Structured NC

B. Scholten and J. van Leeuwen

RUU-CS-89-6
March 1989

Rijksuniversiteit Utrecht

Vakgroep informatica

Padualaan 14 3584 CH Utrecht
Corr. adres: Postbus 80.089, 3508 TB Utrecht
Telefoon 030-531454
The Netherlands

Structured NC

B. Scholten and J. van Leeuwen

Technical Report RUU-CS-89-6
March 1989

Department of Computer Science
University of Utrecht
P.O.Box 80.089, 3508 TB Utrecht
The Netherlands

Structured NC

Bertha Scholten and Jan van Leeuwen

Abstract

In this paper we introduce the class Structured NC (or SNC for short) which is a subclass of NC with the property that any circuit in it should be structured. A structured circuit is a circuit in which the gates are organised in a number of layers, each with the same number of gates and the same connections between gates in a layer. We will distinguish several subclasses of SNC differing in depth of the circuits and fan-in and fan-out of the gates. Relations among these subclasses and between these subclasses and NC and AC will be derived and we will show how to transform circuits from one class into equivalent circuits in other classes. Furthermore for some problems we present structured circuits having fan-in as well as fan-out ≤ 2; for parallel prefix and addition these circuits are of depth $O(\log n)$ with only $O(n)$ gates in a layer, which is optimal for structured circuits. For multiplication we will consider three methods and derive structured circuits for them.

1 Introduction

In the past few years a lot of research has been done on the complexity of parallel algorithms. Complexity classes like NC and AC (classes of problems solvable by logspace uniform families of circuits) have been introduced and studied by various authors (see e.g. [Col, Co2, KaRa, Ru2]). In this paper a new class is introduced. This new class, Structured NC (or SNC for short), is a subclass of NC with the property that any circuit in it should be structured. A structured circuit is a circuit in which the gates are organised in a number of layers, each with the same number of gates and the same connections between gates in a layer. This restriction greatly simplifies the circuits and their description, as a circuit can be represented by the connections of one layer together with its depth and the functions of all gates in the circuit. This means that if the gates are programmable, then we do not have to build the whole circuit; instead it suffices to build just one layer and feed the output of the layer back into this layer. This time the gates are programmed according to the next layer of the circuit. This process is repeated as many times as the depth of the original circuit. Hence the final output is equal to the output of the original circuit and is thus the solution to the problem. It is readily seen that this approach

greatly simplifies the actual construction of the circuit, especially when the circuit is large.

In SNC, we can distinguish several subclasses, according to the depth of the circuits and the fan-in and fan-out of the gates. Relations among these subclasses and between these subclasses and NC and AC are derived. For some problems structured circuits are given with only a small increase in the number of gates compared to the unstructured circuits.

The paper is organized as follows. In section 2 we state the basic definitions about NC, AC and SNC which will be used throughout the rest of the paper.

In section 3 we will compare different SNC classes. Circuits with different fan-in or fan-out will be considered and it will be shown how to transform circuits of one class into equivalent circuits in other classes.

In section 4 we will show how AC^0-circuits can be transformed into $SNC^1_{2,2}$-circuits without using the general method of section 3. This results in circuits using less gates in a layer than if we would have relied on the general method.

In section 5 we will show some problems to be in $SNC^1_{2,2}$, the class of problems solvable with uniform circuit families of structured circuits having depth $O(\log n)$ and with gates with maximum fan-out and fan-in of 2. The problems considered are: parallel prefix, addition and multiplication. For parallel prefix and addition we find structured circuits with $O(n)$ gates in a layer, which is optimal for structured circuits. For multiplication we consider three methods: the 'three for two trick', the method of Karatsuba and Ofman, and the algorithm from Schönhage and Strassen.

In the last section we give a summary of the relationships between different NC, AC and SNC classes and indicate some directions for further research.

2 Preliminaries

In this section we present some basic definitions which will be used throughout this paper.

As usual a (boolean) circuit can be seen as a directed acyclic graph with labelled nodes (called gates). The label of a node indicates the function that the gate performs. Nodes with indegree (fan-in) zero are called input gates or constant gates (constant gates are labelled either 0 or 1). Nodes with outdegree (fan-out) zero are called output gates. The other gates are labelled NOT if the fan-in equals one, and OR or AND for gates with fan-in ≥ 2. The depth of a circuit is the length of the longest path from some input node to some output node. The depth of a gate g in a circuit is the length of the longest path from an input gate to this gate g. In the remainder the size of the input (i.e. the number of input gates) will be denoted by n.

The class AC^k, $k \geq 0$ is the class of all problems solvable by a logspace uniform family of circuits with a polynomially bounded number of gates, and depth $O(\log^k n)$. Furthermore AC is defined as $AC = \bigcup_{k \geq 1} AC^k$. (A family of circuits is logspace uniform if the description of the circuit for inputs of size n can be generated by a

Turing machine using $O(\log n)$ workspace.)

The class NC^k, $k \geq 1$ is the class of all problems solvable by a logspace uniform family of circuits with a polynomially bounded number of gates, all gates with fan-in ≤ 2, and depth $O(\log^k n)$. NC is defined as $NC = \bigcup_{k \geq 1} NC^k$.

A structured circuit is a circuit built from a number of layers, each with the same number of gates and the same lay-out. The first layer consists of the input gates (gates at depth 0) all gates that can be reached from the input gates using only one connection, and the connections between them. We can number these gates, $g_1^0, \ldots, g_n^0$ (input gates) and $g_1^1, \ldots, g_n^1$. All other layers will have the same connections as this one and thus there is a numbering of gates possible where the gates at depth d are numbered $g_1^d, \ldots, g_n^d$, such that there are connections from g_i^d to $g_{j_1}^{d+1}, \ldots, g_{j_r}^{d+1}$ then for all d' ($0 \leq d' <$ the depth of the circuit) $g_i^{d'}$ will have connections only to $g_{j_1}^{d'+1}, \ldots, g_{j_r}^{d'+1}$. Note that the functions (labels) of the gates g_i^d and $g_i^{d'}$ with $d \neq d'$ may still differ. A layer at depth d consists of the nodes at distance d from the input gates, the nodes at distance $d+1$, and the edges (sometimes called *connections* or *lines*) between these nodes. When no confusion may arise we generally will omit the upper-index d (indicating the depth) when numbering the gates.

Definition 1 *(i) SNC^k, $k \geq 1$ is the class of all problems solvable by a logspace uniform family of structured circuits of depth $O(\log^k n)$ with a polynomially bounded number of gates, all gates with fan-in ≤ 2.*

(ii) $SNC = \bigcup_{k \geq 1} SNC^k$.

Like in NC^k, in SNC^k all gates have fan-in ≤ 2 and arbitrary fan-out (i.e. at most polynomial, because there are a polynomially bounded number of gates in the circuit). Structured circuits were introduced because of their simplicity. Gates with unbounded fan-out are hard to build and hence we would also like to have some restriction on the fan-out of the circuits. We define:

Definition 2 *The class $SNC^k_{p(n),q(n)}$, $k \geq 1$ is the class of all problems solvable by a logspace uniform family of structured circuits with a polynomially bounded number of gates, of depth $O(\log^k n)$ and with gates with fan-out $\leq p(n)$ and fan-in $\leq q(n)$.*

For the sake of generality we have also added an index to indicate the maximum fan-in. If unbounded fan-out (fan-in) is allowed, we write $\cdot$ for $p(n)$ ($q(n)$). Thus by definition $SNC^k = SNC^k_{\cdot,2}$

We allow the gates in SNC-circuits to take the following functions (which are a little different from the usual functions): SELECT (one of the inputs and ignore the others), NOT (one of the inputs), AND (some of the inputs and ignore the others), and OR (some of the inputs and ignore the others). Besides these gates there will be input-gates (fan-in = 0) and output-gates (fan-out = 0), some of which are dummy, and constant-gates, which are a special kind of input-gates. The dummy gates are needed because the number of input and output gates should be equal to the number

of gates in the other layers and this might be larger than n. The possiblility for gates to ignore some of the inputs is introduced because as we will see, gates will often have fan-in larger than two, where in different layers gates actually need different inputs. Because of the structure property all layers have to be the same, so all inputs will be the same in each layer. Each gate can choose in each layer which inputs and which function it will use there.

With these functions selecting which inputs to use, the gates in SNC circuits might seem more powerful than those in NC or AC circuits. However in situations where gates in NC or AC would profit from using this selection property of the functions, the input lines that are not selected might as well be left out of the circuit as they have no function at all. In SNC circuits this is not possible as all layers have to be the same and in some other layer one of the not selected inputs might be useful. Note that this means that $SNC^k_{\cdot,\cdot} \subseteq AC^k$ and $SNC^k = SNC^k_{\cdot,2} \subseteq NC^k$.

By allowing the selection property to the functions of the gates we do not make circuits in $SNC_{\cdot,2}$ more powerful than structured circuits without this selection property, because the fan-in is bounded by 2:

Lemma 1 *All functions allowed by gates in $SNC_{\cdot,2}$ circuits can be simulated by gates with the functions AND, OR, NOT and gates where input=output, hereby increasing the size and depth of a circuit only by some constant.*

Proof: Input, constant, NOT gates with one input and gates where input=output have fan-in ≤ 1 and do therefore not use the selection property. Gates with fan-in 2 may use the selection property and next we will show how to avoid using this property by using only ordinary AND, OR, NOT, constant gates and gates where input=output.

We replace all layers in the circuit by 3 new layers, with for all gates with fan-in 2 five extra gates in each layer. The gates with fan-in 1 will in these two extra layers have gates where input=output. This will not change the order of magnitude of the problem in either depth or width (i.e. the number of gates in a layer). The layers will be as in Figure 1. We see that all operations can be simulated using only AND, OR, NOT, constant gates and gates where input=output. As we can easily verify from Figure 1 this will not increase either fan-out or fan-in of the original gates, and the added gates have fan-out, fan-in ≤ 2.

The result of this is that the selection property allowed for gates in SNC-circuits does not really make circuits in $SNC_{\cdot,2}$ more powerful than if these functions were not allowed. □

When unbouded fan-in is allowed the structured circuits with gates which do have the selection property generally cannot be replaced by structured circuits with gates without this property (without increasing the size or depth of the circuit). However, as we will see later on, we are able to transform structured circuits with unbounded fan-in (fan-out) into structured circuits with bounded fan-in (fan-out). These circuits can be replaced as above by circuits with gates with functions not containing the selection property.

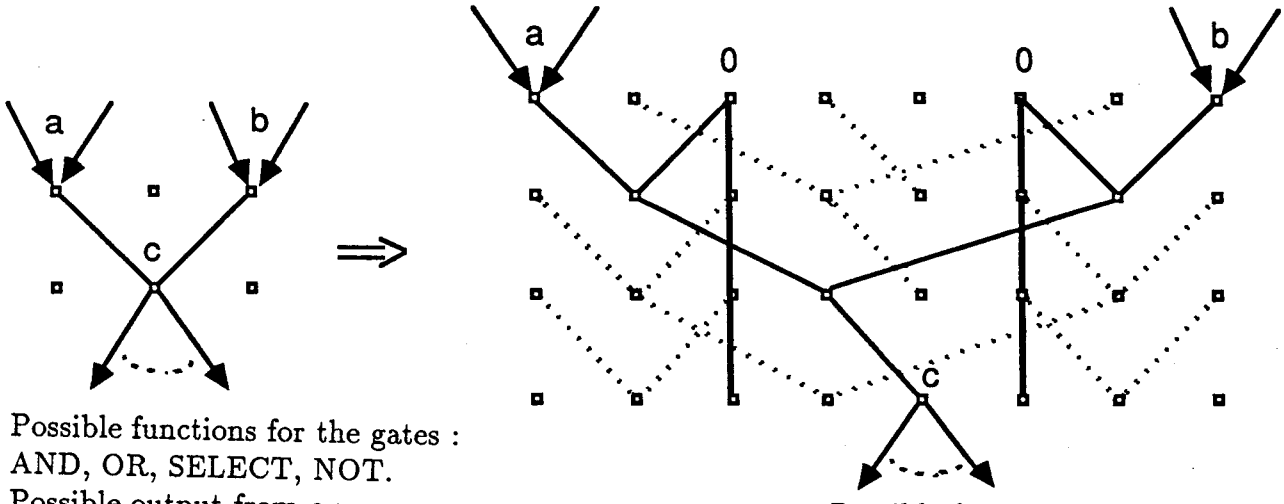

Figure 1: Replacing select gates by a number of ordinary gates.

3 Comparing different SNC classes

In this section we will state some useful theorems dealing with the fan-out and fan-in of gates in structured circuits: by increasing depth and size of a circuit we are able to decrease the number of connections in a layer. This also helps us to transform unstructured circuits into structured ones with resonable fan-out and fan-in. The constructions used to transform the circuits will later on prove to be quite useful in transforming circuits for specific computations into circuits that are in other (more practical) complexity classes. From these theorems we also derive relations between specific complexity classes.

Theorem 1 *Let c_n be a structured circuit of depth $D(n)$ and with $S(n)$ gates each layer, for $S(n)$ some polynomial in n, in which each gate has fan-out $\leq p(n)$ and fan-in $\leq q(n)$. Then there exits an equivalent ciruit c'_n of depth $D(n) * (\lceil \log p(n) \rceil + \lceil \log q(n) - 1 \rceil)$, with $S'(n) \leq S(n) * (p(n) + q(n) - 3)$ gates each layer and each gate having fan-out, fan-in ≤ 2.*

Proof: The general idea behind this proof is as follows. We will first only consider decreasing the fan-out of the gates by adding extra gates. For each gate we add about $p(n)$ extra gates. We replace the original connections by connections from about half of the new gates, each with fan-out 2, which together represent all the original connections. All other gates that have been added will be used to reach all these gates in $\log(p(n))$ layers, using a binary tree structure. This way the output

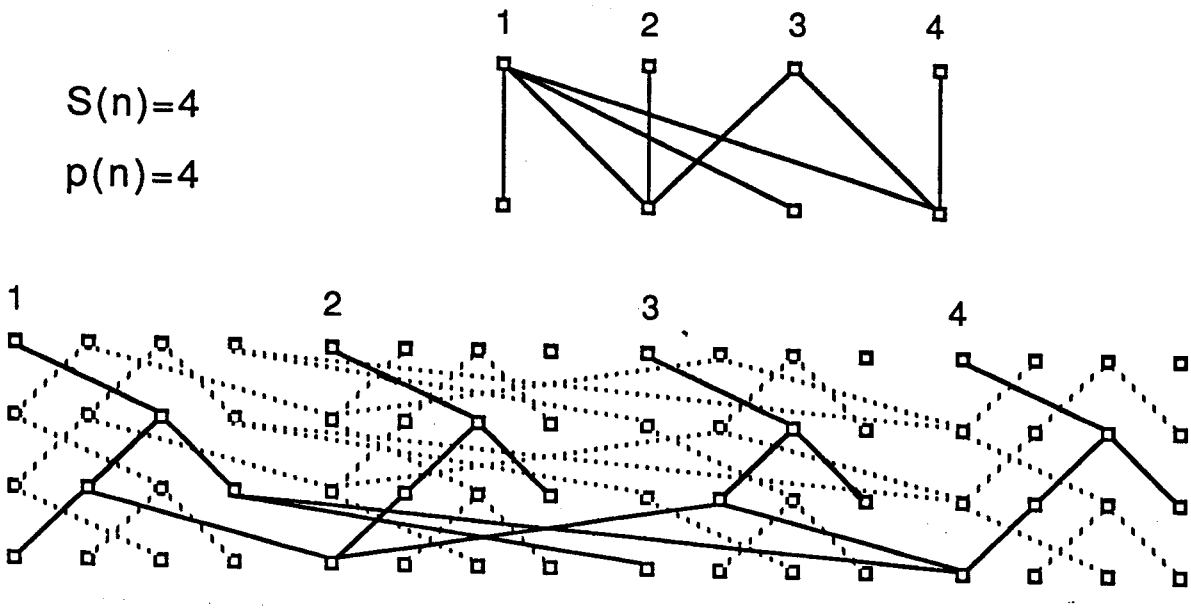

Figure 2: Example of reducing the fan-out.

of each gate of the original circuit can be directed to its original destinations in $\log(p(n))$ layers. See Figure 2.

Now somewhat more formal. We assume $p(n)$ and $q(n)$ to be powers of 2 and show how to transform one layer. As the circuit is structured, the other layers can be treated in exactly the same way. Let the gates from the original circuit be $g_1(0), \cdots, g_{S(n)}(0)$, with outgoing lines from gate $g_i(0)$ to gates $g_{i(j)}(0)$ ($1 \leq j \leq p(n)$ and $1 \leq i \leq S(n)$). We omit the upper-index in the numbering of the gates as we only consider one layer. First we will reduce the fan-out (maximal $p(n)$) per gate to 2. For this purpose we add gates $g_i(1), \cdots, g_i(p(n)-1)$ to the right of the gates $g_i(0)$. The outgoing lines from $g_i(0)$ to gates $g_{i(j)}(0)$ (for $1 \leq j \leq p(n)$) will be replaced by the following connections:

$$
\begin{array}{lll}
(1) & & : g_i(0) \longrightarrow g_i(\frac{p(n)}{2}). \\
(2) & j \neq 0, \textit{ even} & : g_i(j) \longrightarrow g_i(j + 2^{k-1}), \\
 & & \quad g_i(j) \longrightarrow g_i(j - 2^{k-1}), \\
 & & \quad \text{with } k \text{ such that } 2^k \mid j \text{ and } 2^{k+1} \nmid j. \\
(3) & j \neq 0, \textit{ odd} & : g_i(j) \longrightarrow g_{i(j)}(0), \\
 & & \quad g_i(j) \longrightarrow g_{i(j+1)}(0).
\end{array}
$$

Now the fan-out of the original gates has been reduced to 1, and that from the new gates has become 2. The fan-in of the original gates ($g_i(0)$) will not increase because of this operation, while the fan-in of the new gates is 1. This follows from (1)-(3), and the observation that $j \pm 2^{k-1} = j' \pm 2^{k'-1}$ (with j, k and j', k' satisfying the constraint given in (2)) iff $j = j'$. Next we will replace each layer by $\log p(n) + 1$ layers. Now consider a connection $g_i(0) \longrightarrow g_{i(j)}(0)$ in the original circuit. We will show that, using the above connections, $g_{i(j)}(0)$ can also be reached from $g_i(0)$ and that thus our new circuit correctly simulates the old circuit. Because of connection (3) and since we have $\log p(n) + 1$ layers, we must show that $g_i(j)$ (j odd) can be

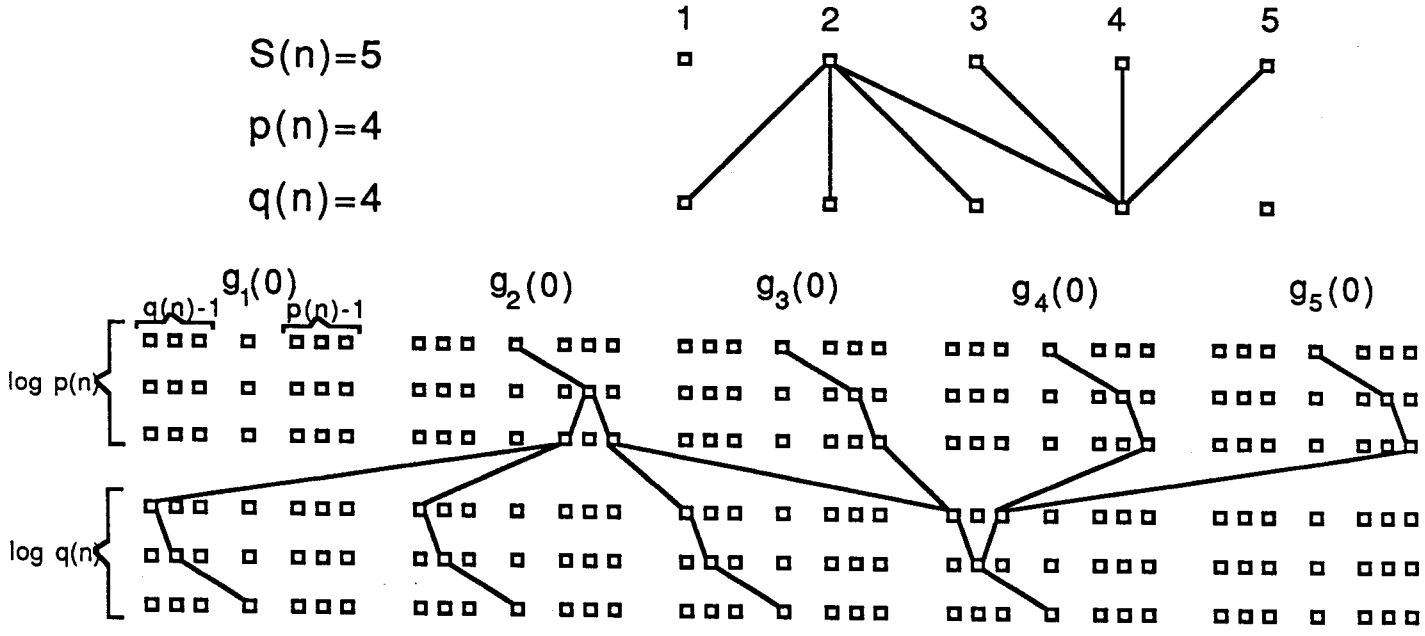

Figure 3: Example of the reduction of both fan-out and fan-in.

reached from $g_i(0)$ in $\log p(n)$ layers. This follows from the fact that in l layers the gates $g_i(m\frac{p(n)}{2^l})$ for $m = 1, 3, \ldots, 2^l - 1$ can be reached, which is easily proved by induction on l. Observe that we can identify $g_i(0)$ with $g_i(\frac{p(n)}{2})$, thereby saving one layer in the new circuit.

Analogous to reducing the fan-out we can reduce the fan-in of a maximum of $q(n)$ to 2. The formal proof will be the same as that for the fan-out and is therefore omitted. Figure 3 gives an example of the above described transformation.

We have shown how to reduce the fan-in and fan-out, and used a factor $p(n) + q(n) - 2$ extra gates per layer and a factor $\log p(n) + \log q(n)$ extra layers compared to the original circuit. Having $D(n)$ as the depth of the original circuit, the depth $D'(n)$ of our newly formed circuit will be:

$$\begin{aligned} D'(n) &= D(n) * (\log p(n) + \log q(n) - 1) \\ &= O(D(n) * \log n) \end{aligned}$$

And the number of gates $S'(n)$ of each layer:

$$S'(n) = S(n) * (p(n) + q(n) - 3)$$

It is straightforward to extend the proof for $p(n)$ and $q(n)$ not powers of 2. We will again take $p(n)+q(n)-3$ times as many gates each layer. To reach the $\lceil \frac{p(n)}{2} \rceil$ and $\lceil \frac{q(n)}{2} \rceil$ gates that will represent the old connections, we need a depth of $\lceil \log p(n) \rceil + \lceil \log q(n) \rceil - 1$ for each layer in the original circuit. In this way we find a circuit with depth $D(n) * (\lceil \log p(n) \rceil + \lceil \log q(n) \rceil - 1)$. □

The transformations as used in the proof of Theorem 1 are such that they will not harm the uniformity of a family of circuits. They do not make the construction

of the circuits of different input size significantly more difficult. Using this theorem we may conclude:

Corollary 1 $SNC^k \subseteq SNC^{k+1}_{2,2}$

Proof: For any family of circuits in SNC^k we can find some polynomial $p(n)$ such that the number of gates in each circuit is smaller than $p(n)$. Thus the maximum fan-out of the gates in each of these circuits can not be larger than $p(n)$. Using Theorem 1, we can reduce fan-out of these circuits to 2, which will result in circuits of depth $O(\log^k n * \log p(n)) = O(\log^{k+1} n)$. The number of gates in a layer will be $O(p(n) * p(n)) = O(p(n)^2)$, which again is bounded by some polynomial in n. We now have a family of structured circuits, each with bounded fan-in and fan-out and of depth $O(\log^{k+1} n)$. □

Corollary 2 $SNC^k_{c_1,c_2} = SNC^k_{2,2}$ *(for constants* $c_1, c_2 \geq 2$*).*

Proof: It is clear that $SNC^k_{2,2} \subseteq SNC^k_{c_1,c_2}$ and the other way around follows from Theorem 1: take $p(n) = c_1$ and $q(n) = c_2$. The $SNC^k_{2,2}$ circuit uses $O(c_1+c_2) = O(1)$ extra gates each layer and the depth will increase no more than $O(\log c_1 + \log c_2) = O(1)$. □

Theorem 1 only tells us about relationships between different structured circuits. It is also interesting to have some general theorem which shows how unstructured circuits can be turned into structured ones with bounded fan-in and fan-out. For this purpose we may use the former theorem to reduce the number of connections once we have found a structured circuit.

Lemma 2 *The Structuring Lemma.*
Let $S(n)$ *be the size and* $D(n)$ *the depth of a circuit with fan-out* $\leq p(n)$ *and fan-in* $\leq q(n)$ *we can build an equivalent circuit which is structured and has depth* $O(D(n) * (\log(D(n) * p(n)) + \log(D(n) * q(n))))$ *and* $O(S(n) * D(n) * (p(n) + q(n)))$ *gates each layer. In the so formed structured circuit all gates will have fan-in, fan-out* ≤ 2.

Proof: The first task is to 'square' the circuit, i.e. to make layers where each layer has the same number of gates. A gate in an unstructured circuit is in layer d if the longest path from the input gate to that gate has length d. Now we simply make the number of gates in each layer equal to the number of gates in the largest layer by adding dummy gates (without connections). As the total (unstructured) circuit needs $S(n)$ gates we will never need more than $S(n)$ gates in each layer to 'square' the circuit. Although by now all layers have the same number of gates, the circuit is still not structured, because the connections differ per layer. To solve this we project the layers onto each other, i.e. we will use all connections from each layer in all the other layers. But now a new problem arises: a gate may have fan-out $O(D(n)*p(n))$ and fan-in $O(D(n) * q(n))$ (the old fan-in/fan-out times the number of layers in the

circuit). Now we can use the method of the former theorem to reduce the number of connections (per layer), leaving us with a structured circuit with bounded fan-in and fan-out of depth $O(D(n) * (\lceil \log(D(n) * p(n)) \rceil + \lceil \log(D(n) * q(n)) \rceil))$ and $O(S(n) * D(n) * (p(n) + q(n)))$ gates each layer. □

Corollary 3 $NC^k \subseteq SNC^{k+1}_{2,2}$.

Proof: For any family of circuits in NC^k we can find some polynomial $p(n)$ such that the number of gates in each circuit is smaller than $p(n)$ (fan-in ≤ 2). Thus the maximum fan-out of the gates in each of these circuits can not be larger than $p(n)$. Using the Structuring Lemma, we can structure these circuits which will result in circuits with depth $O(\log^k n * \log(p(n) * \log^k n)) = O(\log^{k+1} n)$. The number of gates in a layer will be $O(p(n) * \log^k n * p(n)) = O(p(n)^2 * \log^k n)$, which again is bounded by some polynomial in n. We now have a family of structured circuits, each with bounded fan-in and fan-out and of depth $O(\log^{k+1} n)$. □

In the same way we can prove:

Corollary 4 $AC^k \subseteq SNC^{k+1}_{2,2}$.

4 AC^0 circuits in $SNC^1_{2,2}$

We have seen before that $AC^0 \subseteq SNC^1_{2,2}$ as a result of the Structuring Lemma. The so formed SNC^1-circuits need in each layer the square of the total number of gates needed in the corresponding AC^0 circuits. This large increase in the number of gates can be diminished somewhat as our next theorem will show.

Theorem 2 *For any family of circuits $C = \{c_i\}$, $i \geq 1$ with $C \in AC^0$ and each $c_n \in C$ having $O(s(n))$ gates, there exists an equivalent family of circuits $C' = \{c'_i\}$, $i \geq 1$ and $C' \in SNC^1_{2,2}$ with each circuit having $O(\frac{s(n)^2}{\log n})$ gates in a layer.*

Proof: We will restrict ourselves to one layer of the AC^0-circuit transforming it into a $SNC^1_{2,2}$ circuit. For the other layers we can do the same and the constant number of different layers of structured circuit we get can be projected onto each other, forming a layer for the structured circuit representing the total AC^0 circuit. The so formed structured circuit will be in $SNC^1_{c_1,c_2}$, for some constants c_1, c_2. Using Corollary 2 we may transform this into a $SNC^1_{2,2}$ circuit with the number of gates and the depth the same within some constant factor.

With the following we will reduce both fan-out and fan-in as well as structure the circuit. For this we first define the problem a little different. The layer we examine cannot have more than $s(n)$ gates (the total number of gates in the circuit), and thus we can see this layer as $s(n)$ subsets of $\{1, \ldots, s(n)\}$. The subsets S_i $(1 \leq i \leq s(n))$ represent the sets of incoming lines of gate i: $j \in S_i$ if and only if there is a connection from gate j to gate i in the layer under consideration.

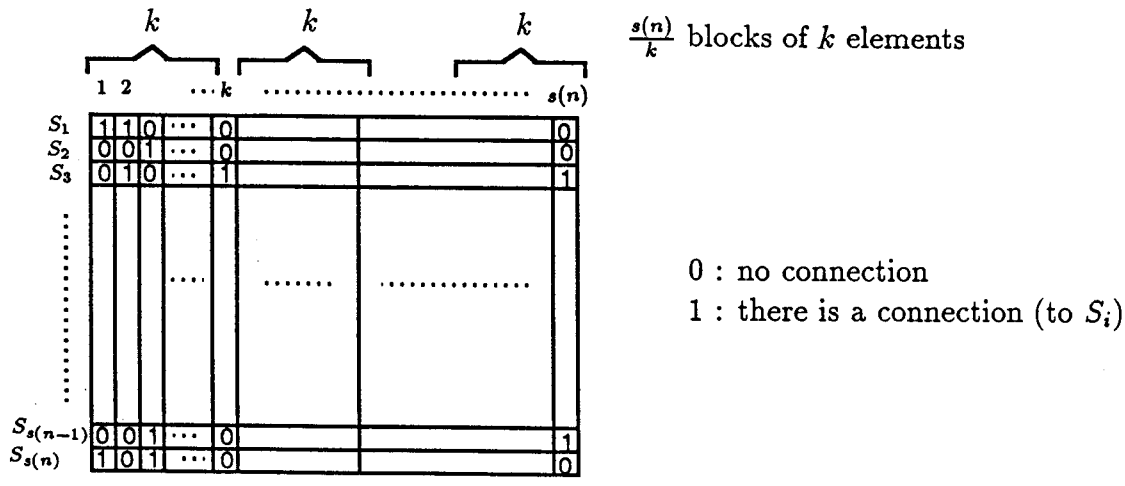

Figure 4: How we can picture the different S_i's.

Since in an AC-circuit every gate uses only one operation on all its elements (AND, OR, or NOT), we will devide the S_i's into three groups:

- a group of S_i's where gate i uses the operation NOT,
- another group of the S_i's where gate i uses the operation AND and
- the S_i's where gate i uses the operation OR.

We will show how to reduce the fan-in of the AND gates; the OR gates may be treated in the same way (increasing the size of the circuit only by a factor two). To this end we devise a structured circuit, such that the fan-out of the "input gates" of the layer is reduced at the same time. We only consider the groups of S_i where the operation used is AND. The order in which the elements of S_i are considered is of no importance (AND and OR are associative). We group the elements of S_i in blocks of size k. The sets S_i now consist of a maximum of $\frac{s(n)}{k}$ blocks of k elements each. We can picture this as in Figure 4.

Now consider one block, say the block corresponding to the first k (possible) input gates. Every S_i $(1 \leq i \leq s(n))$ has such a block, but there are only 2^k different blocks possible. For each column we thus only need to assemble 2^k groups of k elements, instead of $s(n)$ groups of k elements, and when finished every S_i can pick the values (blocks) it needs. Every (input) gate has to be able to reach (or not reach) the 2^k blocks. This can be arranged with a structured circuit of depth $O(\log 2^k) = O(k)$ in the following way. We direct all k inputs to 2^k blocks by each gate having connections to its neighbours. After $\log 2^k$ layers all 2^k blocks will be reached by all k inputs (Figure 5a). The blocks will now have k gates each. Composing the k values in each column (and discarding some on the way) can be done by a structured circuit of depth $O(\log k)$, for all $\frac{s(n)}{k} * 2^k$ blocks in parallel, also just using connections to neighbours (Figure 5b).

For the circuit described above the maximum width (corresponding to the number of gates in a layer of the structured circuit) will be $O(\frac{s(n)}{k} * 2^k * k)$. The depth

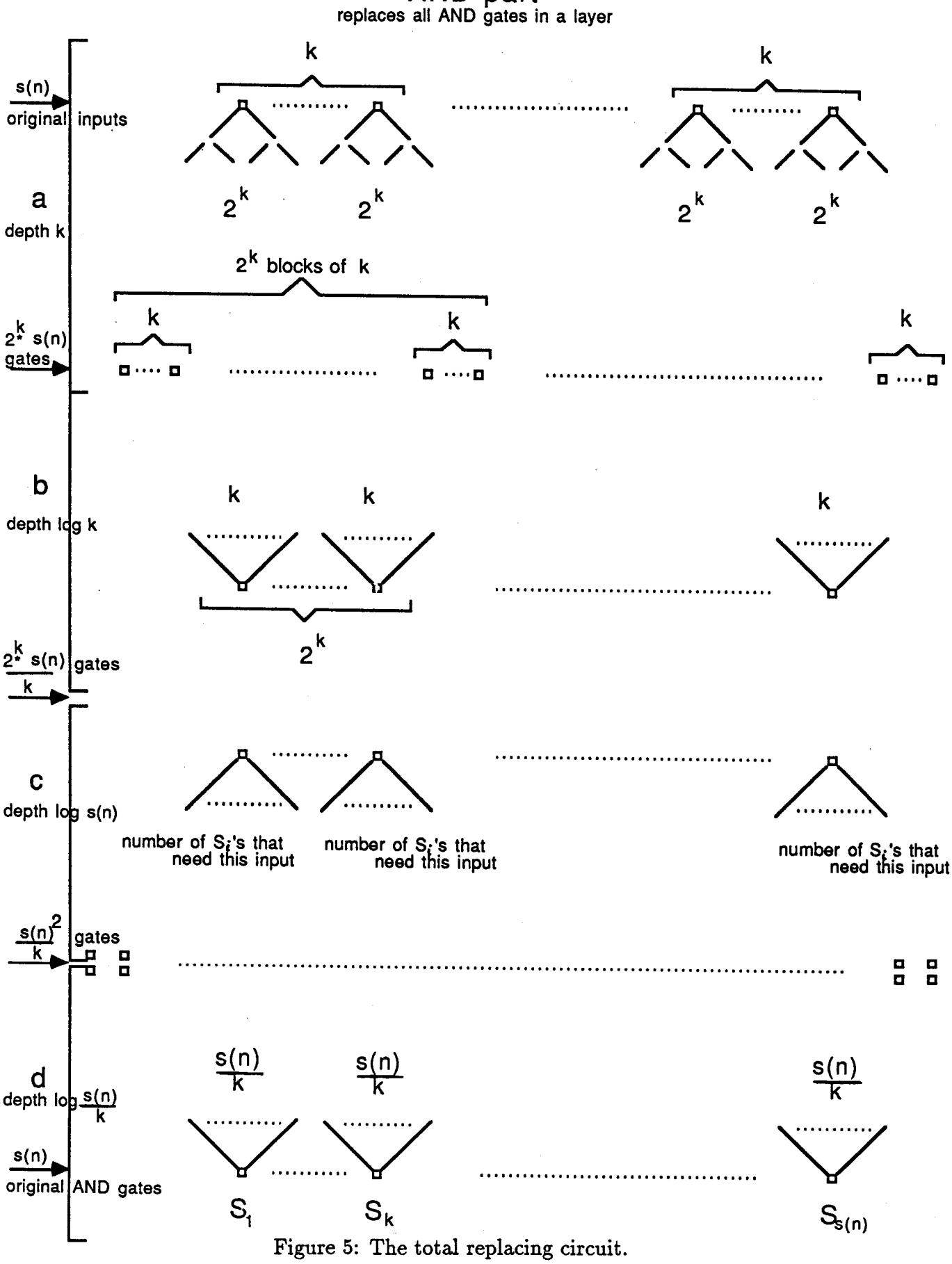

Figure 5: The total replacing circuit.

is $O(k)$ (Figure 5a,b) .

Next we will need a structured circuit to direct the output of the $\frac{s(n)}{k} * 2^k$ blocks (of by now only 1 bit each) to the different gates S_i $(1 \leq i \leq s(n))$. The $s(n)$ different S_i's each need $\frac{s(n)}{k}$ blocks and this leads to a total number of $s(n) * \frac{s(n)}{k}$ necessary input lines. The sum of the fan-out of all blocks together will be smaller than this ($\frac{s(n)^2}{k}$). For each block we add a number of gates equal to the number of S_i's to which the output of the block should be directed. (We cannot say exactly how many this will be for each block seperately but the total number will not exceed $\frac{s(n)^2}{k}$.) For each block this will be no more than $s(n)$ (the total number of S_i's), and these gates can be reached with a structured circuit of depth $O(\log s(n))$, all gates having constant fan-out and fan-in. This part of the circuit can be pictured as in Figure 5c.

Now each S_i can select the neccesary gates as input. In $O(\log \frac{s(n)}{k})$ layers (the number of inputs for each S_i is $\frac{s(n)}{k}$) these inputs can be directed with a structured circuit to each S_i (Figure 5d).

We know that $s(n)$ is some polynomial in n (follows from the definition of AC) and $s(n) = \Omega(n)$ (the circuit has at least as many gates as the number of inputs) and thus $O(\log s(n)) = O(\log n)$. Thus this last part of the circuit has depth $O(\log n)$ and width $O(\frac{s(n)^2}{k})$.

In the above described manner we may replace a layer of the AC^0 circuit by $O(k+\log n)$ layers each being the same and of which the gates have fan-in and fan-out bounded by some constant. (Project the constant number of described layers on each other.) The number of gates (each layer) that we use for this is $\max(s(n)*2^k, \frac{s(n)^2}{k})$. If we take $k = \frac{1}{2}\log s(n)$ then the depth of the structured circuit will be $O(\log n)$ and the number of gates needed for each layer will be $O(\max(s(n)*2^{\frac{1}{2}\log s(n)}, \frac{s(n)^2}{\frac{1}{2}\log s(n)}) = O(\frac{s(n)^2}{\log n})$.

We repeat this for the operation OR and also for the operation NOT. It is neccesary to consider the operation NOT also to make sure that we reduce the fan-out of all the gates (also the ones that have connections with many NOT gates). After working this out for the other layers of the AC^0 circuit we compose them, thus getting a layer with for all gates fan-in and fan-out bounded by some constant and $O(\frac{s(n)^2}{\log n})$ gates. Using Theorem 1 we can find for each circuit an equivalent structured circuit with for all gates fan-in, fan-out ≤ 2, leaving the size of a layer and the depth the same within some constant. When using $O(\log n)$ of these layers, we can simulate the original AC^0 circuit and have found an equivalent $SNC^1_{2,2}$ circuit with $O(\frac{s(n)^2}{\log n})$ gates in a layer.

All operations on the circuits are quite straightforward. Thus the construction of the circuits doesn't become essentially harder and the family will therefore remain uniform. □

Remark: The total number of connections in an AC^0 circuit can be $\Theta(s(n)^2)$. In our SNC^1 circuit replacing an AC^0 circuit we use a total number of connections

that is $O(s(n)^2)$ and hence we probably cannot expect to do better.

Generalizing Theorem 2 to AC^k circuits for arbitrary k, leads to a result that is slightly different from the result when using the Structuring Lemma from section 3. Using the Structuring Lemma we can transform AC^k circuits with $s(n)$ gates into $SNC^{k+1}_{2,2}$ circuits with $O(s(n)^2 \log^k n)$ gates in a layer. We can see an AC^k circuit also as $O(\log^k n)$ subsequent AC^0 circuits, which can each be transformed using Theorem 2 into $SNC^1_{2,2}$ circuits, each having $O(\frac{s(n)^2}{\log n})$ gates in a layer. Putting these $O(\log^k n)$ different layers together by projecting the layers onto eachother will result in a layer with gates with fan-in and fan-out $= O(\log^k n)$. Thus we have transformed the AC^0 circuit into a structured circuit of depth $O(\log^{k+1} n)$ with $O(\frac{s(n)^2}{\log n})$ gates in a layer, all gates with fan-in, fan-out $= O(\log^k n)$. Using Theorem 1 we can reduce the fan-in and fan-out of this structured circuit to 2, increasing the size and depth of the circuit. Finally this results in a stuctured circuit of depth $O(\log^k n \log\log n)$ (a little larger than when using the straight forward method), with $O(s(n)^2 \log^{k-1} n)$ gates in a layer (a factor $\log n$ less) all gates with fan-in, fan-out ≤ 2. thus the total number of gates is $O(s(n)^2 \log^k n \log\log n)$ instead of $O(s(n)^2 \log^{k+1} n)$.

5 Some problems in SNC^1

In this section we will show some basic arithmetic operations (that are in NC^1) to be in SNC^1. In doing so we will try to keep the number of gates necessary in the SNC^1-circuits as small as possible.

5.1 Parallel Prefix

The Parallel Prefix problem is as follows: Given $x_1, x_2, \cdots, x_n$, and some associative operation $*$ (in our case generally $\wedge$ or $\vee$), calculate the n prefix products: $x_1, x_1 * x_2, x_1 * x_2 * x_3, \cdots, x_1 * x_2 * \cdots * x_n$. This is possible with a (unstructured) circuit of depth $O(\log n)$ and size $O(n)$ as follows [LaFi]: (We assume n to be a power of 2)
Input is $x_1, x_2, \cdots, x_n$. We form the products $x_{i-1} * x_i$ for i even, save the value x_{i-1}, and go into recursion with the products $x_{i-1} * x_i$. As we go deeper in recursion the size of the products will increase. When returning from this recursion we will find the products $x_1 * x_2 * \cdots * x_i$ for i even, and we can easily form the products $x_1 * x_2 * \cdots * x_{i+1}$ from $x_1 * x_2 * \cdots * x_i$ and x_{i+1}. See Figure 6. We can easily structure this circuit when we are not concerned about increasing the number of connections per gate. For this purpose we move the inner block ($C(\frac{n}{2})$) over to the right and fill the left half with the remaining lines. Now we can take the number of gates per layer n (less is impossible because the first and the last layers of the circuit need at least n gates). We can achieve this by adding extra (dummy) gates on the lines that have less gates than others, and thus filling up the number of gates per layer to n (this way a lot of gates will be formed where fan-in = fan-out = 1). Our next move will be to project all layers onto one another to assure that all layers will be the same. With this kind of layers we can built a structured circuit (all layers are

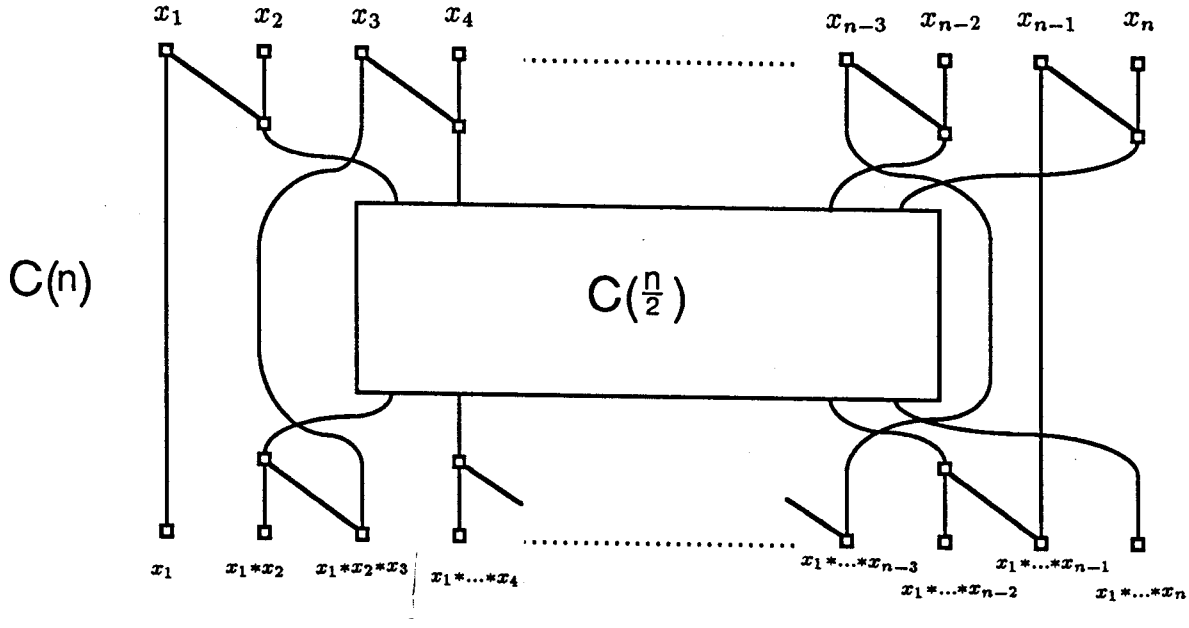

Figure 6: Circuit for parallel prefix.

the same) of depth $O(\log n)$ and width n.

The correctness of this (structured) circuit follows from the correctness of the unstructured circuit, as we have not changed the original circuit essentially. Thus this structured circuit solves the Parallel Prefix problem. In the original circuit all gates have fan-in, fan-out ≤ 2 and we only add gates with fan-in=fan-out=1 and, considering that we take $O(\log n)$ of these layers together we can easily see that we will get fan-in and fan-out of $O(\log n)$ for all gates in the circuit.

Using Theorem 1 from section 3 we can reduce the fan-in, fan-out of $O(\log n)$ to 2. This will give us a structured circuit for the Parallel Prefix problem of depth $O(\log n \log\log n)$ and with $O(n \log n)$ gates each layer.

Theorem 3 *There exists a structured circuit for the Parallel Prefix problem with depth $O(\log n \log\log n)$ and $O(n \log n)$ gates each layer, and with for each gate both fan-in and fan-out ≤ 2.*

Comparing this result with that of the unstructured circuit we see that the depth has been increased with a factor $\log\log n$ and that the total number of gates of $O(n)$ of the unstructured circuit has become $O(n \log^2 n \log\log n)$ in the structured one. When we examine the original Parallel Prefix circuit closer, we notice that as we go deeper in recursion we will find a decreasing number of gates, and less and less action takes place. The exact position of the lines without gates in those layers is not important, as long as they will be back in position upon returning from recursion. We can use this observation to make our structured circuit more economical.

Theorem 4 *Parallel Prefix $\in SNC^1_{2,2}$ with $O(n)$ gates each layer.*

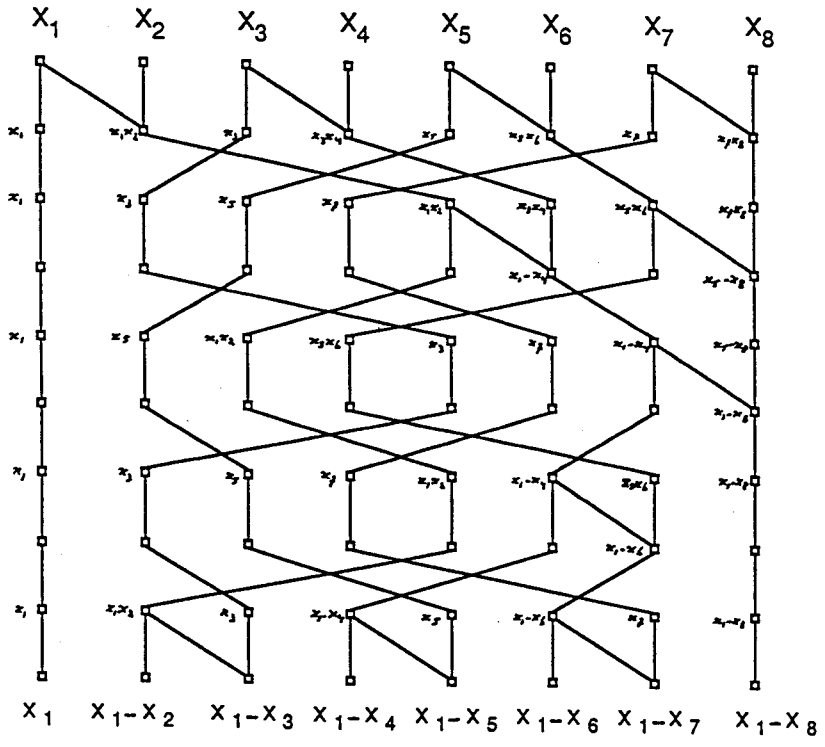

Figure 7: A structured circuit for parallel prefix. ($n = 8$)

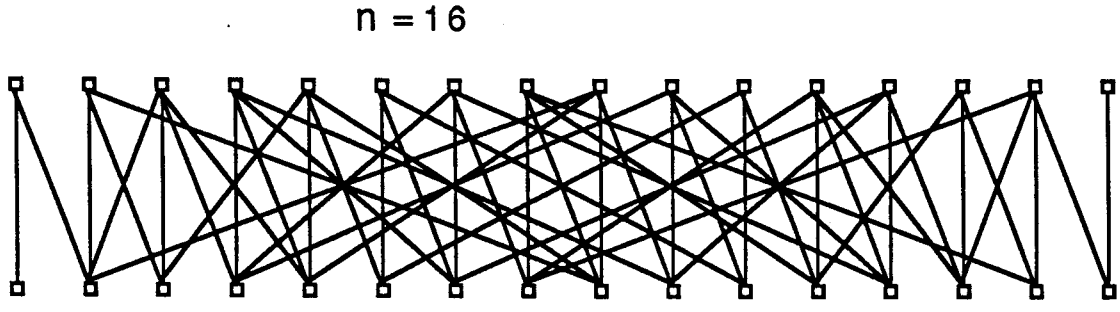

Figure 8: Layout of a layer of the structured circuit for parallel prefix.

Proof: We will show that we can make a circuit for Parallel Prefix with $4\log n - 3$ layers in $SNC^1_{4,4}$ with n gates each layer. From Corollary 2 it then follows that circuit is also in $SNC^1_{2,2}$. The idea here is that in the first $2\log n$ layers the right half of the circuit (and next the right half from that half etc.) is the part where work is done. There increasingly large products of x_i's will be formed. In the other (left) part of the circuit, data will be moved around to make room for the results from the right part. Halfway we reverse this process: the left part brings data to the (increasing) right part, where they will be assembled. We can accomplish this by using in the lower half of the circuit the connections which are reflections of those used in the upper half. This way the results will be moved back, to be at the right place exactly at the right moment. Figure 7 gives an example for $n = 8$.

To be precise, we have a structured circuit of depth $4\log n - 3$, width n, and the

following connections:

$$
\begin{array}{lcll}
g_i & \longrightarrow & g_i & 1 \leq i \leq n \\
g_i & \longrightarrow & g_{i+1} & 1 \leq i \leq n-1 \\
g_{2i} & \longrightarrow & g_{\frac{n}{2}+i} & 1 \leq i \leq n/2 \\
g_{2i-1} & \longrightarrow & g_i & 1 \leq i \leq n/2 \\
g_i & \longrightarrow & g_{2i-1} & 1 \leq i \leq n/2 \\
g_{\frac{n}{2}+i} & \longrightarrow & g_{2i} & 1 \leq i \leq n/2
\end{array}
$$

It is easy to prove that with these connections the maximum fan-in and fan-out of all gates g_i $(1 \leq i \leq n)$ will be four. We have pictured a layer of the so formed structured circuit in Fig 8 (for $n = 16$).

In the following the variable m has been added for notational reasons only. We start the circuit with $m_i = x_i$ $(1 \leq i \leq n)$ and build the results in the m_i's such that at the bottom of the circuit $m_i = x_1 * \ldots * x_i$ $(1 \leq i \leq n)$. All m_i's will start at g_i and be shifted to different gates in the circuit, finally returning to gate g_i. We will use *level* to denote a number of consecutive layers, usually two except for the middle ($\log n$-th) level which consists of one layer and level 0 which consists of zero layers.

Claim 1 *After j levels of the circuit the following holds:*

a. $j = 0$:	input, 0 layers	
	g_i	contains $m_i \quad 1 \le i \le n$
	m_i	$= x_i \quad 1 \le i \le n$
b. $1 \le j < \log n$:	2 layers per level	
	g_i	contains $m_{2ji-(2j-1)n}$ $i > \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{2^j} = n - \frac{n}{2^j}$
	m_{2i-1}	$= x_{2i-1} \quad 1 \le i \le n/2$
	m_{4i-2}	$= x_{4i-3} * x_{4i-2} \quad 1 \le i \le n/4$
	m_{8i-4}	$= x_{8i-7} * x_{8i-6} * x_{8i-5} * x_{8i-4} \quad 1 \le i \le n/8$
	$\vdots$	$\vdots$
	$m_{2ji-2j-1}$	$= x_{2ji-(2j-1)} * x_{2ji-(2j-2)} * \cdots * x_{2ji-2j-1}$ $1 \le i \le n/2^j$
	m_{2ji}	$= x_{2ji-(2j-1)} * x_{2ji-(2j-2)} * \cdots * x_{2ji}$ $1 \le i \le n/2^j$
c. $j = \log n$:	1 layer per level	
	g_n	contains m_n
	m_{2i-1}	$= x_{2i-1} \quad 1 \le i \le n/2$
	m_{4i-2}	$= x_{4i-3} * x_{4i-2} \quad 1 \le i \le n/4$
	$\vdots$	$\vdots$
	$m_{\frac{n}{2}i}$	$= x_1 * \cdots * x_{\frac{n}{2}i} \quad 1 \le i \le 2$
d. $\log n < j \le 2\log n - 1$:	2 layers per level	
	g_i	contains $m_{2^{2\log n-(j+1)}i-(2^{2\log n-(j+1)}-1)n}$ $i > n - \frac{n}{2^{2\log n-(j+1)}}$
	m_n	$= x_1 * \cdots * x_n$
	$m_{\frac{n}{2}}$	$= x_1 * \cdots * x_{\frac{n}{2}}$
	$m_{\frac{n}{4}i}$	$= x_1 * \cdots * x_{\frac{n}{4}i} \quad 1 \le i \le 4$
	$\vdots$	$\vdots$
	$m_{2^{2\log n-(j+1)}i}$	$= x_1 * \cdots * x_{2^{2\log n-(j+1)}i}$ $1 \le i \le \frac{n}{2^{2\log n-(j+1)}}$
	$m_{2^{2\log n-(j+1)}i-2^{2\log n-j}}$	$= x_{2^{2\log n-(j+1)}i-(2^{2\log n-(j+1)}-1)} * \cdots *$ $x_{2^{2\log n-(j+1)}i-2^{2\log n-j}}$ $1 \le i \le \frac{n}{2^{2\log n-(j+1)}}$
	$\vdots$	$\vdots$
	m_{4i-2}	$= x_{4i-3} * x_{4i-2} \quad 1 \le i \le n/4$
	m_{2i-1}	$= x_{2i-1} \quad 1 \le i \le n/2$

Proof: Level 0 represents the input, g_i gets x_i as input, so Claim 1 is true for $j = 0$.

We prove part (b) of the claim by induction on j. At the start of level 1: g_i contains m_i and $m_i = x_i$ (end of level 0). We will use connections $g_i \longrightarrow g_i$ and $g_i \longrightarrow g_{i+1}$. We can now arrange that $m_{2i} = x_{2i-1} * x_{2i}$ (and $m_{2i-1} = x_{2i-1}$). After

this layer g_i still contains m_i . In a next layer we will use the connections $g_{2i} \longrightarrow g_{\frac{n}{2}+i}$ and $g_{2i-1} \longrightarrow g_i$ and thus after this layer g_i will contain m_{2i-n} for $i > \frac{n}{2}$. Thus (b) is true for $j = 1$. Suppose that (b) is true for j, we will show it to be true for $j+1$.

We will leave $m_{2i-1}, m_{4i-2}, \ldots, m_{2^j i-2^j-1}$ unchanged. When observing $m_{2^{j+1}i-2^j}$ and $m_{2^{j+1}i}$ for $1 \leq i \leq n/2^{j+1}$, we find that these are exactly the $m_{2^j i-2^j}$ for $1 \leq i \leq n/2^j$ (as specified by the claim). Thus $m_{2^{j+1}i-2^j} = x_{2^{j+1}i-(2^{j+1}-1)} * \cdots * x_{2^{j+1}i-2^j}$ and $m_{2^{j+1}i} = x_{2^{j+1}i-(2^j-1)} * \cdots * x_{2^{j+1}i}$. We will leave $m_{2^{j+1}i-2^j}$ unchanged and try to arrange:

$m_{2^{j+1}i} := m_{2^{j+1}i-2^j} * m_{2^{j+1}i} = x_{2^{j+1}i-(2^{j+1}-1)} * \cdots * x_{2^{j+1}i}$.

The contents of the different m_i's will then be as according to the Claim (b) with for j the value $j+1$ substituted. We will only be able to arrange this if there is a connection from gate $g_{i''}$ containing $m_{2^{j+1}i-2^j}$ to gate $g_{i'}$ containing $m_{2^{j+1}i}$. We know that gate $g_{i'}$ has $m_{2^j i'-(2^j-1)n}$ and gate $g_{i''}$ contains $m_{2^j i''-(2^j-1)n}$ for $i', i'' > n - \frac{n}{2^j}$. Now suppose that

$2^j i' - (2^j - 1)n = 2^{j+1} i$ then $i' = \frac{(2^j-1)n}{2^j} + 2i$ and
$2^j i'' - (2^j - 1)n = 2^{j+1} i - 2^j$ then $i'' = \frac{(2^j-1)n}{2^j} + 2i - 1$.

It follows that $i' = i'' - 1$ so there is a connection from gate $g_{i''}$ to gate $g_{i'}$, namely $g_{i''} \longrightarrow g_{i''+1}$ for $i', i'' > n - \frac{n}{2^j}$. Now all m_i's have values as they should have at the end of level $j+1$. An extra layer is needed to direct the right values to the different g_i's. At the end of level j g_i contains $m_{2^j i-(2^j-1)n}$ for $i > n - \frac{n}{2^j}$. Consider i with $i = 2i' - n$ (thus i will be even) and $i > n - \frac{n}{2^j}$. We will use the connections

$g_{2k} \longrightarrow g_{\frac{n}{2}+k}$ $(1 \leq k \leq \frac{n}{2})$
$g_{2k-1} \longrightarrow g_k$ $(1 \leq k \leq \frac{n}{2})$.

Because i is even we will use the connections $g_i \longrightarrow g_{\frac{n}{2}+\frac{i}{2}}$. Thus $g_{\frac{n}{2}+\frac{i}{2}}$ gets the m from gate g_i which is $m_{2^j i-(2^j-1)n}$ (at level $j+1$), for $i > n - \frac{n}{2^j}$. Now gate $g_{\frac{n}{2}+\frac{2i'-n}{2}} = g_{i'}$ contains $m_{2^j(2i'-n)-(2^j-1)n} = m_{2^{j+1}i'-2^{j+1}n}$ for $i' > n - \frac{n}{2^{j+1}}$. We may now conclude that at the end of level $j+1$ the data are organized as specified by Claim 1(b), and thus the claim is true for $1 \leq j < \log n$. Note: it is easy to verify that in case we use all the connections $g_{2i} \longrightarrow g_{\frac{n}{2}+i}$ $1 \leq i \leq n/2$ and $g_{2i-1} \longrightarrow g_i$ $1 \leq i \leq n/2$ (see (b)) or all the connections $g_i \longrightarrow g_{2i-1}$ $1 \leq i \leq n/2$ and $g_{\frac{n}{2}+i} \longrightarrow g_{2i}$ $1 \leq i \leq n/2$ (as will be needed for part (d)) no 'collisions' of data will occur.

Part (c) of the claim follows easily by taking $j = \log n - 1$ in (b). All gates then have the right m_i's for after level $\log n$, the only action that remains to be taken is changing m_n. That is $m_n := m_{\frac{n}{2}} * m_n = x_1 * \cdots * x_{\frac{n}{2}} * x_{\frac{n}{2}+1} * \cdots * x_n$. We will use the connection $g_{n-1} \longrightarrow g_n$, because g_{n-1} will contain $m_{\frac{n}{2}}$ and g_n will contain m_n.

The proof of part (d) of the claim is analog to that from part (b) and for that reason left out. □

With this claim it is easy to show that Parallel Prefix $\in SNC^1_{4,4}$. It follows from the used connections that the fan-in and fan-out for each gate have a maximum of 4. We can also see that the above described circuit realises the Parallel Prefix

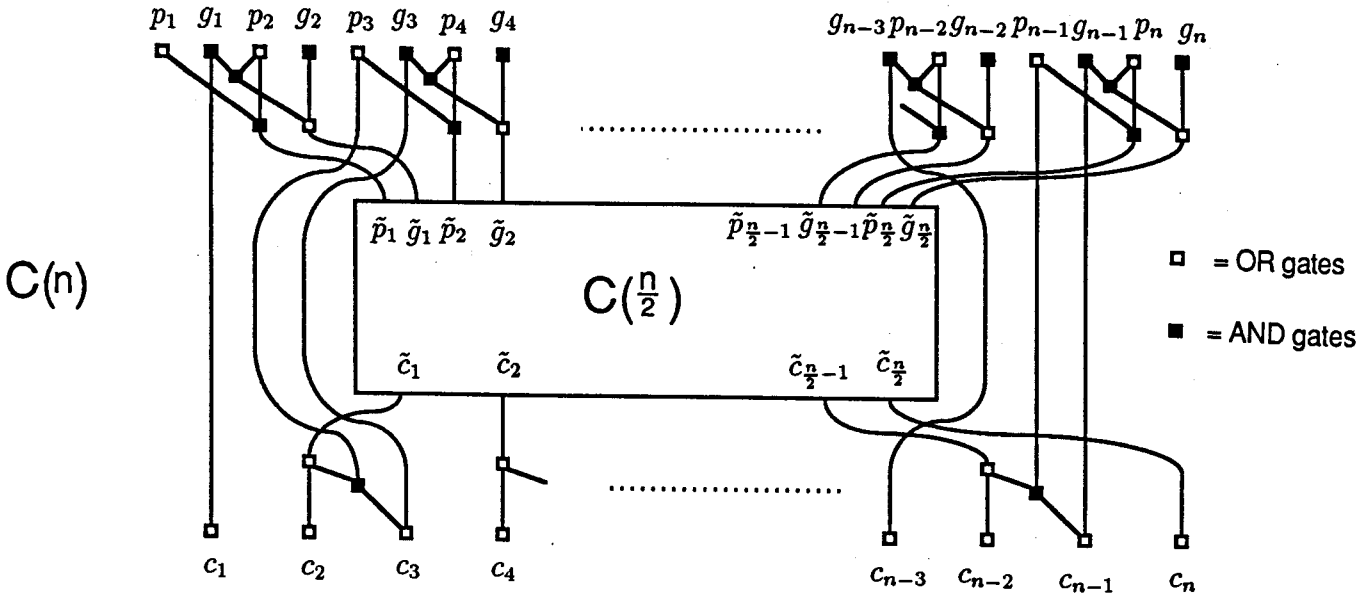

Figure 9: Circuit to produce the carry bits.

problem: take $j = 2\log n - 1$, then with input $x_1, x_2, \ldots, x_n$ the output of gate g_i will be: $x_1 * x_2 * \ldots * x_i$ $(1 \leq i \leq n)$. The width of the circuit is n and the depth is $2(\log n - 1) + 1 + 2(\log n - 1) = O(\log n)$ layers.

Combining this result with Corollary 2 completes the proof. □

Remark: $O(\log n)$ depth and $O(n)$ gates per layer is optimal for circuits in $SNC^1_{2,2}$ as the unstructured circuit already needs depth $O(\log n)$ (and we only contrain ourselves by requiring the circuit to be structured). The width of the structured circuit has to be at least the maximum of the number of in- and output gates, which is n.

5.2 Addition

The addition problem takes two n-bit binary numbers $x_1 \ldots x_n$ and $y_1 \ldots y_n$, and produces its $n+1$-bit binary sum $z_1 \ldots z_{n+1}$.

Theorem 5 *Addition* $\in SNC^1_{2,2}$ *with* $O(n)$ *gates each layer.*

Proof: To add x and y we first form p and g such that $p_i = x_i \vee y_i$ and $g_i = x_i \wedge y_i$. From these we calculate the carry bits c_i $(1 \leq i \leq n)$ with a circuit similar to that for the parallel prefix problem (see Figure 9), knowing that we are able to transform that circuit into a $SNC^1_{2,2}$ cicuit with $O(n)$ gates per layer. We want c_i to be $(c_{i-1} \wedge p_i) \vee g_i$ for all $1 \leq i \leq n$ (where $c_0 = 0$).

First we form $\tilde{p}_i$, $\tilde{g}_i$ $(1 \leq i \leq \frac{n}{2})$ with $\tilde{p}_i = p_{2i-1} \wedge p_{2i}$ and $\tilde{g}_i = (g_{2i-1} \wedge p_{2i}) \vee g_{2i}$. With these $\tilde{p}_i$ and $\tilde{g}_i$ we will produce the corresponding $\tilde{c}_i$, which can be used to calculate the c_i that corresponds to p_i and g_i. We define $\tilde{c}_i$ as $\tilde{c}_i = (\tilde{c}_{i-1} \wedge \tilde{p}_i) \vee \tilde{g}_i$

(and $\tilde{c}_0 = 0$). The $\tilde{c}_i$ that will be formed in this way will be the c_{2i} as can be shown by induction on i in the following way:

$$\begin{aligned}
\tilde{c}_1 &= (\tilde{c}_0 \wedge \tilde{p}_1) \vee \tilde{g}_1 \\
&= (c_1 \wedge p_2) \vee g_2 \\
&= c_2 \\
\tilde{c}_i &= (\tilde{c}_{i-1} \wedge \tilde{p}_i) \vee \tilde{g}_i \\
&= (c_{2i-2} \wedge p_{2i} \wedge p_{2i-1}) \vee (g_{2i-1} \wedge p_{2i}) \vee g_{2i} \\
&= (((c_{2i-2} \wedge p_{2i-1}) \vee g_{2i-1}) \wedge p_{2i}) \vee g_{2i} \\
&= (c_{2i-1} \wedge p_{2i}) \vee g_{2i} \\
&= c_{2i}
\end{aligned}$$

Thus after this step we have computed c_i for i even. Computing c_i for i odd is now simple, since $c_{2i+1} = (c_{2i} \wedge p_{2i+1}) \vee g_{2i+1}$.

Once the carry bits are known it is easy to form the output bits z_i from the p_i and g_i and these carry bits (c_{i-1}).

The explained parts can be put together to form a structured circuit of depth $O(\log n)$ and with $O(n)$ gates in a layer and fan-in and fan-out bounded by some constant. Using Corollary 2 we can reduce the fan-in and fan-out to 2 without changing the bounds on depth and size in order of magnitude. □

Note that this result is optimal as for a circuit with gates with fan-out and fan-in ≤ 2 we need depth at least $O(\log n)$ to compose all inputs and for structured circuits $O(n)$ gates in a layer is minimal.

5.3 Multiplication

So far we have only considered problems which are not just in NC^1, but in AC^0 as well. Multiplication however is in NC^1 and not in AC^0 as shown in [FuSaSi], and we will show it to be in SNC^1. In section 3 we have shown AC^0 to be a subset of SNC^1 (Corollary 4, take $k = 0$), and thus when multiplication is in SNC^1 we may conclude that AC^0 is a proper subset of SNC^1.

The multiplication problem takes two n-bit binary numbers as input and produces its $2n$-bit binary product as output. There are several circuits of depth $O(\log n)$ known to solve this problem, each using a different method to produce the product, resulting in a different number of necessary gates. We list three of them, the Three for two trick ([Of, Wa, KaRa, We]), the method from Karatsuba and Ofman ([Sa, We]) and the best bounded fan-in circuit known, from Schönhage and Strassen([ScSt, AhHoUl, We]). See Table 5.3. The second column gives the size of the circuits needed when the circuit is allowed $O(\log n)$ depth. The third column gives the necessary number of gates in a layer when we structure these circuits using the Structuring Lemma from section 3, which gives us circuits of depth $O(\log n \log\log n)$. The last column indicates the necessary number of gates when structuring the circuits using depth only $O(\log n)$ by making use of the special properties of the circuits considered. So far we only have been able to do this for the Three for two trick-circuits.

Method Depth →	Unstructured $O(\log n)$ number of gates	Structured $O(\log n \log\log n)$ gates/layer	Structured $O(\log n)$ gates/layer
Three for two trick	$O(n^2)$	$O(n^2 \log n)$	$O(n^2)$
Karatsuba and Ofman	$O(n^{\log 3})$	$O(n^{\log 3} \log n)$	?
Schönhage-Strassen	$O(n \log n \log\log n)$	$O(n \log^2 n \log\log n)$	?

5.3.1 Three for two trick

We can transform the multiplication problem into the problem of n additions of $2n$-bit numbers. When taking three $2n$-bit numbers at a time and adding them bit by bit we get 2-bit numbers for each three bits we add. The upper bits together form a $2n$-bit number (the upperbit number) , and so do the lower bits (the lowerbit number). Adding the upperbit number followed by a zero to the lowerbit number will give us the sum of the original three numbers. Thus we have transformed (in a constant number of steps) the addition of three numbers to the addition of two numbers.

Using the Structuring Lemma we easily get a $O(\log^2 n)$ depth, structured circuit with $O(n^3 \log n)$ gates each layer. We can fairly easily reduce this to a structured circuit with $O(\log n \log\log n)$ depth and $O(n^2 \log n)$ gates each layer by adjusting the original circuit a little: reducing the fan-out of the gates to a constant by replacing the first layer which directs the inputs to the n different sums by $O(\log n)$ layers, and some tree structure to direct all inputs to the numbers to be added.

To obtain a structured circuit of depth $O(\log n)$ we study the NC^1 circuit more closely, and we are particularly interested in layers where the fan-out is not bounded. The first layers are concerned with forming the n numbers to be added. This results in a fan-out of $O(n)$ for the input gates, and as we have seen before we can replace this layer by $O(\log n)$ layers, where we direct each input only to a constant number of gates, giving the same total result. This can easily be a structured circuit, of $O(n^2)$ gates each layer and depth $O(\log n)$. We are left with n numbers to be added. We use $2n$ bits to represent each of these numbers. This is neccesary for structuring the circuit, as the addition of small (n bits) and large ($2n-1$ bits) numbers has to fit within the same connections. For the next $O(\log n)$ layers (three for two trick) we can easily use a type of layer which takes three numbers ($2n$ bits each) and makes two out of them, shifting them to the left as far as possible. After $O(\log n)$ layers just two numbers will be left, and we can use a stuctured circuit depth $O(\log n)$, size $O(n)$ to add these two (as in section 5.2). In total we get a constant number of structured circuits of depth $O(\log n)$ and gates with constant fan-in and fan-out. We can according to Theorem 1 (using projections, adding gates where input=output, etc.) make a structured circuit of $O(\log n)$ depth out of these with bounded fan-in and fan-out and $O(n^2)$ gates each layer. This leads to:

Theorem 6 *For the Three for two trick algorithm for multiplication there exist structured circuits with bounded fan-out and fan-in of depth $O(\log n)$ and with $O(n^2)$ gates each layer.*

5.3.2 Karatsuba and Ofman

The number of gates needed in the former circuit for multiplication can be reduced by using a different algorithm for multiplication. Suppose we want to multiply $x = x_1 \dots x_n$ and $y = y_1 \dots y_n$. Let $x' = x_1 \dots x_{\frac{n}{2}}$, $x'' = x_{\frac{n}{2}+1} \dots x_n$ and $y' = y_1 \dots y_{\frac{n}{2}}$ and $y'' = y_{\frac{n}{2}+1} \dots y_n$. The algorithm for multiplication from Karatsuba and Ofman is based on a divide and conquer approach and the observation that:

$$x * y = 2^n * p_1 + 2^{\frac{n}{2}} * (p_3 - (p_1 + p_2)) + p_2$$
$$\text{where} \quad p_1 = x' * y', \quad p_2 = x'' * y'', \quad p_3 = (x' + x'') * (y' + y'')$$

With this a multiplication of size n can be replaced by three multiplications of size $\frac{n}{2}$. The multiplications can be done in parallel, for the additions we can do something special as to make them work not only in size $O(n)$, but also in constant depth (the representation of numbers needed for this has been studied by Melhorn and Preparata [MePr]).

Theorem 7 *(Wegener 1987) The Karatsuba and Ofman algorithm for multiplication can be implemented such that the resulting circuit has size $O(n^{\log 3})$ and depth $O(\log n)$.*

When examining the circuit closer we observe that all gates have bounded fan-in and bounded fan-out. As the circuit has depth $O(\log n)$, we can transform this into a structured circuit of depth $O(\log n)$ with $O(n^{\log 3} \log n)$ gates in a layer and fan-out, fan-in $= O(\log n)$ and thus:

Lemma 3 *Multiplication $\in SNC^1_{O(\log n), O(\log n)}$ with $O(n^{\log 3} \log n)$ gates each layer.*

Using the Structuring Lemma we can conclude the following.

Theorem 8 *For the Karatsuba and Ofman algorithm for multiplication there exist structured circuits with bounded fan-out and fan-in of depth $O(\log n \log\log n)$ and with $O(n^{\log 3} \log n)$ gates each layer.*

The algorithm for multiplication from Karatsuba and Ofman uses a divide and conquer approach to solve the problem. It looks as though circuits using this cannot easily be structured without increasing the depth of the circuit, unless only one part of the division is not trivial (that is, really is a smaller instance of the whole problem). For the circuit for the parallel prefix problem this is the case so structuring that is possible without significantly increasing the depth of the circuit. For the Karatsuba and Ofman algorithm however, all parts in which we divide the problem are non-trivial instances of the original problem and (even if this would only be two subproblems) thus we cannot easily structure the circuit without increasing the depth of the circuit.

5.3.3 Schönhage-Strassen

The Schönhage and Strassen algorithm for multiplication makes use of the Fast Fourier Transform.

Theorem 9 *(Wegener 1987) The algorithm of Schönhage and Strassen leads to a circuit for multiplication of size $O(n \log n \log\log n)$ and depth $O(\log n)$.*

The circuit of Schönhage and Strassen as described in [We] uses gates with bounded fan-out as well as fan-in only. Using the Structuring Lemma once more we conclude the following:

Theorem 10 *For the Schönhage and Strassen algorithm for multiplication there exist structured circuits with bounded fan-out and fan-in of depth $O(\log n \log\log n)$ and with $O(n \log^2 n \log\log n)$ gates each layer.*

6 Summary and directions for further research

In this paper we have introduced an interesting class of problems solvable with structured circuits, called SNC. In the general case, using SNC circuits instead of NC or AC circuits will give an increase in depth of $O(\log n)$. This is not a very high price compared to the amount of simplicity using structured circuits introduces. Remember that we just needed one layer of the structured circuit with programmable gates. Then we would direct the output of the layer back into the layer itself and after repeating this process a number of times the problem will be solved. Thus the $O(\log n)$ increase in the depth of the circuits is in fact only an increase in time and not in the number of gates. In the special cases we have considered (Parallel Prefix, Addition, Multiplication) even this extra factor $\log n$ of depth can be avoided.

Within the class SNC there are several subclasses which differ in the maximum fan-out and fan-in of the gates and in the depth of the circuits: we defined $SNC^k_{p(n),q(n)}$ to be the class of families of structured circuits with depth $O(\log^k n)$ and fan-out $\leq p(n)$ and fan-in $\leq q(n)$. We have related different SNC classes to each other and showed how to transform non-structured circuits into structured ones, thus being able to relate SNC to NC and AC. We can summarize the relationships among the classes as below:

$$AC^k \subseteq SNC^{k+1}_{2,2} \subseteq SNC^{k+1} \subseteq NC^{k+1} \subseteq SNC^{k+1}_{\cdot,\cdot} \subseteq AC^{k+1} \ (\mathrm{k} \geq 0)$$

From these relationships we may conclude that $SNC = AC = NC$.

Moreover we showed multiplication to be in $SNC^1_{2,2}$ while we know that multiplication is not in AC^0. Thus we may conclude that $AC^0 \neq SNC^1_{2,2}$.

Besides multiplication we have shown some other useful problems to be in $SNC^1_{2,2}$. These are parallel prefix and addition, for both of which we can build structured circuits with only $O(n)$ gates in a layer. This number of gates in a layer is minimal

for structured circuits as the size of the input (n) puts a minimum on the width of the circuits.

Next we will indicate some directions for further research concerning classes of SNC.

In [Ru2] we find that NC^k for $k \geq 2$ is the same class when defined with U_E, U_D, U_{BC} or U_{E^*} uniformity, where U_{BC}-uniform means logspace uniform which is the uniformity that we have used to define SNC. It would be interesting to investigate if the same holds for SNC^k.

Furthermore we could study the class of structured Alternating Turing machines ($SATM$ for short). This would be the equivalent of the ATM's for which $NC^k = ATM^k$ such that $SNC^k = SATM^k$. For an explanation of ATM's see e.g. [Ru2, Ru1, ChKoSt, KaRa].

Finally for NC^1 (U_{E^*}-uniform) there is a complete problem known (under AC^0 reduction), namely *the boolean formula value problem* ([Bu]). A question still open is whether there are (natural) complete problems for SNC^1 as well.

References

[AhHoUl] Aho, A.V., J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[Bu] Buss, S.R., The boolean formala value problem is in ALOGTIME, *Proc 19th Annual ACM Symp. on Theory of Computing*, 1987, pp.123-131.

[ChKoSt] Chandra, A.K., D.C. Kozen, L.J. Stockmeyer, Alternation, *JACM* 28(1981), pp.114-133.

[Co1] Cook, S.A., Towards a complexity theory of synchronous parallel computation, *Enseign. Math.* 27(1981), pp.99-124.

[Co2] Cook, S.A., A taxonomy of problems with fast parallel algorithms, *Inform. and Control* 64(1985), pp.2-22.

[FuSaSi] Furst, M., J.B. Saxe and M. Sipser, Parity, circuits and the polynomial time hierarchy, *Math. Systems Theory* 17(1984), pp.13-28.

[KaRa] Karp, R.M. and V.L. Ramachandran, A survey of parallel algorithms for shared-memory machines, in: J. van Leeuwen (Ed.)*Handbook of Theoretical Computer Science*, North-Holland Publ. Comp., Amsterdam (to appear).

[LaFi] Ladner, R.E. and M.J. Fischer, Parallel prefix computation, *JACM* 27(1980), pp.831-838.

[MePr] Mehlhorn, K. and F.P. Preparata, Area-time optimal VLSI integer multiplier with minimum computation time, *IC* 58(1983), pp.137-156.