Circuit Complexity:

from the Worst Case to the Average Case*

Andreas Jakoby †

Rüdiger Reischuk

Christian Schindelhauer

Technische Hochschule Darmstadt ‡

May 1994

Abstract

In contrast to machine models like Turing machines or random access machines, circuits are a rigid computational model. The internal information flow of a computation is fixed in advance, independent of the actual input. Therefore, in complexity theory only worst case complexity measures have been used to analyse this model. Concerning the circuit size this seems to be the best one can do.

The delay between feeding the input bits into a circuit and being able to read off the output bits at the output gates is usually measured by the depth of the circuit. One might try to take advantage of favourable cases in which the output values are obtained much earlier. This will be the case when critical paths, e.g. paths between input and output gates of maximal length, have no influence on the final output.

Inspired by recent successful attempts to develop a meaningful average case analysis for TM computations [Levi86, Gure91, BCGL92, ReSc93a], we follow the same goal for the circuit model. For this purpose, a new comlexity measure for the internal delay is defined, called *time*. This may be given implicitely or explicitely, where in the latter case a gate has to signal that it is "ready", e.g. has computed the desired result. Using a simple coding both models are shown to be equivalent. By doubling the number of gates any circuit with implicit timing can easily be transformed to an equivalent circuit with explicit time signals.

Based on the notion of *time,* two average case measures for the circuit delay are defined. The analysis is not restricted to uniform distributions over the input space, instead a large class of distributions will be considered. For this purpose, a complexity notion is needed for distributions. We define a measure based on the circuit model by considering the complexity of circuits with uniform distributed random input bits that generate such distributions.

Finally, this new approach is applied to concrete examples. We derive matching lower and upper bounds for the average circuit delay for basic functions like the `OR`, `ADDITION`, `THRESHOLD` and `PARITY`. It will be shown that for `PARITY` the average delay remains logarithmic. In many cases, however, an exponential speedup compared to the worst case can be achieved. For example, the average delay for n-Bit-`ADDITION` is of order $\log\log n$. The circuit designs to achieve these bounds turn out to be very different from the standard ones for optimal worst case results.

* A preliminary version was presented at 26th STOC'94

† supported by DFG Research Grant Re 672-2

‡ Institut für Theoretische Informatik, Alexanderstraße 10, 64283 Darmstadt, Germany
email: jakoby / reischuk / schindel @ iti.informatik.th-darmstadt.de

1 Introduction

For combinatorial circuits the depth is usually taken as a measure for the computational delay. A close relationship between circuit depth and the time of several parallel machine models has been shown. Note that depth as well as the parallel time considered so far is a worst case measure.

For sequential machines and algorithms a meaningful concept of average case complexity has been proposed recently, which differs from the simple approach by taking the expectation [Levi86, Gure91, BCGL92, ReSc93a]. Distributional and average case complexity classes based on the Turing machine model have been defined and equipped with a notion of reducibility with similar properties as for worst case classes. It was even possible to obtain the same tight hierarchies as in the worst case [ReSc93b].

Restricting an average case analysis to just the uniform distribution is of limited value. But it has been observed that one should not allow arbitrary distributions since there exist distributions for which the average case equals the worst case [LiVi92, Milt91]. Thus, when considering average case complexity classes one has to restrict the set of possible distributions somehow. For this purpose, the notions of *computable* [Levi86], *sampleable* [BCGL92], and *rankable* [ReSc93a] distributions have been introduced, which set a time limit for a deterministic or probabilistic Turing machine to provide information about the individual distribution.

Since circuit complexity is another fundamental concept we like to investigate average case complexity also with respect to this model. Immediately the question arises whether this makes sense at all since the circuit model is a rigid computational model, which in contrast to machines with a flexible control structure cannot take advantage in favourable situations in an obvious way. Thus, the first thing to be done is to define a notion for the average delay of circuits. For this purpose, we introduce the new complexity measure *time* for the circuit model (compare [Krap78]).

1.1 An Implicit Definition of Time

Definition 1 *Let B_n^m denote the set of Boolean functions $f : \{0,1\}^n \to \{0,1\}^m$ and $B_n = B_n^1$. $\mathcal{D}_n$ denotes the set of all probability distributions μ on $\{0,1\}^n$. The uniform distribution on $\{0,1\}^n$ that gives equal probability to each of the 2^n possible input vectors is denoted by $\mu_n^{\mathbf{uni}}$. For $\mu \in \mathcal{D}_n$ $\mathrm{Supp}(\mu)$ is the set of all vectors $x \in \{0,1\}^n$ with a nonzero probability $\mu(x)$. Let us call a distribution in $\mathcal{D}_n$ strictly positive if $\mathrm{Supp}(\mu) = \{0,1\}^n$.*

Circuits may be defined over an arbitrary finite basis. Let $Cir(f)$ denote the set of all circuits over the given basis that compute f. For a circuit in $Cir_\mu(f)$ it is only required that its results equal $f(x)$ for input vectors $x \in \mathrm{Supp}(\mu)$. Circuits that compute the same function will be called equivalent.

Information can be propagated faster in a circuit if, for example, one of the inputs of an `OR`-gate is already available and has the value 1. Then its output is determined as 1 independent of the value of the other input. More formally, we define a function $time : \{0,1\}^n \to \mathbb{N}$ for each gate v of a circuit C. It specifies for each input x the step when v can compute its result $\mathrm{res}_v(x)$ using the values of its predecessors.

Definition 2 *Let C be a circuit and v be a gate of C. For input gates and constant gates v set $time_v(x) := 0$. For an internal nonconstant gate v with k direct predecessors $v_1, \ldots, v_k$ define $time_v(x) \;:=\; 1 + \min\{t \mid$ the values $res_{v_i}(x)$ with $time_{v_i}(x) \le t$ uniquely determine $res_v(x)\}$. For the circuit C itself with output gates $y_1, \ldots, y_m$ we define the global time function by*

$$time_C(x) \;\;:=\;\; \max_i \; time_{y_i}(x) \; .$$

For example, the delay of an `OR`-gate v with predecessors v_1 , v_2 is given by

$$time_v(x) \;:=\; 1 + \begin{cases} \max\{time_{v_1}(x), time_{v_2}(x)\} & \text{if } \mathrm{res}_{v_1}(x) = \mathrm{res}_{v_2}(x) = 0, \\ \min\{time_{v_i}(x) \mid \mathrm{res}(v_i) = 1\} & \text{else.} \end{cases}$$

This model specifies the delay, how long an internal gate has to wait until it is able to compute its value, only implicitly. A gate does not signal the fact that it has finished to its successors or to the outside world. We will call such circuits **implicitly timed circuits, IT-circuits**.

For the following analysis we will choose the standard basis of `AND`, `OR` and `NOT` gates. Although for a `PARITY`-gate there is no improvement of the average delay compared to the worst case – one always needs both inputs to determine the value of the gate – any basis must contain gates with a speedup for the average case, otherwise it would not be complete. In fact, an independence property of the delay with respect to the chosen basis similar to the depth for the worst case can be shown:

Theorem 1 *For any pair of finite bases B_1, B_2 there exists a constant k with the following property: For every circuit C_1 built from gates of B_1 there exists an equivalent circuit C_2 over B_2 with*

$$time_{C_2} \;\leq\; k \cdot time_{C_1} \;.$$

Proof: It suffices to show that an arbitrary basis B_1 can be simulated with constant delay by the standard basis and that the standard basis can be simulated with constant delay by an arbitrary basis B_2 .

The first property can be seen as follows: For a Boolean function f corresponding to a gate in B_1 compute each prime implicant by a tree of $\wedge$-gates and take the conjunction over all prime implicants (by a tree of $\vee$-gates) to get a circuit C_f .

Whenever a partial input assignment uniquely determines the value of f – either as 0 or as 1, let us first assume as 1 – there is a prime implicant of f for which all variables are defined. Thus the corresponding $\wedge$-tree evaluates to 1. This implies that the $\vee$-tree for the prime implicants of f evaluates *early* to 1.

In the other case, when the value of f is 0 each prime implicant of f must contain a literal with value 0. Thus each $\wedge$-tree can be evaluted early to 0 and therefore also the $\vee$-tree of all prime implicants of f. Therefore, whenever a partial input assignment determines the value of f the circuit C_f generates an output and thus computes f for this partial assignment.

Replacing for each function f in B_1 all gates of that type by C_f the delay of this new circuit over the standard basis is at most a constant factor larger, where the constant is bounded by the maximal depth of the circuits C_f .

Now consider a basis B_2 that has to simulate the standard basis. The 16 Boolean functions f of 2 inputs can be subdivided into 3 different types: the `AND`-type ($f(x_1, x_2) = (x_1^\alpha \wedge x_2^\beta)^\gamma$ for some $\alpha, \beta, \gamma \in \{0,1\}$), the `PARITY`-type ($f(x_1, x_2) = x_1 \otimes x_2 \otimes \alpha$ with $\alpha \in \{0,1\}$), and the elementary functions (0, 1, x_1 , x_2 , $\overline{x_1}$, $\overline{x_2}$).

B_2 must contain a function $b : \{0,1\}^q \to \{0,1\}$ such that $b(p_1(x_1, x_2), \ldots, p_q(x_1, x_2)) = a(x_1, x_2)$, where the p_i are functions of elementary or `PARITY`-type and a is an `AND`-type function. Otherwise, B_2 could not compute any function of type `AND` since the elementary and `PARITY`-type functions are closed under composition.

For such a b there even exist functions e_i such that $b(e_1(x_1, x_2), \ldots, e_q(x_1, x_2)) = a'(x_1, x_2)$, for elementary function e_i and a' of type `AND`. This can be seen by selecting `PARITY` and elementary functions in an appropriate way.

By definition of *time* the function $b \circ (e_1, \ldots, e_q)$ has the same delay behavior as a'. Applying De-Morgans rules yields realization of AND- and OR-gates. ∎

If a circuit is fed with a distribution μ at its input gates that is not strictly positive an internal gate may be able to determine its value even faster because certain input patterns cannot occur at its direct predecessors. In this case the timing function may yield smaller values and should include μ as an additional parameter. For example, given a distribution with $\mu(0^n) = 0$ the computation of the OR-function becomes trivial. We will not elaborate on such effects here.

To realize a combinatorial circuit in practice a simple way is to construct a corresponding synchronous acyclic VLSI-circuit. In every step of the computation each gate v performs a state transition depending on the current states of its predecessors. Note that by step $time_v(x)$ its Boolean state has converged to the value $\mathrm{res}_v(x)$. Little effort seems to have been done by hardware designers so far to achieve speedups based on these observations.

1.2 Gates with Explicit Timing Information

There are several ways to make this timing information explicit. On a conceptual level the easiest would be to extend the model to a three-valued logic with an additional symbol "?" meaning "I don't know yet". In case of the ORand AND-function, for example, we could get the following function tables

AND	?	0	1
?	?	0	?
0	0	0	0
1	?	0	1

OR	?	0	1
?	?	?	1
0	?	0	1
1	1	1	1

All nonconstant internal gates are initialized by the value "?". As soon as a gates changes this value to either 0 or 1 this will correctly specify $\mathrm{res}_v(x)$. Let us call such three-valued circuits **explicitly timed circuits, ET-circuits**.

In a similar way, asynchronous hardware has been synchronized by introducing the notion of *self-timed circuits,* see for example [DGY89, LBS93]. Since our main goal is to analyse the average delay with respect to a distribution on the input space to simplify the exposition we assume in the following a synchronous clocked system of Boolean gates. With the obvious modifications the results can also be extended to asynchronous systems. Formally, the result function for each gate v computing the Boolean function φ_v gets as a second parameter the time step t and for $t = 1, 2, \ldots$

$$\mathrm{res}_v(x,t) \;=\; \varphi_v(\mathrm{res}_{v_1}(x,t-1), \ldots, \mathrm{res}_{v_k}(x,t-1)) \;.$$

To realize an ET-circuit in practice one could use a binary encoding of the three-valued logic. This encoding maps 0,1,? to disjunct sets of binary tupels. A straightforward solution is to add one extra bit, the OK-bit, which indicates whether the other bit, the Data-bit, already specifies the correct result. This way one gets $\mathrm{code}(?) = \{(0,0),(0,1)\}$, $\mathrm{code}(0) = \{(1,0)\}$, $\mathrm{code}(1) = \{(1,1)\}$).

1.3 Equivalence of the Timing Models

Using this encoding it is fairly easy to realize the extended 3-valued OR- and AND-functions by circuits of small size and depth. However, there is another code which is much more efficient, it only requires 2 parallel gates for computing the coded version of OR or AND (compare [DGY89]). Instead of the pair of bits (OK,Data) we choose (OK $\wedge$ Data, OK $\wedge \overline{\text{Data}}$). Thus 0 is coded as $(0,1)$, the value 1 as $(1,0)$ and ? as $(0,0)$. The tuple $(1,1)$ does not occur. The following table describes the realiziation of OR and AND, whereas the negation is given for free. For $z = x \vee y$ the codeword (z_1, z_2) is computed from the codewords $(x_1, x_2), (y_1, y_2)$ of x and y as

$$z_1 = x_1 \vee y_1 \quad \text{and} \quad z_2 = x_2 \wedge y_2 \ .$$

x	(x_1, x_2)	y	(y_1, y_2)	z	(z_1, z_2)
0	(0,1)	0	(0,1)	0	(0,1)
1	(1,0)	*	$\neq$ (1,1)	1	(1,0)
?	(0,0)	$\neq$ 1	$\neq$ (1,*)	?	(0,0)

Here, $*$ denotes an arbitrary value. For $z = x \wedge y$ we use

$$z_1 = x_1 \wedge y_1 \quad \text{and} \quad z_2 = x_2 \vee y_2 \ .$$

x	(x_1, x_2)	y	(y_1, y_2)	z	(z_1, z_2)
0	(0,1)	*	$\neq$ (1,1)	0	(0,1)
1	(1,0)	1	(1,0)	1	(1,0)
?	(0,0)	$\neq$ 0	$\neq$ (*,1)	?	(0,0)

This coding leads to

Theorem 2 *Every 3-valued ET circuit C (over the standard basis of extended* AND, OR, NOT*-gates) can be simulated by a conventional 2-valued circuit C' of the same depth and delay as C and at most twice its size.*

Because of this property we can restrict the following investigations to IT-circuits over the standard Boolean domain. If necessary the timing information can be obtained by a simple modification.

2 Average Case Measures

Definition 3 *For a function $t : \{0,1\}^n \to \mathbb{N}$ and a probability distribution $\mu : \{0,1\}^n \to [0;1]$ let*

$$\mathrm{E}_\mu(\mathrm{t}) \ := \ \sum_{\mathrm{x} \in \{0,1\}^n} \mathrm{t}(\mathrm{x}) \ \mu(\mathrm{x})$$

be the expectation of t with respect to μ. If D is a set of probability distributions we define

$$etime(f, D) \ := \ \max_{\mu \in D} \ \min_{C \in Cir_\mu(f)} \ \mathrm{E}_\mu(\mathrm{time_C})$$

as the optimal expected circuit delay of f with respect to distributions in D.

A simple example of a Boolean function that can be computed significantly faster in the average case compared to the worst case (for which the trivial logarithmic lower bound for the depth holds) is the OR-function $\mathtt{OR}_n(x_1, \ldots, x_n) \ := \ x_1 \vee x_2 \vee \cdots \vee x_n$. Any time optimal circuit with respect to the expectation for this function has not only linear depth, but also a linear time complexity in the worst case. We prove this claim by showing that the circuit in the middle of figure 1 is the only one – up to a permutation of the input gates – that is expected time optimal.

Theorem 3 $\quad etime(\mathtt{OR}_n, \mu_n^{\mathrm{uni}}) \ = \ 2 - 2^{-(n-2)}$.

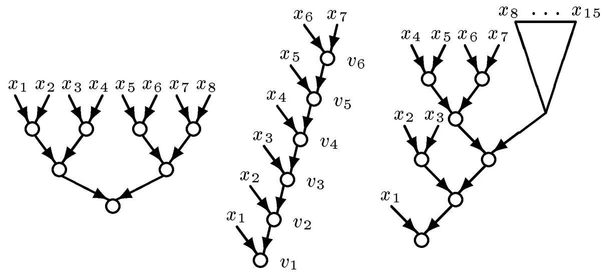

Figure 1: Different circuit designs to compute the OR-function

Proof: Consider the circuit in the middle of figure 1 and let C be corresponding version for n inputs. Let $time_C$ be the random variable which gives the time complexity of C with respect to the uniform distribution. It is easy to see that

$$\Pr[time_C(x) = t] \quad = \quad \begin{cases} 2^{-t} & \text{for } t \leq n-2\ , \\ 2^{-(t-1)} & \text{for } t = n-1\ . \end{cases}$$

Therefore,

$$\begin{aligned} E(time_C) \quad &= \quad \sum_{t=1}^{n-2} \frac{t}{2^t} + \frac{n-1}{2^{n-2}} = \sum_{t=1}^{n-2} \left(\frac{2}{2^t} - \frac{1}{2^{n-1}} \right) + \frac{n-1}{2^{n-2}} \\ &= \quad 2 - \frac{2}{2^{n-2}} - \frac{n-2}{2^{n-2}} + \frac{n-1}{2^{n-2}} \quad = \quad 2 - 2^{-(n-2)}\ . \end{aligned}$$

To prove the uniqueness of this circuit we will show that any structurally different circuit has a larger expected delay. Denote the internal gates of this circuit by $v_1, v_2, \ldots, v_{n-1}$ starting the numbering at the output gate.

Assume that – except a permutation of the input gates – a circuit S differs from C the first time at gate v_d. This means that S develops the OR-function as $x_1 \vee (x_2 \vee \ldots (x_{d-1} \vee v_d(x)) \ldots)$ for some function $v_d(x) = g_1(x)\ op\ g_2(x)$ with $op \in \{\vee, \wedge\}$.

If $d < n-1$ then either one of the subfunctions $g_i(x)$ equals x_j for some $j < d$, or each g_i depends on more than one variable. In both cases holds $\Pr[time_S(x) = d] = 0$. Thus,

$$\begin{aligned} E(time_S) \quad &= \quad \sum_{k=1}^{\mathrm{depth}(S)} \Pr[time_S(x) = k] \cdot k \\ &= \quad \sum_{k=1}^{d-1} \frac{k}{2^k} + \sum_{k=d+1}^{\mathrm{depth}(S)} \Pr[time_S(x) = k] \cdot k \\ &\geq \quad \sum_{k=1}^{d-1} \frac{k}{2^k} + \frac{d+1}{2^{d-1}} = 2 - \frac{d+1}{2^{d-1}} + \frac{d+1}{2^{d-1}} = 2\ . \end{aligned}$$

If $d = n-1$ then either $\Pr[time_S(x) = n-1] = 0$ or $v_d(x) = x_{n-1} \vee g(x)$ with $g(x) \neq x_n$. In the second case, $\Pr[time_S(x) = n-1] = 2^{-(n-1)}$. Similar as above, we can conclude

$$\begin{aligned} E(time_S) \quad &= \quad \sum_{k=1}^{n-2} \frac{k}{2^k} + (n-1) \cdot \Pr[time_S(x) = n-1] + \sum_{k=n}^{\mathrm{depth}(S)} \Pr[time_S(x) = k] \cdot k \\ &\geq \quad \sum_{k=1}^{n-1} \frac{k}{2^k} + \frac{n}{2^{n-1}} = 2 - \frac{n+1}{2^{n-1}} + \frac{n}{2^{n-1}} = 2 - 2^{-(n-1)}\ . \end{aligned}$$

∎

The expectation has certain drawbacks when used as an average case measure. As an alternative we consider a technically more complex measure similarly to the case of uniform average case complexity classes [Levi86, Gure91, ReSc93a] obtained as follows.

Definition 4 *For a monotone complexity bound $T : \mathbb{N} \to \mathbb{N}$ we define the inverse by*

$$T^{-1}(\tau) \;\; := \;\; \min\{m \mid T(m) \geq \tau\} \; .$$

A function $t : \{0,1\}^n \to \mathbb{N}$ is called average T-bounded with respect to a probability distribution $\mu : \{0,1\}^n \to [0;1]$, denoted by $(t, \mu) \in Av(T)$, if

$$\sum_{x \in \{0,1\}^n} T^{-1}(t(x)) \; \mu(x) \;\; \leq \;\; n \; .$$

For a Boolean function $f \in B_n^m$, a set of probability distributions $D \subseteq \mathcal{D}_n$, and a complexity bound T we define the relation

$$avtime(f, D) \leq T \qquad := \qquad \forall \mu \in D \quad \exists C \in Cir_\mu(f) \; (time_C, \mu) \in Av(T) \; ,$$

which means that for any distribution in D the function f can be computed by a circuit with **average delay** *bounded by T. The negation of this relation is written as: $T < avtime(f, D)$.*

Note that the expected delay as defined above yields a single real number $t = \mathrm{E}_\mu(\mathrm{time_C})$ for every circuit C of fixed input size n, whereas in order to bound the average delay we need a function T such that the inverse $T^{-1}(\tau)$ exists for every value $\tau = time_C(x)$. As usual in complexity theory, we are interested in the asymptotic behaviour of such functions T and restrict the analysis to nicely growing functions like the logarithm, powers of the logarithm, and iterated logarithms. For ease of notation let $\mathrm{llog} : \; n \;\to\; \log \log n$ denote the twice iterated logarithmic function with respect to base 2.

Circuits are a nonuniform model and we will derive sharp average bounds for each input size n. These results can be translated into a uniform setting by considering family of circuits, one for each input size.

In the following when using the term average bound we always refer to the second definition with the help of the inverse function T^{-1}, the other will be called an expected bound. As already mentioned it is justified to base an average case complexity analysis on the technical more complicated average time measure, and not on the simple expected time measure by results obtained in the analog case of machines [Levi86, Gure91, ReSc93b]. For ease of notation we will use $\exp n$ to denote 2^n.

For the average case measure it holds

Theorem 4

$$\mathrm{llog} - 1 \;<\; avtime(\mathtt{OR}_n, \mu_n^{\mathrm{uni}}) \;\leq\; 2 \cdot \mathrm{llog} + 1 \; .$$

Proof: A circuit C achieving the upper bound is shown in figure 1 in the right. The average time complexity can be bounded as follows for the uniform distribution. For $1 \leq t \leq \log n$ it holds

$$\begin{array}{rcl} \Pr[time_C(x) = 2t - 1] & = & 2^{-2^t + 1} \; , \\ \Pr[time_C(x) = 2 \log n] & \leq & 2^{-(n-1)} \; . \end{array}$$

In order to get $(time_C, \mu^{\mathrm{uni}}) \in \mathrm{Av}(2 \cdot \mathrm{llog} + 1)$ we have to show for $T = 2 \cdot \mathrm{llog} + 1$

$$\sum_{t \geq 1} \frac{T^{-1}(t)}{n} \; \Pr[time_C(x) = t] \;\leq\; 1 \; .$$

This follows for $n \geq 4$ from

$$\begin{aligned}
& \sum_{t=1}^{\log n} \frac{T^{-1}(2t-1)}{n} \frac{2}{2^{2^t}} + \frac{T^{-1}(2\log n)}{n} \frac{2}{2^n} \\
\leq \; & \sum_{t=1}^{\log n} \frac{2^{2^t}}{n} \frac{2}{2^{2^t}} + \frac{2^{2^{\log n}}}{n} \frac{2}{2^n} \;\leq\; \sum_{t=1}^{\log n} \frac{2}{n} + \frac{2}{n} \;\leq\; 1 .
\end{aligned}$$

The lower bound can be shown as follows. For any circuit C and every input x with $time_C(x) = t$ the result is determined by at most 2^t input variables. For $t < \log n$ this implies that at least one input must have value 1, which for the uniform distribution happens with probability $1 - 2^{-2^t}$. Thus

$$\Pr[time_C(x) < \log n] \;\leq\; 1 - 2^{-n} .$$

For $T = \text{llog} - 1$ the inverse is given by $T^{-1}(t) = 2^{2^{t+1}}$. Thus, using the fact that T^{-1} is monotone increasing we can estimate the sum by

$$\begin{aligned}
& \sum_{t \geq 1} \frac{T^{-1}(t)}{n} \quad \Pr[time_C(x) = t] \\
\geq \; & \frac{T^{-1}(\log n)}{n} \; \Pr[time_C(x) \geq \log n] \\
\geq \; & \frac{2^{2^{\log n + 1}}}{n} \; 2^{-n} \;\; = \; \exp(2n - \log n - n) \; > \; 1 .
\end{aligned}$$

∎

The right circuit in figure 1 has nearly optimal expected time (≤ 2.3) and logarithmic depth. A simple calculation shows that the optimal circuit for the `OR` with respect to the expected delay of linear depth has an average delay of logarithmic order. Thus the average measure may yield much larger values than the expected measure. In the other direction the increase can be shown to be bounded.

Lemma 1 *Let T be a monotone complexity bound such that the function $n \mapsto \frac{n}{T(n)}$ is monotone increasing then*

$$(t, \mu) \in Av(T) \implies \mathrm{E}_\mu(\mathrm{t}) \leq 2 \cdot \mathrm{T(n)} .$$

In particular, this implies:

$$avtime(f, D) \leq T \implies etime(f, D) \leq 2 \cdot T(n) .$$

Proof: For $(f, \mu_n) \in \mathrm{Av}(T)$ the definition yields

$$\sum_{|x|=n} \mu_n(x) \cdot \frac{T^{-1}(f(x))}{n} \;\leq\; 1 .$$

Consider a partition I_1, I_2 of the input set $\{0,1\}^n$

$$\begin{aligned}
I_1 &:= \{x \in \{0,1\}^n |\; f(x) \;>\; T(n)\} , \\
I_2 &:= \{x \in \{0,1\}^n |\; f(x) \;\leq\; T(n)\} .
\end{aligned}$$

I_1: If $f(z) > T(n)$ then $T^{-1}(f(z)) \geq T^{-1}(T(n)+1) \geq n$. For the monotone increasing function $g(n) := n/T(n)$ one can then draw the following conclusions:

$$g(T^{-1}(f(z))) \;\geq\; g(n) \implies \frac{T^{-1}(f(z))}{n} \;\geq\; \frac{f(z)}{T(n)} .$$

Then,

$$\sum_{x \in I_1} \mu_n(x) \cdot \frac{f(x)}{T(n)} \;\leq\; \sum_{x \in I_1} \mu_n(x) \cdot \frac{T^{-1}(f(x))}{n} \;\leq\; 1 \;.$$

I_2 : $f(x) \leq T(n)$ for $x \in I_2$ implies

$$\sum_{x \in I_2} \mu_n(x) \cdot \frac{f(x)}{T(n)} \;\leq\; 1$$

and thus

$$\sum_{|x|=n} \mu_n(x) \cdot \frac{f(x)}{T(n)} \;\leq\; 2 \;.$$

∎

Furthermore, contrary to the expected bound, it can be shown that a circuit with an optimal asymptotic average bound has always small depth.

3 The Complexity of Distributions

For an average case analysis of circuits we also have to define a complexity measure for the distributions that generate the random inputs. The complexity will be measured by the circuit model itself, that is by the complexity of a circuit that starting with a vector of truly random bits generates the specific distribution.

Definition 5 *A circuit C with r input gates and n output gates performs a transformation of a random variable Z defined over $\{0,1\}^r$ into a random variable X over $\{0,1\}^n$ 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.*

If Z is the uniform distribution over $\{0,1\}^r$ such a circuit will be called a **distribution generating circuit, DG-circuit**, *that generates the distribution of X.*

In the following we will identify a distribution μ with the random variable X that represents μ. Let $X = X_1, \ldots, X_n$.

No interesting results can be obtained if we consider distribution generated by DG-circuits with unbounded fan-out, which means that every output may depend on a single random bit. The example in figure 2 shows that in such a case even very simple circuits can generate highly unbalanced distributions.

If in this figure we choose $d(n) = 1$, that means the big rectangle in the middle simply routes the input wires to the $\wedge$-gates then the output distribution is given by $\Pr[X = 0^n] = \frac{1}{2} + 2^{-(n+1)}$ and $\Pr[X = y] = 2^{-(n+1)}$ for $y \neq 0^n$. For the $\mathtt{OR}_n$-function, for example, this distribution implies a lower bound of $\frac{1}{2}\log n$ for the expected delay. However, it will be shown that for any probability distribution that is generated by a constant depth circuit with bounded fan-out the expected delay is constant. Therefore, we restrict DG-circuit to a constant fan-out, let us say fan-out 2.

With the notion of DG-circuits we classify distributions as follows:

Definition 6

$\mathcal{D}Depth_n(d) := \{\ \mu \in \mathcal{D}_n \mid \exists$ *an r-input and n-output DG-circuit C of depth d that transforms a random variable Z over $\{0,1\}^r$ with distribution μ_r^{uni} into a random variable X over $\{0,1\}^n$ with distribution μ* $\}$.

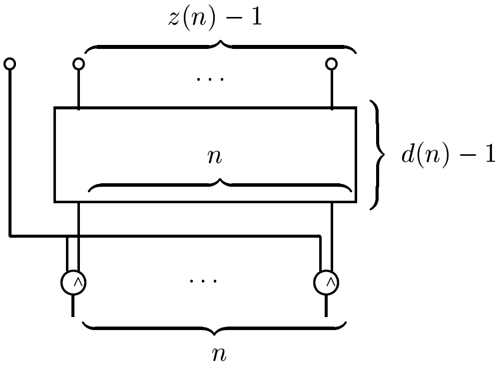

Figure 2: An example of a DG-circuit with $z(n) = n+1$ random bits as inputs and an n-bit output vector.

The basic properties for achieving a substantial speedup in the average case for many Boolean functions are the following. The speedup results for the average case complexity presented below will hold for any distribution fulfilling these properties.

Lemma 2 *Let μ, resp. X belong to $\mathcal{D}Depth_n(d)$ and define*

$$I_X := \{ i \mid 0 < \Pr[X_i = 1] < 1 \} \ .$$

Then

1.) for all $i \in I_X$ holds $\quad 2^{-2^d} \ \leq \ \Pr[X_i = 1] \ \leq \ 1 - 2^{-2^d}$,

2.) there exists a set of $k \geq |I_X| \cdot 2^{-2d}$ bit positions $X_{i_1}, \ldots, X_{i_k}$ that are independent.

Proof: By induction on the depth it can easily be shown that in a DG-circuit any gate in depth d takes a value with a probability that must be a multiple of 2^{-2^d}. In particular, if the probability of a certain output value is not zero it must be at least 2^{-2^d}.

Choose any bit position X_i (see figure 3). This random variable depends on at most 2^d random

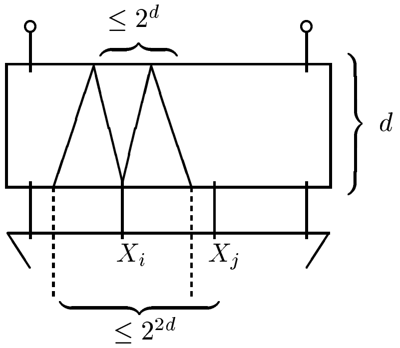

Figure 3: The random variables that determine X_i can only influence at most 2^{2d} output bits.

input bits Z_j. Because of the fanout restriction each such Z_j can influence at most 2^d random variables X_k. Thus, there are at most $2^{2d} - 1$ other X_k that may not be independent of X_i. ∎

4 Upper and Lower Bounds for Basic Boolean Functions

4.1 Functions with Sublogarithmic Average Time Complexity

We start with one of the simplest nontrivial Boolean function, the OR and show that it can be computed much faster on the average for a large class of distributions. For both, the expected and the average measure, the upper and lower bounds almost match.

Theorem 5 *For* $0 \leq d < \operatorname{llog} n$ *holds*

$$\begin{array}{rcll} 2^{d-1}-1 & \leq & etime(\mathtt{OR}_n, \mathcal{D}Depth(d)) & \leq\ 2^d+3\ , \\ \operatorname{llog} n + 2^d - 2 & \leq & avtime(\mathtt{OR}_n, \mathcal{D}Depth(d)) & \leq\ 2\operatorname{llog} n + 2^{d+1} + 4d + 2\ . \end{array}$$

Proof: In order to prove the lower bounds it suffices to find a specific distribution in $\mathcal{D}\mathrm{Depth}(d)$ for which the average delay of the OR-function becomes large. Let $X = X_1, \ldots, X_n$ be a random variable with $\Pr[X_i = 1] = 2^{-2^d}$ and such that the X_i are independent. It is easy to see that such an X can be generated in depth d. For this X the probability that a subsequence of length 2^{2^d-1} is identical to $0^{2^{2^d-1}}$ is at least $\frac{1}{2}$. These sequences contained in an input string cannot solely determine the result of any circuit computing $\mathtt{OR}_n$. The lower bounds for the expected measure immediately follow because all gates have bounded fanin.

$$etime(\mathtt{OR}_n, \mathcal{D}\mathrm{Depth}(d)) \ \geq\ \frac{1}{2} \cdot (2^d - 1) \ \geq\ 2^{d-1} - 1.$$

For the lower bound of the average measure we consider the same random variable X. Note, that

$$\Pr[X = 0^n] \ = \ (1 - 2^{-2^d})^n \ \geq\ \exp\left(-2\frac{n}{2^{2^d}}\right)\ .$$

Let C be a circuit computing $\mathtt{OR}_n$ and assume $avtime(\mathtt{OR}_n, \mathcal{D}\mathrm{Depth}(d)) \leq T = \operatorname{llog} n + 2^d - 2$. Then

$$\begin{array}{rcl} \displaystyle\sum_t \frac{T^{-1}(t)}{n} \Pr[time_C(X) = t] & \geq & \displaystyle\frac{T^{-1}(\log n)}{n} \Pr[time_C(X) = \log n] \\ & \geq & \displaystyle\frac{\exp\exp(\log n - 2^d + 2)}{n \ \cdot \exp\left(2\frac{n}{2^{2^d}}\right)} \ > \ 1\ , \end{array}$$

since $n/2^{2^d} \geq \log n$ for $d < \operatorname{llog} n$.

For the upper bounds we have to consider an arbitrary random variable X over $\{0,1\}^n$ in $\mathcal{D}\mathrm{Depth}(d)$.

First assume that $X \in \mathcal{D}\mathrm{Depth}_n(d)$ is strictly positive. By lemma 2 there exist at least $n \cdot 2^{-2d}$ independent input bits X_i. Partition them into groups Y_j of size 2^{2^d} and compute for each Y_j the $\mathtt{OR}(Y_j)$ of all its bits in depth 2^d. Let W be the remaining bits. Their disjunction can be computed by a binary tree of depth at most $\log n$. Finally, realize the OR-function as

$$\mathtt{OR}(X) \ = \ \mathtt{OR}(Y_1) \vee (\mathtt{OR}(Y_2) \vee (\cdots (\mathtt{OR}(Y_p) \vee \mathtt{OR}(W)) \cdots))\ ,$$

where $p \geq n \cdot 2^{-2d-2^d} \geq 1 + \operatorname{llog} n$ for n large enough. Let $time(X)$ be the random variable that describes the time complexity of this circuit C. Because of lemma 2 it holds $\Pr[X_i = 1] \geq 2^{-2^d}$ and thus

$$\Pr[\mathtt{OR}(Y_j) = 1] \geq 1 - (1 - 2^{-2^d})^{2^{2^d}} \ \geq\ \frac{1}{2}\ .$$

This implies for $1 \le t \le \operatorname{llog} n + 1$

$$\Pr[time(X) = 2^d + t] \;\ge\; 2^{-t} \;.$$

Straightforward calculation shows, that the expectation of $time(X)$ is bounded by $2^d + 3$.

$$E(time_C) \;\le\; \sum_{t=1}^{\operatorname{llog} n+1} \frac{2^d+t}{2^t} + \frac{\log n + \operatorname{llog} n + 1}{2^{\operatorname{llog} n+1}} \;\le\; 2^d + \sum_{t=1}^{\operatorname{llog} n+1} \frac{t}{2^t} + 1 \;\le\; 2^d + 3.$$

Using the outputs of the circuits $\mathtt{OR}(Y_i)$ and $\mathtt{OR}(W)$ as inputs of the right circuit in figure 1 ($\mathtt{OR}(\mathtt{OR}(Y_1), \ldots, \mathtt{OR}(Y_p), \mathtt{OR}(W))$) a straightforward calculation leads to

$$\sum_{i=1}^{p} \frac{T^{-1}(2 \cdot \log i + 1 + 2^d)}{n \cdot 2^{i-1}} + \frac{T^{-1}(2 \cdot \log n + 1)}{n \cdot 2^p} \;\le\; 1.$$

So an upper bound of the Av-measure is given by

$$T \;\le\; 2 \cdot (\operatorname{llog} n + 2 \cdot d + 2^d + 1).$$

If X is not strictly positive either there exists an X_i with $\Pr[X_i = 1] = 1$ and the result will always be 1 and the $\mathtt{OR}$ becomes trivial. Otherwise, let m be the number of X_i with $\Pr[X_i = 0] < 1$. Since all inputs bits with $X_i \equiv 0$ can be ignored the problem reduces to computing $\mathtt{OR}_m$ for a strictly positive distribution on $\{0,1\}^m$. Hence, a circuit for $\mathtt{OR}_m$ computes $\mathtt{OR}_n$ for such a special distribution within the same upper bound. ∎

Note, that the circuit shown on the right of figure 1 is asymptotically optimal with respect to depth, expected time und average time bound for all probability distributions. These bounds can be translated to functions similar to the $\mathtt{OR}$ as follows.

Lemma 3 *Let $f \in B_q^m$, $g \in B_p^q$ and $h \in B_n^p$ be Boolean functions and assume that h can be computed by a circuit with bounded fanout in depth δ. Then*

$$\begin{aligned} etime(f \circ g \circ h, \mathcal{D}Depth(d)) \;&\le\; depth(f) + depth(h) + etime(g, \mathcal{D}Depth(d+\delta)) \;, \\ avtime(g, \mathcal{D}Depth(d+\delta)) \;\le\; T \;&\Longrightarrow\; avtime(f \circ g \circ h, \mathcal{D}Depth(d)) \;\le\; depth(f) + depth(h) + T \;. \end{aligned}$$

Proof: Let $\mathcal{D}$ be a circuit, that generates a probability distribution in $\mathcal{D}\mathrm{Depth}(d)$. Then $h(\mathrm{res}(\mathcal{D}))$ denotes the function obtained by connecting $\mathcal{D}$ with a circuit for h. It holds $h(\mathrm{res}(\mathcal{D})) \in \mathcal{D}\mathrm{Depth}(d+\delta)$. By definition,

$$etime(g, \{h(\mathrm{res}(\mathcal{D}))\}) \le etime(g, \mathcal{D}\mathrm{Depth}(d+\delta)) \;.$$

Compute $g \circ h$ by composing circuits for g and h. Here the random input variables of g in both circuits (for $g \circ h$ and g) have the same probability. Therefore,

$$etime(g \circ h, \{\mathcal{D}\}) \le etime(g, \{h(\mathrm{res}(\mathcal{D}))\}) + \mathrm{depth}(h) \;.$$

$\mathcal{D}$ was chosen arbitrarily from the set $\mathcal{D}\mathrm{Depth}(d)$. Furthermore, the expected time for g is increased if one allows all DG-circuits of depth $d+\delta$, instead of DG-circuits computing $h(\mathrm{res}(\mathcal{D}))$ only. Therefore

$$etime(g \circ h, \mathcal{D}\mathrm{Depth}(d)) \quad\le\quad etime(g, \mathcal{D}\mathrm{Depth}(d+\delta)) + \mathrm{depth}(h) \;.$$

In order to compute $f \circ g \circ h$ append a circuit for f.

The bound for *avtime* can be proved the same way. ∎

Let $\mathtt{AND}_n$ denote the n-ary $\mathtt{AND}$-function and $\mathtt{EQUAL}_n$ the $2n$-ary function which decides whether two n-bit strings are equal. Using $\mathtt{OR}_n(x_1, \ldots, x_n) = \neg\mathtt{AND}_n(\overline{x_1}, \ldots, \overline{x_n})$ and $\mathtt{EQUAL}_n(x_1, y_1, \ldots, x_n, y_n) = \neg\mathtt{OR}_n(x_1 \oplus y_1, \ldots, x_n \oplus y_n)$, a simple application of the lemma above yields

Theorem 6

$$\begin{array}{rcl} etime(\mathtt{OR}_n, \mathcal{D}Depth(d)) \le & etime(\mathtt{EQUAL}_n, \mathcal{D}Depth(d)) & \le etime(\mathtt{OR}_n, \mathcal{D}Depth(d+2)) + 1 \ , \\ etime(\mathtt{OR}_n, \mathcal{D}Depth(d)) \le & etime(\mathtt{AND}_n, \mathcal{D}Depth(d+1)) + 1 & \le etime(\mathtt{OR}_n, \mathcal{D}Depth(d+2)) + 2 \ . \end{array}$$

Similar relations hold for the measure $avtime$.

There are more basic Boolean functions that have a similar expected and average time complexity as the OR, for example the comparison of two binary numbers.

4.2 Logarithmic Average Lower Bounds: The Parity Function

On the other hand, we have found only few examples of functions for which the average case is almost as complex as the worst case. The parity function, denoted by PARITY, is one of them. Intuitively this seems clear because no matter how the actual input looks like the final output depends on every single bit. In order to prove this rigorously, we will introduce a general lower bound technique.

Let $\mathtt{THRESHOLD}_n^a$ denote the n-ary Boolean function which equals 1 if at least a input bits are 1, for $0 \le a \le n$, and $\mathtt{MAJORITY}_n$ the special case $a = \lfloor \frac{n}{2} \rfloor$. $\mathtt{BITSORT}_n$ denotes the sorting of n Bits, and $\mathtt{MULTIPLY}_n$ the multiplication of two n-bit numbers.

Definition 7 *To measure the input bit dependency of an n-ary Boolean function f we define*

$$depen(f) \quad := \quad \min\{k \mid \exists\ a_{i_1}, \ldots, a_{i_k} \ \textit{such that} \ f_{|\ x_{i_1}=a_{i_1}, \ldots, x_{i_k}=a_{i_k}} \equiv \text{const}\}$$

that is the minimal length of a prime implicant or a prime clause of f .

Thus $depen(f)$ is the co-dimension of the largest Boolean subcube on which f is constant.

Proposition 1

$$\begin{array}{rcl} depen(\mathtt{OR}_n) & = & 1 \ , \\ depen(\mathtt{PARITY}_n) & = & n \ , \\ depen(\mathtt{MULTIPLY}_n) & \ge & \sqrt{n} \ , \\ depen(\mathtt{THRESHOLD}_n^a) & = & \min(a, n-a+1) \ , \\ depen(\mathtt{MAJORITY}_n) & = & \lceil \frac{n}{2} \rceil \ , \\ depen(\mathtt{BITSORT}_n) & = & n \ . \end{array}$$

Lemma 4 *Let $f \in B_n$ and $C \in Cir(f)$ then $\forall x \in \{0,1\}^n \quad time_C(x) \ \ge \ \log \ depen(f)$.*

Proof: Let $x \in \{0,1\}^n$ be arbitrary. By definition of $depen(f)$ any subset of less than $depen(f)$ many input bits x_i does not uniquely determine the value of $f(x)$. Since the fan-in is bounded by 2 any computation of a circuit C for f that generates the final result in less than $\log depen(f)$ time does not read enough inputs. ∎

Over the standard basis $\mathtt{PARITY}_n$ can be computed in worst case time $2 \log n$. For the average case even restricting to the uniform distribution the Lemma above implies for this function and similarly for the others with large dependency.

Theorem 7

$$\begin{array}{rcl} etime(\texttt{PARITY}_n, \mu_n^{\text{uni}}) & \geq & \log n\ , \\ etime(\texttt{MAJORITY}_n, \mu_n^{\text{uni}}) & \geq & \log n - 1\ , \\ etime(\texttt{BITSORT}_n, \mu_n^{\text{uni}}) & \geq & \log n\ , \\ etime(\texttt{MULTIPLY}_n, \mu_n^{\text{uni}}) & \geq & \frac{1}{2} \log n\ , \\ avtime(\texttt{PARITY}_n, \mu_n^{\text{uni}}) & > & \log n - 1\ , \\ avtime(\texttt{MAJORITY}_n, \mu_n^{\text{uni}}) & > & \log n - 2\ , \\ avtime(\texttt{BITSORT}_n, \mu_n^{\text{uni}}) & > & \log n - 1\ , \\ avtime(\texttt{MULTIPLY}_n, \mu_n^{\text{uni}}) & > & \frac{1}{2} \log n - 1\ . \end{array}$$

Proof: follows directly from proposition 1 and lemma 4. ∎

The same lower bounds hold for any strictly positive distribution.

4.3 Threshold

If the threshold value a is small the dependency provides only a lower bound on the average time of $\log a$. We can show that this bound can actually be achieved and generalize the bounds to more complex distributions.

Theorem 8 *For* $d \leq \operatorname{llog} n - 1$ *and* $a \leq n \cdot 2^{-(2^d + 2d + O(1))}$ *holds:*

$$\begin{array}{lcl} \frac{1}{2}\ (\log a + 2^d) - O(1) & \leq etime(\texttt{THRESHOLD}_n^a, \mathcal{D}Depth(d)) \leq & 2 \cdot (\log a + 2^d) + O(1)\ , \\ \frac{1}{2}(\operatorname{llog} n + \log a + 2^d) - O(1) & \leq avtime(\texttt{THRESHOLD}_n^a, \mathcal{D}Depth(d)) \leq & 2\ (\operatorname{llog} n + \log a + 2^d) + O(1). \end{array}$$

Proof: It suffices to consider $a \geq 2$. As in the upper bound proof for the OR-function we may restrict to random variables X with $0 < \Pr_\mu[X_i = 0] < 1$ for all $i \in [1 \ldots n]$, that means $\Pr[X_i = 1] \geq q := 2^{-2^d}$.

To achieve the upper bound for the threshold function partition the independent inputs into groups G_j of size $l = \gamma \cdot a \cdot 2^{2^d}$ for some $\gamma \geq 1$. Let

$$Z_j \ := \sum_{X_i \in G_j} X_i\ .$$

Z_j is lower bounded by the binomial distribution $B_{l,q}$ with expectation $E_{l,q} = l \cdot q = \gamma \cdot a$. Thus,

$$\begin{array}{rcl} \Pr[Z_j \leq a]\ \leq\ \Pr[B_{l,q} \leq a] & = & \Pr\left[E_{l,q} - B_{l,q}\ \geq\ (\gamma - 1)\, a\ =\ \dfrac{\gamma - 1}{\gamma} E_{l,q}\right] \\ & \leq & 2 \exp(-\frac{1}{3}(\frac{\gamma - 1}{\gamma})^2\ \cdot\ E_{l,q}) \\ & \leq & 2 \exp(-\frac{3}{4} a)\ \leq\ \frac{1}{2} \end{array}$$

for $\gamma = 4$. Thus, each group G_j has at least a ones with probability at least 1/2. There are at least

$$\frac{n \cdot 2^{-2d}}{l} \ = \ \frac{n \cdot 2^{-(2^d + 2d)}}{\gamma \cdot a} \ =: \ m$$

many groups. With probability at most 2^{-m} no group has enough ones.

For each group compute the function $\texttt{THRESHOLD}_l^a$ by a standard circuit of depth $2\log l$. In parallel, $\texttt{THRESHOLD}_n^a$ is also computed by a standard circuit. These subcircuits are combined by an average case efficient circuits for the $\texttt{OR}$-function (see figure 4). The expected delay of this circuit is given by

$$2\log l + O(1) + \Pr[\text{no } \texttt{THRESHOLD}_l^a \text{ evaluates to } 1] \cdot 2\log n \ .$$

If $m \geq \operatorname{llog} n$ the last term of this sum adds only a constant since the probability is small. Otherwise, from the upper bound on d we can conclude

$$a \ = \ \frac{n \cdot 2^{-(2^d+2d)}}{\gamma m} \ \geq \ n^{\Omega(1)}$$

and thus a single $\texttt{THRESHOLD}_l^a$ already evaluates to 1 with high probability.

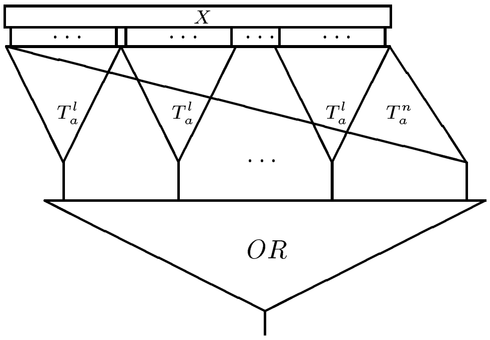

Figure 4: $\texttt{THRESHOLD}_n^a$.

For the lower bound we choose the same probability distribution as for the $\texttt{OR}$. Consider delay $\delta := \log a + 2^d - \gamma$ for some $\gamma \geq 0$. At most $l = 2^{-\gamma} \cdot a \cdot 2^{2^d}$ many inputs X_i can influence the output gate by that time. For $l \leq n - a$ one can conclude that termination of the output gate implies that among these inputs bits there must be at least a many ones, otherwise the output is not guaranteed to be correct. Let $Z := \sum X_i$, where we sum over the l inputs X_i closest to the output gate. Z is distributed according to $B_{l,q}$ for $q = 2^{-2^d}$. Hence, for $\beta := 2^\gamma(1 - 2^{-\gamma})$

$$\begin{aligned}
\Pr[Z \geq a] \ &= \ \Pr[B_{l,q} - E_{l,q} \ \geq \ a\,(1 - 2^{-\gamma}) \ = \ \beta \cdot E_{l,q}] \\
&\leq \ \exp\left(-E_{l,q} \cdot \beta \log \frac{\beta}{2}\right) \\
&\leq \ \exp\left(-(\gamma - 1)\,(1 - 2^{-\gamma})\right) \ \leq \ \epsilon
\end{aligned}$$

for γ sufficiently large. Thus with probability at least $1 - \epsilon \geq \frac{1}{2}$ the delay has to be larger than δ. This yields $\mathit{etime} \ \geq \ \frac{1}{2}\,(\log a + 2^d - \gamma)$. The analysis even gives a factor $1 - \epsilon$ instead of $\frac{1}{2}$ for arbitrary small ϵ.

The bounds for the average delay can be obtained similarly.

For the lower bounds of the $\texttt{THRESHOLD}_n^a$-function in the Av-measure we have to examine

$$T \ := \ \frac{1}{2}\operatorname{llog} + \frac{1}{2}\log a + 2^d - 2$$

and

$$T^{-1} = \exp\exp(2\mathcal{N} - \log a - 2 \cdot 2^d + 4) \ .$$

Here we have to consider the two cases $a \geq \log n$ and $a < \log n$.

1. $a \geq \log n$: For $a \leq n/2$ we want to show that for $l \leq n - a$

$$\Pr[Z_l \leq a] \cdot \frac{T^{-1}(\log n)}{n} > 1.$$

Since for $l = \frac{1}{2} \cdot a \cdot 2^{2^d}$ and $a \geq 4$ $\Pr[Z_l \leq a] \geq \frac{1}{2}$ it holds

$$\exp(2\log l - \log a - 2 \cdot 2^d + 4) - \log n - 1 \;=\; \exp(\log a + 2) - \log n - 1 \;>\; 0.$$

2. $a < \log n$: The claim follows from the observation that for the $\texttt{THRESHOLD}_n^a$ -function the lower bounds of the $\texttt{OR}_n$ -function in the Av-measure are also valid:

$$\text{llog} + 2^d - 2 \;\leq\; AvTime(\texttt{OR}_n, \mathcal{D}\text{Depth}(d))$$

∎

4.4 Addition

It is well known that the problem of adding two binary numbers $x, y \in \{0,1\}^n$ is basically equivalent to an efficient computation of the carries. By adding the two values at each bit position one can derive a string $z \in \{\texttt{pro}, \texttt{gen}, \texttt{del}\}^n$ indicating whether a carry is propagated, deleted or generated. It remains to compute all prefixes $p_i = z_1 \otimes z_2 \otimes \ldots \otimes z_i$ of z where $\otimes$ is a binary associative operator that combines the carry information.

Instead of considering random variables for the distribution of x and y we might as well consider a random variable Z over $\{\texttt{pro}, \texttt{gen}, \texttt{del}\}^n$ for the corresponding distribution of the z-values. It is easy to see that the complexity of corresponding distributions is almost identical (up to a permutation of each pair x_i, y_i).

If $\Pr[Z_i = \texttt{pro}] = 0$ then p_i is independent of the value of $Z_1, \ldots, Z_{i-1}$ which simplifies the problem. If $\Pr[Z_i = \texttt{pro}] = 1$ then it suffices to compute all prefixes of $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$ and set $p_i = p_{i-1}$. In this case the size of the problem can be reduced. Therefore, we can assume that for all i holds:

$$0 \;<\; \Pr[Z_i = \texttt{pro}] \;<\; 1\;.$$

For a vector $z \in \{\texttt{pro}, \texttt{gen}, \texttt{del}\}^n$ define $\texttt{pro}(z)$ as the maximal length of a contiguous block of symbols $\texttt{pro}$ in z . The importance of this quantity is shown by the following

Lemma 5 *For any circuit C that solves the prefix problem and any input z holds*

$$time_C(z) \;\geq\; \log \texttt{pro}(z)\;.$$

Furthermore, there exists a circuit that achieves this lower bound within a factor of 4.

Proof: To prove the lower bound assume that a circuit C finishes for an input z before time $\log\ \texttt{pro}(z)$. Let $S = z_i, \ldots, z_j$ be a block of inputs with value $\texttt{pro}$ of length $\texttt{pro}(z)$. Then the fanin bound implies that the output gate p_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 p_j for this modified input. This input may actually occur with non-zero probability for any distribution in $\mathcal{D}_n$.

This simple lower bound for the prefix problem can be achieved by a special circuit design within a factor of 4 as illustrated in figure 5.

This circuit computes an output bit p_i by partition the input string $z_0 \ldots z_{n-1}$ into exponential increasing blocks $(z_i)\,, (z_{i-1}z_{i-2})\,, (z_{i-3} \ldots z_{i-6})\,, \ldots$ and evaluates them by using complete binary trees.

If the input is coded as $v_i = (x_j \vee y_j)$ and $w_i = (x_j \wedge y_j)$ we get $\texttt{del} = 00$, $\texttt{gen} = 11$, and $\texttt{pro} = 10$. Then one can realize a single $\otimes$-operator by a circuit of depth 2 . Therefore

$$time_C(z) \leq 4 \cdot \log \texttt{pro}(z)\;.$$

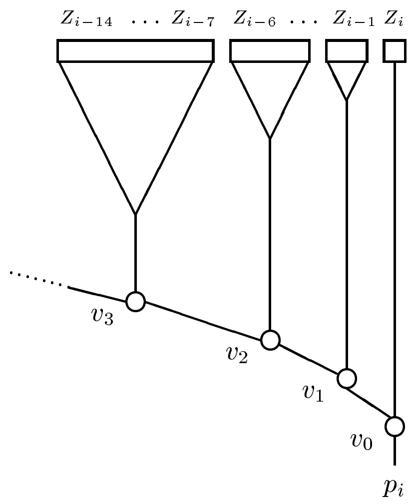

Figure 5: A circuit for carry propagation with optimal average delay

∎

In order to generate the random variable Z we generalize the notion of DG-circuits to arbitrary finite domains in the obvious way, in this case to $\{\texttt{pro}, \texttt{gen}, \texttt{del}\}$. Such a circuit may use as internal gates arbitrary binary operators over this domain. For ease of notation we define $\gamma \;:=\; \log_2 e$.

Lemma 6 *For $Z \in \mathcal{D}Depth(d)$ with $Z_i \not\equiv \texttt{pro}$ for all i holds for any l:*

$$\Pr[\texttt{pro}(Z) \geq 2l-1] \;\leq\; \frac{n}{l} \cdot \exp(-l\; 2^{-(2d+2^d)}) \;.$$

Proof: We partition the Z_i into blocks B_j of length l. Lemma 2 yields that in each such block there exist at least $l/2^{2d}$ independent inputs Z_i and $\Pr[Z_i = \texttt{pro}] \;\leq\; 1 - 2^{-2^d}$.

If Z has a pro-chain of length $2l-1$ at least one block B_j must be identical to $\texttt{pro}^l$. For each B_j this happens with probability at most

$$(1 - 2^{-2^d})^{l\;2^{-2d}} \leq \exp(-\gamma\; l\; 2^{-(2d+2^d)}) \;.$$

This gives

$$\Pr[\texttt{pro}(Z) \geq 2l-1] \;\leq\; \frac{n}{l} \cdot \exp(-\gamma\; l\; 2^{-(2d+2^d)}) \;.$$

∎

Lemma 7 *For $d < \operatorname{llog} n$ there exists $Z \in \mathcal{D}Depth(d)$ such that*

$$\Pr\left[\texttt{pro}(Z) \;\geq\; \exp(\frac{1}{2} \cdot (2^d + \operatorname{llog} n))\right] \geq \frac{1}{2} \;.$$

Proof: Let the Z_i be independent random variables such that $\Pr[Z_i = \texttt{pro}] \geq 1 - 2^{-2^d}$. Then for any fixed block of Z of length l the probability that it contains only the symbol pro is given by $p \;:=\; (1 - 2^{-2^d})^l$. So

$$\begin{aligned}\Pr[\texttt{pro}(Z) \geq l] \quad &\geq \quad \sum_{i:=0}^{n/l-1} (1-p)^i \cdot p \;=\; 1 - (1-p)^{n/l} \;=\; 1 - (1 - (1 - 2^{-2^d})^l)^{n/l} \\ &\geq \quad 1 - (1 - \exp(-2 \cdot l \cdot \exp(-2^d)))^{n/l} \;\geq\; 1 - \exp(-\exp(-2 \cdot l \cdot \exp(-2^d)) \cdot n/l).\end{aligned}$$

For $l = \exp(\frac{1}{2} \cdot (2^d + \operatorname{llog} n))$ we get:

$$\Pr\left[\texttt{pro}(Z) \; > \; \exp(\frac{1}{2} \cdot (2^d + \operatorname{llog} n))\right] \geq \frac{1}{2} \; .$$

∎

Using these estimations we can show

Theorem 9 *For $d < \operatorname{llog} n$ holds*

$$\begin{array}{lcl} \frac{1}{4}(\operatorname{llog} n + 2^d) & \leq \; etime(\texttt{ADDITION}_n, \mathcal{D}Depth(d)) \; \leq & O(\operatorname{llog} n + 2^d) \; , \\ \operatorname{llog} n + 2^d - 2 & < \; avtime(\texttt{ADDITION}, \mathcal{D}Depth(d)) \; \leq & O(\operatorname{llog} n + 2^d) \; . \end{array}$$

Proof: The lower bounds for the expected measure follow from Lemma 5 and 7. Choose Z as in Lemma 7. Then

$$\sum_t t \cdot \Pr[time_C(Z) \leq t] \; \geq \; \frac{1}{4}(\operatorname{llog} n + 2^d) \; .$$

For the average measure the lower bound follows from the lower bound of the $\texttt{OR}_n$ -function (see theorem 5 and lemma 3), and the observation that

$$(\texttt{ADDITION}_n(X, 1^n))_n = \texttt{pro} \quad \Leftrightarrow \quad \texttt{OR}(X) = 1$$

Lemma 5 and 6 imply

$$\begin{array}{rcl} \Pr[\texttt{pro}(Z) \geq 2l - 1] & \leq & \frac{n}{l} \cdot \exp\left(\frac{-l}{2^{2^d + 2d}}\right) \; , \\ \Pr[time(X) \geq 8t + 5] & \leq & \frac{n}{2^t} \cdot \exp\left(\frac{-2^t}{2^{2^d + 2d}}\right) \; . \end{array}$$

Choosing $\tau \; := \; \operatorname{llog} n + 2^d + 2d$ the expected delay of the circuit C described in figure 5 can be bounded as

$$\begin{array}{rcl} etime(C) & \leq & \sum_t \Pr[time(X) = 8t + 5] \cdot (8 \cdot t + 5) \\ & \leq & \Pr[time(X) \leq 8 \cdot \tau + 5] \cdot (8 \cdot \tau + 5) \; + \; \Pr[time(X) > 8 \cdot \tau + 5] \cdot (4 \cdot \log n + 3) \\ & \leq & 8\tau + 5 + (4 \log n + 3) \cdot n \cdot 2^{-2^\tau} \cdot \exp\left(\frac{-2^\tau}{2^{2^d + 2d}}\right) \\ & \leq & 8 \cdot \tau \; + \; (4 \cdot \log n + 3) \cdot n \cdot 2^{-2^\tau} \cdot \frac{1}{n} + 5 \\ & \leq & 8 \cdot (\operatorname{llog} n + 2^d + 2d) + 6 \; . \end{array}$$

For the Av-measure we choose $T := 8(\operatorname{llog} + 2^d + 2d) + 5$ with $T^{-1}(n) = \exp 2^{\left(\frac{1}{8}(n-5) - 2^d - 2d\right)}$. This gives

$$\begin{array}{rcl} \sum_t \Pr[time(X) = 8 \cdot t + 5] \cdot \frac{T^{-1}(8t + 5)}{n} & \leq & \sum_{t=1}^{\log n} 2^{-t} \exp\left(-\frac{2^t}{2^{2d} 2^{2^d}}\right) \cdot T^{-1}(8 \cdot t + 5) \\ & \leq & \sum_{t=1}^{\log n} 2^{-t} \exp\left(\frac{-2^t}{2^{2^d + 2d}}\right) \exp\left(\frac{2^t}{2^{2^d + 2d}}\right) \; \leq \; 1 \; . \end{array}$$

∎

5 Conclusion

The following table summarizes the average case complexity results shown for basic Boolean functions.

function	probability distribution set	Depth	ECirTime	AvCirTime
`OR`, `AND`,	uniform	$\log n$	$2 - \frac{1}{2^{n-2}}$	$\Theta(\log\log n)$
`OR`, `AND`, `EQUAL`	$\mathcal{D}\mathrm{Depth}(d)$	$\log n$	$\Theta(\min(2^d, \log n))$	$\Theta(\min(\log\log n + 2^d, \log n))$
`PARITY`, `MAJORITY`	uniform	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$
`ADDITION`	$\mathcal{D}\mathrm{Depth}(d)$	$\Theta(\log n)$	$\Theta(\min(\log\log n + 2^d, \log n))$	$\Theta(\min(\log\log n + 2^d, \log n))$
$\mathtt{THRESHOLD}_a^n$	$\mathcal{D}\mathrm{Depth}(d)$	$\Theta(\log n)$	$\Theta(\min(\log a + 2^d, \log n))$	$\Theta(\min(\log\log n + \log a + 2^d, \log n))$

The upper bounds match the lower bounds in a strong sense. According to the definition it is only required that for each distribution μ computable in depth d the average delay complexity is small, that is for each μ one can find an average delay efficient circuit. Our construction actually yield a single circuit design that means efficient for all μ in $\mathcal{D}\mathrm{Depth}(d)$ (up to a permutation of the inputs). For the addition we have even obtained a single circuit that is optimal for all d simultaneously.

These new circuit designs may have practical applications since the constant factors are small. Most notably, we have shown that for a large class of distributions the addition can be performed with an average delay of order $\mathrm{llog}\, n$. These techniques can also be exploited to design algorithms for parallel machines achieving average time bounds of sublogarithmic order.

By extending these methods we can achieve an optimal average delay for addition and similar prefix problems by circuits of linear size [JRSW93] as it was obtained by Ladner/Fischer for the worst case [LaFi80].

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 [Reif93]. He observed that circuit depth $O(\mathrm{llog}\, n)$ is sufficient if one allows a small portion of input vectors for which a wrong result will be obtained. Conbining such a circuit with a worst case circuit of depth $O(\log n)$ that is always correct and using one gate of unbounded fanin a circuit with expected delay $O(\mathrm{llog}\, n)$ can be constructed. Thus only two cases occur, the optimal averge case behaviour or the worst case. Our circuits dynamically adapt to each specific input and always use not much more than the minimal set of input bits necessary to compute a prefix correctly.

We have also studied the behaviour of asymptotic average case circuit delay and the question whether malign distributions exist for these nonuniform complexity classes. Results are reported in [JRS94].

Acknowledgement
We thank Thomas Zeugmann, Maciej Liśkiewicz, Michael Goedecke, Stephan Weis, and Ingo Wegener for fruitful discussions, helpful hints and pointers to the literature.

References

[BCGL92] S. Ben-David, B. Chor, O. Goldreich, M. Luby, *On the Theory of Average Case Complexity*, J. CSS 44, 1992, 193-219.

[DGY89] I. David, R. Ginosar, M. Yoelli, *An Efficient Implementation of Boolean Functions and Finite State Machines as Self-Timed Circuits*, ACM SIGARCH, 1989, 91-104.

[Krap78] V. Krapchenko, Depth and Delay in a Network, Soviet Math. Dokl. 19, 1978, 1006-1009.

[Gure91] Y. Gurevich *Average Case Completeness*, J. CSS 42, 1991, 346-398.

[JRS94] A. Jakoby, R. Reischuk, C. Schindelhauer, *Malign Distributions for Nonuniform Average Case Complexity,* Technical Report, TH Darmstadt, 1994.

[JRSW93] A. Jakoby, R. Reischuk, C. Schindelhauer, S. Weis, *The Average Case Complexity of the Parallel Prefix Problem,* Technical Report, TH Darmstadt, 1993, to be presented at ICALP, 1994.

[LaFi80] R. Ladner, M. Fischer, *Parallel Prefix Computation,* J. ACM 27, 1980, 831-838.

[LBS93] W. Lam, R. Brayton, A. Sangiovanni-Vincentelli, *Circuit Delay Models and Their Exact Computation Using Timed Boolean Functions*, ACM/IEEE, Design Automation Conference, 1993, 128-133.

[Levi86] L. Levin, *Average Case Complete Problems*, SIAM J. Computing 15, 1986, 285-286.

[LiVi92] M. Li, P. Vitanyi, *Average Case Complexity under the Universal Distribution Equals Worst-Case Complexity,* IPL 42, 1992, 145-149.

[Milt91] P. Miltersen, *The Complexity of Malign Ensembles,* SIAM. J. Comput. 22, 1993, 147-156.

[Reif93] J. Reif, *Probabilistic Parallel Prefix Computation,* Comp. Math. Applic. 26, 1993, 101-110.

[ReSc93a] R. Reischuk, C. Schindelhauer, *Precise Average Case Complexity,* Proc. 10. GI-AFCET Symposium on Theoretical Aspects of Computer Science, STACS 1993, Springer Lecture Notes, 650-661.

[ReSc93b] R. Reischuk, C. Schindelhauer, *Precise Average Case Complexity Measures,* Technical Report, ICSI Berkeley, TR-93-049, August 1993.