The Average Case Complexity of the Parallel Prefix Problem

Andreas Jakoby[*] **Rüdiger Reischuk**
Christian Schindelhauer **Stephan Weis**[**]

Technische Hochschule Darmstadt[***]

Abstract. We analyse the average case complexity of evaluating all prefixes of an input vector over a given semigroup. As computational model circuits over the semigroup are used and a complexity measure for the average delay of such circuits, called *time*, is introduced. Based on this notion, we then define the average case complexity of a computational problem for arbitrary input distributions.
For highly nonuniform distributions the average case complexity turns out to be as large as the worst case complexity. Thus, in order to make the average case analysis meaningful we also develop a complexity measure for distributions.
Using this framework we show that two n-bit numbers can be added with an average delay of order $\log\log n$ for a large class of distributions. We then give a complete characterization of the average case complexity of the parallel prefix problem with respect to the underlying semigroup. By considering a related reachability problem for finite automata it is shown that the complexity only depends on a property of the semigroup we will call a *confluence*.
Our analysis yields that only two different cases can arise for the reachability question. We show that the parallel prefix problem either can be solved with an average delay of order $\log\log n$, that means with an exponential speedup compared to the worst case, or in case of nonconfluent semigroups that no speedup is possible. Circuit designs are presented that for confluent semigroups achieve the optimal double logarithmic delay while keeping the circuit size linear.
The analysis and results are illustrated at some concrete functions. For the n-ary Boolean `OR`, `THRESHOLD` and `PARITY`, for example, the average case circuit delay is determined exactly up to small constant factors for arbitrary distributions.
Finally, we determine the complexity of the reachability problem itself and show that it is at most quadratic in the size of the semigroup.

1 Introduction

The parallel prefix problem is a fundamental task with a lot of applications such as addition of binary numbers or solving linear recurrences. To each such problem one

[*] supported by DFG Research Grant Re 672-2
[**] supported by GIF Research Grant
[***] Institut für Theoretische Informatik, Alexanderstraße 10, 64283 Darmstadt, Germany
email: jakoby / reischuk / schindel / weis @ iti.informatik.th-darmstadt.de

associates a specific semigroup that describes the algebraic structure of the problem. The complexity of the parallel prefix problem has extensively been investigated in the circuit model. Every prefix can obviously be computed in logarithmic depth. Ladner/Fischer have shown how this can be done in parallel using only linear circuit size [LF80]. Snir has obtained exact bounds for the tradeoff between the depth and the size of prefix circuits [S86].

Bilardi/Preparata have studied time-size tradeoffs for this problem. They have given a complete characterization of the semigroups with respect to this question [BP89] by providing tight lower and upper bounds. The complexity depends on algebraic properties of the semigroup. It is shown that only two cases are possible: within a wide range either the product of time and size grows only linearly or as $n \log n$. In [BP90] the same authors study constant depth unbounded-fanin circuits and classify semigroups according to the property of having linear size prefix circuits in this model.

All these investigations consider the worst case complexity of the parallel prefix problem. In [JRS93] we have defined an average measure for the delay of Boolean circuits called *time*. The idea is to take advantage of favourable cases in which the value of a prefix can be computed faster than within the trivial lower bound of logarithmic delay.

We have shown that for several semigroups like the Boolean semigroup with the `OR` or the `AND`-operator or for the semigroup corresponding to the addition of binary numbers one can compute all prefixes with an average delay $\log \log n$. Futhermore, the circuit size can be kept linear. That means there is an exponential speedup from the worst case to the average case. On the other hand, for functions like `PARITY` or `MAJORITY` no speedup is possible. Expected case upper bounds of order $O(\mathrm{llog}\, n)$ for the addition and other prefix problems have also been obtained by Reif for a different model [Rei93]. He observed that circuit depth $O(\mathrm{llog}\, n)$ ist sufficient if one allows a small portion of input vectors for which a wrong result will be obtained.

The aim of this paper is to study the relation between the average case complexity of the parallel prefix problem and the underlying semigroup in detail. Again, using the notion of **confluent semigroups** a complete characterization will be given saying that only two substantially different cases are possible. Either the average delay is of order $\log \log n$, or it is of logarithmic order, that means equal to the worst case. To obtain these results we translate the problem into a reachability problem for finite automata. It will be shown that the reachable sets depend on algebraic properties of the semigroup.

The paper is organized as follows. The next section provides a formal setting to discuss the parallel prefix problem. Section 3 briefly repeats the formal model for an average case analysis of circuit complexity presented in [JRS93]. For a motivation and more details we refer the reader to that paper.

Using the `ADDITION` as an example we then show that circuits with a good average case delay are structurally quite different from the well known design for the worst case. Section 5 introduces the automata model and defines the reachability problem and the notion of a confluence. We then relate the average case complexity of the parallel prefix problem to the confluence properties of the underlying semigroup. A logarithmic lower delay bound for nonconfluent semigroups is obtained in section 6. The following section shows the upper bound of order $\log \log n$ for con-

fluent semigroups. Finally, we consider the complexity of the reachability problem itself.

2 The Parallel Prefix Problem

Definition 1 *Let G with binary operator $\otimes$ be a finite semigroup, that is $\otimes : G \times G \to G$ is associative. Applying $\otimes$ to a pair of arguments will be written as $\otimes(g_1, g_2)$ or simply as $g_1 \otimes g_2$. Let Σ be a subset of G, which does not necessarily have to be a subsemigroup. The parallel prefix function $\mathtt{PP}_{\Sigma,n} : \Sigma^n \to G^n$ maps an input vector $x = x_1, \ldots, x_n \in \Sigma^n$ to the vector $y = y_1, \ldots, y_n$ where $y_i = x_1 \otimes x_2 \otimes \ldots \otimes x_i$.*

The reason for generalizing the parallel prefix problem by introducing the subset Σ is due to the average case analysis. In some applications, one is interested in the average case complexity where the input distribution is restricted to a subset of G.

Depending on the operator $\otimes$ sometimes it may not be necessary to know both arguments to determine the value of $\otimes(g_1, g_2)$.

Definition 2 *The set of left and right arguments that uniquely determine $\otimes$ is given by $L_\otimes := \{a \in G \mid \otimes(a, G) \equiv const\}$ resp. $R_\otimes := \{a \in G \mid \otimes(G, a) \equiv const\}$.*

We say that G has **a speedup from the left** *iff $L_\otimes \neq \emptyset$, resp.* **from the right** *iff $R_\otimes \neq \emptyset$.*

Obviously, the Boolean semigroup with `OR`-operator allows a speedup from the left and from the right, while with `PARITY`-operator it does not have any speedup.

It is not obvious whether the average case complexity of evaluating the whole input vector, that means computing $y_n = x_1 \otimes \ldots \otimes x_n$, is smaller compared to the worst case for arbitrary input distributions. We will consider this task and also the simultaneous evaluation of all prefixes y_i. It will be shown that the average case complexity can be dramatically smaller and that this also holds for the parallel prefix version. Furthermore, the asymptotically optimal delay can be achieved by circuits of linear size.

3 An Average Case Measure for Circuit Delay

Definition 3 *A circuit over a semigroup G is a directed acyclic graph C with fanin and fanout bounded by 2. An internal gate of fanin 2 represents the $\otimes$-operator. It computes an element of G from the values of its two direct predecessors. In order to get more than 2 copies of the value obtained at a gate simple duplication gates of fanin 1 and fanout 2 may also be used in C. Input gates are initialized with elements from Σ, and certain gates are marked as output gates.*

It should be obvious how such a circuit C computes a function $f : \Sigma^n \to G^m$. Using a suitable binary encoding of G any circuit over G can be replaced by an equivalent Boolean circuit substituting an $\otimes$-gate by a fixed Boolean subcircuit of constant size. In the following we will only consider circuits over G.

A speedup of the semigroup can be exploited by a circuit as follows.

Definition 4 *Let x be an input vector for a circuit C over G, and let v be an internal gate of C with predecessors v_1, v_2. Let $res_v(x)$ denote the value computed by v on x. Then the evaluation time of v for x is given by*

$$\boldsymbol{time_v(x)} \;:=\; 1 + \begin{cases} \max_i time_{v_i}(x) & \text{if } res_{v_1}(x) \notin L_\otimes \;\wedge\; res_{v_2}(x) \notin R_\otimes, \\ time_{v_1}(x) & \text{if } res_{v_1}(x) \in L_\otimes \;\wedge\; res_{v_2}(x) \notin R_\otimes, \\ time_{v_2}(x) & \text{if } res_{v_1}(x) \notin L_\otimes \;\wedge\; res_{v_2}(x) \in R_\otimes, \\ \min_i time_{v_i}(x) & \text{if } res_{v_1}(x) \in L_\otimes \;\wedge\; res_{v_2}(x) \in R_\otimes. \end{cases}$$

For input gates v we set $time_v(x) := 0$. The delay of C for input x, $\boldsymbol{time_C(x)}$, is defined as the maximum of $time_v(x)$ over all output gates v of C.

Observe that although circuits are a rigid computational model the complexity measure *time* is data dependent. Imagine the gates of a circuit working in synchronous phases, where in each phase a gate takes the values of its direct predecessors of the previous phase to compute its new value. If the input values are fed into the input gates at time 0 then independent of the values the internal gates start with by $time_v(x)$, a gate v has converged to its final value $\mathrm{res}_v(x)$. Thus at phase $time_C(x)$ one can read off the result of the computation of circuit C on input x from the output gates. Therefore, $time_C(x)$ can be taken as the actual delay that occurs when C performs the computation for x.

So far, the timing information $time_v(x)$ is given only implicitly. In [JRS93] it is shown how this can be made explicit without increasing the delay (the size doubles at most). Therefore, without loss of generality an average case complexity analysis can be based on the implicitly timed circuit model, which is technically simpler.

Definition 5 *Let $\boldsymbol{\mathcal{D}_{n,\Sigma}}$ denote the set of all probability distributions $\mu_{n,\Sigma}$ that are strictly positive on Σ^n and 0 elsewhere. The uniform distribution on Σ^n that gives equal probability to each of the $|\Sigma|^n$ possible input vectors is denoted by $\mathbf{uni_{n,\Sigma}}$. Let $\boldsymbol{Cir_\Sigma(f)}$ denote the set of all circuits over G that compute f restricted to inputs from Σ^n.*

For a function $f : \Sigma^n \to G^m$ and a set D of probability distributions we define the average case complexity of f with respect to D as

$$etime(f, D) \;:=\; \max_{\mu \in D} \min_{C \in Cir_\mu(f)} \sum_{x \in \Sigma^n} time_C(x)\ \mu(x)\ .$$

In [JRS93] we have defined a second average case measure which is technically more involved than the expectation, but has certain advantages when performing a complexity analysis. In many applications, in particular in the analysis below, both measures do not differ by much. Therefore, to keep the presentation simpler and shorter, we will consider only the expectation here.

We are able to perform an average case analysis not only for the uniform distribution, but for a rather large class of distributions. On the other hand, we will see that for certain distributions the average case complexity is almost as large as the worst case complexity. Thus, in order to specify restrictions we will also define a complexity measure for distributions. The complexity will be measured by the circuit model itself, that is by the complexity of a circuit that starting with elements choosen uniformly at random generates the specific distribution.

Definition 6 *A* **distribution generating circuit (DG-circuit)** *over Σ is a dag C of fanin and fanout at most 2 where an internal node v may represent an arbitrary function $\varphi_v : \Sigma \times \Sigma \to \Sigma$. If C has r input gates and n output gates it performs a transformation of a random variable Z uniformly distributed over Σ^r into a random variable X over Σ^n as follows. The input vector for C is chosen according to Z. Then X equals the distribution of the values obtained at the output gates. Such a C is said to generate the distribution of X.*

In the following we will identify a distribution μ with a random variable X having that distribution. Let $X = X_1, \ldots, X_n$. Using the notion of DG-circuits distributions that are strictly positive on Σ^n can be classified as follows:

Definition 7

$\mathcal{D}\mathbf{Depth}_{n,\Sigma}(d) := \{\mu \in \mathcal{D}_{n,\Sigma} \mid \exists$ *an r-input and n-output DG-circuit C of depth d that transforms a uniformly distributed random variable Z over Σ^r into a random variable X with distribution μ* $\}$.

The restriction to distributions that generate all vectors in Σ^n with positive probability simplifies the analysis. If, for example, one input is fixed to a certain value the dimension of the problem decreases. But it may then be necessary to handle individual prefixes slightly differently.

Distributions that can be generated in small depth share the properties stated in the lemma below. These will turn out to be sufficient to achieve a substantial speedup in the average case for the parallel prefix problem whenever a speedup is possible at all.

Lemma 1 *Let $X \in \mathcal{D}Depth_{n,\Sigma}(d)$. Then for $i \neq j \in [1 \ldots n]$ and all $\sigma \in \Sigma$ and all $w \in \Sigma^{n-1}$ holds* $|\Sigma|^{-2^d} \leq \Pr[X_i = \sigma] \leq 1 - |\Sigma|^{-2^d}$ *and* $|\Sigma|^{-2^{3d}} \leq \Pr[X_i = \sigma \mid X_1 \cdots X_{i-1} X_{i+1} \cdots X_n = w] \leq 1 - |\Sigma|^{-2^{3d}}$.

4 Addition, an Example of a Semigroup with an Average Case Speedup

Before studying the average case complexity of the parallel prefix problem in full generality we will discuss a specific and important example of a Boolean function that can be computed with much smaller delay on the average, the addition. This problem has already been analysed in [JRS93], where matching upper and lower bounds for the average delay have been proved. For the upper bound we will use a different circuit design here. Instead of computing each prefix separately, which results in superlinear circuit size, all prefixes will be computed simultaneously. This way the size will grow only linearly.

The problem of adding two binary numbers basically reduces to an efficient computation of the carries. Carry propagation is the parallel prefix problem for the semigroup $G_{\text{carry}} = (\{\texttt{del}, \texttt{gen}, \texttt{pro}\}, \otimes_{\text{carry}})$. The elements (`delete, generate, propagate`) combine as $u \otimes_{\text{carry}} v = u$, if $v = \texttt{pro}$ and v elsewhere.

In [LF80] Ladner/Fischer describe a design for parallel prefix circuits that achieve logarithmic depth and linear size simultaneously. It has the property that any output

gate y_i has at least distance $\log i$ from any input gate. This immediately implies that for any input x the delay $time(x)$ is of logarithmic order. We will show below that by another design the average delay can be reduced to $\mathrm{llog}\, n$, where **llog** denotes the twice iterated logarithm function. Contrary to the worst case, circuits for a good average delay are not nicely balanced, but extremly unbalanced.

For carry propagation the input alphabet Σ equals the semigroup G_{carry} itself. For $x \in \Sigma^n$ let $\mathtt{pro}(x)$ denote the largest substring containing only the symbol $\mathtt{pro}$. The importance of this quantity comes from the following

Lemma 2 *If C is a circuit over G_{carry} that computes $\mathtt{PP}_{\Sigma,n}$ then for all $x \in \Sigma^n$ holds:* $\log \mathtt{pro}(x) \leq time_C(x)$.

Proof: To prove the lower bound assume that a circuit C finishes for an input x before time $\log \mathtt{pro}(x)$. Let $S = x_i, \ldots, x_j$ be a block of inputs with value $\mathtt{pro}$ of length $\mathtt{pro}(x)$. Then the fanin bound implies that the output gate y_j does not depend on all inputs in S. It is easy to see that changing such an input in an appropriate way C will compute a wrong result at y_j for this modified input. This input may actually occur with non-zero probability for any distribution in $\mathcal{D}_{n,\Sigma}$. ∎

This simple lower bound for $\mathtt{PP}(G_{\mathrm{carry}}, n)$ can be achieved by a special circuit design within a factor of 3 and linear circuit size as illustrated in figure 1.

The circuit G_k for size $n = 2^{k+2} - 2$ has input nodes x_i, output nodes y_i and three types of internal gates. The upper part, let us denote it by A_k, consists of nodes u_i^j with $0 \leq j \leq k$ and $0 \leq i \leq 2^{k-j+2} - 3$. The lower part B_k has nodes v_i^j, w_i^j with $1 \leq j \leq k+1$ and $0 \leq i \leq 2^{k-j+2} - 2$.

Thus, G_k has less than $3 \cdot n$ internal gates. These gates are defined by

$$\begin{array}{lll} u_i^0 = x_i & v_0^j = u_0^{j-1} & w_i^j = v_i^j \otimes u_{2i+1}^{j-1} \\ u_i^j = u_{2i}^{j-1} \otimes u_{2i+1}^{j-1} & v_{2i+1}^j = v_i^{j+1} \otimes u_{4i+2}^{j-1} & y_{2i+1} = w_i^1 \\ & v_{2i}^j = w_{i-1}^{j+1} \otimes u_{4i}^{j-1} & y_{2i} = v_i^1 \end{array}$$

Every node u_i^j is the root of a complete binary tree of height j with leaves $x_{i \cdot 2^j}, \ldots, x_{(i+1) \cdot 2^j - 1}$. As a whole the upper part of G_k is a forest of complete binary trees, in which each height up to k appears exactly twice and the heights decrease when going from left to right.

The v- and w-nodes come in pairs where the w-node is always the right son of the matching v-node. If each pair (v_i^j, w_i^j) is collapsed to a single node the resulting topology of the lower part is a collection of complete binary trees. The output nodes $y_{(i+1) \cdot 2^j - 2}, \ldots, y_{(i+2) \cdot 2^j - 3}$ are the leaves of the subtree with "root" (v_i^j, w_i^j).

The connection between the upper and lower part is chosen such that for each output y_l going from right to left we choose a sequence of u-subtrees whose heights increase slowly. The leaves of these trees cover exactly the inputs $x_l, \ldots, x_1$. Their roots are connected – and this is most important for a good average case behaviour – in a sequential fashion starting with the smallest subtree of height 0. For an example consider output y_8 in figure 1. The corresponding sequence consists of u-subtrees with roots u_8^0, u_3^1, u_2^1 and u_0^2 and leaves $\{x_8\}$, $\{x_7, x_6\}$, $\{x_5, x_4\}$ and $\{x_3, x_2, x_1, x_0\}$.

Theorem 1 *The circuit G_k computes $\mathtt{PP}(G_{\mathrm{carry}}, 2^{k+2} - 2)$ and for all $x \in \Sigma^{2^{k+2}-2}$ its delay is bounded by $time_{G_k}(x) \leq 3 \cdot \log \mathtt{pro}(x) + 6$.*

Proof: The correctness of G_k can be seen easily. To estimate the delay note that $a \otimes_{\text{carry}} b = \mathtt{pro}$ holds only if $a = b = \mathtt{pro}$. Therefore no $\mathtt{pro}$ will appear at a u_i^j-gate of height j larger than $\log \mathtt{pro}(x)$. A node u_i^j only connects to a v- or w-node in the lower part with distance at most $2(j+1)$ to any output gate. Therefore, the delay is bounded by $3 \cdot \log \mathtt{pro}(x) + 6$. ∎

These circuits can also be described by a recursive block structure using the upper and lower parts A_k, B_k as follows. In addition, a block C_k is used which consists of 2 complete binary trees with 2^{k+1} inputs each, and a binary tree D_k with 2^{k+2} outputs. The internal nodes of C_k are gates of type u_i^j, whereas the nodes of D_k are pairs (v_i^j, u_i^j). These blocks are connected as shown in figure 2.

Using another design described in [JRS93], which is based on average case optimal circuits for the $\mathtt{OR}$-function one can achieve an upper bound $2 \log \mathtt{pro}(x)$ for the delay, but this requires $O(n \log n)$ gates. This bound we can also obtain with linear size if the fanout may be increased to 3 successors per gate.

Theorem 1 shows that the average case analysis for the addition reduces to computing the distribution of the values $\mathtt{pro}(x)$.

Lemma 3 *Let X be a random variable over G_{carry} with distribution in $\mathcal{D}Depth(d)$. Then for any l it holds:* $\Pr[\mathtt{pro}(X) \geq 2l-1] \leq \frac{n}{l} \cdot \exp(-l \cdot 3^{-2^{3d}})$.

We can also provide a lower bound for the probability of a long chain of carries.

Lemma 4 *For all n larger than some constant and $d \geq \log\log_3 n$, there exists a random variable $X \in \mathcal{D}Depth(d)$ such that*

$$\Pr\left[\mathtt{pro}(X) > \exp(\tfrac{1}{2} \cdot (2^d \cdot \log 3 + \operatorname{llog} n))\right] \geq \tfrac{1}{2} .$$

These bounds allow us to determine the average case delay of the addition up to a small constant factor.

Theorem 2 *For $d \leq \operatorname{llog} n - 2$ holds*

$$\tfrac{1}{4}(\operatorname{llog} n + \log 3 \cdot 2^d) \leq etime(\mathtt{PP}_{G_{\text{carry}},n}, \mathcal{D}Depth(d)) \leq 3 \cdot (\operatorname{llog} n + \log 3 \cdot 2^{3d}) + 7 .$$

The upper bound can be achieved by circuits of linear size.

Proof: To obtain the lower bound we combine Lemma 2 and Lemma 4 to get a distribution $X \in \mathcal{D}\text{Depth}(d)$ such that for all circuits C

$$\Pr[time_C(X) \geq \tfrac{1}{2}(\operatorname{llog} n + \log 3 \cdot 2^d)] \geq \tfrac{1}{2} .$$

Therefore, $\sum_t t \cdot \Pr[time_C(X) = t] \geq \frac{1}{4}(\operatorname{llog} n + \log 3 \cdot 2^d)$.

The upper bound follows from Theorem 1. By lemma 3 it holds

$$\Pr[\mathtt{pro}(X) \geq 2l-1] \leq \frac{n}{l} \cdot \exp\left(-l \cdot 3^{2^{3d}}\right) ,$$

$$\Pr[time_C(X) \geq 3 \cdot t + 3] \leq \frac{n}{2^t} \cdot \exp\left(-2^t \cdot 3^{2^{3d}}\right) .$$

Choosing $t' := \operatorname{llog} n + \log 3 \cdot 2^{3d}$ we can bound the expectation by

$$\begin{aligned} &\mathrm{E}(\mathrm{time_C(X)}) \\ &\leq \Pr[time_C(X) \leq 3 \cdot t' + 6] \cdot (3 \cdot t' + 6) + \Pr[time_C(X) > 3 \cdot t'] \cdot (3 \cdot \log n + 6) \\ &\leq 3 \cdot t' + 6 + 3 \cdot \log n \cdot \tfrac{n}{2^{t'}} \cdot \exp\left(-2^{t'} 3^{2^{3d}}\right) \leq 3 \cdot (\operatorname{llog} n + \log 3 \cdot 2^{3d}) + 7. \end{aligned}$$

∎

5 The Reachability Problem for Semigroups

After we have seen for a specific example that an average case delay of order $\mathrm{llog}\, n$ can be achieved the parallel prefix problem will now be investigated for arbitrary semigroups. We will classify semigroups according to whether a substantial speedup in the average case is possible or not. It will be shown that only two cases are possible, an exponential speedup like for the `ADDITION`, or no speedup. The answer depends on algebraic properties of the semigroup which can be modelled as a reachability problem for a corresponding finite automata, whose transition graph corresponds to the Cayley-graph of the semigroup.

Given a semigroup G and a subset Σ of G we define a deterministic finite automata $A = A_{G,\Sigma} = (Q, \Sigma, v_0, \Delta)$ with set of states $Q = G \dot\cup \{v_0\}$, starting state $v_0 \notin G$, input alphabet Σ and transition function $\Delta : Q \times \Sigma \to Q$ by

$$\Delta(u, w) := \begin{cases} w, & \text{if } u = v_0 \ , \\ u \otimes w, & \text{if } u \neq v_0 \ . \end{cases}$$

Figure 4 illustrates this automata for G_{carry}.
For $w_1, \ldots, w_m \in \Sigma$ let $\Delta(u, w_1 \ldots w_m)$ denote $\Delta(\Delta(\cdots \Delta(u, w_1) \cdots, w_{m-1}), w_m)$. Obviously, $\Delta(v_0, x_1 \ldots x_t) = y_t = x_1 \otimes \ldots \otimes x_t$.

The set of states that can be reached by A in exactly t steps, where $t \in \mathbb{N}$, is given by $R_A(t) := \{v \in Q \mid \exists x \in \Sigma^t : \Delta(v_0, x) = v\}$.

It is easy to see that these reachability sets share the following property for all $t_1, t_2 \in \mathbb{N}$: $\quad R_A(t_1) = R_A(t_2) \iff \forall i \in \mathbb{N} : \quad R_A(t_1 + i) = R_A(t_2 + i)$.

Since for $t \geq 1$ there are at most $2^{|G|}$ different possibilities for $R_A(t)$ this property implies that there exist numbers τ, π in $[1 \ldots 2^{|G|}]$ with $R_A(\tau) = R_A(\tau + i \cdot \pi)$ for all $i \in \mathbb{N}$. Let us call the smallest such τ the **start of periodicity $\tau(A)$**, and the smallest such π the **period $\pi(A)$** of A.

Definition 8 *Let v_1, v_2 be states of A. A string $w \in \Sigma^*$ such that $\Delta(v_1, w) = \Delta(v_2, w)$ is called a* **confluence** *of these states. A has a* ***t*-confluence** *if there exists a $w \in \Sigma^*$ such that for all states v in $R_A(t)$ the state $\Delta(v, w)$ is the same. A $\tau(A)$-confluence is called a* **canonical** *confluence of A.*

For an illustration of these concepts see figure 3. If G contains a subgroup then obviously a confluence is impossible. Observe that in the special case $\Sigma = G$ the existence of a confluence $w = w_1 \ldots w_l$ implies that Σ also contains a confluence of length 1 since w can be replaced by $w_1 \otimes \ldots \otimes w_{|w|}$. In general, this property does not hold, but we can show:

Lemma 5 *If A has a canonical confluence then there also exists a confluence of length at most $|G|$.*

The example of the threshold semigroup defined below shows that this bound is best possible in general.

6 The Lower Bound for Nonconfluent Semigroups

The last lemma implies that either

- there exists a pair of states v_1, v_2 in $R_A(\tau(A))$ without a confluence, that means there are strings u, v of length $\tau(A)$ such that $\forall w \in \Sigma^* \quad \Delta(v_0, uw) \neq \Delta(v_0, vw)$,
- or all states in $R_A(\tau(A))$ have a common confluence of length $\alpha \leq |G|$.

In the first case, in order to compute $y_t = x_1 \otimes \ldots \otimes x_t$ for $t \geq \tau(A)$ one has to distinguish whether the prefix with length $\tau = \tau(A)$ of x equals u or v.

Theorem 3 *If the automata A for a semigroup G and input alphabet Σ does not have a canonical confluence then for all circuits C computing $\mathtt{PP}_{\Sigma,n}$ it holds for every $x \in \Sigma^n \quad time_C(x) \geq \log(n-\tau) - \log\tau$, where $\tau = \tau(A)$.*

Proof: We have just seen that every output gate y_t with $t \geq \tau$ cannot finish its computation until it has seen at least one of the inputs $x_1, \ldots, x_\tau$. Thus τ input gates have to provide information for at least $b = n - \tau$ output gates. Because of the fanout restriction this implies that within a delay less than $\log \frac{n-\tau}{\tau}$ not every output gate can be reached. ∎

Corollary 1 *For a semigroup G and alphabet Σ without a canonical confluence for the average delay of the parallel prefix problem with respect to any distribution μ that is strictly positive on Σ^n it holds $etime(\mathtt{PP}_{\Sigma,n}, \mu) \geq \log n - O(1)$.*

7 Using a Confluence to Decrease the Average Delay

We will now show how a canonical confluence for the semigroup makes it possible to design parallel prefix circuits with an average delay of order $\mathrm{llog}\, n$. Let $\overline{\boldsymbol{w}}$ be a τ-confluence of length $\boldsymbol{\alpha}$ that forces the result of $y_{\tau+\alpha}$ into the state $\overline{\boldsymbol{v}}$ in $R_A(\tau+\alpha)$. First we will show that for any distribution of bounded complexity the probability to generate $\overline{w}$ as a substring is bounded away from 0. Exploiting Lemma 1 it is not hard to show that $\forall X \in \mathcal{D}\mathrm{Depth}(d)\ \forall\, i \in [\alpha \ldots n] \quad \Pr[X_{i-\alpha+1} \ldots X_i = \overline{w}] \geq |\Sigma|^{-2^{3d}\,\alpha}$.

The design of delay efficient circuits based on a confluence $\overline{w}$ can conveniently be described by a state shift that occurs when the automata has found $\overline{w}$ as a substring in the input sequence x. To model the state shift we duplicate the semigroup.

These considerations now lead to

Lemma 6 *The parallel prefix problem for G can be solved by a circuit C of linear size with delay $time_C(x) \leq 3\log((p+2)\cdot \mathtt{pro}(x) + 2\tau) + \log\alpha + 6$.*

Theorem 4 *If the automata for a semigroup G with alphabet Σ has a τ-confluence of length α and $p \geq \alpha$ is a multiple of its period then the parallel prefix problem for G, Σ can be solved by a circuit of linear size with an average delay of order $\mathrm{llog}\, n$. More precisely,*
$etime(\mathtt{PP}_{\Sigma,n}, \mathcal{D}Depth_{n,\Sigma}(d)) \leq 3\cdot(\mathrm{llog}\, n + 2^{3d}\alpha\log|\Sigma| + \log p + \log\tau) + 7 + \log\alpha$.

These results provide a complete characterization for the average case complexity of the parallel prefix problem. The confluence property is a necessary and sufficient condition for a semigroup to allow an exponential speedup compared to the worst case. Let us illustrate these results at the semigroups for `PARITY`, $\mathtt{THRESHOLD}^a$, `OR`, `AND` and `ADDITION`, where for a number a between 0 and n $\mathtt{THRESHOLD}^a_n$ denotes the n-ary Boolean function which equals 1 if at least a input bits are 1. The semigroups

for $\texttt{THRESHOLD}^3$ and PARITY are described in figure 5 and 6. For the $\texttt{THRESHOLD}^a$-function in general the semigroup is given by $G := \{0, \ldots, a\}$, the input alphabet by $\Sigma := \{0, 1\}$ and the operator $\otimes$ by $r \otimes s := \min\{a, r + s\}$. The important properties of these semigroups are characterized by the following table.

function	G	Σ	τ	π	$\overline{w}$
OR	$\{0,1\}$	$\{0,1\}$	1	1	1
$\texttt{THRESHOLD}^a$	$\{0,\ldots,a\}$	$\{0,1\}$	a	1	1^a
PARITY	$\{0,1\}$	$\{0,1\}$	1	1	—
ADDITION	{gen, del, pro}	{gen, del, pro}	1	1	gen

Corollary 2

$$etime(\texttt{PARITY}_n, uni_{n,\Sigma}) \geq \log n - O(1) \ ,$$
$$etime(\texttt{PP}_{\Sigma,\texttt{OR}_n}, \mathcal{D}Depth_{n,\Sigma}(d)) \leq 3 \cdot (\mathrm{llog}\, n \ + \ 2^{3d}) + 7 \ ,$$
$$etime(\texttt{PP}_{\Sigma,\texttt{THRESHOLD}^a_n}, \mathcal{D}Depth_{n,\Sigma}(d)) \leq 3 \cdot (\mathrm{llog}\, n \ + \ a \cdot 2^{3d}) + 4 \cdot \log a + 7 \ ,$$
$$etime(\texttt{ADDITION}_n), \mathcal{D}Depth_{n,\Sigma}(d)) \leq 3 \cdot (\mathrm{llog}\, n \ + \ 2 \cdot 2^{3d}) + 7 \ .$$

The upper bounds can be improved a little bit by a special analysis depending on the specific function. This and matching lower bounds of order $\mathrm{llog}\, n + 2^d$ for OR, THRESHOLD and ADDITION are shown in [JRS93].

8 The Complexity of the Reachability Problem

Finally, we want to show that the confluence property can be decided efficiently, that means polynomial in the size of the semigroup G. For this purpose we have to analyse the structure of the reachability sets more carefully.

Lemma 7 *For each state q in $R_A(\tau)$ let $\ell(q)$ be the length of a minimal string w with $\Delta(q, w) = q$, if such strings exist, and 1 otherwise. Then the least common multiple of all $\ell(q)$ for $q \in R_A(\tau)$ is a multiple of $\pi(A)$.*

Lemma 8 $\tau \leq |G|^2 + |G|$.

Theorem 5 *There exists an $O(|G|^2 \log |G|)$-time bounded algorithm that given an automata $A = (Q, \Sigma, v_0, \Delta)$ decides whether for $\tau = |G|^2 + |G|$ A has a τ-confluence and, if yes, outputs such a confluence w of length at most $|G|$, and a multiple of the period of A.*

Proof: The following procedure performs the required task.

1. Construct a spanning tree T of the transition graph of A with state v_0 as root. This will be done by a breadth-first-search strategy in time $O(|G|)$. Furthermore we label the nodes of T with their depth.
2. For each state q in T determine $\ell(q)$ as defined above. For this we select from elements v in G, such that $q \otimes v = q$, one with minimal depth $d(v)$ in T. If there exists such v then $\ell(q) := d(v)$. The least common multiple p of these numbers can then be computed in time $O(|G|^2)$.
3. Compute all $R_A(2^i)$ for $i \leq 2 \log |G|$. Since $R_A(0) = \{v_0\}$ and $R_A(i) = \otimes(R_A(i-1), R_A(i-1))$. Now compute $R_A(|G|^2 + |G|)$ in $O(|G|^2 \log |G|)$ steps.
4. Compute a confluence w for $R_A(2 \cdot |G|)$. Using lemma 5 the total amount of steps is bounded by $O(|G|^2)$. ∎

9 Conclusion

Our results exactly characterize the semigroups that allow a significant speedup of the parallel prefix problem in the average case. We can further show that the $\mathrm{llog}\, n + \exp d$ upper bound obtained for confluent semigroups are best possible, except for certain trivial cases like constant or right-identical semigroups. Moreover, the circuit family achieving this bound is easy to construct and universal in the sense that up to a constant factor it yields minimal average delay for arbitrary input distributions (assuming strict positivity). Thus, for the parallel prefix problem a single circuit design turns out to be distribution independent average case optimal.

We have also considered the prefix problem for the circuit model without fanout restriction. Again we can give a complete characterization of the semigroups with an average case speedup, which now becomes slightly more difficult.

References

[BP89] G. Bilardi, F. Preparata, *Size-Time Complexity of Boolean Networks for Prefix Computations,* J. ACM 36, 1989, 362-382.

[BP90] G. Bilardi, F. Preparata, *Characterization of Associative Operations with Prefix Circuits of Constant Depth and Linear Size,* SIAM J. Comput. 19, 1990, 246-255.

[JRS93] A. Jakoby, R. Reischuk, C. Schindelhauer, *Circuit Complexity: from the Worst Case to the Average Case,* Technical Report, TH Darmstadt, 1993, to be presented at STOC'94.

[LF80] R. Ladner, M. Fischer, *Parallel Prefix Computation,* J. ACM 27, 1980, 831-838.

[Rei93] J. Reif, *Probabilistic Parallel Prefix Computation,* Comp. Math. Applic. 26, 1993, 101-110.

[S86] M. Snir, *Depth-Size Trade-offs for Parallel Prefix Computation,* J. Alg. 7, 1986, 185-201.

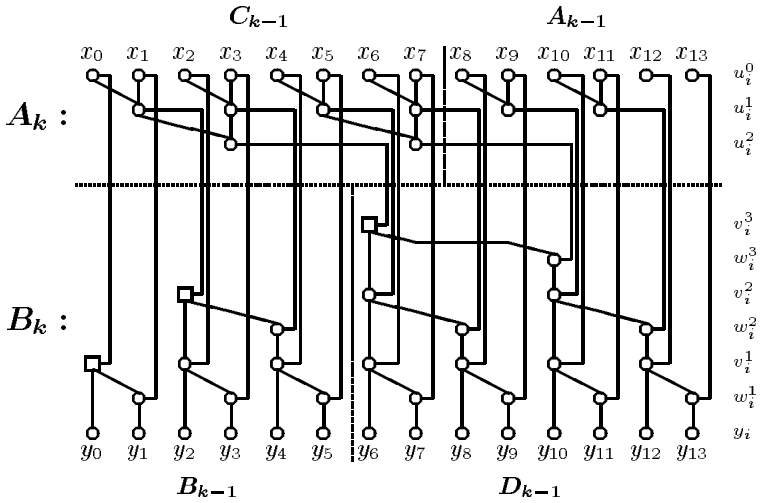

Fig. 1. An average case optimal parallel prefix circuit G_k with $k = 2$ for input vectors of length 14.

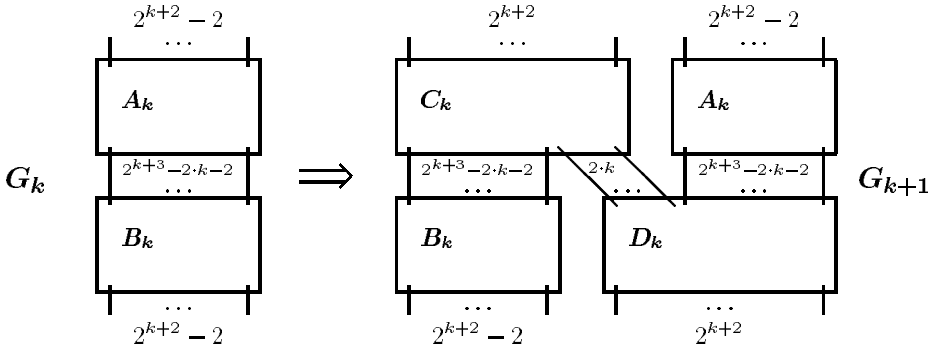

Fig. 2. Recursive construction of an average optimal parallel prefix circuit G_{k+1} with input length $2^{k+3}-2$ from G_k with input length $2^{k+2}-2$.

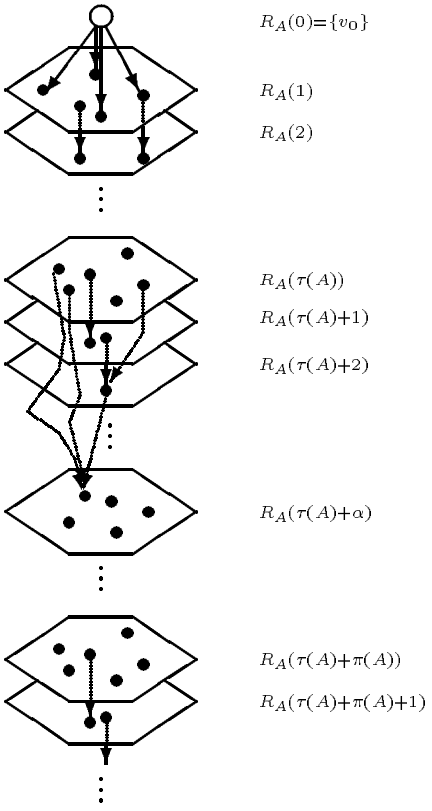

Fig. 3. Sequence of reachability sets with a confluence of length α.

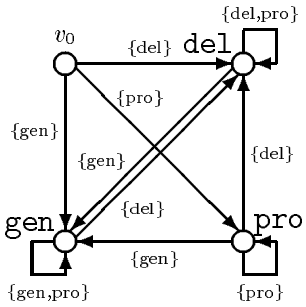

Fig. 4. Semigroup G for the addition.

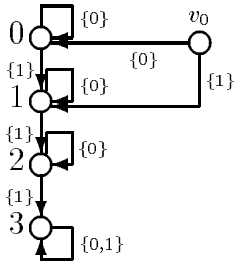

Fig. 5. Semigroup for $\texttt{THRESHOLD}_n^3$.

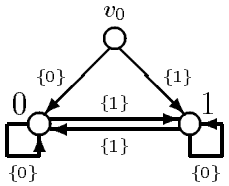

Fig. 6. Semigroup for $\texttt{PARITY}_n$.