AT^2 Bounds for a Class of VLSI Problems and String Matching

Micah Adler John Byers

Abstract

We present a class of boolean functions which have *regional mappings*, a generalization of the *transitivity* property defined in [12]. We prove that all transitive problems have regional mappings, as do a variety of interesting computational problems such as merging two sorted lists of arbitrary length, generalized integer multiplication and matrix-vector products. We present a general area-time lower bound for VLSI implementations of problems with regional mappings and confirm that the lower bound matches the previously known bound for transitive problems. For generalized integer multiplication, we present a custom VLSI implementation which provides a matching upper bound. The results improve AT^2 bounds on a number of open problems.

In related work, we consider the problem of finding occurrences of a P-bit pattern in an N-bit text string. We show that every chip capable of solving this string matching problem requires $AT^2 = \Omega(PN)$. A custom VLSI design, using an idea similar to that used for generalized integer multiplication, provides a matching upper bound when N is polynomial in P.

1 Introduction

Vuillemin [12] introduces a property of boolean functions called transitivity common to well-known problems such as the cyclic shift problem, integer multiplication, and matrix-matrix multiplication. Using the realistic model for designing VLSI circuits described by Brent, Kung and Thompson [4, 10], Vuillemin presents area-time lower bounds for transitive problems. Much of the work in VLSI circuit design has focused on implementation of area-time optimal circuits to solve problems in integer arithmetic [3, 4] and problems of ordering such as merging and sorting [1, 2, 6]. We employ the methodology used by Vuillemin, seeking to characterize a wide range of problems whose area-time complexity can be lower bounded by a general technique. The focus of this paper will be on the minimum amount of information exchanged across boundaries of small regions of the chip. Using the convention that A denotes the area of the chip, and T denotes the running time of an algorithm on the chip, our technique gives us bounds on the AT^2 measure of complexity.

Transitive problems have the property that given a sufficiently large set of input bits and a set of output bits of equal cardinality, there exists a control specification such that a constant fraction of the input bits in the set must be output as a subset of the output set. Vuillemin exploits this property to produce a general and powerful area-time lower bound for transitive problems. We generalize the class of transitive problems by removing the restriction that the input and output are of the same length, yet we maintain the property common to transitive problems stated above. We confirm the following important points about the generalized class, which we describe as problems having regional mappings:

- The unrestricted class strictly contains the class of transitive problems.
- No power is lost by generalizing transitivity, i.e. the area-time lower bounds for transitive problems are still achieved.
- The general lower bound is useful, i.e. the results improve AT^2 bounds on a number of open problems.

The proof of the area-time lower bound is derived from the tesselation technique of Bilardi and Preparata [2], which proves that a large number of bits must be transmitted across the boundary of a small rectangle of the VLSI chip for problems having regional mappings.

In related work, we show that the techniques used to prove that a given problem has regional mappings and which prove the lower bound for problems with regional mappings are useful in other contexts. We present an AT^2 lower bound of $\Omega(PN)$ to solve the pattern matching problem with a P-bit pattern and an N-bit text string. Then, we extend early work by Galil [7] and Vishkin [11] to prove a matching AT^2 upper bound when the length of the pattern is polynomial in N.

1.1 Summary of Results

We present improved AT^2 bounds for the following problems:

- **String Matching**: Given a P-bit pattern string and an N-bit text string, report the location of all occurrences of the pattern in the text. $AT^2 = \Omega(PN)$. $AT^2 = O(PN)$ when N is polynomial in P.

- **Generalized Integer Multiplication**: Compute the product of a V-bit integer and a P-bit integer. $\mathrm{AT}^2 = \Omega(PV)$. $\mathrm{AT}^2 = \mathrm{O}(PV)$ when V is polynomial in P.
- **Matrix-Vector Product**: Given an M by N matrix R and an N vector V, compute $R \times V$. $\mathrm{AT}^2 = \Omega(MN)$.
- **Merging Two Sorted Lists**: Given two sorted lists of M and N elements respectively, $M < N$, merge the two lists, producing a sorted list of $M + N$ elements. $\mathrm{AT}^2 = \Omega(MN)$ when each element is of length at least $\log M$.

The latter three problems are contained in the class of problems having regional mappings, as are a wide variety of problems whose area-time complexity is known.

1.2 Overview of the Paper

The rest of the paper proceeds as follows. In Section 2, we review the VLSI model proposed by Thompson [10]. In Section 3, we define the class of problems which have regional mappings and provide a lower bound on those problems using the tesselation lower bound technique of Bilardi and Preparata [2]. We then provide the definition of transitive problems and give a proof that the class of transitive problems is a subset of the class of problems having regional mappings. Section 4 examines two problems, generalized integer multiplication and variable-length merging; regional mappings produce improved lower bounds for these problems. In Section 5, we present another main result: an AT^2 lower bound for the string matching problem which incorporates the techniques presented in Sections 3 and 4 and a fooling set argument. A matching AT^2 upper bound for string matching follows before we conclude with possibilities for future work.

2 The VLSI Model

In the synchronous VLSI model defined in [4] the VLSI chip is an embedding of a computation graph with bounded fan-in and fan-out on a 2-dimensional mesh. The vertices of the graph correspond to processors and I/O ports, which lie at the mesh interconnections. The edges of the graph correspond to wires on the chip and are subject to the technological constraint that ν wires are allowed to overlap at any point in the chip. For the purpose of this paper, we make the simplifying assumption that $\nu = 2$. Wires can only transmit 1 bit of information per unit time. Wires can be of any length, and propagation across a wire of arbitrary length is assumed to take a fixed amount of time (*synchronous* system). In this model, it is assumed that the chip is *when and where-oblivious*, i.e. I/O bits conform to a fixed schedule, arriving and departing at a fixed time and location for all problem instances. Using terminology introduced in [2], the I/O schedule is *semellective* and *unilocal*, i.e. each input bit is received at exactly one time and location. The running time of an algorithm on a VLSI chip is denoted by T, and the area of the chip is denoted by A.

3 Problems with Regional Mappings

For the purpose of analysis, a problem on our VLSI chip is set up in the following framework:

Input bits: $X = x_1, \ldots, x_s$

Control bits: $C = c_1, \ldots, c_t$

Output bits: $Z = z_1, \ldots, z_k$

If, for all subsets of the inputs and subsets of the outputs, both of size $\frac{s}{3}$, there exists a setting of the control bits such that $\Omega(s)$ inputs are mapped to the outputs, then we will say that a problem has regional mappings. More formally:

For control bit specification C, we define $D_{i,j}(C) = 1$ iff $z_j = x_i$ for every set of inputs X.

Now, let $I = x_{i_1}, \ldots, x_{i_{\frac{s}{3}}}$ be a subset of X with cardinality $\frac{s}{3}$ and let $J = z_{j_1}, \ldots, z_{j_{\frac{s}{3}}}$ be a subset of Z with cardinality $\frac{s}{3}$.

Definition 1 *A boolean function $f(X, C) = Z$ has* **regional mappings** *if there exists a constant γ such that for all I, J as defined above there exists $\overline{C}$, an instantiation of the control bits and g, a permutation on $\frac{s}{3}$ elements such that for D as defined above*

$$\sum_{v=1}^{\frac{s}{3}} D_{i_v, j_{g(v)}}(\overline{C}) \geq s\gamma$$

Theorem 1 *A problem with regional mappings can be solved only on a VLSI chip with $AT^2 = \Omega(\gamma^2 sk)$.*

For the proof of this theorem, we use the following three lemmas.

Lemma 1 *For any chip that produces k output bits and for any value of $s \leq k$, either $AT^2 \geq sk$ or every input pad of that chip receives $\leq s$ inputs.*

<u>Proof:</u> All k bits must be produced at an output pad of the chip, and thus $\mathrm{AT} > k$. If a single pad receives more than s inputs, then $\mathrm{T} > s$ and thus $\mathrm{AT}^2 > sk$. So, either $\mathrm{AT}^2 = \Omega(sk)$ or every pad receives less than s inputs. □

Lemma 2 (Brent, Kung) *Given an s-input, k-output VLSI chip such that no I/O port receives or produces more than s bits, there exists a line with a jog which partitions the I/O ports into two regions: L, containing input ports receiving at least $\frac{s}{3}$ input bits, and R, containing output ports which produce at least $\frac{k}{3}$ output bits.*

Lemma 3 (Bilardi, Preparata) *Given an s-input, k-output VLSI chip such that no I/O port receives or produces more than s bits, there exists a partition of the chip into connected regions $L, R_1, R_2, \ldots, R_m$ such that:*

- *L receives at least $\frac{s}{3}$ input bits*
- *For all i, R_i is rectangular with side-length $O(\sqrt{\frac{A_r s}{k}})$, where $A_r = \sum_i Area(R_i)$*
- *There exists a j such that R_j produces at least $\frac{s}{3}$ output bits*
- *The number of regions i with $Area(R_i) \geq Area(R_j)$ is $\Omega(\frac{k}{s})$*

<u>Proof (Sketch):</u> The conditions of Lemma 3 are consistent with those of Lemma 2, so we can partition the chip into regions L and R. We refer to R as the right-hand side of the chip. Next, place a grid over R ensuring that the edges of the grid lie between the mesh edges of the VLSI chip. The side-length of the grid squares is $\sqrt{\frac{A_r s}{k}}$. If we assume that R has height and width at least $\sqrt{\frac{A_r s}{k}}$, the

placement of the grid induces a partition of the right-hand side of the chip into $\Theta(\frac{k}{s})$ regions, which we label $R_1, R_2, \ldots$. The cases where either the height or width less than $\sqrt{\frac{A_r s}{k}}$ are dealt with in the appendix. On average, each of the $\frac{k}{s}$ regions contains output ports which produce $\frac{s}{3}$ output bits, so by an averaging argument, some region R_j must produce at least $\frac{s}{3}$ bits. □

Proof of Theorem 1: [1] Assume an adversary provides us with a chip which solves an instance of problem P with regional mappings, yet purportedly uses $\mathrm{AT}^2 = o(\gamma^2 sk)$. From Lemma 1, we can assume each I/O port only receives or produces s bits. The conditions of Lemma 3 are now satisfied so we form the described partition in which we will pay particular attention to the induced regions L and R_j. First, let any $\frac{s}{3}$ output bits from region R_j form the set J in the definition of problems with regional mappings. Likewise, let any $\frac{s}{3}$ input bits on the left-hand side of the chip (region L) form the set I in the definition. Since P has regional mappings, $s\gamma$ bits in I are directly mapped to $s\gamma$ outputs in J under some control specification $\overline{C}$, and so we can show with a fooling set argument that $s\gamma$ bits must cross the boundary of the region R_j. Details of a similar argument are presented in Section 5.1 for the string matching lower bound. Now since the number of wires bisected by the perimeter of R_j is at most $O(\sqrt{\frac{A_r s}{k}})$, this acts as an upper bound on the number of bits which can cross the boundary of R_j at each time step. Therefore, all $s\gamma$ bits can cross the line only when T is large enough to satisfy $\sqrt{\frac{A_r s}{k}}\ \mathrm{T} = \Omega(s\gamma)$, or, $\frac{A_r s}{k}\ \mathrm{T}^2 = \Omega(\gamma^2 s^2)$. Finally, the area of the chip is at least as large as the area of the right-hand side of the chip, so $\mathrm{AT}^2 = \Omega(\gamma^2 sk)$. □

3.1 The Relation Between Transitive Problems and Problems with Regional Mappings

In order to prove our second main theorem, we first need to give a formal definition of a transitive problem. The same general framework as before applies, provided that we now have the additional constraint that the input and output are of equal length.

$$\text{Input bits: } X = x_1, \ldots, x_n$$

$$\text{Control bits: } C = c_1, \ldots, c_t$$

$$\text{Output bits: } Z = z_1, \ldots, z_n$$

Definition 2 (Vuillemin) *A problem P is* **transitive** *if there exists a transitive group G of permutations on $1 \ldots k$ such that for all $g \in G$, there exists a control specification $\overline{C}$ such that for D as defined above:*

$$\sum_{v=1}^{n} D_{v,g(v)}(\overline{C}) = n$$

Theorem 2 *All transitive problems have regional mappings, with $\gamma = \frac{1}{9}$.*

Proof: For a transitive problem, both the input X and the output Z have size n. Select arbitrary subsets $I \subseteq X$ and $J \subseteq Z$ such that $|I| = |J| = \frac{n}{3}$ as in the definition of regional mappings. For each element $g_i \in G$, where G is the transitive group, let there exist an i colored edge between $x_j \in I$ and any output z_k if $g_i(x_j) = z_k$. Since $|I| = \frac{n}{3}$, we

[1] This proof technique also draws from ideas presented in [2].

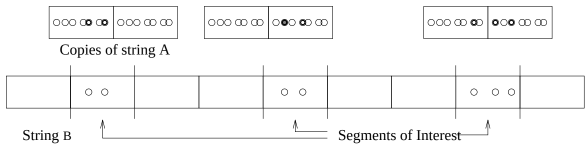

Figure 1: Aligning Marked Bits

will have a total of $|G| \cdot \frac{n}{3}$ edges between I and all outputs. But, in any transitive group of permutations, the number of elements such that $g(u) = v, g \in G$ is the same for all u and v. Since $|J| = \frac{n}{3}$, the total number of edges that represent a map of an input in I to an output in J is $|G| \cdot \frac{n}{9}$. But, we only have a total of $|G|$ colors, and therefore, there must be some color common to at least $\frac{n}{9}$ edges. Now since each vertex adjoins at most one edge of a given color, there must exist some element $g \in G$ which maps $\frac{n}{9}$ inputs in I to outputs in J. □

Note that the lower bound proved in Theorem 1 is asymptotically equivalent to the $\Omega(n^2)$ bound derived by Vuillemin for transitive problems.

4 Examples of Problems with Regional Mappings

Before presenting problems which have regional mappings, we prove the following two lemmas which will be used to prove the existence of regional mappings in a subsequent example. Later on, these lemmas will be reused in our derivation of a lower bound for string matching.

Lemma 4 *Given source and target bit strings, both of length k, of which x bits of the source and y bits of the target are marked, there exists a cyclic shift of the source which aligns $\frac{xy}{k}$ of the marked bits.*

Proof: Draw an edge with color s from x_i to y_i if an s-bit shift aligns key bit x_i with y_i. Since xy edges are created, one for each pair of bits, and only k colors are used, there exists a color common to at least $\frac{xy}{k}$ of these edges (a shift that aligns that many bits). □

Lemma 5 *Given a P-bit string A and a N-bit string B, $P < N$, of which x bits of string B and y bits of string A have been marked, $x \leq y \leq P$, there exists a placement of non-overlapping copies of string A on string B which aligns $\frac{x^2}{4P}$ of the marked bits in B with marked bits in A.*

Proof: We divide string B into segments of length P, numbering the segments $1, \ldots, \lceil \frac{N}{P} \rceil$. Next, we determine which value of $k \in \{0, 1, 2\}$ maximizes the number of marked text bits in segments with numbers congruent to k mod 3. Since the total number of marked bits in B is x, this maximizing value of k identifies segments which contain a total of at least $\frac{x}{3}$ marked bits. We denote the segments with numbers congruent to k mod 3 *segments of interest*, and we restrict attention to the marked bits in those segments, while disregarding other segments.

Denote the number of marked bits in segment i by α_i. If for any i, $\alpha_i \geq \frac{x}{4}$, then we immediately apply Lemma 4 to this segment and string A to align at least $\frac{xy}{4P} \geq \frac{x^2}{4P}$ bits. So, we assume without loss of generality that for all i, $\alpha_i < \frac{x}{4}$. Next, let m_i denote the number of aligned marked bits from string A after the first i segments of string B have been processed and let u_i denote the number of remaining unaligned marked bits. If for any i, $m_i \geq \frac{x}{4}$, then we have aligned the requisite number of bits, since $\frac{x}{4} \geq \frac{x^2}{4P}$. Therefore, we assume that for all i, $m_i \leq \frac{x}{4}$ and therefore $u_i \geq y - \frac{x}{4} \geq \frac{3x}{4}$. Now, by applying Lemma 4, $\frac{u_i \times \alpha_i}{P}$ marked bits are aligned when processing the ith segment of interest. To compute the number of marked bits which are successfully aligned, sum over all segments of interest i:

$$\sum_i \frac{u_i \times \alpha_i}{P} \geq \sum_i \frac{3x \times \alpha_i}{4P}$$

$$= \frac{3x}{4P} \sum_i \alpha_i \geq \frac{3x}{4P} \times \frac{x}{3} = \frac{x^2}{4P}$$

□

Before discussing two examples in detail, we note that matrix-vector multiplication has regional mappings, giving us the easy lower bound of $\Omega(sk)$ for the problem of multiplying an s by k matrix with an s vector.

4.1 Generalized Integer Multiplication

Multiplying two k-bit integers is one of the original transitive problems described in Vuillemin's paper. A great deal of work on this problem has been presented in the literature [3, 4], culminating in an asymptotically optimal AT^2 bound of $\Theta(k^2)$ for a wide range of T: $\mathrm{T} \in (\log k, \sqrt{k})$. Here we consider the unrestricted problem of multiplying two integers of possibly different length.

Theorem 3 *The area-time complexity of computing the product of a P-bit integer and a V-bit integer is $AT^2 = \Theta(PV)$.*

Proof: The lower bound on AT^2 of $\Omega(PV)$ is immediate once we establish that integer multiplication has regional mappings. Given a chip that computes $Z = A \times B$, we can fit this into the regional mappings framework as follows.

$$\text{Input bits: } A = a_1, \ldots, a_P$$

$$\text{Control bits: } B = B_1, \ldots, B_V$$

$$\text{Output bits: } Z = z_{P+1}, \ldots, z_{P+V}$$

Without loss of generality, we assume that $P < V$. [2] To show that generalized integer multiplication has regional mappings, mark an arbitrarily chosen subset of $\frac{P}{3}$ input bits (from string A) and an arbitrarily chosen subset of $\frac{P}{3}$ of the V highest-order output bits (from string Z). Note that the P lowest order bits of the output are not included in the regional mapping framework. Now we apply Lemma 5 to this pair of strings, where $x = y = \frac{P}{3}$, giving us a non-overlapping placement of copies of string A onto string Z which aligns $\frac{P}{36}$ of the marked bits. To simulate this placement using multiplication, we set the control string B as follows. Consider the placement of non-overlapping copies of A as produced by Lemma 5. For each bit i of output string Z which is aligned with a low-order bit of a copy of string A, we set the ith bit of B to 1. All other bits of B are set to 0. In this manner each of the bits set to 1 in B produces a copy of A on Z. From the definition of problems with regional mappings, it follows that generalized integer multiplication has regional mappings with $\gamma = \frac{1}{36}$. From Theorem 1, generalized integer multiplication requires $\mathrm{AT}^2 = \Omega(PV)$. [3] □

[2] We assume further that V is an integer multiple of P in an effort to omit easy technical details from the presentation.

The stated lower bound is asymptotically optimal, as we exhibit a custom VLSI chip for generalized integer multiplication using $\mathrm{AT}^2 = \mathrm{O}(PV)$. Since this chip's construction is similar to the chip presented in detail in Section 5.2 to solve the string matching problem, we provide only a quick sketch. The chip consists of $\frac{V}{P}$ multipliers which compute a $2P$ bit product from two P bit integers. Each such multiplier can be constructed using $\mathrm{A} = \frac{P^2}{\log^2 P}$ and requiring time $\mathrm{T} = \log P$ [3]. With only a constant factor increase in area, these multipliers can be joined by an H-tree with bandwith $\frac{P}{\log P}$. The short multiplicand is input at the root of this tree while the long multiplicand is distributed among the $\frac{V}{P}$ multipliers. In time $2 \log \frac{V}{P}$, the short multiplicand is transmitted to all the leaves (multipliers), which produce a $2P$ bit result in an additional $\log P$ time. Using the H-tree, carry-propagate addition of the intermediate results (shifted appropriately) can be completed in a final stage again using time $\mathrm{O}(\log \frac{V}{P})$. Total time is $\mathrm{O}(\log \frac{V}{P})$ and area is $\mathrm{O}(\frac{V}{P} \times \frac{P^2}{\log^2 P}) = \mathrm{O}(\frac{VP}{\log^2 P})$. Therefore, $\mathrm{AT}^2 = \mathrm{O}(VP)$ when V is polynomial in P.

4.2 Merging Two Sorted Arrays

Baudet and Chen [1] prove a lower bound for merging two sorted arrays of lengths k and s, $k \geq s$, of $\mathrm{AT}^2 = \Omega(s^2 \log^2 s)$. When k is much larger than s, this bound is not tight, and they state as an open problem obtaining a better lower bound. A better bound can be acheived by showing that the problem of merging two sorted arrays has regional mappings.

Theorem 4 *Any VLSI chip that merges arrays of length s and k, $k \geq s$ where the elements are chosen from the range $\{0, 1, \ldots, D-1\}$, $D \geq \lceil \log s \rceil$ requires $AT^2 = \Omega(sk)$*

Note that for any $k \geq s \log^2 s$ this is an improvement on the previous result.

Proof: We will prove this by showing that the lowest order bits of the array element have regional mappings. Assume X is a sorted array containing k elements, and Y is a sorted array containing s elements, with $k \geq s$. [4] We need to compute Z, a sorted list containing all the elements of X and Y.

To show that this problem has regional mappings, consider the input bits to be the least significant bit of each element of X. The least significant bits of the $s+1$ through k elements of Z are considered the output bits. The other bits of X, as well as all the bits of Y are the control bits.

We now need to show that if we mark a subset of the input bits of size $\frac{s}{3}$ and a subset of the outputs of size $\frac{s}{3}$, we can map $\frac{s}{9}$ of the marked inputs to the marked outputs. First note that the ith input bit can be mapped to the jth output bit by setting the control bits so that exactly $j-1$ elements from X and Y have a value that is less than the

[3] Note that for generalized integer multiplication, γ can be improved to $\frac{2}{27}$ by a slightly more complicated and specialized version of Lemma 5.

[4] Again, we assume that k is an integer multiple of s to avoid technical details.

ith element of X. Similarly, we can set the control bits so that the s input bits are mapped to any s output bits. [5]

To map any $\frac{s}{3}$ marked inputs bits to any $\frac{s}{3}$ marked output bits, let us first consider the unlikely case where all the marked output bits are contiguous. We divide X into three segments of length $\frac{s}{3}$. We know that one of the three segments contains at least $\frac{s}{9}$ marked input bits. This segment, call it the *dense third*, can then be placed directly over the $\frac{s}{3}$ contiguous output bits with a setting of the control bits, thus aligning at least $\frac{s}{9}$ marked input bits with marked output bits.

When the marked output bits are not contiguous, we can align the first bit of the dense third with the first marked output bit. Then, the bits from the dense third are aligned with the marked output bits by inserting a control bit at every output bit that is not marked. This will again align at least $\frac{s}{9}$ marked input bits with marked output bits. Therefore, the problem of merging two sorted arrays has regional mappings with $\gamma = \frac{1}{9}$, and thus from Theorem 1, $\mathrm{AT}^2 = \Omega(ks)$ for any chip that solves this problem. $\square$.

5 String Matching

The string matching problem can be stated simply as follows. Given a binary pattern string E of length P, and a binary text string S of length N, $P \leq N$, generate a binary output string Q of length $N - P + 1$, such that $Q_i = 1$ if and only if there is an occurrence of the string E beginning at position i of S.

Vishkin [11] provides a good background in what is known about string matching for various PRAM models. Galil [7] provides an algorithm that can be implemented on a fixed connection network, yielding an area-time upper bound of $\mathrm{AT}^2 = O(N^2)$. A straight-forward fooling set argument can be used to show a lower bound of $\mathrm{AT}^2 = \Omega(P^2)$, thus leaving us a gap of $\Theta\left(\frac{S^2}{P^2}\right)$ for cases where $S = \omega(P)$.

We present here asymptotically optimal AT^2 bounds of $\Theta(PN)$ for the string matching problem. Although the string matching problem does not have regional mappings, techniques similar to those used in the proof of the regional mapping lower bound are applicable, as are many of the same lemmas. The upper bound is from a custom VLSI design similar to those presented for problems with regional mappings.

5.1 The Lower Bound

Theorem 5 *Any chip that performs string matching on a N-bit text string and a P-bit pattern requires $AT^2 = \Omega(PN)$.*

Thinking of the pattern bits as the input bits and the text bits as the control bits, the proof idea is similar to the proof provided for problems with regional mappings. As in the lower bound proof for regional mappings, we partition the string-matching chip to focus attention on a small region. By showing that the communication complexity across the boundary of this region is large, we lower bound the area-time complexity of string matching. [6]

[5] This assumes that the size of each element in the array is larger than $\lceil \log s \rceil$, which is necessary in order to be able to place a control bit between every two chosen input bits.

[6] Again, we restrict ourselves to the case where N is an integer multiple of P deferring discussion of the technical details of the other case to the full paper.

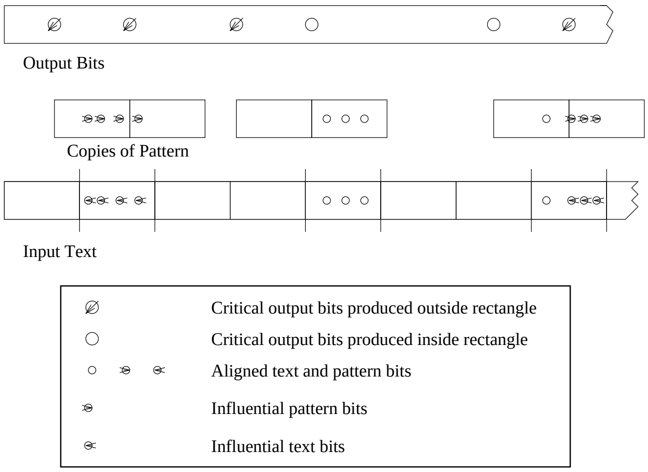

Figure 2: Influential Bits

Lemma 6 *Given R_j, a region of a string matching chip such that non-overlapping copies of the pattern string can be placed on the text string so as to align $\Omega(P)$ text bits received at I/O ports inside R_j, with $\Omega(P)$ distinct pattern bits received at I/O ports outside of R_j, then $\Omega(P)$ bits must be transmitted across the boundary of R_j.*

Proof: Our first goal is to focus on only those pattern and text bits which must cross the boundary of the region R_j. This will require some terminology. First, consider the copies of the pattern which are placed upon the text string. Let the bit location at which the first bit of the kth copy of the pattern is placed be referred to as h_k. Define the *critical output bit* for either a pattern or a text bit aligned using the kth copy of the pattern as z_{h_k}. In the following exposition, we assume the case that more aligned bits have critical output bits which are produced outside of region R_j than are produced inside region R_j. The proof in the alternative case is similar.

A pattern bit is *influential* iff it is an aligned bit described in the statement of the lemma and its critical output bit is produced outside of the region R_j. A text bit is *influential* iff the pattern bit with which it is aligned is influential. Note that an influential text bit need not have been one of the aligned bits.

Also, note that the number of influential text and pattern bits is still $\Omega(P)$, since at least half of the aligned pattern bits will have critical output bits produced outside the rectangle. Now, set all non-influential pattern and text bits to 0, including those unimportant text bits which are not covered by the non-overlapping placement of copies of the pattern string.

Next, construct input specification pairs (x, y). The string x instantiates all influential pattern bits and those text bits which are influential, yet were received outside of the region R_j. The string y instantiates all remaining influential text bits. By specifying x and y we complete the chip's input specification. We say that an x string is *valid* if each text bit set in x is instantiated with the same value as the corresponding influential pattern bit in x. All y strings are valid. By our construction, the value of a given critical output bit z_{h_i} now depends only on bits in x and y. We denote the values of these critical output bits by $g(x, y) = (z_{h_1}, z_{h_2}, \ldots)$. Furthermore, it is easy to verify that for each valid x string, there is a unique y string which sets $g(x, y) = (1, 1, \ldots, 1)$, i.e. creates matches at all critical outputs produced outside region R_j.

Now enumerate all valid strings x and pair them up with their corresponding y strings. The set $\{(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)\}$ is a fooling set, since $g(x^i, y^i) = (1, \ldots, 1)$, yet $g(x^i, y^l) \neq (1, \ldots 1)$, for all $i \neq l$. The size of the fooling set is $2^{\Omega(P)}$, since there are $\Omega(P)$ influential pattern bits. We know that the communication complexity of a problem is lower bounded by the logarithm of the cardinality of a fooling set for that problem. Therefore, $\Omega(P)$ bits must cross the boundary of the region R_j. □

Proof of Theorem 3: Given a chip that solves the string matching problem, from Lemmas 1, 2, and 3, we can divide the chip into regions in a manner analogous to the technique used for problems with regional mappings. In particular, this division will cause one of the regions, the "left-hand side" (region L), to contain at least $\frac{P}{3}$ of the pattern bits. The "right-hand side" of the chip will be subdivided into smaller square regions, one of which is guaranteed to receive $\frac{P}{3}$ text bits, and is denoted by R_j.

After marking at least of $\frac{P}{3}$ the pattern bits received on the left-hand side of the chip and marking $\frac{P}{3}$ of the text bits received inside region R_j, we can apply Lemma 5 to construct a placement of non-overlapping copies of the pattern on the text which aligns $\left(\frac{P}{3}\right)^2 \cdot \frac{1}{4P} = \frac{P}{36}$ marked text bits with distinct marked pattern bits. Region R_j is now known to satisfy the conditions of Lemma 6, so $\Omega(P)$ bits must be transmitted across its boundary. Letting s denote the side-length of region R_j, we see that $\Omega(P)$ bits must cross a perimeter of length $4s$ in time T. If we denote the area of R_j by $a(R_j)$, then $a(R_j) = \Theta(s^2) = \Omega(\frac{P^2}{T^2})$. By Lemma 3 we are guaranteed to have $\Omega(\frac{N}{P})$ disjoint regions on the right-hand side of the chip which have area $\geq a(R_j)$. The total chip area $A = \Omega(\frac{N}{P}) \times \Omega(\frac{P^2}{T^2})$. Therefore, $\mathrm{AT}^2 = \Omega(PN)$. □

5.2 The Upper Bound

Lemma 7 (Galil) *A K-node butterfly can solve an instance of the string matching problem in which the pattern is of length K and the text is of length $2K$ in time $O(\log K)$.*

Theorem 6 *If $P = O(N^\epsilon)$ for $0 < \epsilon \leq 1$, there exists a VLSI chip which solves the string matching problem with $AT^2 = O(PN)$.*

Proof: We adapt Galil's algorithm to work on a VLSI chip with considerably smaller area than the N-node butterfly. We will break the text string into substrings of length $\mathrm{O}(P)$ and apply Galil's lemma locally on those substrings. Our custom VLSI design does this by employing many small butterflies, each containing $\Theta(P)$ nodes. In order to have a total of N nodes across all the butterflies, we require $\Theta\left(\frac{N}{P}\right)$ butterflies. Our custom VLSI design appears as drawn in Figure 3.

The pattern bits are broadcast from a central source along an H-tree of thickness $\frac{P}{\log P}$ to the leaves of the tree. Each of the $\frac{N}{P}$ H-tree leaves corresponds to a P-node butterfly, which can be embedded in a $\frac{P}{\log P}$ x $\frac{P}{\log P}$ square. The total area of the chip is less than 4 times the area of the embedded butterflies and each P-node butterfly requires area $\Theta\left(\frac{P^2}{\log^2 P}\right)$. We therefore have an upper bound on area:

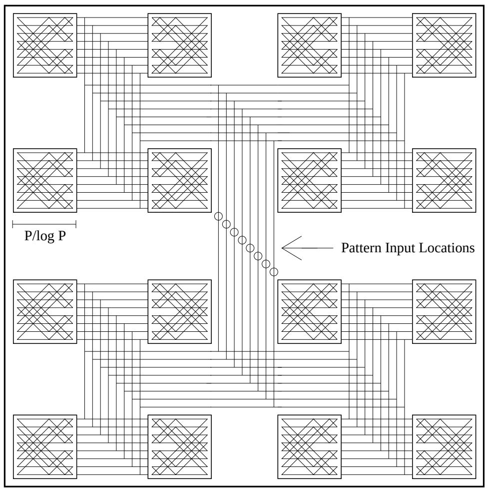

Figure 3: Our Custom VLSI Chip

$$\mathrm{A} = \mathrm{O}\left(\frac{N}{P}\right) \times \mathrm{O}\left(\frac{P^2}{\log^2 P}\right) = \mathrm{O}\left(\frac{PN}{\log^2 P}\right)$$

If each butterfly is to determine P output bits correctly, each butterfly must have access to $2P$ text bits. In particular, since input bits arrive at exactly one location, adjacent leaves must share P text bits. We can use an in-order walk on the H-tree to give us paths between all adjacent leaves such that the maximum path length is $\mathrm{O}(\log N)$ and the maximum congestion is 2. We can then pipeline the P inputs along the $\frac{P}{\log P}$ wires in time $\mathrm{O}(\log N)$. With this pipelining, the bits arrive at each butterfly at a rate of $\frac{P}{\log P}$ per time unit. Therefore, the inputs can be routed to the correct node within each butterfly at an additional cost of $\mathrm{O}\left(\frac{P}{\log P}\right)$.

A sketch of our complete algorithm is as follows:

- Broadcast all bits of the pattern to the small butterflies.
- Input P text bits to each of the small butterflies.
- Transmit P text bits from each butterfly to its left neighbor.
- Each butterfly receiving text input bits $i, i+1, \ldots, i+2P-1$ solves the string matching problem as in Galil's algorithm, reporting the output bits for locations $i, \ldots, i+P-1$.

Pipelining the broadcast of the pattern through the H-tree in step 1 takes $\mathrm{O}(\log \frac{N}{P} + \log P)$ time. Inputting the text bits takes $\mathrm{O}(1)$ time, but copying them to the second butterfly at which they are needed also takes $\mathrm{O}(\log N)$ time. Finally, performing Galil's algorithm requires $\mathrm{O}(\log P)$ time. The total running time T is $\mathrm{O}(\log N)$.

We have an AT^2 upper bound of $\mathrm{O}\left(\frac{NP\log^2 N}{\log^2 P}\right)$, which reduces to a tight bound of $\mathrm{O}(PN)$ when P is polynomial in the size of N. □

6 Conclusions and Future Work

We have proven a general AT^2 lower bound for the class of problems with regional mappings, and for the string matching problem. For several of the problems, we have provided tight upper bounds provided that the number of input bits in the output is polynomial in the length of the short input. Of interest may be the remaining gap of $\Theta(\frac{\log^2 N}{\log^2 P})$ when the discrepancy is super-polynomial. Also, the VLSI designs presented as upper bounds rely on *a priori* knowledge of the length of the input strings. It would be interesting to see area-time optimal VLSI designs that match the lower bounds for variable-length inputs.

Another interesting extension of our work would be to further generalize the class of problems whose area-time complexity can be lower bounded by information-barrier techniques. For example, the lower bound proof for regional mappings does not require that output subsets are exact copies of input subsets; only that the set of possible outputs has cardinality $2^{\Omega(s)}$. It remains to show that naturally occurring problems occur in these more general classes. In the realm of string matching, area-time bounds on the variant in which the output is a single bit – a 1 if any substring of the text matches the pattern and a 0 otherwise, remains open to our knowledge.

7 Acknowledgements

We would like to thank Abhiram Ranade for suggesting work on both the string matching problem and the merging problem, and for his many useful ideas. Also thanks to Dick Karp for helpful conversations, and to David Blackston and Mike Mitzenmacher for their comments on earlier versions of this paper.

References

[1] G. Baudet and W. Chen. *Area-Time Tradeoffs for Merging,* in VLSI: Algorithms and Architectures, pp. 61-68, 1985.

[2] G. Bilardi and F. Preparata. *Area Time Lower Bound Techniques with Applications to Sorting,* in Algorithmica 1, pp. 65-91, 1986.

[3] G. Bilardi and M. Sarrafzadeh. *Optimal VLSI Circuits for the DFT,* in Advances in Computing Research, Volume 4, pp. 87-101, 1987.

[4] R. Brent and H. Kung. *The Chip Complexity of Binary Arithmetic,* in Journal of the ACM, 28(3):521-534, July 1981.

[5] B. Codenotti and M. Leoncini. *Introduction to Parallel Processing.* Addison-Wesley Publishing, London, 1993.

[6] R. Cole and A. Siegel. *On Information Flow and Sorting; New Upper and Lower Bounds for VLSI Circuits,* in Proc. 26th FOCS, pp. 208-221, 1985.

[7] Z. Galil. *Optimal Parallel Algorithms for String Matching,* in Information and Control 67, pp. 144-157, 1985.

[8] J. JaJa. *An Introduction to Parallel Algorithms.* Addison-Wesley, Reading, MA, 1992.

[9] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[10] C. Thompson. *A Complexity Theory for VLSI.* PhD Thesis. Carnegie-Mellon University, Pittsburgh, PA, 1980.

[11] U. Vishkin. *Optimal Parallel Pattern Matching in Strings,* in Information and Control 67, pp. 91-113, 1985.

[12] J. Vuillemin. *A Combinatorial Limit to the Computing Power of VLSI Circuits,* in Proc. 21st FOCS, 1980.

8 Appendix

In the lower bound proofs, we assumed that the division of the chip into left and right sides leaves regions that are not extremely elongated. Here, we deal with extreme chip shapes for the regional mappings lower bound. The arguments for string matching are similar.

In one extreme case, the right side of the chip is very narrow, i.e. has height less than $\sqrt{\frac{A_r s}{k}}$, where A_r is the area of the chip, and k is the length of the input string. But, by lengthening some wires, we can bend the right-hand side of the chip into a snakelike region which fits in a $2\sqrt{A_r}$ by $2\sqrt{A_r}$ square. This procedure can easily be seen to produce at most a four-fold increase in area.

The width of the right side of the chip may also be less than $\sqrt{\frac{A_r s}{k}}$. In this case, we first ensure that the line dividing the right and left sides runs in the same direction as the shorter dimension of the chip. If the width is still not large enough, we divide the right hand side into $\frac{k}{s}$ rectangles, each of width w, the width of the right hand side of the chip, and height $x = \frac{A_r s}{kw}$. We can again argue that some rectangle R_j must contain output ports outputting at least $\frac{s}{3}$ output bits. Recall that $s\gamma$ bits can be mapped from the input pads in the left side of the chip to the output pads in R_j. Therefore, $\Omega(s\gamma)$ bits must cross the perimeter of R. But, we know that $x > w$, and therefore, $x\mathrm{T} = \Omega(s\gamma)$. Let H be the height of the chip. Since we have $\frac{k}{s}$ rectangles, we know that $\mathrm{HT} = \Omega(k\gamma)$, and since the height of the chip is greater than the width of the right hand side of the chip, we know that $\mathrm{AT}^2 = \Omega(\gamma^2 k^2)$. Since $k \geq s$, we again have $\mathrm{AT}^2 = \Omega(\gamma^2 sk)$.