Automatic Design of Low-Power Encoders Using Reversible Circuit Synthesis

Robert Wille[1] Rolf Drechsler[1,2] Christof Osewold[3] Alberto Garcia-Ortiz[3,4]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
[3]ITEM, University of Bremen, 28359 Bremen, Germany
[4]TZI, University of Bremen, 28359 Bremen, Germany
{rwille,drechsle}@informatik.uni-bremen.de {osewold,agarcia}@item.uni-bremen.de

***Abstract*—The application of coding strategies is an established methodology to improve the characteristics of on-chip interconnect architectures. Therefore, design methods are required which realize the corresponding encoders and decoders with as small as possible overhead in terms of power and delay. In the past, conventional design methods have been applied for this purpose.**

This work proposes an entirely new direction which exploits design methods for reversible circuits. Here, much progress has been made in the last years. The resulting reversible circuits represent one-to-one mappings which can inherently work as logical descriptions for the desired encoders and decoders. Both, an exact and a heuristic synthesis approach, are introduced which rely on reversible design principles but also incorporate objectives from on-chip interconnect architectures.

Experiments show that significant improvements with respect to power consumption, area, and delay can be achieved using the proposed direction.

I. Introduction

With the rise of very deep sub-micron and nanometric technologies, interconnects are increasingly affecting the overall power consumption, performance, and reliability of the chip. As a result, interconnect-centric design [1], a paradigm which sets the *interconnect architecture* in the center of the design process, is gaining relevance. To address the aforementioned issues, concepts from communication system engineering are getting introduced and adapted for on-chip communication architectures. Networks-on-chip and coding applications are two examples of this trend.

An established methodology is thereby the application of coding strategies in order to improve the on-chip interconnections [2], [3], [4], [5], [6]. Fig. 1 illustrates the basic idea. Instead of simply transmitting the desired data (see top of Fig. 1), the architectures are extended by an encoder and a decoder (as shown in the bottom of Fig. 1). Thanks to the additional flexibility provided by the coding, it is possible to achieve important improvements on the overall communication architecture and additionally consider the trade-off between power consumption, delay, and reliability.

In the past, the first broad use of (on-chip) coding was aimed at reducing the power consumption [2], [3]. As technology improved, coupling capacitances and the related timing and noise issues had to be addressed. This led to the coupling aware codes [4], [7], [6]. Recently, reliability is emerging as an important issue where coding plays a major role [7], [8]. A *key issue* in any case is the reduction of the overhead imposed by the additional hardware of the encoder and decoder.

All the codings proposed in the past have been determined by means of conventional design methods. In this work, we take another direction. In order to determine "good" codings for the above mentioned communication architectures, we propose the exploitation of design methods for reversible

978-3-9810801-8-6/DATE12/©2012 EDAA

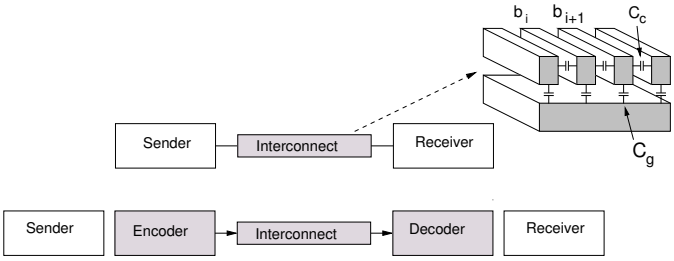


Fig. 1. Optimization of interconnect architectures through coding

logic [9]. Here, functions and circuits with an equal number of input and output signals are considered, whereby each input assignment maps to a unique output assignment. In the last decade, much progress has been made in the development of corresponding design methods for this kind of circuits. As a result, various methods for the synthesis of reversible circuits are already available [10], [11], [12], [13], [15], [16]. Since the considered encoders and decoders are also realizing reversible one-to-one mappings, it is reasonable to exploit this progress.

This is done in this work. More precisely, we introduce an exact approach and a heuristic approach, respectively, aiming at the determination of feasible codings to be used in order to improve the on-chip interconnections. For this purpose, the general idea of existing synthesis methods for reversible circuits work as "blueprint". In doing so, the past progress in the development of reversible synthesis methods is exploited while, at the same time, objectives from on-chip interconnect architectures are incorporated. We focus thereby on the development of coding strategies aiming at the improvement of the power-consumption. However, other objectives (e.g. delay) can be considered similarly.

Experiments confirm the benefits of the codings obtained by our approaches: On average, the corresponding encoder and decoder require 23% less static power, 25% less dynamic power, are 25% smaller, and have 17% less delay than their counterparts obtained by a conventional method. In the best cases, even up to 70% improvements are possible.

The remainder of this paper is structured as follows: The next section provides a brief review of the background on low-power coding for on-chip interconnections as well as on reversible logic and circuits. In Section III, the problem addressed in this paper is explicitly formulated and the general idea of the proposed solution is sketched. Afterwards, the proposed synthesis algorithms are introduced in Section IV. Finally, Section V provides experimental results and Section VI concludes the paper.

II. Background

In order to keep the remaining paper self-contained, in this section we briefly review the basics on the considered low-power coding approach. Afterwards, reversible circuits are introduced.

A. Low-Power Encoding

Power consumption and propagation delay are pattern dependent phenomena.

Consider an m-bit Boolean stationary random signal $\mathcal{X}$ which has to be transmitted over a bus with ground capacitance C_g and coupling capacitance $C_c = \kappa C_g$ (see Fig. 1). Furthermore, let the current and previous digital value of the j-th line be denoted as b_j^+ and b_j^- respectively and let $\Delta b_j = b_j^+ - b_j^-$ be the value of the temporal variation of that bus. The difference between two neighboring lines is defined as $e_j = b_j - b_{j+1}$ and the temporal variation between two consecutive lines is defined as $\Delta e_j = e_j^+ - e_j^- = \Delta b_j - \Delta b_{j+1}$.

Then, the power dissipated when loading or un-loading a capacitor C_g is given by $E_0 = \frac{1}{2} C_g V_{dd}^2$. Thus, the power dissipation depends on the probability of having a transition. In the presence of coupling capacitances, the situation gets more intricate. Then, the power extracted from the supply voltage of the driver corresponding to the line j is given by [17]

$$E_{vdd\_j} = 2E_0 \; [\Delta b_j + \kappa(2\Delta b_j - \Delta b_{j+1} - \Delta b_{j-1})] b_j^+ . \quad (1)$$

That is, the mean power dissipated in the form of heat during a transition can be calculated by adding the contribution of each bit and averaging (statistically) the cost associated with each pattern. Using the expectation operator $\mathbb{E}[\,\cdot\,]$, we get

$$E_d = E_0 \left(T_t + \kappa T_e\right), \quad (2)$$

where the total (self) transition activity, T_t, and the total coupling activity, T_e, are given by

$$T_t = \sum_{j=0}^{m-1} \mathbb{E}[\Delta b_j^2] \qquad T_e = \sum_{j=0}^{m-2} \mathbb{E}[\Delta e_j^2]. \quad (3)$$

As a consequence, the goal of a standard low-power code either is to minimize T_t (i.e. the number of transitions in the wires of a bus) or to minimize T_e (i.e. the number of transitions in opposite directions for neighbor wires if coupling is considered).

In buses where C_c is dominant, it is common to use a coupling aware code (e.g. edge coding [6]). The idea behind edge coding consists of delaying the rising edge of the signals. Then, because of the temporal shift, the lines on the bus cannot switch simultaneously in opposite directions. Therefore, the worst-case delay is avoided. This compensates the delay introduced by the edge coding while improvements with respect to the power consumption are achieved. After the incorporation of this technique, the power consumption is mostly dependent on T_t and can be optimized with a standard low-power coding scheme.

Templates for such a low-power coding scheme are illustrated in Fig. 2. One possibility is to consider the coding scheme as an XOR-decorrelator together with a Boolean function $E(x[n], x[n-1]) : \mathbb{B}^m \times \mathbb{B}^m \rightarrow \mathbb{B}^m$ which is decodable and minimizes the expected number of ones at the output of the decoder (see left-hand side of Fig. 2). An optimal implementation of such a coder E has been proposed in [3]. However, the hardware complexity of the resulting circuit is very high – particularly with increasing bit-widths. Thus, more structured solutions are usually applied.

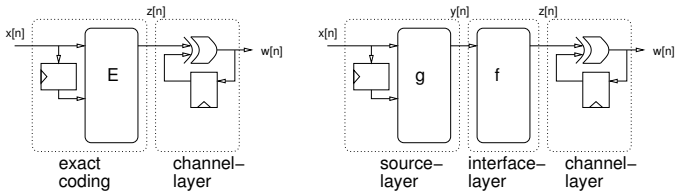


Fig. 2. Template for a low-power coding.

Similar to the coding template proposed in [2], we can describe a low-power coding scheme as composed of three layers (see the right-hand side of Fig. 2): the *source-layer*, the *interface-layer*, and the *channel-layer*. The goal of the source-layer is to detect the redundancy of the input data (denoted by $x[n]$) using the previous values of the data (denoted by $x[n-1]$). The channel-layer aims to simplify and, if possible, improve the data with respect to the characteristics of the physical medium. Typically, the channel-layer is composed by an XOR-decorrelator. By this, T_t is minimized with respect to the number of ones (i.e. the Hamming weight) of $z[n]$. Finally, the interface-layer implements a decodable map (denoted by f) between $y[n]$ and $z[n]$. More precisely:

$$E_d = E_0 T_t = E_0 \, \mathbb{E}[\mathrm{Ham}(z)] = E_0 \, \mathbb{E}[\mathrm{Ham}(f(y))], \quad (4)$$

where $\mathrm{Ham}(\,\cdot\,)$ represents the Hamming distance.

If the probability of $y[n]$ is known, an optimal decoder is obtained by mapping the most probable input values to codewords with a small Hamming weigh. This is called, *probability based mapping* (pbm) [2]. This is later illustrated by means of Table I in Section III.

In the remainder of this paper, we assume implicitly the templates of Fig. 2 and, thus, we address the question of how to automatically design efficient hardware implementations for that pbm-mapping using design methods for reversible circuit synthesis.

B. Reversible Logic and Reversible Circuits

A logic function $f : \mathbb{B}^m \rightarrow \mathbb{B}^{m'}$ over inputs $X = \{x_0, \ldots, x_{m-1}\}$ is *reversible* if and only if

- its number of inputs is equal to its number of outputs (i.e. $m = m'$) and
- it maps each input pattern to a unique output pattern.

In other words, a reversible function represents a bijection and, therefore, a valid coding.

A reversible function can be realized by a circuit $G = g_0 g_1 \ldots g_{d-1}$ comprised of a cascade of reversible gates g_i, where d is the number of gates. Fanouts and feedback are not directly allowed [18]. Several different reversible gates have been introduced including the Toffoli gate [19], the Fredkin gate [20], and the Peres gate [21]. In the following, we focus on Toffoli gates which are universal gates, i.e. all reversible functions can be realized by means of this gate type alone [19].

A *multiple control Toffoli gate* has a *target line* x_j and *control lines* $\{x_{i_0}, x_{i_1}, \ldots, x_{i_{k-1}}\}$. This gate maps $(x_0, x_1, \ldots, x_j, \ldots, x_{m-1})$ to $(x_0, x_1, \ldots, (x_{i_0} x_{i_1} \ldots x_{i_{k-1}}) \oplus x_j, \ldots, x_{m-1})$. That is, the target line is inverted if all control lines are set to 1; otherwise the value of the target line is passed through unchanged. A Toffoli gate with no control always inverts the target line and is a *NOT gate*. A Toffoli gate with a single control line is called a *controlled-NOT gate* (also known as the CNOT gate). The case of two control lines is the original gate defined by Toffoli. For brevity, we refer to a multiple-control Toffoli gate as a Toffoli gate.

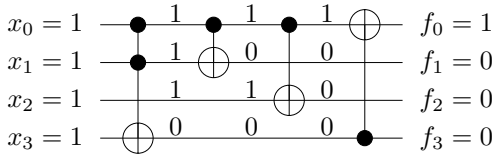


Fig. 3. Reversible circuit

TABLE I
ILLUSTRATING THE OBJECTIVE OF AN ENCODER

(a) Pattern probability

Inputs	Prob.
000	8%
001	8%
010	10%
011	10%
100	40%
101	10%
110	8%
111	6%

(b) Desired encoding

Inputs	Weight (H)
000	2
001	2
010	1
011	1
100	0
101	1
110	2
111	3

(c) Possible encoding

Inputs	Encoding
000	101
001	011
010	010
011	001
100	000
101	100
110	110
111	111

In the following, a Toffoli gate is denoted by the tuple $T(C, t)$ where $C \subset X$ is the possibly empty set of control lines and $t \in X \setminus C$ is the target line. Note that the control lines and unconnected lines pass through a gate unchanged. For drawing circuits, we follow the established convention of using the symbol $\oplus$ to denote the target line and solid black circles to indicate control connections for the gate.

Example 1. *Fig. 3 shows a reversible circuit composed of* $m = 4$ *circuit lines and* $d = 4$ *Toffoli gates. This circuit maps e.g. the input pattern* 1111 *to the output pattern* 1000 *(as shown in Fig. 3). Inherently, every computation can be performed in both directions (i.e. computations towards the outputs* and *towards the inputs can be performed).*

III. Automatic Design of Low-Power Encoders Using Reversible Circuit Synthesis

In this section, first the problem addressed in this paper is explicitly formulated. Afterwards, the general idea of the proposed solution is illustrated.

A. Problem Formulation

In this work, we focus on coding strategies to improve on-chip interconnect architectures based on the coding template shown in the right-hand side of Fig. 2 and briefly reviewed in Section II-A. Therefore, implementations of encoders and decoders are required that realize the pbm-mapping, i.e. that link the most frequently occurring data inputs to patterns with a low Hamming weight. The following example illustrates the objective.

Example 2. *Table I(a) shows a set of data inputs with their corresponding probability of occurrence. Based on that, an encoder should map the most frequently occurring data input (i.e. 100) to a bit-string with the lowest Hamming weight (i.e. 000). Then, the second-most frequently occurring data inputs should be mapped to a bit-string with the second-lowest Hamming weight and so on. That is, a coding is desired which leads to patterns with Hamming weights as shown in Table I(b). A precise coding satisfying this property is given in Table I(c).*

However, determining the best possible or even just one "good" coding is a non-trivial task. Already for 3-bit data inputs as considered in Example 2, eight different codings are possible. In the general case with m-bit data inputs, this number significantly increases to $\prod_{i=0}^{m} \binom{m}{i}$.

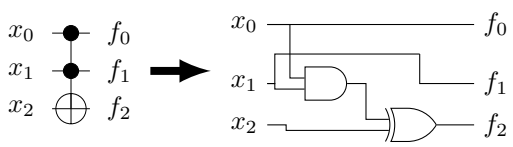


Fig. 4. Mapping Toffoli gate to conventional circuit

Motivated by this, the problem addressed in this paper is formulated as:

How can we automatically determine codings (as well as corresponding encoders and decoders) that map frequently occurring data inputs to patterns with low Hamming weight while keeping the hardware overhead as small as possible?

B. General Idea

Since encoders realize reversible one-to-one mappings, the application of synthesis approaches for reversible logic is a reasonable choice. Here, many achievements have been made in the last decade. Mainly motivated by the application of reversible logic in the design of quantum circuits [18], various efficient synthesis methods have been introduced. This include exact methods that guarantee minimality with respect to the number of gates (e.g. [10], [11]), more scalable heuristic approaches (e.g. [12], [13]), and first hardware description languages [16].

All these methods realize circuits composed of Toffoli gates as introduced in Section II-B. But since Toffoli gates represent a logic description, they can easily be mapped to a conventional gate library. As an example, Toffoli gates with two control lines can be mapped to a netlist composed of one AND-gate and one XOR-gate as shown in Fig. 4. From such a netlist, the established optimization and technology mapping steps can be performed.

While this already motivates the consideration of reversible synthesis methods for the design of low-power encoders, no existing approach does explicitly address the "Hamming weight"-objective so far. As in conventional circuit design, synthesis methods for reversible circuits get a function description with a fixed input-output mapping. However, the underlying reversibility enables to address this issue in a more elegant and powerful way than their conventional counterparts.

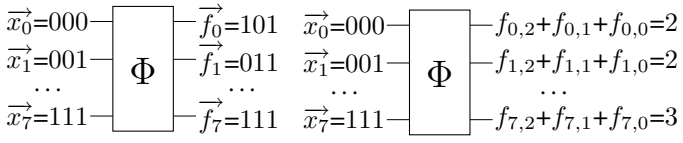
IV. Proposed Synthesis Approaches

In this section, two approaches for synthesis with respect to the "Hamming weight"-objective are introduced. This includes one method for exact synthesis and one method for heuristic synthesis.

A. Exact Approach

Exact synthesis algorithms determine a *minimal* circuit realization for a given function with respect to a given cost metric (e.g. with respect to the number of gates). Ensuring minimality often causes large computation times and, thus, exact approaches are applicable to relatively small functions only. Nevertheless, it is worth to consider exact methods, since e.g. they allow an evaluation of the quality of heuristic approaches or they generate minimal building blocks for larger circuits.

In the past, exact approaches synthesizing reversible circuits with respect to different cost metrics have been introduced [10], [11]. In this work, we are proposing an approach similar to the one introduced in [11]. Here, the exact synthesis problem is formulated as a sequence of decision problems. Given a reversible function $f : \mathbb{B}^m \rightarrow \mathbb{B}^m$, it is checked whether f can be synthesized using $d = 1$ Toffoli gates.



(a) Original approach [11] (b) With respect to Hamming weights

Fig. 5. Encoding for exact synthesis

If this is not the case, d is increased until a realization is determined. Since d is iteratively increased starting with $d = 1$, minimality is ensured. The respective checks are thereby performed by (1) formulating the synthesis problem as a satisfiability instance and (2) using satisfiability solvers to solve this instance.

Due to page limitation, we refer to [11] for a detailed description of the satisfiability formulation. In this work, the precise formulation is simplified as follows:

Definition 1. *Let $f : \mathbb{B}^m \rightarrow \mathbb{B}^m$ be a reversible function to be synthesized. Then, the satisfiability instance of the respective synthesis problem is given as*

$$\Phi \wedge \bigwedge_{i=0}^{2^m-1} ([\overrightarrow{x_i}]_2 = i \wedge [\overrightarrow{f_i}]_2 = f(i)),$$

where

- *$\overrightarrow{x_i} = (x_{i,m-1} \dots x_{i,0})$ is a Boolean vector representing the inputs of the circuit to be synthesized for truth table line i,*
- *$\overrightarrow{f_i} = (f_{i,m-1} \dots f_{i,0})$ is a Boolean vector representing the outputs of the circuit to be synthesized for truth table line i and,*
- *Φ is a set of satisfiability constraints representing the synthesis problem according to [11].*

Example 3. *Figure 5(a) shows the abstracted formulation of the synthesis problem applied for the function specified in Table I(c).*

To apply the "Hamming weight"-objective, all possible codings leading to the same Hamming weight have to be considered. In a simple way, this can be done by separately considering all possibilities resulting in $\prod_{i=0}^{m} \binom{m}{i}$ single synthesis calls. However, exploiting the reversibility of the considered circuit descriptions enables a more efficient approach. Therefore, the formulation is modified as follows:

Definition 2. *Let $H_0, \dots, H_{2^m-1}$ be the desired Hamming weights for each truth table line. Then, the satisfiability instance formulating the synthesis problem is given as*

$$\Phi \wedge \bigwedge_{i=0}^{2^m-1} \left(([\overrightarrow{x_i}]_2 = i) \wedge (\sum_{j=0}^{m-1} f_{i,j} = H_i) \right).$$

Example 4. *Figure 5(b) shows the new formulation of the synthesis problem for the Hamming weights provided in Table I(b).*

As can be seen, the output patterns $\overrightarrow{f_i} = f_{i,m-1} \dots f_{i,0}$ for each truth table line i are not set to fixed values in the new formulation. Instead, they can be arbitrarily chosen by the satisfiability solver. The additional constraints guarantee thereby that the solver only picks values which are consistent with respect to the desired Hamming weight. The reversibility of the considered circuit structure further ensures that the resulting mapping is always unique, i.e. that each data input always is mapped to one unique pattern. This is an essential property of the applied synthesis methods which can not directly be exploited in conventional synthesis techniques.

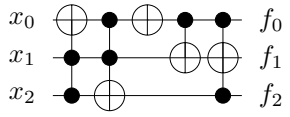


Fig. 6. Circuit obtained by exact synthesis

TABLE II
OBTAINED ENCODING

Inputs	Encoding
000	110
001	101
010	100
011	010
100	000
101	001
110	011
111	111

If the solver determines a satisfying assignment for this formulation, the precise coding can be obtained from the respective assignments to $\overrightarrow{f_i}$. If the solver proves that no such assignment exists, it is shown that no such coding can be generated by means of the given number d of Toffoli gates. Consequently, d is iteratively increased by 1 eventually leading to a *minimal* circuit.

Applying the proposed exact synthesis approach for the Hamming weights provided in Table I(b) leads to the reversible circuit shown in Fig. 6. From this circuit, the encoding as shown in Table II can be derived. This is the best coding with respect to the number of Toffoli gates (considering *all* possibilities).

B. Heuristic Approach

While exact approaches guarantee minimality, usually they do not scale well. Thus, heuristic synthesis methods are applied if larger functions should be realized. A very well-known heuristic synthesis approach for reversible circuits has been introduced in [12]. Here, the basic idea is

- to traverse each line of the truth table representing the function to be synthesized and
- add gates to the circuit until the output values of each line match the corresponding input values (i.e. until the identity of both is achieved).

Gates are thereby chosen so that they don't alter already considered lines. Furthermore, gates are added starting at the output side of the circuit (this is, because *output* values are transformed until the identity is achieved).

In this work, we adapt this idea. But instead of traversing a truth table with fixed output patterns, we exploit the fact that output patterns with the same Hamming weights can be interchanged. This requires a new synthesis algorithm which is formulated as follows:

Algorithm *(Heuristic Synthesis). Synthesizes a reversible circuit with respect to the desired Hamming weights from which a coding can be obtained.*

1) *Generate an arbitrary coding satisfying the desired "Hamming weight"-objective.*
2) *Traverse all lines i ($0 \leq i < 2^m$) of the corresponding truth table. In each iteration, perform the following steps:*
   a) *If the currently considered input $\overrightarrow{x_i}$ is equal to the currently considered output $\overrightarrow{f_i}$, continue with the next input.*
   b) *Consider all lines $j \geq i$ with $H_i = H_j$, i.e. all non-traversed lines whose coding finally should have the same Hamming weight.*
   c) *From this set, choose the output pattern $\overrightarrow{f_j}$ with the lowest Hamming distance to $\overrightarrow{x_i}$. If $\overrightarrow{f_i} \neq \overrightarrow{f_j}$, swap $\overrightarrow{f_i}$ and $\overrightarrow{f_j}$.*
   d) *Add gates which map $\overrightarrow{x_i}$ to $\overrightarrow{f_i}$ while, at the same time, do not alter previously considered lines.*

TABLE III
APPLYING HEURISTIC SYNTHESIS

No. (i)	Input abc	Weight H_i	Initial abc	0^{th} abc	1^{st} abc	abc	...	7^{th} ...	abc
0	000	2	101	**000**	000	000		...	000
1	001	2	011	**110**	**011**	**001**		...	001
2	010	1	010	**111**	111	**101**		...	010
3	011	1	001	**100**	100	100	...	...	011
4	100	0	000	**101**	101	**111**		...	100
5	101	1	100	**001**	001	**011**		...	101
6	110	2	110	**011**	**110**	110		...	110
7	111	3	111	**010**	010	010		...	111

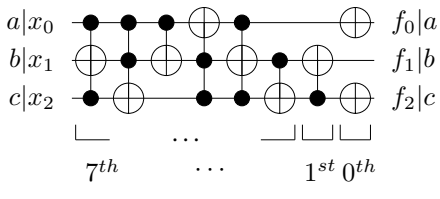


Fig. 7. Circuit obtained by heuristic synthesis

TABLE IV
OBTAINED ENCODING

Data	Encoding
000	101
001	110
010	100
011	010
100	000
101	001
110	011
111	111

In the following, the application of this algorithm is illustrated by means of the Hamming weights provided in Table I(b). The respective steps are illustrated in Table III. Here, the first columns denote the respective line numbers i of the truth table, the inputs, and the desired Hamming weight H_i for each line i (taken from Table I(b)). For brevity, the single bits are denoted by a, b, and c, respectively. The remaining columns provide the output patterns $\overrightarrow{f_i}$ for each line i as obtained by performing the respective steps of the algorithm.

First, the algorithm arbitrarily generates an initial coding with respect to the given Hamming weights (Step 1). In this example, we take the coding already given in Table I(c). Then, the 0^{th} line is considered. Since 000 is not equal to 101 (Step 2a) and no proper output pattern with a lower Hamming distance is available (Step 2b and 2c), gates are added which map $\overrightarrow{x_0} = 000$ to $\overrightarrow{f_0} = 101$ (Step 2d). To this end, two *NOT* gates as shown in Fig. 7 are applied[1]. This affects the output values of all the remaining lines as shown in the fifth column of Table III. Changes to the previous values are highlighted in bold.

Next, the mapping from $\overrightarrow{x_1} = 001$ is considered. Also here, 001 is not equal to 110 (Step 2a). But, a better output pattern is available. In fact, the current output of the 6^{th} line (011) has a lower Hamming distance to $\overrightarrow{x_1}$ (001) than the current output of the 1^{st} line (110). While the coding of both, the 1^{st} line and the 6^{th} line, finally should have the same Hamming distance, the precise pattern can arbitrarily be chosen. Since a smaller Hamming distance requires less gates to be added, the pattern from the 6^{th} line is the better choice. Consequently, the outputs of line 1 and line 6 are swapped as shown in the sixth column of Table III (Step 2b and 2c). Now, just one single gate inverting b and, at the same time, not altering the previously considered line has to be added (Step 2d). To this end, a Toffoli gate $T(\{c\}, b)$ is applied. The target line b performs the desired invertion, while the control line c ensures that this gate is only performed if $c = 1$. Therefore, the previous line (with $c = 0$) is not affected by this (see also the 7^{th} column of Table III).

This procedure is continued until all lines have been considered and, thus, an input-output identity is achieved. Then, a circuit has been generated which realizes a coding with the desired Hamming weights. For the considered example, a circuit as shown in Fig. 7 results from which the coding as shown in Table IV can been obtained. Since a heuristic has been applied, this circuit is larger than the one obtained by the exact synthesis approach (compare to Fig. 6). However, as confirmed by the experiments in this next section, the circuits and codings generated by the proposed heuristic approach are still better than the ones obtained from conventional synthesis methods.

[1]Note that all gates are added at the output side of the circuit, since output values are transformed so that they match the input values.

V. EXPERIMENTAL EVALUATION

In this section, we present results which have been obtained by the proposed design methods. Therefore, the exact as well as the heuristic synthesis approach have been implemented in C++ on the top of RevKit [22]. Then, the obtained reversible solutions have been implemented in an industrial 65nm-technology using Synopsys' *Design Compiler*. Results are thereby generated by both, keeping the structure of the reversible circuit or just using the coding obtained by the proposed approaches. The best realization generated by this was then compared to the best realization obtained by applying a coding derived from a conventional design method.

For the evaluation, we used a set of benchmarks representing a variety of different input probabilities, i.e. *increasing* (the probability increases steadily), *decreasing* (the probability decreases steadily), *gauss* (the probability follows the Gaussian distribution), and *inv_gauss* (the probability follows the inverse of the Gaussian distribution). Besides that, some randomly generated distributions are considered (denoted by *random_1* to *random_6*).

The obtained results are summarized in Table V(a) (considering the codings obtained by the exact approach) and Table V(b) (considering the codings obtained by the heuristic approach), respectively. The first columns denote thereby the name of the benchmark and their corresponding bit-width m. Afterwards, the delay (in ns), the area (in μm^2), as well as the dynamic and static power consumption (in nW and mW/MHz, respectively) of the encoder are provided. As a reference, the number of Toffoli gates from the obtained reversible circuits as well as the respective number of their AND- and XOR-gates (derived as illustrated in Fig. 4) are also reported (denoted by #Tof and #gates, respectively). Finally, the differences of the results obtained by the proposed approach in comparison to the results obtained by the conventional approach are provided.

Note that all reported synthesis steps have been executed in negligible run-time, i.e. in less than one minute. Thus, we waived reporting the run-times in Table V. However, due to the limited scalability of the exact approach (see also discussion in Section IV-A), the exact approach was not able to generate a coding for *increasing*, *decreasing*, *gauss*, and *inv_gauss* with a bit-width larger than 4. Furthermore, the heuristic approach was not able to improve the results of the conventional approach for the benchmarks *random_1* to *random_6*. Thus, we omitted the corresponding (redundant) numbers in Table V(b) and refer to the same numbers in Table V(a).

The results clearly show the benefits of the proposed approaches. In more than half of the cases, improvements can be observed. These improvements are thereby significant, i.e. the static power consumption can be improved by up to 80% and the dynamic power consumption, the area, and the delay can be improved by up to 70% in the best cases. Considering all benchmarks (also those without any improvement), on average static power consumption is reduced by 23%, dynamic power consumption by 25%, area by 24%, and delay by 17%. That is, applying design methods for reversible circuits indeed leads to much better codings and, thus, represents a promising alternative to conventional design methods.

TABLE V
EXPERIMENTAL RESULTS
(a) Exact approach from Section IV-A

Benchmark		Conventional approach				Proposed approach						Difference (in %)			
Name	m	delay	area	dyn. power	stat. power	#Tof	#gates	delay	area	dyn. power	stat. power	delay	area	dyn. power	stat. power
3_increasing	3	0.05	75.60	0.032	6.774	6	7	0.05	32.40	0.014	2.757	0.0	-57.1	-55.8	-59.3
3_decreasing	3	0.05	49.68	0.020	3.986	4	6	0.05	49.68	0.020	3.986	0.0	0.0	0.0	0.0
3_gauss	3	0.05	59.40	0.023	4.322	5	8	0.05	59.40	0.023	4.322	0.0	0.0	0.0	0.0
3_inv_gauss	3	0.05	18.00	0.007	1.586	2	2	0.05	18.00	0.007	1.586	0.0	0.0	0.0	0.0
4_increasing	4	0.11	133.56	0.051	8.193	7	9	0.05	53.64	0.023	5.137	-54.5	-59.8	-54.0	-37.3
4_decreasing	4	0.08	183.96	0.075	15.404	4	6	0.05	75.24	0.029	6.954	-37.5	-59.1	-61.0	-54.9
4_gauss	4	0.10	136.80	0.054	11.192	5	6	0.08	37.08	0.016	2.102	-20.0	-72.9	-71.2	-81.2
4_inv_gauss	4	0.11	117.72	0.052	8.507	4	5	0.06	108.36	0.040	7.171	-45.5	-8.0	-22.8	-15.7
random_1	3	0.07	64.80	0.029	7.196	4	5	0.09	45.00	0.019	3.304	28.6	-30.6	-35.2	-54.1
random_2	3	0.09	45.72	0.020	4.212	3	4	0.08	51.12	0.020	3.413	-11.1	11.8	-2.1	-19.0
random_3	4	0.13	63.36	0.028	3.790	6	9	0.13	63.36	0.028	3.790	0.0	0.0	0.0	0.0
random_4	5	0.12	76.68	0.034	7.456	7	8	0.12	76.68	0.034	7.456	0.0	0.0	0.0	0.0
random_5	5	0.10	94.68	0.044	8.202	8	11	0.09	26.64	0.011	2.351	-10.0	-71.9	-74.8	-71.3
random_6	5	0.09	118.08	0.054	11.763	5	7	0.10	90.72	0.041	8.265	11.1	-23.2	-23.6	-29.7

(b) Heuristic approach from Section IV-B

Benchmark		Conventional approach				Proposed approach						Difference (in %)			
Name	m	delay	area	dyn. power	stat. power	#Tof	#gates	delay	area	dyn. power	stat. power	delay	area	dyn. power	stat. power
3_increasing	3	0.05	75.60	0.032	6.774	7	10	0.05	32.40	0.014	2.757	0.0	-57.1	-55.8	-59.3
3_decreasing	3	0.05	49.68	0.020	3.986	4	7	0.05	49.68	0.020	3.986	0.0	0.0	0.0	0.0
3_gauss	3	0.05	59.40	0.023	4.322	9	13	0.05	59.40	0.023	4.322	0.0	0.0	0.0	0.0
3_inv_gauss	3	0.05	18.00	0.007	1.586	3	5	0.05	18.00	0.007	1.586	0.0	0.0	0.0	0.0
4_increasing	4	0.11	133.56	0.051	8.193	11	14	0.09	26.64	0.011	2.351	-18.2	-80.1	-77.8	-71.3
4_decreasing	4	0.08	183.96	0.075	15.404	7	10	0.05	75.24	0.029	6.954	-37.5	-59.1	-61.0	-54.9
4_gauss	4	0.10	136.80	0.054	11.192	13	21	0.08	77.76	0.032	5.931	-20.0	-43.2	-41.9	-47.0
4_inv_gauss	4	0.11	117.72	0.052	8.507	9	17	0.08	81.36	0.032	5.824	-27.3	-30.9	-38.1	-31.5
5_increasing	5	0.11	495.36	0.171	34.657	35	80	0.13	388.80	0.135	26.210	18.2	-21.5	-21.3	-24.4
5_decreasing	5	0.38	713.88	0.384	50.471	30	75	0.11	623.88	0.223	46.608	-47.4	-12.6	-41.8	-7.7
5_gauss	5	0.14	366.48	0.138	25.480	34	84	0.14	366.48	0.138	25.480	0.0	0.0	0.0	0.0
5_inv_gauss	5	0.15	303.12	0.117	21.713	29	79	0.15	303.12	0.117	21.713	0.0	0.0	0.0	0.0
6_increasing	6	2.35	2540.14	1.649	124.172	70	190	1.01	1416.24	0.809	79.444	-57.0	-44.2	-50.9	-36.0
6_decreasing	6	2.45	2525.75	1.687	134.835	64	184	0.95	1422.36	0.857	85.457	-61.2	-43.7	-49.2	-36.6
6_gauss	6	2.93	3232.07	2.154	165.499	75	228	0.82	1177.92	0.655	74.829	-72.0	-63.6	-69.6	-54.8
6_inv_gauss	6	3.21	3113.26	2.100	153.345	69	222	0.72	1131.84	0.645	73.762	-77.6	-63.6	-69.3	-51.9
No improvements have been obtained for the benchmarks *random_1* to *random_7*, i.e. all results remained as provided in Table V(a).															

Legend of columns: Conventional approach: Coding obtained by a conventional design method Proposed approach: Coding obtained by exploiting reversible logic design
m: Bit-width of the pattern #Tof: Number of Toffoli gates in the encoder circuit #gates: Number of AND- and XOR-gates in the mapped encoder
delay: Delay of the encoder (in ns) area: Size of the encoder (in μm^2) dyn./stat. power: dynamic/static power consumption of the encoder (in nW and mW/MHz, resp.)

VI. CONCLUSIONS

This paper showed the suitability of reversible synthesis methods in the design of optimized coding architectures. For this purpose, an exact approach and a heuristic approach have been introduced. Both exploited the past progress in the development of reversible synthesis methods while, at the same time, incorporated objectives from on-chip interconnect architectures. Experiments confirmed the benefits of this new design paradigm and disclosed improvements by up to 70% in the best cases.

While in this work, we focused on *probability based mapping* aiming at the reduction of the Hamming weight of frequently occurring patterns, the proposed design direction can similarly be applied for other objectives. A more detailed investigation on this is left for future work. Furthermore, the results from this work motivate a deeper consideration of reversible logic design methods within the development of coding strategies for on-chip interconnections and provides the basis for further work in this direction.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) (DR 287/20-1).

REFERENCES

[1] S. Pasricha and N. Dutt, *On-Chip Communication Architectures*, ser. Systems on Chip. Morgan Kaufman, 2008.

[2] S. Ramprasad, N. Shanbhag, and I. Hajj, "A Coding Framework for Low-Power Address and Data Busses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 2, pp. 212–220, June 1999.

[3] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power-efficient bus interfaces," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 969–980, September 2000.

[4] S. R. Sridhara and N. R. Shanbhag, "Coding for System-on-Chip Networks: A Unified Framework," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 6, pp. 655 – 667, June 2005.

[5] A. García Ortiz, L. S. Indrusiak, T. Murgan, and M. Glesner, "Low-Power Coding for Networks-on-Chip with Virtual Channels," *Journal of Low-Power Electronics*, vol. 5, pp. 1–8, 2009.

[6] J. Seo, H. Kaul, R. Krishnamurthy, D. Sylvester, and D. Blaauw, "A Robust Edge Encoding Technique for Energy-Efficient Multi-Cycle Interconnect," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 2, pp. 264–273, 2011.

[7] A. Ganguly, P. P. Pande, and B. Belzer, "Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 11, pp. 1626–1639, 2009.

[8] A. Ejlali, B. M. Al-Hashimi, P. Rosinger, S. G. Miremadi, and L. Benini, "Performability/energy tradeoff in error-control schemes for on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 1, pp. 1–14, 2010.

[9] R. Wille and R. Drechsler, *Towards a Design Flow for Reversible Logic*. Springer, 2010.

[10] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 710–722, 2003.

[11] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple control Toffoli network synthesis with SAT techniques," *IEEE Trans. on CAD*, vol. 28, no. 5, pp. 703–715, 2009.

[12] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Design Automation Conf.*, 2003, pp. 318–323.

[13] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[14] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Trans. on Design Automation of Electronic Systems*, vol. 12, no. 4, 2007.

[15] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conf.*, 2009, pp. 270–275.

[16] R. Wille, S. Offermann, and R. Drechsler, "SyReC: A programming language for synthesis of reversible circuits," in *Forum on Specification and Design Languages*, 2010, pp. 184–189.

[17] T. Murgan, P. B. Bacinschi, S. Pandey, A. García Ortiz, and M. Glesner, "On the Necessity of Combining Coding with Spacing and Shielding for Improving Performance and Power in Very Deep Sub-Micron Interconnects," in *Integrated Circuit and System Design*, 2007, pp. 242–254.

[18] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[19] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*, W. de Bakker and J. van Leeuwen, Eds. Springer, 1980, p. 632, technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[20] E. F. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3/4, pp. 219–253, 1982.

[21] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A*, no. 32, pp. 3266–3276, 1985.

[22] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: A toolkit for reversible circuit design," in *Workshop on Reversible Computation*, 2010, pp. 69–72, RevKit is available at http://www.revkit.org.