\mathcal{D}}(P)$, other than $\mathcal{M}_{\mathcal{D}}(I)$, the `Initiator`, in the lattice computer, $t(P)$ will denote the number of instruction cycles elapsed, since the beginning of the execution of the algorithm, just before $\mathcal{M}_{\mathcal{D}}(P)$ gets its first message. $t(I)$, by definition, equals 0.

Definition 3.1 *Suppose $\mathcal{D}$ is an n-dimensional lattice, and $D \subseteq \mathcal{D}$. An algorithm is said to be a* perfect wave front broadcast in $\mathcal{M}_{\mathcal{D}}(D)$ *iff, when it's initiated in any processor $\mathcal{M}_{\mathcal{D}}(I)$ in $\mathcal{M}_{\mathcal{D}}(D)$,*

1. *each processor, other than $\mathcal{M}_{\mathcal{D}}(I)$, in $\mathcal{M}_{\mathcal{D}}(D)$ eventually gets a message, and*
2. *there is a real, positive constant c so that for each processor $\mathcal{M}_{\mathcal{D}}(P)$, other than $\mathcal{M}_{\mathcal{D}}(I)$,*
$$t(P) = c \cdot d(P, I).$$

Theorem 3.1 *In any n-dimensional lattice $\mathcal{D}$, $n > 1$, there are points P and Q so that $d(P, Q)$ is an irrational multiple of $N(\mathcal{D})$.*

□

Then, the following theorem is an immediate corollary of Theorem 3.1 since we are counting time as integer multiples of an instruction cycle.

Theorem 3.2 *For any n-dimensional lattice $\mathcal{D}$, $n > 1$, for almost all $D \subseteq \mathcal{D}$, so that D cannot be embedded in any one-dimensional lattice, there is no perfect wave front broadcast in $\mathcal{M}_{\mathcal{D}}(D)$.*

□

4 Nearly Perfect Wave Front Broadcast

Since it is not possible to have a perfect wave front broadcast in most lattice computers, we relax the timing requirement of a perfect wave front broadcast in a nearly perfect wave front broadcast.

Definition 4.1 *Suppose $\mathcal{D}$ is an n-dimensional lattice, and $D \subseteq \mathcal{D}$. An algorithm is said to be a* nearly perfect wave front broadcast in $\mathcal{M}_{\mathcal{D}}(D)$ *iff when initiated in any processor $\mathcal{M}_{\mathcal{D}}(I)$ in $\mathcal{M}_{\mathcal{D}}(D)$,*

1. *each processor, other than $\mathcal{M}_{\mathcal{D}}(I)$, in $\mathcal{M}_{\mathcal{D}}(D)$ eventually gets a message,*
2. *for any two processors, $\mathcal{M}_{\mathcal{D}}(P)$ and $\mathcal{M}_{\mathcal{D}}(Q)$, other than $\mathcal{M}_{\mathcal{D}}(I)$, if $d(P, I) < d(Q, I)$, then $t(P) < t(Q)$, and*
3. *there are real, positive constants c and ϵ so that for each processor $\mathcal{M}_{\mathcal{D}}(P)$, other than $\mathcal{M}_{\mathcal{D}}(I)$,*
$$|t(P) - c \cdot d(P, I)| < \epsilon.$$

Suppose $\mathcal{M}_{\mathcal{D}}(I)$ is the source of a broadcast in a lattice computer $\mathcal{M}_{\mathcal{D}}(D)$, $D \subseteq \mathcal{D}$, an n-dimensional lattice. For any two neighboring processors, $\mathcal{M}_{\mathcal{D}}(P)$ and $\mathcal{M}_{\mathcal{D}}(Q)$, in the lattice computer, such that $d(P, I) \neq d(Q, I)$, $t(P)$ and $t(Q)$ must differ at least by 1 in a nearly perfect wave front broadcast since t is integer valued.

Definition 4.2 *Suppose $\mathcal{D}$ is a lattice.*

1. *For any three points P, Q and R in $\mathcal{D}$,*

$$\delta_{\mathcal{D}}(P, Q, R) = |d(Q, P) - d(R, P)|.$$

2. *For each real number $r > 0$,*

$$\Delta_{\mathcal{D}}(r) = \min\{\ \delta_{\mathcal{D}}(P, Q, R) \mid P, Q, R \in \mathcal{D},\ \text{ and } r \geq d(Q, P) \neq d(R, P) \leq r\ \}.$$

3. *Suppose D is a finite, non-trivial subset of $\mathcal{D}$. Then,*

$$\Delta(D) = \Delta_{\mathcal{D}}(\mathrm{Diam}(D)).$$

Lemma 4.1 *Suppose $\mathcal{D}$ is an n-dimensional, $n > 1$, lattice. Then,*

$$(\forall r > 0)\left[\Delta_{\mathcal{D}}(r) > \frac{N(\mathcal{D})}{2r}\right].$$

□

Then, since the squares of all distances between lattice points in an integrally representable lattice are integers, the following two corollaries are immediate.

Corollary 4.1 *Suppose $\mathcal{D}$ is an n-dimensional, $n > 1$, integrally representable lattice. Then,*

$$(\forall r > 0)\left[\Delta_{\mathcal{D}}(r) > \frac{1}{2r}\right].$$

□

Corollary 4.2 *Suppose $\mathcal{D}$ is an n-dimensional, $n > 1$, integrally representable lattice, and D is a finite, non-trivial subset of $\mathcal{D}$. Then,*

$$\Delta(D) > \frac{1}{2\mathrm{Diam}(D)}.$$

□

This corollary (Corollary 4.2) explains the presence of $2\mathrm{Diam}(D)$ in our algorithms and in Fact 5.6.

Then we have the following theorem, whose proof is given by Algorithm Broadcast (and the remarks following it) presented in Section 5 just below.

Theorem 4.1 *For each $\mathcal{M}_{\mathcal{D}}(D)$ in $\mathbf{M}$, so that D is finite, there is a nearly perfect wave front broadcast in $\mathcal{M}_{\mathcal{D}}(D)$.*

□

5 **Algorithm** Broadcast

We present this algorithm as if it were being executed by a processor $\mathcal{M}_{\mathcal{D}}(P)$ in a lattice computer, $\mathcal{M}_{\mathcal{D}}(D)$ in $\mathbf{M}$. Let n' be the smallest integer so that $\mathcal{D}$ is a subset of $\mathbf{Z}^{n'}$.

The variables *SendDirs* and *InDirs* are used to store the set of basic directions to send messages in, and the set of basic directions from which messages were received, respectively. We assume that the set *InDirs* is automatically updated whenever the processor receives a message. With each basic direction, we associate a buffer, denoted $Buf\langle B\rangle$, which contains three fields: *InMessage*, *OutMessage*, and *SendTime*. A message, in this algorithm, is an n'-tuple of integers. At any time during the execution of Algorithm Broadcast in $\mathcal{M}_{\mathcal{D}}(P)$, for any basic direction B, the field *InMessage* contains only the message received, if any, from the processor in that basic direction; similarly the field *OutMessage* is used to store the message to be sent, if any, to the processor in that basic direction; the field *SendTime* is used to store the time at which to send a message, if any, to the processor in that basic direction. Until $\mathcal{M}(P)$ schedules a message to be sent at $\langle time\rangle$, SendList $(\langle time\rangle) = \emptyset$. *Self* is a variable that stores the message received from a processor in any direction (messages received from processors in all basic directions in *InDirs* are identical; see Fact 5.1 below). `CompLen` is the maximum amount of time required to be by any processor in executing Steps 2 and 3. As can be seen from Fact 5.3 below, the value of `CompLen` *can* be computed and stored as a constant prior to the execution of the algorithm!

begin Broadcast {Clarifications of steps follow algorithm.}

1. Wait for external input or a message;
2. Compute *SendDirs*;

   for each $B \in SendDirs$,

   Compute $Buf\langle B\rangle.OutMessage$;

   Compute $Buf\langle B\rangle.SendTime$;

   endfor;
3. **for each** $B \in SendDirs$,

   Schedule $\langle Buf\langle B\rangle.OutMessage, \mathcal{M}(P+B)\rangle$ **for**

   `SimClock` $=$ `CompLen` $\cdot\, Buf\langle B\rangle.SendTime$;

   endfor;

end Broadcast {Clarifications of steps follow.}

As can be seen from the algorithm described above, the steps that need to be clarified are:

1. computing *SendDirs*,
2. for each basic direction, B, in *SendDirs*, computing $Buf\langle B\rangle.OutMessage$, and
3. for each basic direction, B, in *SendDirs*, computing $Buf\langle B\rangle.SendTime$.

We first make these clarifications, and then note some facts about this algorithm.
$\mathcal{M}(I)$, the initiator, computes the three quantities mentioned just above as

$$SendDirs := \mathrm{Dirs}(\mathcal{D});$$

for each $B \in \mathrm{Dirs}(\mathcal{D})$,

$\mathit{Buf}\langle B\rangle.\mathit{OutMessage} := B$;

$\mathit{Buf}\langle B\rangle.\mathit{SendTime} := \texttt{CompLen} \cdot (2 \cdot \mathrm{Diam}(D))$;

endfor;

Each processor, other than the initiator, in the lattice computer, upon getting messages from processors in basic directions given by *InDirs* computes the three quantities as

$\mathit{Self} := \mathit{Buf}\langle B\rangle.\mathit{InMessage},\ B \in \mathit{InDirs}$;

$\mathit{SendDirs} := \{\ B \in \mathrm{Dirs}(\mathcal{D}) \mid d(\mathit{Self} + B, \vec{0}) > d(\mathit{Self}, \vec{0})\ \}$;

for each $B \in \mathit{SendDirs}(\mathcal{D})$,

$\mathit{Buf}\langle B\rangle.\mathit{OutMessage} := (\mathit{Self} + B)$;

$$\mathit{Buf}\langle B\rangle.\mathit{SendTime} := \texttt{CompLen}\cdot \left[\mathrm{rnd}\left(\frac{2 \cdot \mathrm{Diam}(D) \cdot d((\mathit{Self} + B), \vec{0})}{N(\mathcal{D})}\right)\right];$$

endfor; {rnd is the rounding function.}

We note the following facts about Algorithm Broadcast. If processor $\mathcal{M}_{\mathcal{D}}(P)$ sends a message to one of its neighbors, $\mathcal{M}_{\mathcal{D}}(P+B)$, $B \in \mathrm{Dirs}(\mathcal{D})$, at time $T = \mathit{Buf}\langle B\rangle.\mathit{SendTime}$, then $t(P+B) = T$. The verification of each of these facts is relatively straightforward. Facts 5.4, 5.6 and 5.7 witness the correctness of Algorithm Broadcast as a nearly perfect wave front broadcast in $\mathcal{M}_{\mathcal{D}}(D)$, and, hence, we have the proof of Theorem 4.1.

Fact 5.1 *Each processor $\mathcal{M}(P)$ receives as its message, from a processor in any of the basic directions in InDirs, the coordinates of the vector $(P - I)$.*

Fact 5.2 *All messages that are ever sent to $\mathcal{M}(P)$ are sent at the same time.*

Fact 5.3 *Each* step *in Algorithm* Broadcast*, except for Step 1, can be executed in a bounded amount of time. Furthermore, these bounds are functions of $\mathcal{D}$ and D, and, hence, they can be determined prior to the execution of the algorithm.*

Fact 5.4 *Each processor in $\mathcal{M}(D)$ does get a message.*

Fact 5.5 *Since $\mathcal{D}$ is an integrally representable lattice, $N(\mathcal{D})$ and $d((\mathit{Self}+B), \vec{0})$ are square roots of integers. Thus the computation of SendTime for each basic direction in SendDirs can be performed using integers only.*

Fact 5.6 *Algorithm* Broadcast *satisfies condition (3) of Definition 4.1 with*

$$\begin{aligned} c &= (2 \cdot \texttt{CompLen} \cdot \mathrm{Diam}(D))/N(D), \text{ and} \\ \epsilon &= \texttt{CompLen}/2. \end{aligned}$$

Fact 5.7 *From Fact 5.6 and since the minimal distance between any two points in D is $N(D)$, Algorithm* Broadcast *satisfies Definition 4.1.*

6 Physically Natural Nearly Perfect Wave Front Broadcast

In Algorithm Broadcast, presented in Section 5 above, each processor passes as a message, to each processor in a subset of its neighboring processors, the relative coordinates *with respect to the origin of the wave front broadcast* of the neighboring processor. In *physical* reality, it seems unlikely that a point in Euclidean space "finds out" about its relative coordinates with respect to the origin of a wave. For example, it seems unlikely that a water molecule, on, say, the surface of a lake, "knows" about the origin of an impending wave as it is being affected by that wave. It seems more likely that the point is activated by impulses (informationless, except for magnitude, type, and direction) from neighboring points and these impulses cause a point to be on a wave front at the appropriate time.

The algorithms HexBroadcast and SqBroadcast, presented in Sections 6.1 and 6.2 below, do not have this physically implausible characteristic. In these algorithms, every processor, after receiving a "piece" of wave front, uses the *set of basic directions* (Definition 2.2 above) from which it *received* the piece of wave front in "deciding" which of its neighbors to send the piece of wave front on to. Intuitively, then, one can think of the pieces of wave front passed to the processor as a "push" by the wave emanating from origin. This seems more physically natural than the mechanism used in Broadcast. We discuss a bit more the possible physics of our algorithms in Section 6.3 below.

6.1 Algorithm HexBroadcast

This algorithm is meant to run on lattice computers based on the lattice $\mathbf{A}_n$. This algorithm is similar to Algorithm Broadcast presented in Section 5 above; the *only* differences lie in the method of computing *SendDirs*, the messages that are sent, the method of computing the messages, the method of computing *SendTime* for each processor corresponding to a basic direction in *SendDirs*, and, of course, the actual value of `CompLen`. In Sections 6.1.1, 6.1.2, and 6.1.3 we elaborate upon these computations. We will present the computations as if the algorithm were running in $\mathcal{M}_{\mathbf{A}_n}(A)$ in $\mathbf{M}$. In Section 6.1.5 we show that all the computations involved in carrying out Step 2 of Algorithm HexBroadcast (a suitably modified refinement of Algorithm Broadcast) can be executed in a bounded amount of time and that Algorithm HexBroadcast is a nearly perfect wave front broadcast in $\mathcal{M}_{\mathbf{A}_n}(A)$. We also show that the computations performed, in this algorithm, by any processor $\mathcal{M}_{\mathbf{A}_n}(P)$ do not depend on the coordinates, absolute or relative to the source of the broadcast, of P. (Henceforth, up to Section 6.2, we will be dealing with lattice computers whose underlying set is a subset of $\mathbf{A}_n$. So, as usual, we will drop the subscript $\mathbf{A}_n$.)

Definition 6.1

1. *For each $n \geq 1$, for each pair $\langle i,j\rangle, 0 \leq i \neq j \leq n$, we define λ^n_{ij} to be the $(n+1)$-tuple, $(l_0, l_1, \ldots, l_n)$ so that*
   (a) $l_i = 1$,
   (b) $l_j = -1$, *and*
   (c) $(\forall k \notin \{i,j\}, 0 \leq k \leq n)[l_k = 0]$.
2. *For each $n \geq 1$, we define Λ^n to be the set of all λ^n_{ij}.*

The following lemma follows from the definitions of Λ^n, $\mathbf{A}_n$ and the basic directions in a lattice.

Lemma 6.1

$$Dirs(\mathbf{A}_n) = \Lambda^n.$$

$\square$

We now elaborate upon the computation performed by a processor, $\mathcal{M}(P)$, running Algorithm HexBroadcast.

6.1.1 Computing *SendDirs*

$\mathcal{M}(I)$ sets *SendDirs* to be $\mathrm{Dirs}(\mathbf{A}_n)$.
Each processor, $\mathcal{M}(P)$, other than $\mathcal{M}(I)$, computes *SendDirs* to be the set of basic directions

$$\left[\bigcap_{B \in InDirs} \{B' \in \mathrm{Dirs}(\mathbf{A}_n) \mid (B, B') \leq 0\}\right]$$

6.1.2 Computing $Buf\langle B\rangle.OutMessage$

Each processor, $\mathcal{M}(P)$, sends a 3-tuple of integers, *DSq*, *NormDiff*, and *GPL* as a message to each neighbor $\mathcal{M}(P+B), B$ in *SendDirs*. The messages sent to different neighboring processors could be different.
$\mathcal{M}(I)$ sets the values of *DSq*, *NormDiff*, and *GPL*, in the messages to each of its neighbors to be $2, 2$, and 1, respectively.
Each processor, $\mathcal{M}(P)$, other than $\mathcal{M}(I)$ sets the values of messages to be sent as follows: Let D_P denote the value of *DSq* received as part of the message from any $\mathcal{M}(P+B'), B'$ in *InDirs*. (As shall be seen in Section 6.1.5 below, the value of *DSq* (and *GPL*, respectively) received from all neighbors that sent a message to $\mathcal{M}(P)$ is the same.)
Let G_P denote the value of *GPL* received as part of the message from any $\mathcal{M}(P+B'), B'$ in *InDirs*.
Compute the sets *Opp*, *Rt*, and *Obtuse* given by

$$\begin{aligned} Opp &= \{B \in \mathrm{Dirs}(\mathbf{A}_n) \mid (-B) \in InDirs\ \} \\ Rt &= \{B \in \mathrm{Dirs}(\mathbf{A}_n) \mid \forall B' \in InDirs, (B, B') = 0\} \\ Obtuse &= SendDirs \setminus (Opp \cup Rt) \end{aligned}$$

For each $B \in Obtuse$, compute

$$\begin{aligned} Ob_B &= \{B' \in InDirs \ \mid\ (B, B') = -1\}, \text{ and} \\ S_B &= \sum_{B' \in Ob_B} Buf\langle B'\rangle.InMessage.NormDiff. \end{aligned}$$

Then, for each B in *SendDirs*,

1. $Buf\langle B\rangle.OutMessage.NormDiff$ is assigned the value
   (a) $Buf\langle -B\rangle.InMessage.NormDiff + 4$, if $B \in Opp$,
   (b) 2, if $B \in Rt$, and

(c) $\dfrac{S_B - 2 \cdot G_P}{|Ob_B|}$, if $B \in \mathit{Obtuse}$.

2. $\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{DSq} := D_P +$ $\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{NormDiff}$, and

3. $\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{GPL} := G_P + 1$.

6.1.3 Computing $\mathit{Buf}\langle B\rangle.\mathit{SendTime}$

For each basic direction B, $\mathcal{M}(P)$ computes $\mathit{Buf}\langle B\rangle.\mathit{SendTime}$ to be the unique (see Lemma 6.8 below) integer that is closest to

$$\frac{2 \cdot \mathrm{Diam}(A) \cdot \sqrt{D_P}}{\sqrt{2}}.$$

6.1.4 Technical Lemmas

Lemma 6.2 *Suppose I is in $\mathbf{A}_n$. Then the graphical path length of any point P, in $\mathbf{A}_n$, from I is given by*

$$\mathrm{GraphPL}(P, I) = \sum_{(P-I)_i > 0} (P - I)_i = - \sum_{(P-I)_i < 0} (P - I)_i.$$

Proof. Follows from the definitions of $\mathbf{A}_n$ and the graphical path length between points in $\mathbf{A}_n$. □

Lemma 6.3 *Suppose I is in $\mathbf{A}_n$. Then, for any point P, in $\mathbf{A}_n$,*

$$\begin{aligned} \lambda_{ij} \in \mathit{Inner}(P, I) &\iff [((P - I)_i < 0)\ \&\ ((P - I)_j > 0)], \ \textit{and} \\ \lambda_{ij} \in \mathit{Outer}(P, I) &\iff [((P - I)_i \geq 0)\ \&\ ((P - I)_j \leq 0)]. \end{aligned}$$

Proof. Follows from the definitions of Inner and Outer, and Lemma 6.2. □

Lemma 6.4 *Suppose I in $\mathbf{A}_n$. Then, for any P in $\mathbf{A}_n$,*

$$\mathit{Outer}(P, I) = \left[\bigcap_{B \in \mathit{Inner}(P,I)} \{B' \in \mathrm{Dirs}(\mathbf{A}_n) \mid (B, B') \leq 0\} \right].$$

Proof. This follows from Lemmas 6.1 and 6.3 and a case-by-case analysis of all possible basic directions. □

Lemma 6.5 *The euclidean distance between any two points in $\mathbf{A}_n$ is either an integer, or an irrational number.*

Proof. Follows from the fact that $\mathbf{A}_n$ is an integrally representable lattice. □

Lemma 6.6 *Suppose I is in $\mathbf{A}_n$. For any point $P \neq I$ in $\mathbf{A}_n$, define the following sets.*

$$\begin{aligned}
Op(P,I) &\stackrel{\text{def}}{=} \{\ B \in \mathrm{Dirs}(\mathbf{A}_n) \mid (-B) \in \mathit{Inner}(P,I)\ \},\\
Rt(P,I) &\stackrel{\text{def}}{=} \{\ B \in \mathrm{Dirs}(\mathbf{A}_n) \mid (\forall B' \in \mathit{Inner}(P,I))\ (B,B') = 0\ \},\ \textit{and}\\
Ob(P,I) &\stackrel{\text{def}}{=} \mathit{Outer}(P,I) \backslash (Op(P,I) \cup Rt(P,I)).
\end{aligned}$$

Also, for each $B \in Ob(P,I)$ let

$$\begin{aligned}
Ob_B(P,I) &\stackrel{\text{def}}{=} \{B' \in \mathit{Inner}(P,I) \mid (B,B') = -1\ \},\ \textit{and}\\
S_B &\stackrel{\text{def}}{=} \sum_{B' \in Ob_B(P,I)} \mathrm{NormDiff}(P,(P+B'),I).
\end{aligned}$$

Then,

1. $\forall B \in Ob(P,I)$,
$$\mathrm{NormDiff}((P+B),P,I) = \mathrm{NormDiff}(P,(P-B),I) + 4.$$
2. $\forall B \in Rt(P,I)$, $\mathrm{NormDiff}((P+B),P,I) = 2$.
3. $\forall B \in Ob(P,I)$,
$$\mathrm{NormDiff}((P+B),P,I) = \frac{S_B(P,I) - 2 \cdot GPL(P,I)}{|Ob_B(P,I)|}.$$

PROOF. Follows from Definition 2.2, Lemma 6.1, and the fact that $\mathbf{A}_n$ is a subset of $\mathbf{Z}^{n+1}$. □

6.1.5 Algorithm HexBroadcast *is* a Nearly Perfect Wave Front Broadcast

Lemma 6.7 *The computations in Sections 6.1.1, 6.1.2, and 6.1.3 can be performed, using integers only, in a bounded amount of time, a function of $\mathbf{A}_n$ and A.*

PROOF. Clearly, the computations in Sections 6.1.1 and 6.1.2 can be performed in time bounded above by a function of the number of basic directions in $\mathbf{A}_n$.
Lemma 4.1 implies that for each B in *SendDirs*, *SendTime* can be computed in time bounded above by a function of $\mathrm{Diam}(A)$, a constant with respect to A. Therefore, the computation in Section 6.1.3 can be performed in a bounded amount of time, a function of $\mathbf{A}_n$ and A.
Computations in Sections 6.1.1 and 6.1.2 can be clearly computed using integers only. Furthermore, since D_P, appearing in the displayed expression in Section 6.1.3, is an integer, the computations in Section 6.1.3 can be performed using integers only in a bounded amount of time, a function of A. □

Lemma 6.8 *For each B in SendDirs, SendTime is a unique integer.*

PROOF. Follows from Lemma 6.5 and the expression for computing *SendTime*. □

Theorem 6.1 *If Algorithm* HexBroadcast *is initiated at processor* $\mathcal{M}(I)$ *in* $\mathcal{M}(A)$*, then for each processor* $\mathcal{M}(P)$*,*

1. $\mathit{InDirs} = \mathit{Inner}(P, I)$,
2. $\mathit{SendDirs} = \mathit{Outer}(P, I)$,
3. $(\forall B \in \mathit{SendDirs})[\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{GPL} = \mathrm{GraphPL}((P+B), I)]$, *and*
4. $(\forall B \in \mathit{SendDirs})[D_P + \mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{DSq} = (d((P+B), I))^2]$.

PROOF. Parts (1) and (2) of the theorem can be proved by an induction on the graphical path length of P with respect to I using Lemma 6.4 and the convexity (see Definition 2.3) of A with respect to $\mathbf{A}_n$.
Part (3) can be proved by considering, for each B in *SendDirs*, which set, *Opp*, *Rt*, or *Obtuse*, B belongs to, and then using Lemmas 6.2 and 6.3, and the expression for computing $\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{NormDiff}$. □

Then, we have the following three corollaries. The first two are immediate consequences of Theorem 6.1 above. We sketch the proof of the third one.

Corollary 6.1 *If Algorithm* HexBroadcast *is initiated at processor* $\mathcal{M}(I)$ *in* $\mathcal{M}(A)$*, then for each processor* $\mathcal{M}(P)$*,*

$$(\forall B \in \mathit{SendDirs})[\mathit{Buf}\langle B\rangle.\mathit{OutMessage}.\mathit{NormDiff} = \mathrm{NormDiff}(P, (P+B), I)].$$

□

Corollary 6.2 *Each processor, other than the source of the broadcast, receives the same values in DSq and GPL, respectively, in all its messages.* □

Corollary 6.3 *Algorithm* HexBroadcast *is a nearly perfect wave front broadcast in* $\mathcal{M}(A)$.

PROOF. Algorithm HexBroadcast trivially satisfies Condition (1) in the definition of a wave front broadcast.
A simple induction on the graphical path length of processors from the source of the broadcast, together with Theorem 6.1 and the convexity (see Definition 2.3) of A in $\mathbf{A}_n$, helps us show that messages reach every processor in the lattice computer, i.e., Condition (3).
Further, it can be verified that with the values of α and ϵ given by

$$\begin{aligned} \alpha &= (2 \cdot \mathtt{CompLen} \cdot \mathrm{Diam}(A))/N(D), \text{ and} \\ \epsilon &= \mathtt{CompLen}/2, \end{aligned}$$

condition (4), and, hence, condition (3) of a wave front broadcast (see Definition 4.1) are satisfied by Algorithm HexBroadcast, thus showing that Algorithm HexBroadcast is a nearly perfect wave front broadcast. □

Having shown that Algorithm HexBroadcast is, in fact, a nearly perfect wave front broadcast in $\mathcal{M}(A)$, we note an important fact about this algorithm in the following theorem.

Theorem 6.2 *For each finite, convex subset A of $\mathbf{A}_n$, there is a nearly perfect wave front broadcast where the relative (with respect to the source of the broadcast) coordinates of points are not sent as messages.* □

6.2 Algorithm SqBroadcast

Suppose Z is a finite subset of $\mathbf{Z}^n$ convex with respect to $\mathbf{Z}^n$. It is fairly easy to verify that Lemmas 6.4, 6.5, and 6.6 above are true if $\mathbf{A}_n$ is replaced by $\mathbf{Z}^n$ and A is replaced by Z in the statements of the lemmas.

Next we note that if we compute *SendDirs* to be

$$\left\lfloor \bigcap_{B \in InDirs} \{B' \in \mathrm{Dirs}(\mathbf{Z}^n) \mid (B, B') \leq 0\} \right\rfloor,$$

and define sets *Opp*, *Rt*, and *Obtuse* for $\mathbf{Z}^n$ (similar to Section 6.1.2), then the set *Obtuse* is always empty. Therefore, $SendDirs = Opp \cup Rt$.

Finally we specify the computations of *NormDiff* and *DSq* for each B in *SendDirs* as (we use the same notation as in Section 6.1.2 above):

1. $Buf\langle B\rangle.OutMessage.NormDiff$ is assigned the value
   (a) $Buf\langle -B\rangle.InMessage.NormDiff + 2$, if $B \in Opp$, and
   (b) 1, if $B \in Rt$.
2. $Buf\langle B\rangle.OutMessage.DSq := D_P + Buf\langle B\rangle.OutMessage.NormDiff$.

Algorithm SqBroadcast is, then, essentially Algorithm HexBroadcast, with the modifications mentioned above.

It is easy to verify that all the results in Section 6.1.5, replacing $\mathbf{A}_n$ by $\mathbf{Z}^n$ and A by Z, carry through, thus showing that Algorithm SqBroadcast "physically naturally" generates a wave front broadcast in $\mathcal{M}_{\mathbf{Z}^n}(Z)$.

6.3 Possible Physical Significance

While the contents of messages for the Algorithms HexBroadcast and SqBroadcast do not involve relative coordinates with respect to the origin as does Algorithm Broadcast, it is interesting to consider whether there is physical significance to the actual content of these messages. Many *physical* waves in 3-space carry information about the magnitude of a force. In 3-space the magnitude of many such forces falls off in proportion to the inverse of the square of the distance from the origin of the force wave; also the surface area of the wave front correspondingly increases with this same square of the distance from the origin. One of the message contents for the algorithms of this paper is the square of the distance from the origin, but this for any dimension > 1. Actually, we need to send distance information, and sending squares of distance is our way to work with integers only. The other two message contents for Algorithms HexBroadcast and SqBroadcast, while a bit technical, are also directly or indirectly information about the distance from the origin. One of these message contents is the minimal (possibly meandering) graph path length distance. If real physical space *is* lattice-like, one might expect this non-euclidean lattice distance also to be a factor in the falling off of forces with distance as a force wave progresses. An exact working out of the true physical meaning, if any, of our algorithms is intriguing open problem.

Note added November 3, 1991. We just learned about [Koc91] which essentially uses a discretization of the wave equation from physics to provide an intriguing algorithm for the approximately linear in real-time simulation[5] of the energy dissipation of light in a lattice

[5]Of course his simulation on a Connection Machine CM2 likely can't be carried out approximately linear in real-time because of unpredictable delays in the interconnection network.

computer based on $\mathbf{Z}^3$. We believe that we could carry along a little extra computing with the progression of any of our wavefront algorithms and achieve the effects of his algorithm (also in a wider class of lattice computers).

7 Other Work

In [CRS91b] we discuss lattice computers based on lattices from a large class of lattices we call the *well-represented* lattices. There, we also present an algorithm template for the simulation of constant speed particle motion in lattice computers based on well-represented lattices, and then an instance of this template for the simulation of constant speed particle translation in such lattice computers. In [She91] we present an algorithm, also an instance of the template from [CRS91b], for the simulation of constant angular speed particle revolution. Surprisingly it turns out that the class of well-represented lattices is the class of *root* lattices (see [CS88] for a discussion on root lattices and [RS91] for this and other characterizations of root lattices). Interestingly, it also turns out that a direct consequence of one of these characterizations of root lattices is that the "natural" algorithms, presented in this paper, for uniform radiation from a point source can be generalized to work on a bounded lattice computer based on any root lattice.

References

[CRS90a] J. Case, D. S. Rajan, and A. M. Shende. Optimally representing euclidean space discretely for analogically simulating physical phenomena. In K. V. Nori, editor, *Foundations of Software Technology & Theoretical Computer Science*, volume 472 of *Lecture Notes in Computer Science*, pages 190–203. Springer Verlag, Berlin, 1990. See [CRS91a].

[CRS90b] John Case, Dayanand S. Rajan, and Anil M. Shende. Parallel processor computer, Dec 1990. U.S. Patent Application submitted by SUNY, Research Foundation.

[CRS91a] John Case, Dayanand S. Rajan, and Anil M. Shende. Lattice computers for optimally representing euclidean space. Technical Report 91-15, University of Delaware, June 1991. Expands on and corrects slightly [CRS90a].

[CRS91b] John Case, Dayanand S. Rajan, and Anil M. Shende. Simulating uniform motion in lattice computers I: Constant speed particle translation. Technical Report 91-17, University of Delaware, June 1991.

[CS88] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer Verlag, 1988.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7), 1982.

[FT82] Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4), 1982.

[GL87] Peter M. Gruber and Gerrit Lekkerkerker. *Geometry of Numbers.* North-Holland Mathematical Library, 1987.

[Her64] I. N. Herstein. *Topics in Algebra.* Blaisdell Publishing Co., 1964.

[Hil55] E. L. Hill. Relativistic theory of discrete momentum space and discrete space–time. *Physical Review*, 100(6), December 1955.

[Koc91] P. Kochevar. A simple light simulation algorithm for massively parallel machines. *Journal of Parallel and Distributed Computing*, 13(2):193–201, October 1991.

[Mac86] Thinking Machines. Introduction to data level parallelism. Technical Report 86.14, Thinking Machines, April 1986.

[Mar84] Norman Margolus. Physics–like models of computation. *Physica 10D*, pages 81–95, 1984.

[Min82] Marvin Minsky. Cellular vacuum. *International Journal of Theoretical Physics*, 21(6/7), 1982.

[RS91] Dayanand S. Rajan and Anil M. Shende. Efficiently generated lattices and root lattices. Technical Report 92-02, University of Delaware, August 1991. Journal article under review.

[She91] Anil M. Shende. *Digital Analog Simulation of Uniform Motion in Representations of Physical n-Space by Lattice-Work MIMD Computer Architectures.* PhD thesis, SUNY at Buffalo, Buffalo, N. Y., 1991. TR 91-14, Department of Computer Science.

[Sny47] Hartland S. Snyder. Quantized space–time. *Physical Review*, 71(1):38–41, 1947.

[Svo86] Karl Svozil. Are quantum fields cellular automata? *Physics Letters A*, 119(4), December 1986.

[SW86] James B. Salem and S. Wolfram. Thermodynamics and hydrodynamics with cellular automata. In S. Wolfram, editor, *Theory and Applications of Cellular Automata.* World Scientific, 1986.

[TM87] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines.* MIT Press, 1987.

[Tof77a] Tommaso Toffoli. Cellular automata machines. Technical Report 208, Comp. Comm. Sci. Dept., University of Michigan, 1977.

[Tof77b] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.

[Tof84] Tommaso Toffoli. CAM: A high–performance cellular–automaton machine. *Physica 10D*, pages 195–204, 1984.

[Ung58] S. H. Unger. A computer oriented towards spatial problems. In *Proceedings of the IRE*, volume 46, pages 1744–1750, 1958.

[Vic84] Gérard Y. Vichniac. Simulating physics with cellular automata. *Physica 10D*, pages 96–116, 1984.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983.

[Zus69] K. Zuse. *Rechnender Raum.* Vieweg, Braunshweig, 1969. Translated as Calculating Space, Tech. Transl. AZT–70–164–GEMIT, MIT Project MAC, 1970.