Fault tolerant circuits and probabilistically checkable proofs *

Anna Gál
The University of Chicago
Department of Computer Science
1100 E. 58th Street
Chicago, IL 60637
panni@cs.uchicago.edu

Mario Szegedy
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill NJ 07974
ms@research.att.com

Abstract

We introduce a new model of fault tolerance for Boolean circuits. We consider synchronized circuits and we allow an adversary to choose a small constant fraction of the gates at each level of the circuit to be faulty. We require that even in the presence of such faults the circuit compute a "loose version" of the given function. We show that every symmetric function has a small (size $O(n)$, depth $O(\log n)$) fault tolerant circuit in this model. We also show a perhaps unexpected relation between our model and probabilistically checkable proofs.

1 Introduction

In this paper we study Boolean circuits with gates that may fail. It is important to have constructions that reliably perform a given computation task even in the presence of errors, without increasing the size of the circuit by too much.

Much of the work in this area uses the model of independent random faults. The assumption is that the gates of the circuit may fail with probability bounded by some small constant, and the failures occur independently. The circuit should produce the correct result with high probability on any input. In this model it is known [16], [9], [17] that any function can be reliably computed by $L \log L$ size circuits, where L is the size needed to compute the function without faults. Pippenger [17] proved that almost all functions can be computed with only constant redundancy. By *redundancy* we mean the fraction of the size needed for computation with faults and the size needed to compute the function without faults. These results hold if the probability of a gate being faulty is bounded by some constant $< 1/2$. The logarithmic redundancy is necessary for some functions [8], [10], [19]. Thus, as long as the faults occur randomly and independently, we can build circuits with optimal redundancy that work correctly with high probability, even if a constant fraction of the gates is faulty.

The constructions above do not work if the faults are not random. If an adversary chooses to destroy critical parts of the circuit, for example the output gate, then the circuit cannot give the correct result. Yet in many cases we would like to deal with worst case behavior. Previous attempts to build reliable circuits in the case when the faults are not random allowed using a certain number of absolutely reliable gates. The practical justification for introducing absolutely reliable elements is that circuits can be built using more expensive hardware for certain gates, that will be more reliable than the rest of the circuit. Reliable circuits with small redundancy were obtained but they could tolerate only a negligible (exponentially small) fraction of the gates being faulty [11], [12], [22]. Moreover these constructions have exponential size, thus they give small redundancy only for functions that require exponential size

*Presented at Structure in Complexity Theory '95

circuits even without faults. (Note that almost all Boolean functions require exponential size circuits [20], [15], but there is still no explicit function known to require superlinear size circuits.)

In a recent paper [13] Kleitman, Leighton and Ma investigate the *short-circuit model of gate failure* for Boolean circuits in which a faulty gate is restricted to output one of its input values. They show that in this model it is possible to tolerate worst case faults without introducing absolutely reliable gates.

To our knowledge, if the type of the failures is not restricted, there are no previous results achieving small redundancy tolerating a constant fraction of gates being faulty in the case when the faults are not random. For a survey of related results see [18].

We propose a new model of fault tolerance for Boolean circuits. We consider synchronized (leveled) circuits and let the adversary choose a certain fraction of the gates at each level to be faulty. Our model could be thought of as using a constant number of absolutely reliable gates for the last few levels of the circuit. Thus, for practical purposes it can be justified similarily to the previous approaches to handle non-random faults.

Instead of trying to correctly compute the function on every input we define the *loose computation* of a function (Section 2). We prove that every symmetric function has a synchronized circuit of size $O(n)$ and depth $O(\log n)$ that performs the loose computation of the function even if a constant fraction of the gates at each level is chosen to be faulty by an adversary. This bound is within a constant factor of the complexity of arbitrary symmetric functions (that depend on all n inputs) in the fault free case [23].

It may seem that introducing the loose computation of Boolean functions is a serious relaxation of the problem. In Section 2 we argue that this is not the case.

There is a connection between our model and recently discovered constructions for proof encodings [1], [5] (also see [7] for a survey). In Section 5 we show that if circuits computing certain Boolean functions can be turned into fault tolerant circuits with only a polynomial increase in size and keeping the depth within a constant factor, then $NP = PCP(\log n, \log n)$ [1] follows. The existence of such fault tolerant circuits is verified by Theorem 9. However, combining the two theorems we do not get a simpler proof to the notoriously hard characterization of NP in [1]: our construction uses an even harder result stating that $NP = PCP(\log n, 1)$ [5].

2 The model

In the model introduced by von Neumann the gates of a Boolean circuit fail (produce an incorrect value) independently with a probability bounded by a constant. The results of [16], [9], [17] give a transformation that takes a circuit C into another circuit C' such that:

1. The transformation takes polynomial time in the size of C.
2. The size of C' is $O(S(C) \log S(C))$, where $S(C)$ denotes the size of cicuit C.
3. The depth of C' is $O(D(C))$, where $D(C)$ is the depth of circuit C.
4. There exist $\epsilon > 0$ and $p < 1/2$ such that if the gates of C' fail independently with probability bounded by ϵ then for every input x the probability that $C'(x) \neq C(x)$ is at most p.

In the present paper we raise the question: can we obtain analogous results when the errors are not random? The first problem to overcome is the following. If the adversary biases a few critical gates or the output bit then the circuit inadvertently outputs an incorrect value. We avoid this problem by restricting the number of faulty gates for each level of the circuit to some small percentage of the number of the gates at that level.

All the circuits considered here will be *synchronised*. This means that the gates are classified according to their levels numbered from 0 to $D(C)$, where $D(C)$ denotes the depth of the circuit. Wires go only between consecutive levels (a gate on the i^{th} level is fed from gates on the $(i-1)^{th}$ level for $i = 1, \ldots, D(C)$). We would like to build synchronized fault tolerant circuits that give the correct output even if at each level a small, but maliciously choosen constant fraction of the gates are malfunctioning.

Instead of trying to correctly compute the function on every input we define the *loose computation* of a Boolean function f.

Definition 2.1 *For any computational device M we say that M δ-loosely computes f if*

1. *whenever $f(x) = 1$ then $M(x) = 1$.*
2. *If $f(z) = 0$ for every z with $d(x, z) \leq \delta n$, then $M(x) = 0$.*

Here $d(x, z)$ denotes the Hamming distance between words x and z. We remark that M can output an arbitrary value or no value at all if input x does not belong to the above two categories.

At the first sight it may appear that computational devices that loosely compute a function f can be far weaker than those that are able to get the value of f everywhere. For instance it is well known [2] that there is an AC_0 circuit that loosely computes the majority function, whereas majority itself cannot be computed in AC_0. One may speculate similarly that every function in NP might be loosely computed in P. Proposition 1 together with Remark 1 shows that this is not the case (unless $P = NP$).

Definition 2.2 *For an error correcting code E_n with codewords of length q_n and for a function $f : \{0,1\}^n \to \{0,1\}$ we define the function $f \diamond E_n : \{0,1\}^{q_n} \to \{0,1\}$ as follows.*

- *$(f \diamond E_n)(z) = 0$ for all z where z is not a codeword of the code E_n.*
- *If $z = E_n(x)$ then $(f \diamond E_n)(z) = f(x)$.*

Proposition 1 *If the Hamming distance of any two codewords in E_n is at least δq_n, and M is a computational device that computes $f \diamond E_n$ in a δ-loose manner then $M(E_n(x)) = f(x)$ on any input x.*

Remark 1 *It is known from coding theory that there exist linear binary codes E_n with the following properties.*

1. *The matrix of E_n can be polynomailly computed in n. (This also means that the length of the codeword, q_n, is polynomial in n.)*
2. *The Hamming distance of any two codewords in E_n is at least δq_n, for some small constant δ.*

There are two ways to interpret the assumption of our model about allowing at most a small constant fraction of the gates to be faulty at each level of the circuit. We may or may not allow the adversary to bias gates at the input level. The following proposition shows that we cannot expect to find a simple transformation that turns an arbitrary circuit into a fault tolerant circuit in our model in the harder case, when faults are allowed at the input level.

Proposition 2 *Assume that there is a polynomial time transformation that would turn any circuit C computing a function f into a circuit C' of size polynomial in the size of C such that C' loosely computes f, even if an adversary biases a small constant fraction of the gates at every level of C' including the input level. Then P=NP.*

Proof: We show that the hypothesis implies that the satisfiability of a 3CNF formula could be decided in polynomial time. Let ϕ be a 3CNF formula of n variables. Let us introduce n^2 new variables $y_1, \ldots, y_{n^2}$. Let C be the circuit that computes $f = y_1 \wedge \ldots \wedge y_{n^2} \wedge \phi$. Let C' be the fault tolerant circuit that computes f in a δ-loose manner. By our assumption we can build C' from C in polynomial time. Let z be the input $1 \ldots 1$. We claim that $C'(z)$=1 (without faults) iff ϕ is satisfiable. Indeed, if ϕ is not satisfiable then f is identically 0 and hence $C'(z) = 0$. On the other hand if there is an assignment a such that $\phi(a) = 1$ then by changing only at most $n \leq \delta n^2$ digits of z we can turn z into a z' such that $f(z') = 1$. Observe that loose computation preserves the 1 values. But then by the fault tolerant property of C', $C'(z) = 1$ holds. Thus building C' as above would allow us to differenciate between satisfiable and non satisfiable instances.

An observation of Gábor Tardos [21] shows that a similar statement also holds if we do not allow faults at the input level.

3 Halvers

Our construction for symmetric functions is based on the ϵ-halvers of Ajtai, Komlós and Szemerédi [3]. An ϵ-halver is a bounded depth comparator network with the property that for any set of the l smallest (largest) inputs, where $l \leq n/2$, at most ϵl elements will be among the last (first) $n/2$ outputs. In other words, an ϵ-halver is a halver (a network that separates the $n/2$ largest and the $n/2$ smallest inputs into two disjoint sets) that can misplace at most an ϵ fraction of the elements.

Constant depth linear size ϵ-halvers where constructed in [3]. More efficient ϵ-halvers (with smaller constants) based on units called k-comparators were proven to exist in [4], but no explicit construction has been given. A k-comparator is a unit that sorts its k inputs.

All our constructions work using either ϵ-halver. If the more efficient ϵ-halvers are used, we assume that each k-comparator is realized by depth k 2-comparator networks. We note that the depth of the realization of such ϵ-halvers by 2-comparator networks will be $2k$, since the ϵ-halvers of [4] are depth 2 k-comparator networks. In what follows by the depth of an ϵ-halver we always mean the depth of the realization by 2-comparator networks.

For $0-1$ inputs ϵ-halvers can be realized by monotone Boolean circuits. Each comparator will be realized by an AND - OR gate pair. The AND gate computes the minimum, the OR gate computes the maximum of the two common input bits. The resulting circuit will have the same depth as the original ϵ-halver. In what follows by ϵ-halvers we usually mean the Boolean circuits of the above mentioned form realizing the original ϵ-halvers.

For our purposes we have to consider ϵ-halvers with faulty gates.

Definition 3.1 *Denote by $g(y_1, y_2)$ the function that the gate g is supposed to compute on input y_1, y_2. A gate g is* faulty *if its output is different from $g(y_1, y_2)$.*

Definition 3.2 *Let C be a circuit with no faulty gates, and let $\tilde{C}$ be a copy of the same circuit with possibly faulty gates. The output of a gate of $\tilde{C}$ is* incorrect *if it is different from the value computed by the same gate in C. Note that the output of a gate may be incorrect because the gate is faulty, or because the inputs of the gate are incorrect.*

The following lemma says that in ϵ-halvers the incorrect outputs cannot cause too much damage on subsequent levels.

Lemma 3 *At level d of an ϵ-halver the number of incorrect outputs is at most the number of incorrect outputs at level $d-1$ plus the number of faulty gates at level d.*

Proof: Recall that a comparator in our ϵ-halvers is an AND - OR gate pair with common inputs. Level d of the ϵ-halver consists of disjoint comparators. Thus to prove the lemma it is enough to prove that the number of incorrect output bits of a comparator is at most the number of its incorrect input bits plus the number of its faulty gates. (Thus we only have to deal with a single comparator with two inputs and two outputs at a time.) Let us consider the case when there are no faulty gates and exactly one of the two input bits is incorrect. Notice that since the comparator only permutes the two input bits we can have only one incorrect bit in the output in this case. In all other cases the statement for a single comparator is trivially true.

Definition 3.3 *We say that a circuit is* γ-faulty *if at most* γ *fraction of the gates on each level is faulty.*

When talking about γ-faulty circuits we do not consider the inputs to be gates, but just values fed into the circuit. The reason for this is that our constructions will involve γ-faulty circuits connected to one another such that the outputs of one of them provide the inputs to others. Once we receive a set of values as output of a previous circuit it will be fed to the next circuit without further damage. (Another way of making this clear is noting that the wires of our circuits work correctly no matter what the adversary does.) However we will have to take into account that the adversary may destroy some of the real input gates of the whole construction.

For an ϵ-halver denote by L_1 (L_0) the number of $1's$ ($0's$) in the lower part of the output, and by U_1 (U_0) the number of $1's$ ($0's$) in the upper part of the output.

We shall need the observations below in the next section.

Lemma 4 *Consider a γ-faulty ϵ-halver of depth c with m inputs. Let the number of 0's in the input be a. If $a \geq m/2$ then*

$$L_1 \leq \epsilon(m-a) + c\gamma m$$

and

$$(a - m/2) - c\gamma m \leq U_0 \leq (a - m/2) + \epsilon m/2 + c\gamma m\,.$$

If $a \leq m/2$ then

$$U_0 \leq \epsilon a + c\gamma m$$

and

$$(m/2 - a) - c\gamma m \leq L_1 \leq (m/2 - a) + \epsilon m/2 + c\gamma m\,.$$

Proof: We first estimate what would be the value of L_1 and U_0 in a correct ϵ-halver. Lemma 3 guarantees that the number of incorrect outputs in the last level of a γ-faulty ϵ-halver of depth c is at most $c\gamma m$.

We summerize what we have learned in the following lemma:

Lemma 5 *Consider a 0-1 string of length m, such that there are z 0's followed by $m - z$ 1's. Denote by z_L (z_U) the number of 0's in the lower (upper) part of this ordered string.*

Consider a γ-faulty ϵ-halver of depth c with m inputs. Let the number of 0's in the input be a, such that $\mid a - z \mid \leq r$. Then

$$\mid U_0 - z_U \mid \leq r + (\epsilon + 2c\gamma)m/2$$

and

$$\mid L_0 - z_L \mid \leq r + (\epsilon + 2c\gamma)m/2\,.$$

Note that a similar statement holds for the number of 1's in the output.

4 Symmetric functions

In this section we construct fault tolerant circuits for any symmetric function.

First we show that γ-faulty circuits can compute overwhelming majority functions correctly on all inputs where they are defined.

The overwhelming majority function Maj_k^n has value 1 or 0, respectively, if the number of 1's or the number of 0's in the input is at least k, and it is undefined otherwise.

Lemma 6 *Let $k > 3/4n$. Then for some $\gamma > 0$, there is a γ-faulty circuit of size $O(n)$ and depth $O(\log n)$ computing Maj_k^n correctly on every input belonging to its domain.*

Proof: The building blocks of our construction are triplets of ϵ-halvers with a majority preserving property. A similar component with this property was introduced by Assaf and Upfal in [6] for building fault tolerant sorting networks in the case of random faults. They call a comparator network with m inputs and $m/2$ outputs a *majority preserver* if it guarantees that if at least a given ($> 1/2$) constant fraction of the m inputs have the same value, then this value appears in at least the same given constant fraction of the $m/2$ outputs. We construct a majority preserver that tolerates a small constant fraction of worst case errors at each of its levels as follows.

First we apply an m-input ϵ-halver to the m inputs. Let us call it M-halver. Then we apply two $m/2$-input ϵ-halvers, one to the lower part the other to the upper part of the output of the M-halver. We refer to these two ϵ-halvers as L-halver and U-halver respectively. The output of the majority preserver will consist of the upper part of the output of the L-halver and of the lower part of the output of the U-halver.

We use a family of ϵ-halvers of depth c with the same parameters ϵ and c for all input lengths. For appropriate constants ϵ and c such family exists.

Let a be the number of 0's in the input of the majority preserver. Suppose

$$a \geq (3/4 + c\gamma)m\,. \tag{1}$$

Denote by $L_0(M)$ ($U_0(M)$) the number of 0's in the lower (upper) part of the output of the M-halver. We use similar notation for the L-halver and U-halver. By Lemma 4 and the inequality (1)

$$L_0(M) \geq m/2 - (\epsilon + 2c\gamma)m/2$$

and

$$U_0(M) \geq m/4\,.$$

Applying Lemma 4 to both the L-halver and the U-halver, we get

$$U_0(L) + L_0(U) \geq (1 - 2(\epsilon + 2c\gamma))m/2\,. \tag{2}$$

We have proved that if the number of 0's in the input of a majority preserver is at least $(3/4 + c\gamma)$ fraction of the input then at least a $(1 - 2(\epsilon + 2c\gamma))$ fraction of its output is also 0. A similar statement holds about the number of 1's.

We choose γ small enough to have $k > 3/4 + c\gamma$ and such that the following inequality holds:

$$1 - 2(\epsilon + 2c\gamma) \geq 3/4 + c\gamma\,. \tag{3}$$

Combining j majority preservers by feeding the outputs of one as inputs to the next we compute a set of size $n/2^j$ with the property that its overwhelming majority is the same as the overwhelming majority of the input. For $n/2^j < 1/\gamma$ we can finish the computation with a small circuit that has fewer than $1/\gamma$ gates at each of its levels, thus the adversary cannot destroy any of its gates. This circuit has to compute only the usual majority of its inputs, and that gives us the correct result.

Next we give a construction for any threshold function. The threshold function Th_k^n has value 1 if and only if at least k of the n input bits have value 1.

Theorem 7 *For any $\delta > 0$ there is a $\gamma > 0$ such that for any threshold function Th_k^n there is a synchronized circuit such that*

1. *If an adversary destroys a γ fraction of the gates on every level (including the input level), the circuit still computes Th_k^n in a δ-loose manner.*
2. *The size of the circuit is $O(n)$.*
3. *The depth of the circuit is $O(\log n)$.*

Proof: According to the definition of loose computation we have to build a circuit that outputs

- 1 if the number of 0's is $\leq n-k$
- 0 if the number of 0's is $\geq n-k+\delta n$
- an arbitrary value otherwise.

Let us consider the correctly ordered input and consider the set of elements at positions $n-k+1,\cdots,n-k+\delta n$. Notice that whenever our circuit must output 1 all these elements are 1, and whenever the circuit must output 0 all these elements are 0.

Choose t and l such that $ln/2^t \geq n-k$ and $(l+1)n/2^t \leq n-k+\delta n$. (We note that the value of t depends only on n and δ, but not on k. This will be important for the proof of Theorem 8.) Denote by T the set of elements at positions $ln/2^t+1,\cdots,(l+1)n/2^t$ of the correctly ordered input. Denote by z_T the number of 0's in the set T.

Our construction will consist of two parts. The first part denoted by C_1 will have n inputs and $n/2^t$ outputs. For any permutation of the input the number v of 0's in the output will satisfy

$$\mid v - z_T \mid \leq (\epsilon + 2c\gamma + \gamma)n\,.$$

ϵ and c are parameters of the ϵ-halvers we use. The second part will be the construction from Lemma 6 computing the overwhelming majority Maj_s^m of the outputs of C_1 where $m = n/2^t$ and $s = (1-2^t(\epsilon + 2c\gamma+\gamma))m$.

For applying Lemma 6 we need

$$1 - 2^t(\epsilon + 2c\gamma + \gamma) \geq 3/4 + c\gamma$$

to hold, which is satisfied if $\delta/16 > \epsilon + 3c\gamma$. This inequality determines our possible choices of the parameters ϵ and γ. c will be determined by ϵ and the kind of ϵ-halvers we use. Note that the increase in size and depth relative to the simplest nonfaulty circuits will be decided by c.

The circuit C_1 will be a comparator network consisting of t γ-faulty ϵ-halvers each of depth c. The i-th ϵ-halver will have $n/2^{i-1}$ inputs and $n/2^i$ outputs. Let $l_1\cdots l_t$ be the binary representation of l. If $l_i = 0$ then the input to the $i+1$-st ϵ-halver will be the lower part of the output of the i-th ϵ-halver. If $l_i = 1$ we take the upper part as input to the next ϵ-halver.

Let a_0 be the number of 0's in the correct input. Since the input of C_1 is provided by the input gates of the whole circuit, in our model the adversary is free to destroy a γ-fraction of the inputs of C_1 as well. Let a denote the number of 0-valued input gates. We have $a_0 - \gamma n \leq a \leq a_0 + \gamma n$. We can apply Lemma 5 to the first ϵ-halver with $r = \gamma n$. Repeatedly applying Lemma 5 we get

$$| v - z_T | \leq \gamma n + \sum_{i=1}^{t} (\epsilon + 2c\gamma) n / 2^i \leq (\epsilon + 2c\gamma + \gamma) n$$

which concludes the proof of the theorem.

Theorem 8 *For any $\delta > 0$ there is a $\gamma > 0$ such that for any symmetric function f there is a synchronized circuit with the following properties.*

1. *If an adversary destroys a γ fraction of the gates on every level (including the input level), the circuit still computes f in a δ-loose manner.*
2. *The size of the circuit is $O(n)$.*
3. *The depth of the circuit is $O(\log n)$.*

Proof: Any symmetric function can be represented by a 0-1 string of length $n+1$, where the i-th element of the string is 1 if and only if $f = 1$ on inputs containing exactly i 1's.

If every set of consecutive 0's in this string has length $< \delta n$, then the δ-loose computation of f becomes trivial. (Giving the output 1 for every input is appropriate.)

Suppose the string representing f contains $h \geq 1$ sets of consecutive 0's each of length $\geq \delta n$. Let the i-th set be (l_i, u_i). For the δ-loose computation of f it is enough to δ-loosely compute each $\neg Th^n_{l_i}$ and each $Th^n_{u_i}$. We compute these $2h$ functions in parallel. We use the construction from Theorem 7 for δ-loosely computing threshold functions. The negation of a threshold function can be δ-loosely computed in an analogous way. We choose $\gamma = \tilde{\gamma}/2h$, and we choose $\tilde{\gamma}$ as required for the δ-loose computation of threshold functions in Theorem 7. We note that we choose the parameters δ, ϵ, c and $\tilde{\gamma}$ to have the same value for each of the $2h$ circuits. This guarantees that the number of gates at corresponding levels of the $2h$ circuits is exactly the same. Thus by destroying $\leq \gamma$ fraction of the gates at a given level, it is not possible to destroy more than a $\tilde{\gamma}$ fraction of the gates of a level in any one of the $2h$ circuits. Since $h < 1/\delta < 1/2\gamma$ after computing the $2h$ values in parallel we can combine them without any more faulty gates.

5 Fault tolerant circuits and probabilistically checkable proofs

Recently, the language class NP received a new characterization in terms of probabilistically checkable proofs. This characterization made it possible to show for the first time that to approximate the largest clique size of any graph within a constant factor is NP hard. In [1] it is shown that $NP = PCP(\log n, \log n)$. The class $PCP(f(n), g(n))$ is defined as the set of those languages L for which there is a randomized, polynomial time oracle machine $M^y(x, r)$ with oracle y, input x of length n and a random string r, such that:

1. $|r| = O(f(n))$.
2. For every fixed x and r, M reads only $O(g(n))$ bits of y.
3. If $x \in L$ then there is an oracle y such that $M^y(x, r) = 1$ with probability one over all r's.

4. If $x \notin L$ then for every oracle y the probability over all r's that $M^y(x, r) = 0$ is at least $3/4$.

The result of Arora and Safra [1] was improved by Arora et al. [5] to show that $NP = PCP(\log n, 1)$. Both proofs rely on relatively difficult techniques from classical algebra, coding theory and combinatorics. In what follows, we give a theorem about fault tolerant circuits in our model that imply the result in [1]. As appealing as it may sound, we do not get a simpler proof to the notoriously hard characterization of NP by Arora and Safra. Our construction uses the even harder result of [5].

We would like to emphasize that Theorem 9 is presented in this section because of its connections to probabilistically checkable proofs, but we believe that it is of independent interest for fault tolerant computations in our model.

Theorem 9 *Let C be a Boolean circuit and let $f : \{0,1\}^n \rightarrow \{0,1\}$ be the Boolean function computed by C. There exist a code $E = E_C$ and a circuit C' such that C' computes $f \diamond E$ in a δ-loose manner for every $\delta > 0$, even if an adversary destroys $\gamma(\delta) > 0$ fraction of the gates on every level (including the input level). Moreover E and C' have the following properties:*

1. *$|E(x)| \leq q(|x|)$ for some polynomial q independent of C.*

2. *The Hamming distance between any two codewords of E is at least $\delta_0|E|$ for some $0 < \delta_0 < 1$ independent of C. (δ_0 is a function of δ.)*

3. *$D(C') \leq O(\log S(C))$. (This implies that $S(C')$ is polynomial in $S(C)$.)*

4. *C' can be computed from C in polynomial time and $E(x)$ can be computed from C and x in polynomial time.*

Sketch of the proof: The proof is based on using the results of Arora et al. [5] and Theorem 7.

Let C^1 be the subset of $\{0,1\}^n$ for which C evaluates to 1 and C^0 be the set for which C evaluates to 0. In [5] an algorithm is described for turning a circuit C into a family $\{C_i\}_{i \in I}$ of constant size circuits with input $(G(x), Y)$, where $G(x)$ is an appropriate error correcting encoding of the input x to C and Y is an advise string (both have length polynomial in $S(C)$). This family has the property that

1. For every $x \in C^1$ there is a Y such that for all $i \in I$ we have $C_i(G(x), Y) = 1$.

2. For every $x \in C^0$ and for every Y we have that

$$\mathrm{Prob}_{i \in I}(C_i(G(x), Y) = 0) \geq 3/4 \ .$$

A modification by Lund and Spielmann [14] of the construction in [5] ensures that for every $x \in C^1$ the corresponding Y in condition 1 is unique. More precisely conditions 1 and 2 can be replaced by:

3. For every $x \in C^1$ there is a unique Y_x such that for all $i \in I$ we have $C_i(G(x), Y_x) = 1$. Also, for every $\delta > 0$ there exist $\epsilon > 0$ such that $\mathrm{Prob}_{i \in I}(C_i(H, Y) = 0) \leq \epsilon$ implies that $dist((H, Y), (G(x), Y_x)) \leq \delta|(H, Y)|$ for some $x \in C^1$.

We would like to build circuit C' such that it contains all members of the family $\{C_i\}_{i \in I}$. Because of technical reasons we want the members of the above family to have disjoint inputs. To achieve this we form $G'(x)$ and Y'_x by repeating the bits of $G(x)$ and Y_x as many times as the number of times for which they appear as input for some $\{C_i\}$ $(i \in I)$. We note that this way each bit of $G(x)$ (and Y, respectively) will be repeated for the same number of times because of the symmetry of the construction in [5].

The code E is defined as follows. For $x \in C^1$ we define $E(x) = (G'(x), Y'_x)$. For $x \in C^0$ we define $E(x) = (G'(x), 0 \ldots 0)$.

We build C' as follows: We turn the family $\{C_i\}_{i \in I}$ into a family of synchronized circuits such that its members have all disjoint inputs and all outputs are at the same level. We include additional tests for checking that the groups of repeated bits indeed consist of identical bits. This is done by building a bounded degree expander over each group of input bits that must be identical and checking equality for every edge of these expanders. We denote this additional family of constant size circuits by $\{D_j\}_{j \in J}$. We synchronise these ciruits with the members of the family $\{C_i\}_{i \in I}$ such that all outputs are at the same level. It will be also important for us that the size of I and J are the same within a constant factor ($|J| = \theta(|I|)$).

Observe that $\wedge_{i \in I} C_i(H, Y) \wedge \wedge_{j \in J} D_j(H, Y) = f \diamond E(H, Y)$. Using the construction described in Theorem 7 we build a fault tolerant circuit that computes the AND of the output bits of the circuits C_i for $i \in I$ and D_j for $j \in J$.

By condition 3. and by Theorem 7 this circuit will compute $f \diamond E$ in a δ-loose manner even if γ fraction of the gates is faulty at each level, including the input level, for appropriately chosen $\gamma = \gamma(\delta)$.

Remark 2 *The additional tests $\{D_j\}_{j \in J}$ are only necessary to tolerate faults that may occur at the input level. If we exclude the possibility of faults occuring at the input level, the additional tests may be left out from the construction.*

Theorem 10 *From Theorem 9 the theorem $NP = PCP(\log n, \log n)$ ([1]) follows.*

Proof: It is easy to see that $PCP(\log n, \log n) \subseteq NP$. What we need to show is that $NP \subseteq PCP(\log n, \log n)$ using Theorem 9. For this it is enough to show that 3SAT is in $PCP(\log n, \log n)$. Let ϕ be a 3SAT instance of n variables and C_ϕ be a circuit that computes ϕ. Clearly C_ϕ has size polynomial in n. We use Theorem 9 to turn C_ϕ into a fault tolerant circuit C'_ϕ with depth $O(\log n)$ that computes $\phi \diamond E$ for some code E. Observe that C'_ϕ outputs 1 on some input iff ϕ is satisfiable. We describe a checking procedure with a "prover" and a "verifier" as follows. (The verifier can be looked at as a polynomial time random oracle machine $M^y(x, r)$. The prover is the one to provide the verifier with advice y.)

We denote the i^{th} gate on the j^{th} level by $g(i, j)$. Let k_j denote the number of the gates on the j^{th} level. The prover's intention is to show that C'_ϕ outputs 1 on some input. He presents the verifier with the evaluation of all gates of C'_ϕ for an input that he claims is a satisfying assignment for C'_ϕ (i.e. C'_ϕ outputs 1 on this input).

The verifier chooses a random number r from 1 to $\max_j k_j$ and checks the value of the gates $\{g(r \, mod \, k_j, j)\}_{j=1..D(C'_\phi)}$ and the value of the gates that are inputs to these gates (two for each gate). The number of checkbits is at most $3D(C'_\phi) = O(\log n)$. The verifier accepts if the output bit has value 1 and if none of the gates in the set $\{g(r \, mod \, k_j, j)\}_{j=1..D(C'_\phi)}$ makes an error in the computation. Clearly, the true prover passes the test of the verifier with probability 1.

What is the probability that the cheating prover is caught? If at each level at most γ percent of the gates is faulty, then by the fault tolerant property of C'_ϕ the output is zero, thus the verifier rejects with probability one. Otherwise there is a level on which γ fraction of the gates is faulty. This will be revealed with probability at least $\gamma/2$ (we may loose a factor of at most 2 because k_l may not divide $\max_j k_j$). With a constant number of independent repetitions this probability can be brought above 3/4.

Remark 3 *We note that the proof of Theorem 10 does not require the full strength of Theorem 9. It would be enough to have a circuit C' that δ-loosely computes $f \diamond E$ for just some $\delta > 0$ in the presence of faults. We could have also relaxed the fault tolerant property of C' on the input level when proving Theorem 10. However, the fact that C' can be computed from C in polynomial time is crucial.*

6 Open problems

Theorem 9 (together with Proposition 1) shows that if we allow the input to be given in certain encoded form, we can obtain for any Boolean function f circuits with small redundancy computing f exactly and tolerating worst case faults in our model. However the encoding used in Theorem 9 depends on the function f, thus it does more than just encoding the input, it also encodes information about the value of the function. It would be interesting to achieve an analogous result using a code that does not depend on the function.

It is an interesting question whether a different construction could be used in the proof of Theorem 9. If there was a simpler encoding, that could be produced by an efficient fault tolerant computation, Theorem 9 could provide direct fault tolerant constructions for arbitrary functions.

On the other hand according to Theorem 10, a different proof of Theorem 9, that does not rely on constructions of probabilistically checkable proofs would provide an alternative technique for the area of probabilistically checkable proofs.

Finally, without giving constructions for arbitrary functions, it would be interesting to find direct fault tolerant constructions for other than symmetric functions.

Acknowledgements:

The authors would like to thank János Simon and Jean-Pierre Seifert for helpful remarks on the manuscript. They also thank Carsten Lund and Dan Spielmann for helpful discussions.

References

[1] S. Arora and S. Safra, "Probabilistic checking of proofs," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 2-13.

[2] M. Ajtai, M. Ben-Or, "A theorem on probabilistic constant depth computations," In *Proc. of "16-th ACM Symposium on Theory of Computing"*, 1984, pp. 471-474.

[3] M. Ajtai, J. Komlós, E. Szemerédi, "An $O(n \log n)$ sorting network," In *Proc. of "15-th ACM Symposium on Theory of Computing"*, 1983, pp. 1-9.

[4] M. Ajtai, J. Komlós, E. Szemerédi, "Halvers and expanders," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 686-692.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, "Proof verification and hardness of approximation problems," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 14-23.

[6] S. Assaf, E. Upfal, "Fault tolerant sorting network", In *Proc. of "31-st IEEE Symposium on the Foundations of Computer Science"*, 1990, pp. 275-284.

[7] L. Babai, "Transparent proofs and limits to approximation," University of Chicago Technical Report CS93-15, 1993.

[8] R. L. Dobrushin and S. I. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.* 13, 1977, pp. 59-65.

[9] R. L. Dobrushin and S. I. Ortyukov, "Upper bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.* 13, 1977, pp. 203-218.

[10] P. Gács and A. Gál, "Lower Bounds for the Complexity of Reliable Boolean Circuits with Noisy Gates," *IEEE Trans. Inform. Theory*, Vol. 40, No. 2, 1994, pp. 579-583.

[11] G. I. Kirienko, "On self-correcting schemes from functional elements," *Probl. Kibern.* 12, 1964, pp. 29-37.

[12] G. I. Kirienko, "Synthesis of self-correcting schemes from functional elements for the case of growing number of faults in the scheme," *Diskret. Anal.* 16, 1970, pp. 38-43.

[13] D. Kleitman, T. Leighton and Y. Ma, "On the design of reliable Boolean circuits that contain partially unreliable gates," In *Proc. of "35-th IEEE Symposium on the Foundations of Computer Science"*, 1994, pp. 332-346.

[14] C. Lund, D. Spielmann, Personal communication.

[15] D. E. Muller, "Complexity in electronic switching circuits," *IRE Trans. Electr. Comput.* 5, 1956, pp. 15-19.

[16] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," In *Automata Studies*, C. E. Shannon and J. McCarthy Eds., Princeton University Press, Princeton, NJ, 1956, pp. 329-378.

[17] N. Pippenger, "On networks of noisy gates," In *Proc. of "26-th IEEE Symposium on the Foundations of Computer Science"*, 1985, pp. 30-36.

[18] N. Pippenger, Developments in "The Synthesis of Reliable Organisms from Unreliable Components", In *Proc. of "Symposia in Pure Mathematics"*, Vol. 50, 1990, pp. 311-324.

[19] R. Reischuk, B. Schmeltz, "Reliable Computation with Noisy Circuits and Decision Trees – A General n log n Lower Bound," In *Proc. of "32-nd IEEE Symposium on the Foundations of Computer Science"*, 1991, pp. 602-611.

[20] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell Syst. Techn. J.* 28, 1949, pp. 59-98.

[21] G. Tardos, Personal communication.

[22] D. Uhlig, "On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements," *Math. Notes. Acad. Sci. USSR* 15, 1974, pp. 558-562.

[23] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner, 1987.