[Array 14 (2022) 100165](https://doi.org/10.1016/j.array.2022.100165)



Contents lists available at [ScienceDirect](https://www.elsevier.com/locate/array)

Array

journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array)

Characteristics of reversible circuits for error detection

Lukas Burgholzer [a](#_bookmark0),[∗](#_bookmark4), Robert Wille [b](#_bookmark1),[c](#_bookmark2), Richard Kueng [a](#_bookmark0)

a *Institute for Integrated Circuits, Johannes Kepler University Linz, Austria*

b *Chair for Design Automation, Technical University of Munich, Germany*[1](#_bookmark5)

c *Software Competence Center Hagenberg GmbH (SCCH), Austria*

A R T I C L E I N F O A B S T R A C T

*Keywords:*

Emerging technologies Reversible logic

Error detection Simulation Quantum computing

In this work, we consider error detection via simulation for reversible circuit architectures. We rigorously prove that reversibility augments the performance of this simple error detection protocol to a considerable degree. A single randomly generated input is guaranteed to unveil a single reversible error with a probability that only depends on the size of the error, not the size of the circuit itself. Empirical studies confirm that this behavior typically extends to multiple errors as well. In conclusion, reversible circuits offer characteristics that reduce masking effects – a desirable feature that is in stark contrast to irreversible circuit architectures.

# Introduction

neering and computer science. Given a circuit *𝐶*1 with *𝑛* inputs and *𝑚* The detection of errors is a fundamental problem in electrical engi-

circuit realization *𝐶*2 (the *Design Under Verification*) describes the same outputs (the *Golden Specification*), the task is to decide whether a given

functionality on the logical level.

Many approaches exist that address this important and challenging problem. In this work, we focus on error detection protocols that only require simulation runs of the two circuits—as opposed to formal verification techniques which explicitly utilize structural knowledge about both circuits [[1](#_bookmark30)–[6](#_bookmark31)]. This is a severe restriction, but simulations alone are – in principle – sufficient to solve this task. If the two circuits are equivalent, they have the same input–output behavior. Conversely, suppose that they are functionally distinct. Then, there exists at least one input string for which the two circuits produce *distinct* outputs. In formulas:

∃*𝑥⃗* ∈ {0*,* 1}*𝑛* such that *𝐶*1(*𝑥⃗*) ≠ *𝐶*2(*𝑥⃗*)*.* (1) Such an input successfully detects the discrepancy between *𝐶*1 and *𝐶*2

and serves as a counterexample for the equivalence of the circuits.

The problem, however, is how to find counterexamples ([1](#_bookmark3)). If we only allow simulations of both circuits, i.e., we consider them as black boxes, we do not have actionable advice on how to choose promising input strings and we may as well generate inputs uniformly at random:

*𝑥⃗* ∼ Unif ({0*,* 1}*𝑛*), i.e., we flip an unbiased coin for each input value (*𝑥⃗* = (*𝑥𝑛,* … *, 𝑥*1), where *𝑥𝑛,* … *, 𝑥*1 ∼ *𝑥* and Pr [*𝑥* = 0] = Pr[*𝑥* = 1] = 1∕2). Subsequently, we simulate both circuits with this input and check

whether they produce the same output: *𝐶*1(*𝑥⃗*) =? *𝐶*2(*𝑥⃗*). If the outputs

are distinct, we have found a counterexample. The circuits cannot be

equivalent. But if the outputs are the same, the test is inconclusive. In this case, we must repeat it with new (randomly generated) inputs until we either find a counterexample (non-equivalence) or have exhausted

all 2*𝑛* possible inputs (equivalence). The latter, unfortunately, can be a

very real possibility. The two circuits *𝐶*1 and *𝐶*2 may differ on a single

input only and it is extremely unlikely to quickly find this input by (random) chance.

errors very effectively. For *𝑛* = 8, this is illustrated in [Fig.](#_bookmark6) [1](#_bookmark6). A cascade To make matters worse, classical circuits can mask even ‘‘small’’ of logical AND gates, realizing the functionality *𝑦* = *𝑥𝑛* ⋅ … ⋅ *𝑥*1 (ideal circuit *𝐶*1), is affected by a single bit-flip error (erroneous implemen- tation *𝐶*2) in the second layer. It is easy to check that only 4 out of all 28 = 256 input strings can detect this discrepancy.

Masking is a serious issue for error detection using simulation techniques. No malicious intent is required to fool randomly generated inputs. The circuit may do it all by itself. Needless to say, this issue has been well-known for decades. Error detection based on random inputs (alone) often pales in comparison to other more sophisticated techniques. Today’s state of the art is governed by constrained-based stimuli generation techniques [[7](#_bookmark32)–[11](#_bookmark33)], fuzzing [[12](#_bookmark34)], etc. But on the positive side, error detection using randomly-chosen inputs is based on minimal assumptions, namely the possibility to simulate two circuits as black boxes. Moreover, it is intuitive and individual simulation runs are easy and fast to execute.

∗ Corresponding author.

*E-mail addresses:* [lukas.burgholzer@jku.at](mailto:lukas.burgholzer@jku.at) (L. Burgholzer), [robert.wille@tum.de](mailto:robert.wille@tum.de) (R. Wille), [richard.kueng@jku.at](mailto:richard.kueng@jku.at) (R. Kueng).

1 <https://www.cda.cit.tum.de/research/quantum/>.

<https://doi.org/10.1016/j.array.2022.100165>

Received 31 July 2021; Received in revised form 15 February 2022; Accepted 1 April 2022

Available online 12 April 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).





**/ig. 1.** *Error detection in classical circuits is hard:* Suppose that a cascade of logical AND

gates, realizing the Boolean function *𝑦* = *𝑥*8 ⋅ … ⋅ *𝑥*1, is affected by a single bit-flip error

The first two insights are mathematical statements that address single errors only. They readily follow from reversibility and fundamental properties of uniformly random input strings. We refer to Section [3](#_bookmark9) for details and [Fig.](#_bookmark8) [2](#_bookmark8) for illustrative caricatures. When combined, they imply the following confidence bound for detecting single errors with random inputs.

*reversible error of size 𝑘 and fix 𝛿* ∈ (0*,* 1) *(confidence). Then, at most* **Theorem 1.** *Suppose that a general reversible circuit is affected by a single*

*𝑘*−1

⌈ ⌉

(red) in the second layer. Only 4 out of the 28 = 256 possible input strings can detect

this error.







**/ig. 2.** *Illustration of main rigorous contributions:* Simulations with uniformly random inputs completely expose any single reversible error in a given reversible circuit. The two scenarios are *exactly* equivalent (‘‘no masking’’). In the lower scenario, the

probability of correct distinction is governed by the size *𝑘* of the error, not the total

number of lines.

# Summary of results: Error detection in reversible circuits

We have seen that, in general, simulation with (uniformly) ran- dom inputs is not a viable strategy for detecting errors in classical circuits. Already a single ‘‘small’’ error can be exceedingly difficult to detect (masking). Perhaps surprisingly, this dark picture lightens up considerably if we consider *reversible* implementations of logical functionalities. As the name suggests, reversible circuits are circuits

formally, *𝑛*-bit reversible circuits implement permutations on the set of whose action can be undone by running the circuit backwards. More all 2*𝑛* bit strings. This, in particular, implies that the number of input

log(1∕*𝛿*)2 *randomly selected inputs suffice to witness this error with probability (at least)* 1 − *𝛿.*

For *𝑘* = 1 – a single bit-flip error (NOT) *anywhere* within the circuit

– this statement can be further improved (see [Theorem](#_bookmark19) [2](#_bookmark19)) and actually

becomes deterministic: already a single (random) input is guaranteed to detect this error with certainty. We emphasize that this statement is true irrespective of the number of lines and the circuit’s size. It is simply impossible to hide a single bit-flip inside a reversible circuit. Such a

There it can routinely happen that order 2*𝑛* random inputs are necessary behavior is strikingly different from irreversible circuit architectures.

to detect even a single bit-flip error, see e.g. [Fig.](#_bookmark6) [1](#_bookmark6).

The multiple-error case is much more intricate, because error lo- cations and circuit structure start to matter. This leads to drastically different behaviors of best case (independent errors) and worst case (severe masking) behavior. To better understand the typical behavior of multiple errors, we resort to numerical simulations. These indicate a (close-to) best-case behavior: the probability of failing to detect a

total of *𝑙* reversible errors is exponentially suppressed in *𝑙*, see [Fig.](#_bookmark11) [3](#_bookmark11).

Additional simulation results and details are provided in Section [4](#_bookmark20).

Note that a similar line of thought has recently been presented for the domain of quantum computing (which bears many similarities to reversible circuits). More precisely, a verification scheme heavily based on simulation has been proposed in [[17](#_bookmark38)] and refined in [[18](#_bookmark39)]. A similar theoretical result has been presented in [[19](#_bookmark40)].

# Rigorous theory for single errors

* 1. *Reversible circuits and error model*

We will work in the reversible circuit model for *𝑛* input bits (and *𝑛*

to deduce powerful consequences. An *𝑛*-bit reversible circuit imple- output bits). A high-level of mathematical abstraction already suffices ments a permutation *𝑅* ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* of all 2*𝑛* bit strings. Reversing

mutation *𝑅𝑇* ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* that undoes the original circuit: the circuit, that is running it backwards, produces the unique per-

and output bits must be the same (*𝑛* = *𝑚*). Despite these restrictions,

*𝑅𝑇* ◦*𝑅* = *𝑅*◦*𝑅𝑇* = id, where id(*𝑥⃗*) =

*𝑥⃗* for all *𝑥⃗*

∈ {0*,* 1}*𝑛* is the

reversible circuits are universal, i.e., *any* logical function on *𝑛* bits

can be implemented by a reversible circuit [[13](#_bookmark35)] and efficient mapping techniques are readily available [[14](#_bookmark36)–[16](#_bookmark37)] (this implementation may

require strictly more than *𝑛* bits, though). Negation (NOT), exclusive or

(CNOT) and the Toffoli gate (CCNOT) are examples of simple reversible functionalities. Viewed as a logic gate, CCNOT is also universal. Every reversible circuit can be constructed from Toffoli gates alone [[13](#_bookmark35)].

To summarize, reversible circuits bear strong similarities with clas- sical (irreversible) circuits, but there are some notable additional char- acteristics. Chief among them is reversibility itself which implies that information cannot easily escape. Here, we show that this has profound implications for error detection with random inputs. More precisely,

* + 1. reversible circuits can never mask single reversible errors (rigor- ous result, see [Proposition](#_bookmark14) [1](#_bookmark14))
    2. the probability of detecting a single reversible error only de- pends on its size, i.e., on the number of bits it affects, not the total number of bits (unsurprising rigorous result, see [Lemma](#_bookmark16) [2](#_bookmark16))
    3. multiple reversible errors are typically even easier to detect (empirical studies, see [Fig.](#_bookmark11) [3](#_bookmark11) and discussions in Section [4](#_bookmark20))

identity permutation (‘‘do nothing’’). This defining feature suffices to deduce three elementary properties that will form the basis of our proof strategy.

*circuits 𝑅*1*, 𝑅*2*, 𝑅*3 ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛 and an 𝑛-bit string 𝑥⃗* ∈ {0*,* 1}*𝑛. Then,* **Lemma 1** (*Characteristics of Reversible Circuits*)**.** *Consider reversible*

1. *output equivalence is unaffected by composition:*

*𝑅*1(*𝑥⃗*) = *𝑅*2(*𝑥⃗*) ⇔ (*𝑅*3◦*𝑅*1)(*𝑥⃗*) = (*𝑅*3◦*𝑅*2)(*𝑥⃗*)

1. *invariance of the uniform distribution:*

*𝑥⃗* ∼ Unif({0*,* 1}*𝑛*) *implies 𝑅*1(*𝑥⃗*) ∼ Unif({0*,* 1}*𝑛*)

1. *non-trivial action: suppose 𝑅*1 ≠ id*. Then, there are at least two bit strings such that 𝑅*1(*𝑥⃗*) ≠ *𝑥⃗.*

permutations on the set of all 2*𝑛* bit strings. **Proof.** All proofs utilize the fact that reversible circuits act like

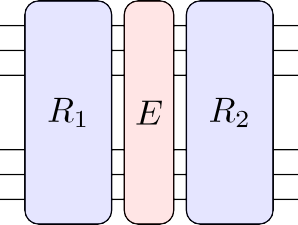
serve equivalence: *𝑦* = *𝑦*′ if and only if *𝑅*(*𝑦*) = *𝑅*(*𝑦*′) for any (i) Permutations are *invertible* transformations. As such, they pre- reversible circuit *𝑅*. The claim follows from setting *𝑦* = *𝑅*1(*𝑥⃗*),

*𝑦*′ = *𝑅*2(*𝑥⃗*) and *𝑅* = *𝑅*3.



**/ig. 3.** *Typical accumulation effects for multiple errors (log–log plot):* number *𝑙* of randomly injected reversible errors (*𝑥*-axis) vs. average number of random inputs required to detect erroneous behavior (*𝑦*-axis) in a generic *𝑛* = 20-bit reversible circuit with 4000 gates. Different colors denote worst-case errors of increasing size *𝑘*. Solid lines track the theoretical best-case behavior (independent errors, see Eq. ([5](#_bookmark27)) below). For small *𝑙*, the plot highlights an excellent agreement between typical (diamonds) and best-case (solid lines) behavior.

**/ig. 4.** *(Single) error model and compatible circuit decomposition:* An ideal reversible



a decomposition of ideal and corrupted circuit into matching constituents: *𝑅* = *𝑅*2◦*𝑅*1 circuit (blue) is corrupted by a single reversible error (red). The error location begets (ideal) and *𝑅̃* = *𝑅*2 ◦*𝐸*◦*𝑅*1 (corrupted).

1. The uniform distribution over *𝑛*-bit strings assigns the same weight to each of the 2*𝑛* bit strings. Permuting the bit strings can-

not affect the weights and, by extension, the uniform distribution itself.

1. The number of invariant bit strings (*𝑥⃗* ∈ {0*,* 1}*𝑛* ∶ *𝑅*1(*𝑥⃗*) = *𝑥⃗*) is

A non-trivial permutation of 2*𝑛* elements can have at most 2*𝑛* − 2 equal to the number of fix points of the underlying permutation.

fix points (transposition). □

Different reversible circuits of compatible bit-size *𝑛* can be combined

to yield another (larger) circuit: (*𝑅* ◦*𝑅* )(*𝑥⃗*) = *𝑅* (*𝑅* (*𝑥⃗*)) for input *𝑥⃗* ∈

(‘‘composition’’). The reverse direction is also possible (‘‘decom-

(iii) *𝑅*2 ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* describes the original functionality from

the error location onwards (‘‘future’’).

In summary,

*𝑅̃* = *𝑅*2◦*𝐸*◦*𝑅*1*,* while *𝑅* = *𝑅*2◦*𝑅*1*,* (2)

and we refer to [Fig.](#_bookmark12) [4](#_bookmark12) for a visual illustration.

* 1. *No masking for random inputs*

We now have all building blocks in place to present and derive the main conceptual result of this work. It addresses the probability of detecting single reversible errors in arbitrary reversible circuits ([2](#_bookmark13))

based on a single random input *𝑥⃗* ∼ Unif({0*,* 1}*𝑛*).

**Proposition 1** (*No Masking*)**.** *Fix 𝑅* = *𝑅*2◦*𝑅*1 *(ideal circuit) and 𝑅̃* =

*𝑅*2◦*𝐸*◦*𝑅*1 *(single, reversible error). Then, the probability of detecting this discrepancy with a random input 𝑥⃗* ∼ Unif({0*,* 1}*𝑛*) *only depends on the error 𝐸, not the actual circuit. More precisely,*

Pr [*𝑅̃*(*𝑥⃗*) ≠ *𝑅*(*𝑥⃗*)] = Pr [*𝐸*(*𝑥⃗*) ≠ *𝑥⃗*] *,*

*all* 2*𝑛 possible input strings. where the probability is taken with respect to the uniform distribution over*

**Proof.** This statement is an immediate consequence of two elementary characteristics of reversible circuit architectures. Apply [Lemma](#_bookmark10) [1](#_bookmark10)(i) to

{0*,* 1 *𝑛*

}

2 1 2 1

remove the effect of *𝑅*2,

position’’) and, arguably, more interesting. Circuit diagrams provide a well-established tool that does precisely that. They decompose a possi- bly complicated circuit into a structured sequence of simpler building blocks. We use circuit decomposition on a rather high level to reason

about *single* reversible errors in reversible circuits. Suppose that an *𝑛*-bit

reversible circuit *𝑅* is affected by a reversible error *𝐸* that produces a functionally different circuit *𝑅̃*. Then, the location of this error within

the circuit suggests a compatible decomposition into three parts:

* + 1. *𝑅*1 ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* describes the original functionality up to the location where the error occurs (‘‘past’’),
    2. *𝐸* ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* captures the error as an additional circuit layer on all *𝑛* bits (‘‘present’’),

Pr [*𝑅̃*(*𝑥⃗*) = *𝑅*(*𝑥⃗*)] = [*𝑅*2◦*𝐸*◦*𝑅*1(*𝑥⃗*) = *𝑅*2◦*𝑅*1(*𝑥⃗*)]

= Pr [*𝐸* (*𝑅*1(*𝑥⃗*)) = *𝑅*1(*𝑥⃗*)] *,*

and note that, according to [Lemma](#_bookmark10) [1](#_bookmark10) (ii), *𝑥⃗* ∼ Unif({0*,* 1}*𝑛*) implies

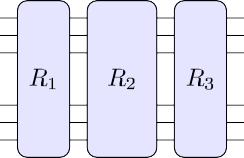
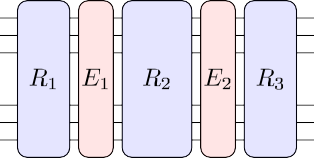
*𝑅*1(*𝑥⃗*) ∼ Unif({0*,* 1}*𝑛*). □

Although simple to prove, [Proposition](#_bookmark14) [1](#_bookmark14) pinpoints remarkable dif- ferences between reversible and irreversible circuits. As illustrated in [Fig.](#_bookmark8) [2](#_bookmark8), the former cannot hide errors from randomly sampled inputs (‘‘no masking’’).

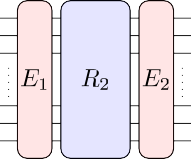
We emphasize that a uniformly random selection of input strings

is enough to ignore the final portion of the circuit *𝑅*2 (after the error is crucial to arrive at such a powerful conclusion. Reversibility alone

to (non-)equal bit strings. In contrast, the first portion of the circuit *𝑅*1 has occurred). Reversible circuits always map (non-)equal bit strings (before the error has occurred) can affect concrete inputs *𝑥⃗* ∈ {0*,* 1}*𝑛*. But if *𝑥⃗* is sampled randomly, then *𝑅*1(*𝑥⃗*) will be a different, but still



that it is invariant under reversible transformations. The circuit *𝑅*1 random, bit string. The uniform distribution is special in the sense

may affect every concrete input, but it does not affect the underlying distribution.



* 1. *Only error size matters*

We have seen that uniformly random inputs can uncover single

[reversible](#_bookmark14) errors in a general reversible circuit. According to [Proposi-](#_bookmark14) [tion](#_bookmark14) [1](#_bookmark14), the probability of witnessing a discrepancy only depends on the error, not the underlying circuit structure.

We say that an error *𝐸* ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛* has *size 𝑘* if it only

affects *𝑘* bits in a nontrivial fashion. The remaining *𝑛* − *𝑘* bits are not

touched at all. We refer to [Fig.](#_bookmark8) [2](#_bookmark8) for a visual illustration of this summary

to detect than ‘‘small’’ ones and that the number of lines *𝑛* plays an parameter. Intuitively, we would expect that ‘‘large’’ errors are easier

active role. However, the following simple statement shows that the probability of detecting an error in the worst case is exponentially

suppressed with respect to the error size *𝑘*, but is independent of the

actual number of bits *𝑛*.

**Lemma 2** (*Only Error Size Matters*)**.** *Suppose that 𝐸* ∶ {0*,* 1}*𝑛* → {0*,* 1}*𝑛*

*is a reversible error that only affects 𝑘 bits in a non-trivial fashion and*

*𝑥⃗* ∼ Unif({0*,* 1}*𝑛*) *is sampled from the uniform distribution. Then,*

Pr [*𝐸*(*𝑥⃗*) ≠ *𝑥⃗*] ≥ 2−(*𝑘*−1)*.*

**Proof.** Suppose, without loss of generality, that the error *𝐸* only affects

the least-significant *𝑘* bits, i.e., *𝐸*(*𝑥⃗*) = *𝐸*(*𝑥𝑛,* … *, 𝑥*1) = (*𝑥𝑛,* … *, 𝑥𝑘*+1*, 𝑦𝑘,*

exposes multiple errors only partially. Everything before the first error (*𝑅*1) and after **/ig. 5.** *Partial simplification for multiple errors:* Simulation with uniformly random inputs the last error (*𝑅*3) can be safely ignored, but the part in between (*𝑅*2) does matter.

Different circuit structures can lead to strikingly different error detection probabilities.

setting *𝑁* = log(1∕*𝛿*)2*𝑘*−1 provides a concrete number of repetitions [Theorem](#_bookmark7) [1](#_bookmark7) above is a streamlined consequence of this observation:

⌈ ⌉

that ensures that we detect the discrepancy with probability (at least)

1 − *𝛿*.

**Proof of** [**Theorem**](#_bookmark19)[**2**](#_bookmark19)**.** For *𝑁* = 1 (one random input), the claim readily follows from combining [Proposition](#_bookmark14) [1](#_bookmark14) and [Lemma](#_bookmark16) [2](#_bookmark16) (more precisely,

their contrapositions):

Pr [*𝑅̃*(*𝑥⃗*1) = *𝑅*(*𝑥⃗*1)] = Pr [*𝐸*(*𝑥⃗*1) = *𝑥⃗*1] ≤ 1 − 2−(*𝑘*−1)*.*

This bound readily extends to the general *𝑁* -case by using the as- sumption that the individual input strings *𝑥⃗*1*,* … *, 𝑥⃗𝑁* are all sampled

… *, 𝑦*1), where (*𝑦𝑘,* … *, 𝑦*1) =

*̃ 𝑘*

*𝐸̃*(*𝑥𝑘,* … *, 𝑥*1). Since *𝐸* is reversible, its

*𝑘*

independently. Joint probabilities of independent events factorize and

we conclude

restriction *𝐸* ∶ {0*,* 1}

→ {0*,* 1}

to the *𝑘* relevant bits must also be

*𝑁*

reversible. Moreover, *𝐸̃* ≠ id, because *𝐸* is non-trivial. [Lemma](#_bookmark10) [1](#_bookmark10) (iii)

Pr[ ⋀

{*𝑅̃*(*𝑥⃗* ) = *𝑅*(*𝑥⃗* )}] = ∏ Pr [*𝑅̃*(*𝑥⃗* ) = *𝑅*(*𝑥⃗* )]

then implies that there must be at least 2 bit strings of size *𝑘* that 1

are affected by *𝐸̃*. Finally, we use the fact that *𝑥⃗* = (*𝑥 ,* … *, 𝑥* ) ∼

Unif({0 1}*𝑛*)

*𝑛*

1

implies that the least-significant *𝑘* bits are also distributed

*,*

≤*𝑖*≤*𝑁*

*𝑖 𝑖*

*𝑖 𝑖*

*𝑖*=1

)

≤ (1 − 2−(*𝑘*−1) *𝑁 .* (3)

uniformly: (*𝑥𝑘,* … *, 𝑥*1) ∼ Unif({0*,* 1}*𝑘*). Therefore,

Apply 1+*𝑥 <* exp(*𝑥*) for all *𝑥* ∈ R (convexity of the exponential function)

−(*𝑘*−1)

Pr [*𝐸*(*𝑥⃗*) ≠ *𝑥⃗*] = Pr [*𝐸̃*(*𝑥 ,* … *, 𝑥* ) ≠ (*𝑥 ,* … *, 𝑥* )] ≥ 2 *.* □

with *𝑥* = −2

to complete the argument. □

*𝑘* 1 *𝑘* 1 2*𝑘*

rors of size *𝑘* permute exactly 2 out of the 2*𝑘* possible *𝑘*-bit inputs on This probability bound is actually sharp. Worst-case reversible er- which they act. Concrete examples of such a behavior are NOT (*𝑘* = 1), CNOT (*𝑘* = 2), CCNOT (*𝑘* = 3) and, more generally, a (*𝑘* − 1)-fold controlled NOT gate on *𝑘* bits (general *𝑘*). The numerical simulations

shown in [Fig.](#_bookmark11) [3](#_bookmark11) are based on injecting such worst-case errors at random circuit locations.

* 1. *General confidence bound for detecting single reversible errors*

We now have all necessary ingredients to establish a rigorous per- formance guarantee for reversible error detection with (uniformly) random inputs. The following statement bounds the number of uni- formly random inputs that may be required to detect a single reversible

error of size *𝑘*.

**Theorem 2.** *Fix 𝑅* = *𝑅*2◦*𝑅*1 *(ideal circuit), 𝑅̃* = *𝑅*2◦*𝐸*◦*𝑅*1 *(sin- gle, reversible error) and 𝐸 has size 𝑘. Suppose that 𝑥⃗*1*,* … *, 𝑥⃗𝑁 are 𝑁*

*(independent) uniformly random inputs. Then,*

equality 1+*𝑥* ≤ exp(*𝑥*) is never tight). As such, it always under-estimates The bound provided in [Theorem](#_bookmark19) [2](#_bookmark19) is simple, but not sharp (the in-

small error sizes *𝑘*. The extreme case is a single NOT error (*𝑘* = 1). For the actual confidence level. This discrepancy is most pronounced for

*𝑘* = 1, the bound in Eq. ([3](#_bookmark18)) becomes (exactly) zero. By contraposition,

*every possible input bit string is guaranteed to detect a single bit-flip error*

*that is hidden anywhere within the circuit.*

# Empirical analysis for multiple errors

In the previous section, we have established strong theoretical support for detecting single reversible errors. At its heart has been the

decomposition *𝑅̃* = *𝑅*2◦*𝐸*◦*𝑅*1 illustrated in [Fig.](#_bookmark12) [4](#_bookmark12). Reversibility and

the circuit portions *𝑅*2 and *𝑅*1 completely. In turn, we were able to uniformly random inputs have subsequently allowed us to discuss away

focus exclusively on the error itself.

For more than one error, this is in general not an option anymore. While we can safely ignore circuit contributions before the first and after the last error, the circuit in between cannot be ignored, see [Fig.](#_bookmark17) [5](#_bookmark17).

Pr[ ⋀

1≤*𝑖*≤*𝑁*

{*𝑅̃*(*𝑥⃗𝑖*) =

*𝑅*(*𝑥⃗𝑖*

)}] ≤

exp (−*𝑁*

∕2*𝑘*−1)

The relation between errors and intermediate circuit parts governs how likely it is to witness the overall error.

In this section, we analyze error accumulation effects in generic

*suppressed in the number 𝑁 of random test inputs. In words, the probability of failing to detect a single error is exponentially*

reversible circuits. To obtain guiding intuition, we will first isolate and discuss the two extreme cases. Independent errors (best case, see



**/ig. 6.** *Best-case scenario for two errors:* One of the errors, say *𝐸*2, commutes with the relevant circuit part *𝑅*2. Reordering allows us to treat the two errors as a single effective error *𝐸̃* = *𝐸*1 ◦*𝐸*2 . In addition, *𝐸*1 and *𝐸*2 affect disjoint bit collections (independence) and *𝐸̃* factorizes nicely into two disjoint components: Pr[*𝑅̃*(*𝑥⃗*) ≠ *𝑅*(*𝑥⃗*)] = Pr[*𝐸̃*(*𝑥⃗*) ≠ *𝑥⃗*] ≥ 1 − (1 − 2−(*𝑘*−1))2 (quadratic improvement).

Section [4.1](#_bookmark23)) and maximal masking (worst case, see Section [4.2](#_bookmark24)) turn out to behave in a radically different fashion. Subsequent numerical studies demonstrate that typical error accumulation effects closely follow the best-case trajectory: Multiple errors are typically *much* easier to detect than a single error.

**/ig. 7.** *Worst-case scenario for two errors:* Two bit-flip errors (*𝑘* = 1) affect one control line of a (*𝑛*−1)-fold controlled NOT-gate. These errors do not commute with the relevant circuit part *𝑅*2. Quite the opposite: two errors with size *𝑘* = 1 produce an effective error



*𝐸̃* of size *𝑘* = (*𝑛* − 1). To make matters even worse, such a (*𝑛* − 2)-fold controlled NOT

error is extremely difficult to detect: Pr[*𝑅̃*(*𝑥⃗*) ≠ *𝑅*(*𝑥⃗*)] = Pr[*𝐸̃*(*𝑥⃗*) ≠ *𝑥⃗*] = 4∕2*𝑛* (masking).

For small *𝑙* (in comparison to 2(*𝑘*−1)), the claim is comparable to We conclude this section with a simplified interpretation of Rel ([5](#_bookmark27)).

*𝑁*

* 1. *Best-case behavior: Commuting and independent errors*

(↓) expect

≈ 2*𝑘*−1∕*𝑙*, which can also be observed in [Fig.](#_bookmark11) [3](#_bookmark11): the slopes of

Let us first discuss *𝑙* = 2 reversible errors of size *𝑘*. An extension to multiple errors (*𝑙* ≥ 3) and different sizes will be straightforward. [Fig.](#_bookmark21) [6](#_bookmark21)

that one of the errors, say *𝐸*2, can be pulled through the central circuit provides valuable guidance for potential best-case behavior. Suppose part *𝑅*2 without affecting it: *𝐸*2◦*𝑅*2 = *𝑅*2◦*𝐸*2. If circuit and error

commute in such a fashion, we can group both errors into a single layer and have effectively reduced the problem to the single-error case which we already understand:

*𝑅̃* = *𝑅*3◦*𝐸*2◦*𝑅*2◦*𝐸*1◦*𝑅*1 = (*𝑅*3◦*𝑅*2)◦(*𝐸*2◦*𝐸*1)◦*𝑅*1*.*

detect the cumulative error *𝐸*2◦*𝐸*1 with a single random input? This The only remaining question is: what is the probability of failing to

the sense that they act on disjoint sets of *𝑘* bits each. A uniformly failure probability is smallest if the two errors are *independent* in random input *𝑥⃗* ∈ Unif({0*,* 1}*𝑛*) then ensures that the failure probability

of errors *𝑙* is small compared to 2(*𝑘*−1). Under best-case assumptions, the solid lines match this estimate rather well whenever the number detecting *𝑙* size *𝑘*-errors is *𝑙*-times easier than detecting a single error

of the same size.

* 1. *Worst-case: anti-commuting errors and masking*

We expect that worst case error accumulation should occur when errors and relevant circuit portion do not commute at all (‘‘anti- commutation’’). If this is the case, the probability of detecting errors can become exponentially small in the total number of bits. We illus-

trate this by means of an example that is illustrated in [Fig.](#_bookmark22) [7](#_bookmark22): *𝐸*1 and

*𝐸*2 are bit-flip errors (*𝑘* = 1) that affect the first bit while *𝑅*2 ∶ {0*,* 1}*𝑛* →

{0*,* 1}*𝑛* is a (*𝑛* − 1)-fold controlled NOT-gate. It is easy to check that

*𝐸*2◦*𝑅*2◦*𝐸*1 = *𝐸̃*◦*𝑅*2*,*

factorizes:

Pr [(

)( ) =

] = ∏

Pr [

( ) = ]

(1 − 2−( −1))2

where *𝐸̃* is a (*𝑛*−2)-fold controlled NOT gate that acts on all bits, except

the very first one (*𝑘* = *𝑛* − 1). This is a single worst-case error of almost

*𝐸*2◦*𝐸*1

*𝑥⃗*

*𝑥⃗*

*𝑖*=1

*𝐸𝑖 𝑥⃗ 𝑥⃗* ≤

*𝑘 .*

maximal size. [Proposition](#_bookmark14) [1](#_bookmark14) and [Lemma](#_bookmark16) [2](#_bookmark16) assert

This argument readily extends to multiple errors (*𝑙* ≥ 3). Taking the

2

complement ensures

Pr [*𝑅̃*(*𝑥⃗*) ≠ *𝑅*(*𝑥⃗*)] =1 − Pr [*𝐸𝑙* ◦ ⋯ ◦*𝐸*1(*𝑥⃗*) = *𝑥⃗*]

Pr[*𝑅̃*(*𝑥⃗*) ≠ *𝑅*(*𝑥⃗*)] = Pr [*𝐸̃*(*𝑥⃗*) ≠ *𝑥⃗*] = 4 *.*

This success probability is exponentially small in the total number of bits and we expect to require a total of

2*𝑛*

*𝑁*

≥1 − (1 − 2−(*𝑘*−1))*𝑙 ,* (4)

(↑) expect

≥ 2(*𝑛*−2)(worst case) (6)

provided that all *𝑙* errors commute with the circuit (first equality) and act on different subsets of *𝑘* bits each (second inequality). Rel. ([4](#_bookmark25))

substantially with the number of errors *𝑙*. Intuitively, this makes sense: highlights that the probability of (best case) error detection increases

for the number *𝑁* of random inputs that are required to detect *𝑙* best- more errors should be easier to detect. This insight has implications case errors of size *𝑘* each. To pinpoint them, it is instructive to view

with probability *𝑝* = Pr[*𝑅̃*2(*𝑥⃗*) ≠ *𝑅*2(*𝑥⃗*)] (‘‘heads’’) and fail to detect it a single simulation run as a biased coin toss: we detect a discrepancy with probability 1 − *𝑝* = Pr *𝑅̃*2(*𝑥⃗*) = *𝑅*2(*𝑥⃗*) (‘‘tails’’). When attempting

[ ]

to detect a discrepancy, we input new randomly generated inputs until

we find a mismatch. This is equivalent to tossing the biased coin

achieve this goal is 1∕*𝑝* (geometric distribution). Together with Rel. ([4](#_bookmark25)), until ‘‘heads’’ appears. The expected number of required coin tosses to

this analogy allows us to conclude that we expect to require

accumulation effects can occur for more errors (*𝑙* ≥ 3) and/or larger random inputs in order to detect the discrepancy. Even worse error error sizes (*𝑘* ≥ 2). But already Rel. ([6](#_bookmark26)) is almost as bad as it can be. It is only a factor of two away from 2*𝑛*−1—the absolute worst case for

distinguishing *any* pair of reversible circuits, see [Lemma](#_bookmark10) [1](#_bookmark10) (iii).

* 1. *Empirical studies*

The multiple-error case is intricate by comparison, because the interplay between error (locations) and underlying circuit geometry starts to matter. We have seen that this leads to strikingly different best- (commuting errors, Sub. [4.1](#_bookmark23)) and worst-case (anticommuting errors, Sub. [4.2](#_bookmark24)) behavior. Concrete problem instances fall into the wide range between these extreme cases. In this section, we employ numerics to

(↓) expect

*𝑁*

≤ 1

1 − (1 − 2−(*𝑘*−1))*𝑙*

(best case) (5)

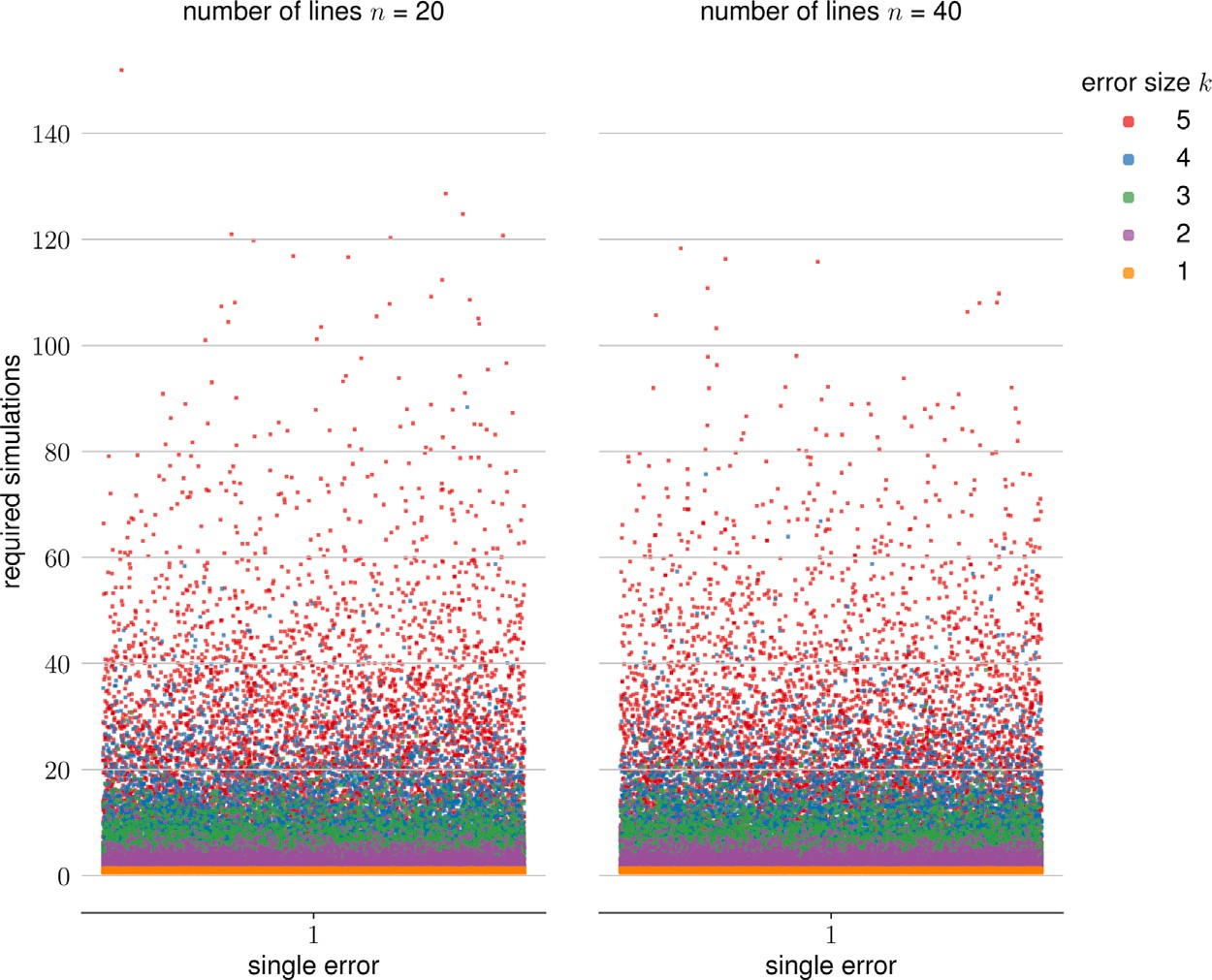
delineate *typical* behavior.

We study the effect of size-*𝑘* reversible errors in reversible circuits with *𝑛* lines. For a given number of lines *𝑛*, we construct random

random inputs to detect *𝑙* commuting and independent errors of size *𝑘* each. This bound is sharp. It holds with equality if each of the *𝑙* errors is a worst-case error of size *𝑘*, e.g. a (*𝑘* − 1)-fold controlled NOT gate.

reversible circuits with *𝑔* ≈ U(*𝑛*2) arbitrary multi-controlled NOT gates. When injecting errors of size *𝑘*, we always consider (*𝑘* − 1)-

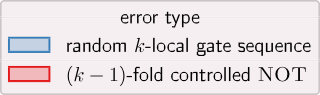
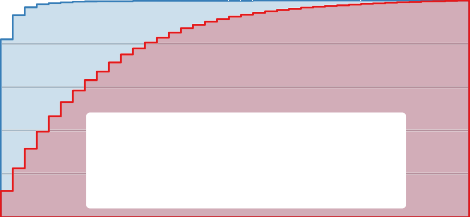
fold controlled NOT gates which represent the worst case behavior,



**/ig. 8.** *Confirmation of theoretical results*: Scatter-plot of required simulations (*𝑦*-axis) for detecting a single reversible error of size *𝑘* in a circuit with *𝑛* = 20 (left plot) and *𝑛* = 40

(right plot) lines. Different colors denote varying values of *𝑘* ∈ {1*,* 2*,* 3*,* 4*,* 5} . This experimentally confirms that the distribution of simulations does not depend on the number of

lines and that the number of required simulations grows exponentially with the size of the error.







**/ig. 9.** *Comparison of worst-case and average-case errors* : performed simulations (*𝑥*-axis) vs. cumulative distribution function (cdf) for detecting *𝑙* = 1*,* 2*,* 4*,* 6 reversible errors of size

*𝑘* = 5 (*𝑦*-axis). The red curve corresponds to injecting worst-case errors, while the blue curve delineates the cdf for detecting *randomly generated* errors of the same size. This goes



to show, that average-case errors require far less simulations than worst-case ones.

as discussed in Section [3.3](#_bookmark15). Without loss, we assume that these errors

experiments were repeated 10 000 times with different random seeds are geometrically local, i.e., they only affect neighboring lines. All

in order to ensure adequate statistical uniformity.

smaller the error, the greater its impact. This is in excellent agreement

the predicted 2*𝑘*−1 trajectory with no apparent variation. Additionally, with [Theorem](#_bookmark19) [2](#_bookmark19). On average, the required simulations exactly follow

the distributions of results is the same when simulating the circuits

First and foremost, we confirm interesting aspects of the theory developed in Section [3](#_bookmark9). To this end, we considered the injection of a

*𝑅* = *𝑅*2◦*𝑅*1 and *𝑅̃*

error *𝐸* itself.

= *𝑅*2◦*𝐸*◦*𝑅*1 as compared to only simulating the

single size-*𝑘*-error and count the required number of simulations for

detecting this error. The results are depicted in [Fig.](#_bookmark28) [8](#_bookmark28). In contrast to

of size *𝑘* is (1) completely independent of the circuit under consid- classical intuition, the probability of detecting a single reversible error eration, and (2) diminishes exponentially in the error size *𝑘*, i.e., the

The next set of numerical experiments pilots us in more interesting territory. Namely, the multiple-error case. We have already teased the results in the introduction and summarized them in [Fig.](#_bookmark11) [3](#_bookmark11). The aver- aged number of inputs highlights an excellent agreement between the observed behavior and the best-case scenario discussed in Section [4.1](#_bookmark23).

The deviation from this optimum for higher numbers of errors can be explained by accumulation affects of errors not acting independently (see Section [4.2](#_bookmark24)).

Last but not least, we emphasize that – up to this point – theoretical and empirical results have been contingent on a worst-case assumption:

each injected size-*𝑘* error is a (*𝑘* − 1)-fold controlled NOT-gate. In a

final series of evaluations, we analyzed the success probability after

*random*. More precisely, each size-*𝑘* error is a randomly selected gate conducting a certain number of simulations when choosing errors *at* sequence with the additional constraint that none of the *𝑘* relevant lines

most) (*𝑘*−1)). We expect that this error model captures typical behavior remain unaffected (such a scenario would produce an error of size (at

in a more accurate fashion. The results are shown in [Fig.](#_bookmark29) [9](#_bookmark29) and highlight a considerable discrepancy between random (blue) and worst-case (red)

errors. This is not at all surprising. Random errors of size *𝑘* tend to

factorize into several independent contributions and the probabilities of

detecting them with random inputs factorizes accordingly, see Sub. [4.1](#_bookmark23). Such factorizations lead to an increased error detection probability within (very) few simulation runs.

# Conclusion

In this work, we have shown the impact of the reversible circuit paradigm on the probability of detecting errors in circuits. Our rigor- ous analysis shows, that, as opposed to classical/irreversible circuits, reversible circuits can never mask single errors and, that the probability of detecting a single reversible error only depends on the error’s size and not at all on the surrounding circuit. Empirical evaluations have shown that, in case of multiple errors, the detection probability is very close to the theoretical best case. Finally, we have observed that, in case the assumption of worst-case errors is dropped, the probability of detecting these errors is increased even more.

# CRediT authorship contribution statement

**Lukas Burgholzer:** Investigation, Software, Validation, Writing – original draft, Visualization. **Robert Wille:** Conceptualization, Funding acquisition, Writing – review & editing, Supervision. **Richard Kueng:** Methodology, Formal analysis, Writing – original draft, Supervision, Project administration.

# Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

# Acknowledgments

The authors want to thank J. Küng for inspiring discussions through- out the early stages of this project and W. Schreiner for further valuable feedback.

This work received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and in-

novation program (grant agreement No. 101001318), was part of the

Munich Quantum Valley, which is supported by the Bavarian state

government with funds from the Hightech Agenda Bayern Plus, and has been supported by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

# References

1. [Disch S, Scholl C. Combinational equivalence checking using incremental](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb1) [SAT solving, output ordering, and resets. In: Asia and South Pacific Design](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb1) [Automation Conference. 2007, p. 938–43.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb1)
2. [Marques-Silva Ja, Glass T. Combinational equivalence checking using satisfi-](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb2) [ability and recursive learning. In: Design, Automation and Test in Europe.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb2) [1999.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb2)
3. [Molitor P, Mohnke J. Equivalence checking of digital circuits: fundamentals,](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb3) [principles, methods. Springer; 2010.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb3)
4. [Jha S, Lu Y, Minea M, Clarke EM. Equivalence checking using abstract BDDs.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb4) [In: Int’l Conference on Comp. Design. 1997.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb4)
5. [Clarke EM, Grumberg O, Kroening D, Peled DA, Veith H. Model checking. MIT](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb5) [Press; 2018.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb5)
6. [Biere A, Cimatti A, Clarke EM, Zhu Y. Symbolic model checking without BDDs.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb6) [In: Tools and algorithms for the construction and analysis of systems. 1999, p.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb6) [193–207.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb6)
7. [Yuan J, Pixley C, Aziz A. Constraint-based verification. Springer; 2006.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb7)
8. [Biere A, Kunz W. SAT and ATPG: boolean engines for formal hardware](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb8) [verification. In: Int’l Conference on CAD. 2002, p. 782–5.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb8)
9. [Wille R, Große D, Haedicke F, Drechsler R. SMT-based stimuli generation in the](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb9) [systemc verification library. In: Forum on specification and design languages.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb9) [2009.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb9)
10. [Kitchen N, Kuehlmann A. Stimulus generation for constrained random simulation.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb10) [In: Int’l Conference on CAD. 2007, p. 258–65.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb10)
11. [Gent K, Hsiao MS. Fast multi-level test generation at the RTL. In: IEEE annual](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb11) [symposium on VLSI. 2016, p. 553–8.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb11)
12. [Laeufer K, Koenig J, Kim D, Bachrach J, Sen K. RFUZZ: coverage-directed fuzz](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb12) [testing of RTL on FPGAs. In: Int’l Conference on CAD. 2018.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb12)
13. [Toffoli T. Reversible computing. In: Automata, languages and programming, vol.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb13)

[85. Springer; 1980, p. 632–44.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb13)

1. [Zulehner A, Wille R. Make it reversible: efficient embedding of non-reversible](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb14) [functions. In: Design, automation and test in Europe. 2017.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb14)
2. [Maslov D, Dueck GW. Reversible cascades with minimal garbage. IEEE](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb15) [Transactions on CAD of Integrated Circuits and Systems 2004;23(11):1497–509.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb15)
3. [Zilic Z, Radecka K, Kazamiphur A. Reversible circuit technology mapping from](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb16) [non-reversible specifications. In: Design, Automation and Test in Europe. 2007.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb16)
4. [Burgholzer L, Wille R. The power of simulation for equivalence checking in](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb17) [quantum computing. In: Design Automation Conference. 2020.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb17)
5. [Burgholzer L, Kueng R, Wille R. Random stimuli generation for the verification](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb18) [of quantum circuits. In: Asia and South Pacific Design Automation Conference.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb18) [2021.](http://refhub.elsevier.com/S2590-0056(22)00028-5/sb18)
6. Linden N, Wolf Rd. Lightweight detection of a small number of large errors in a quantum circuit, [arXiv:2009.08840](http://arxiv.org/abs/2009.08840), 2020.