DORCIS: Depth Optimized Quantum Implementation of Substitution Boxes

Matthew Chun; Amherst College, Massachusetts, USA; Email: machun24@amherst.edu
Anubhab Baksi; Nanyang Technological University, Singapore; Email: anubhab001@e.ntu.edu.sg
Anupam Chattopadhyay; Nanyang Technological University, Singapore; Email: anupam@ntu.edu.sg

***Abstract*—In this paper, we present the "DORCIS" tool, which finds depth-optimized quantum circuit implementations for arbitrary 3- and 4-bit S-boxes. It follows up from the previous LIGHTER-R tool (which only works for 4-bit S-boxes) by extending it in multiple ways. LIGHTER-R only deals at the top level (i.e., Toffoli gates), whereas DORCIS takes quantum decomposition (i.e., Clifford + T gates) into account. Further, DORCIS optimizes for quantum depth and T depth. We match, if not surpass, other optimized quantum circuit implementations put forth in the other papers. Similar to LIGHTER-R, our tool is also easy to use, and we provide an extended interface to IBM's Qiskit.**

***Index Terms*—S-Box, Quantum Computing, Optimized Implementation, LIGHTER-R**

I. Introduction

In recent times, quantum computing is being considered as a serious threat to the integrity of the security/privacy algorithms used in our regular communication. In particular, it is causing accelerated research efforts to defend cryptography for a post-quantum world. In the effort of learning which of our current in-use cryptographic protocols are weak to such attacks, we aim to use or simulate quantum computers ourselves to diagnose encryption weaknesses. Recent research works like [5], [15], [16] reflect this.

To optimize a Grover's search algorithm key recovery attack against a given protocol, we require a minimally burdensome quantum circuit implementation of the protocol. One such metric that scales with computational burden is the circuit's depth. In a quantum computer simulation, a depth-optimized quantum circuit decreases the time it takes to compute the outcome of the simulated attack. In a physical realization of a quantum computer, a depth-optimized circuit reduces the proximity between components, decreasing the amount of noise in the circuit.

Our Contribution

Previous work used an efficient meet-in-the-middle algorithm proposed in [18] to create a reversible circuit optimization tool for gate cost optimization [12]. However, unlike the classical computing paradigm, the state in a single quantum wire cannot be measured at points both before and after a quantum gate acts on the wire, which complicates the design process for a depth-optimization tool.

We therefore decided to port the meet-in-the-middle algorithm from [18] and built in a depth-searching metric into it. To see the difference in performance before and after our modifications, we weighted gate-costs the same as our depth gate weights for DORCIS and asked each one to implement the GIFT S-Box.

DORCIS also comes with a tool that decomposes non-Clifford gates to a T-depth minimizing decomposition. We use DORCIS here to find depth-optimizations of other circuits superior to all other depth-optimization attempts.

All the code built for this publication, as well as all implementation examples created, are available as an open-source project[1].

The remainder of this paper is organized as follows. First, we give some background for S-boxes, depth, and T-depth in the context of circuit optimization in Section II. Next, we give an overview for our presented tool, "DORCIS" in Section III. Then, we show some optimized implementations for several 3-bit and 4-bit S-boxes, and compare our implementations to ones found in other papers in Section IV. Conclusion is given in Section V. Additionally, Section VI shows the Qiskit implementation in terms decomposed quantum gates obtained from DORCIS.

II. Background & Prerequisites

In reversible circuit optimization, depth and T-depth are two metrics that impact the cost and reliability of a quantum computer attack against a given encryption protocol. S-boxes describe how we want our circuits to behave given a particular input, and LUT/bit-slice formats are ways we concisely describe the total functionality of

[1] https://github.com/matthewchunqed/dorcis-public.

a given S-box. Here, we give some background on each of the relevant topics for this paper.

A. S-Box

An S-box is a function $f : \{0,1\}^M \mapsto \{0,1\}^N$ which, when converted to decimal representations, is a function $S : [0, 2^M - 1] \mapsto [0, 2^N - 1]$. Typically, block ciphers require the non-linearity from S to defend against well-known cryptanalysis attacks. In quantum computing, we must have f be bijective, since information loss is irreversible in the reversible computing paradigm. Thus, for this paper, we refer to a bijective $f : \{0,1\}^N \mapsto \{0,1\}^N$ as an N-bit S-box.

S-boxes ensure the property of confusion for a block cipher. In other words, S-boxes ensure that multiple parts of the key are used to create the ciphertext, therefore obscuring the relationship between the plaintext/key and the ciphertext.

Since S-boxes are non-linear functions at a high-level of inspection, there is a need to describe them concisely rather than explicitly writing the function definitions out. For an N-bit S-box, call its input X. Though X is the input to the S-box in binary, so $X \in \{0,1\}^N$, we can equivalently describe it in decimal form with $X \in [0, 2^N - 1]$. Henceforth, we will let the binary form of some number K be given as $K_0K_1...K_{N-1}$ with K_i indicating the $(i+1)^{\text{th}}$ most significant digit of K when written in binary.

In the look-up table (LUT) format, an N-bit S-box is described as a 2^N digit long string $L_0L_1...L_{2^N-1}$, with each digit in base 2^N. Then, the S-box is given by

$$f(i) = L_i \tag{1}$$

Concretely, the identity function would be given by $012\ldots2^N - 1$ since $L_i = i$.

In the bit-slice format, an N-bit S-box is described as a $N \times \frac{2^N}{\log_2 \beta}$ digit long number, with each digit in a predetermined base β that is a power of 2, and with $(N-1)$ spaces typically separating each of the N "blocks" of $\frac{2^N}{\log_2 \beta}$ digits. Denote each digit $B_{i,j}$, where $0 \leq i < N$ and $0 \leq j < \frac{2^N}{\log_2 \beta}$. Then, $B_{i,j}$ is the (j+1)-th digit of the $(i+1)^{\text{th}}$ block and

$$B_{i,j} = \sum_{n=0}^{(\log_2 \beta)-1} \left(\frac{\beta}{2^{n+1}}\right) \left(L_{n+j(\log_2 \beta)}\right)_i \tag{2}$$

A visualization of Equation (2) with $\beta = 16$ is shown in Table I. While both the S-box formats can be passed into DORCIS, we will refer to S-boxes only in LUT form for the remainder of this paper.

Additionally, for the rest of this paper, we will assume $\beta = 16$ for bit-slice formats. So for $\beta = 16$, the 3-bit identity S-box in bit-slice format would be given by `0F 33 55` and that of the 4-bit identity S-box would be given by `00FF 0F0F 3333 5555`.

TABLE I: 3-bit identity S-box bit-slice calculation with $\beta = 16$

LUT →	0	1	2	3	4	5	6	7	↓ bit-slice
L_0	0	0	0	0	1	1	1	1	`0F`
L_1	0	0	1	1	0	0	1	1	`33`
L_2	0	1	0	1	0	1	0	1	`55`

B. Depth

The depth of a quantum circuit (also known as the "quantum depth") is defined as the number of gates in a circuit, counting any set of parallelizable gates as a single gate.

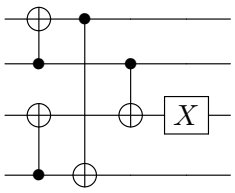


Fig. 1: A basic 4-bit circuit with quantum depth 3

In Figure 1, there are two pairs of parallelizable CNOT gates near the entry point of the circuit, followed by one additional gate thereafter. Hence, the quantum depth is 3.

Quantum depth is notably different from classical depth, which is the maximum number of gates which can be entirely contained in a left-to-right path through the circuit. In other words, the classical depth is the maximum number of gates a single source of electrical charge can pass through, entering from one of the lines at the left and exiting from any of the lines on the right. In Figure 1, the classical depth is 2. Henceforth, when we refer to "depth", we refer to quantum depth.

C. T-Depth

The T-depth of a quantum circuit is defined as the number of non-Clifford gates in a circuit, counting any set of parallelizable non-Clifford gates as a single gate. Both the NOT and the CNOT gates are Clifford gates, but the Toffoli (CCNOT) and CCCNOT gates are not.

In order to optimize for T-depth, we needed to decide on a T-gate decomposition for our non-Clifford gates that minimizes T-depth. The most T-depth minimal decomposition of a Toffoli gate has T-depth of 1, and requires four extra ancilla lines [21]. The decomposition has depth of 7, and it is shown in Figure 2.

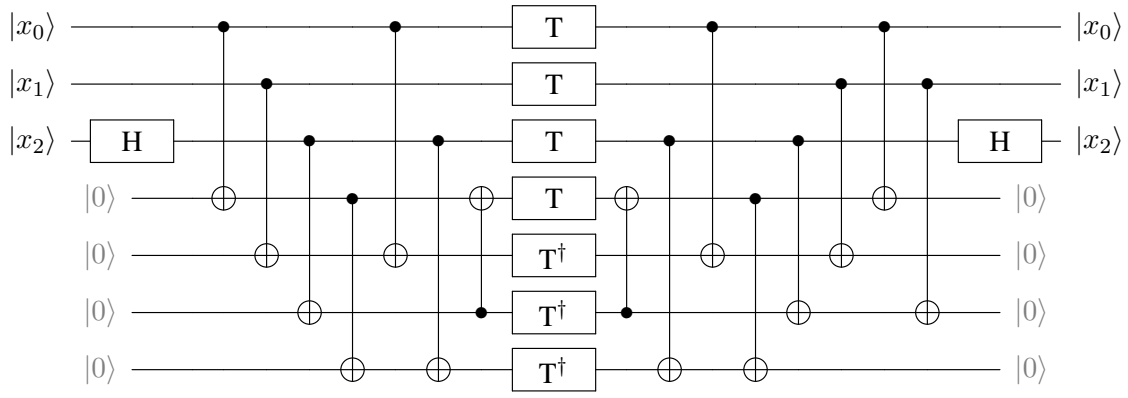


Fig. 2: Toffoli decomposition with T-depth 1, depth 7, and 4 ancilla qubits

To the best of our knowledge, the best decomposition for a CCCNOT gate has T-depth of 3, with the decomposition having depth of 21, using the generalization for the T-depth minimal representation given in [21]. We therefore choose these representations for the Toffoli and CCCNOT gates for DORCIS.

We acknowledge that, in choosing decompositions that optimize T-depth, we incur the cost of adding ancilla lines for each T-gate to use.

A notable alternative candidate has 3 T-depth Toffoli decomposition without ancillas, as depicted in Figure 3. Regardless of decomposition choice, various decompositions can be adapted into DORCIS by changing the depth and T-depth costs given to each gate.

By choosing the decomposition in Figure 2, we are saying that every Toffoli gate in every circuit henceforth will contribute to the depth on its three acting lines by 7, and to the T-depth by 1. We choose this decomposition over the decompositions with no ancilla lines, because we aim to find theoretical minimum T-depth implementations for arbitrary S-boxes.

We say a circuit is a 'top level' (or sometimes simply referred to as 'reversible') circuit if its non-Clifford gates have not been decomposed to its elementary T gates, H gates, and accompanying Clifford gates. We say a circuit is decomposed if all the non-Clifford gates have been decomposed.

D. Related Works

Proposed in the context of classical computing, LIGHTER [18], only accepts bijective S-boxes. However, LIGHTER has no built-in functionality for depth optimization. Also, LIGHTER search generates in-place implementation (i.e., without using any extra variable), which makes it compatible for reversible computing. This was studied in LIGHTER-R [12].

Speaking about other tools, one may note that [24] works by solving the shortest linear program (SLP) problem over a path between each input and its corresponding output. However, this approach assumes that states in a wire before and after a gate can both be accessed by later points in the circuit, which is not directly compatible for the quantum computing paradigm.

III. Tool Overview

A. LIGHTER Details

LIGHTER uses a graph-based meet-in-the-middle algorithm. It creates a start point at the identity function, and an end point at the output of a given S-box. It then repeatedly expands two graphs, starting at each point, by adding gates. As LIGHTER expands both graphs, it continually searches for collisions between the graphs, returning the connecting path as an implementation.

Curiously, the search strategy of the tool described by Dansarie in [10], [11] (which considers the algorithm due to Kwan [19][2], and an extension of it) was ignored by the LIGHTER designers. It offers the following advantages over LIGHTER:

1) It is heuristic, so it can be run multiple times and the best one can be taken as the final implementation (LIGHTER is deterministic and does exhaustive search). The former can even work with 8-bit S-boxes (albeit slowly), but LIGHTER does not scale up beyond 4-bit S-boxes.
2) It does not force reversibility, so it will likely to find improved implementations in the classical computing paradigm, and can deal with non-bijective S-boxes (the original work [19] presented implementation of the DES S-boxes). LIGHTER only looks for reversible implementation (and hence does not work with a non-bijective S-box).

[2]The blog by Matthew Kwan contains relevant information: https://darkside.com.au/bitslice/index.html.

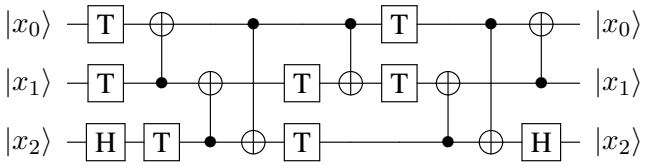


Fig. 3: Toffoli decomposition with T-depth 3, depth 9; from [2]

3) It does not assume a minimum three-step solution unlike LIGHTER (which comes from LIGHTER's meet-in-the-middle search strategy). This is useful when an S-box is implementable at two or less steps. LIGHTER will either force some redundant step (only to cancel it later) or will crash in a similar situation. All the examples of the first case (i.e., redundant computation) we found have to do with some trivial S-box (such as, identity or any other linear S-box); it is unclear whether this case happens with any non-trivial S-box.

Indeed, the only algorithmic advantage that LIGHTER has over the Kwan's algorithm is the reversible implementation, but the authors [18] seemed to be unaware of that.

Some of the problems/features of LIGHTER are shown in Codes 1, 2 and 3. In Code 1, LIGHTER implements the 4-bit identity S-box (`0123456789ABCDEF`); where its 3-step strategy forces to do a redundant XOR operation which is then canceled. Another example (with S-box `1032547698BADCFE`) where the least significant coordinate function is XORed with 1 is given in Code 2, where LIGHTER crashes. The same thing happens if one tries to generate; e.g., the SKINNY S-box [7] from the PICCOLO S-box [22]; despite one S-box being apart from another just by an XOR operation at one coordinate function. Finally, Code 3 shows the feature of the reversible implementation (with the S-box `103254EF98BADC67`). As LIGHTER does not use any extra variable, it has to recompute the same thing if that is needed more than once. In this example, the same NAND operation is computed twice (lines 6 and 7). This is useful in quantum computing (as no ancilla qubit is used); but causes extra cost in the classical computing paradigm.

Code 1: LIGHTER implementation of `0123456789ABCDEF` (C file content)

```
1  F[0] = X[0];
2  F[1] = X[1];
3  F[2] = X[3];
4  F[3] = X[2];
5
6  F[3] = XOR2 (F[3], F[2]);
7  F[3] = XOR2 (F[3], F[2]);
8
9  X[0] = F[0];
10 X[1] = F[1];
11 X[2] = F[3];
12 X[3] = F[2];
```

Code 2: LIGHTER result for `1032547698BADCFE` (terminal content)

```
From : 00FF 0F0F 3333 5555
To   : 00FF 0F0F 3333 AAAA
terminate called after throwing an instance of
↪  'std::out_of_range'
  what():  map::at
Aborted (core dumped)
```

Code 3: LIGHTER implementation of `103254EF98BADC67` (C file content)

```
F[0] = X[0];
F[1] = X[1];
F[2] = X[2];
F[3] = X[3];

F[3] = MAOI1(F[3], NAND2(F[1], F[2]      ),
 ↪  F[3], NAND2(F[1], F[2]      ));
```

```
F[0] = MOAI1(F[0], NAND2(F[1], F[2]      ),
 ↪  F[0], NAND2(F[1], F[2]      ));

X[0] = F[0];
X[1] = F[1];
X[2] = F[2];
X[3] = F[3];
```

B. Modifications from LIGHTER

Unlike LIGHTER, DORCIS optimizes with respect to depth and T-depth, not gate costs. Therefore, instead of associating a cost with each gate, we need to store the depth and T-depth contribution of each gate to the circuit.

To calculate this, we create an N-bit array $depths[N]$, where $depths[i]$ is the number of gates which use line i as either an input or an output. Whenever a new gate is added, we increment the input lines $depths[i]$ by the appropriate depth contributions. We then modify the MITM algorithm and LIGHTER's base code and data structures to accommodate a new, and qualitatively different, metric to optimize against.

For example, depth has to be calculated by adding gates starting near the circuit entry point before adding gates sequentially in the direction of the circuit exit. In the LIGHTER's base code, addition of gate costs is done from the collision point of the two graph expansion paths towards each respective end of the circuit. Since depth's gate contributions may be different dependent on the order the gates come in, we incorporated a stack and a queue to store the collision path's gates. Then, we could correctly compute depth by calculating gate contributions sequentially from circuit start to end.

C. Search Differences from LIGHTER-R

Unlike LIGHTER and LIGHTER-R, the parameter to optimize against has changed. We still use the graph expansion and BFS algorithm originally described in LIGHTER, due to its generality and verified efficiency, but the interpretation on what to search for has significantly changed. This introduced an additional challenge which required the aforementioned algorithmic modifications.

To verify the differences in searching algorithms, we searched for an optimized S-box implementation of the GIFT block cipher with both programs (cost for LIGHTER, depth for DORCIS), weighting the LIGHTER costs the same as our depth search weights. We can see that the they create very different implementations. Notably, LIGHTER-R's proposed implementation (Figure 5b) has a depth of 32, while ours (Figure 5c) has a depth of 31. This comparison demonstrates the algorithmic modifications we made to suit the qualitative differences between cost and depth.

DORCIS, by default, outputs implementations in terms of the top level gates, to maintain the convention of LIGHTER. However, we included a tool to decompose top level gates output by DORCIS.

D. Comment on PEIGEN

PEIGEN has two components: An extension of the LIGHTER code, to 3-8 dimensions, and a circuit depth optimization tool. The depth optimization tool is unfit for adaptation to quantum circuits because it assumes that a signal in a wire can be measured from both before and after a gate acting upon it. Thus, it reduces the depth-circuit problem to a shortest-linear-path problem. But this assumption, which motivates the design of the solving algorithm, cannot be translated into a quantum circuit tool without undue and significant ancilla line additions.

The LIGHTER component from PEIGEN optimizes with respect to gate costs, and uses the same searching method. However, attempts to search for implementations of non-trivial 5-bit or larger S-boxes never returned an output. Since PEIGEN included code from other applications as well, each gate programmed in had several other necessary dependencies or inherited attributes from other files in the PEIGEN tool. So modifications of the code to add new quantum gates did not create operational gates.

Though PEIGEN could compile with a 5-bit search request, its exhaustive search time scales exponentially with S-box dimension. Hence, we chose to build off of LIGHTER rather than PEIGEN.

IV. Implementation/Result

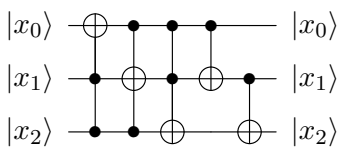
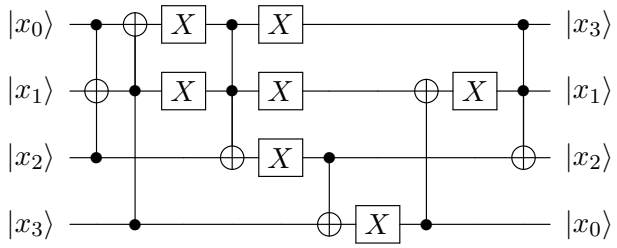
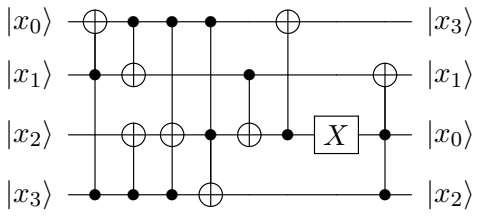


Fig. 4: LOWMC S-box implementation by DORCIS

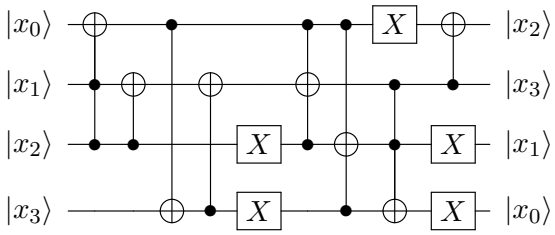
Unlike LIGHTER-R, DORCIS also works with 3-bit S-boxes. Figure 4 shows a depth optimized implementation of LOWMC (which is given by `01367452`).



(a) Provided by [17], found empirically (depth 34)



(b) LIGHTER-R (depth 32)



(c) DORCIS (depth 31)

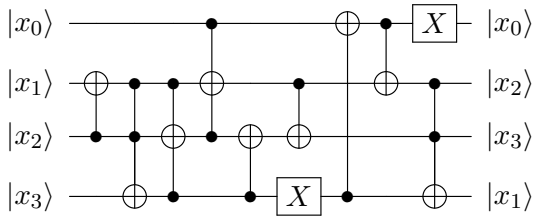
Fig. 5: GIFT S-box implementation

The implementation in Figure 5b shows a LIGHTER implementation without any modifications. The implementation in Figure 5c shows a DORCIS provided implementation.

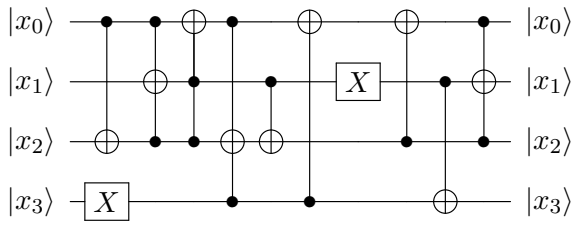
The key difference in our tool is how the metric we optimize against is calculated, given an arbitrary circuit. The aforementioned prior work optimized off of gate costs and total circuit costs, and it searched for circuits using as few costly gates as possible. Our software searches for circuits with a high number of gates running in parallel, which reduces depth without necessarily reducing circuit cost.

In [17], the authors provide an implementation of the GIFT S-box (`1A4C6F392DB7508E`) that is shown in Figure 5a. Using the same non-Clifford gate decompositions as our implementations assume, their implementation of the GIFT S-box has depth 34 and T-depth 4. In the same paper, the authors also put forth an implementation for the PRESENT S-box (`C56B90AD3EF84712`), which was generated using LIGHTER-R, and is shown in Figure 6a.

From Figure 6a, we can see the LIGHTER-R generated implementation of the PRESENT S-box has depth 33 and T-Depth 4. In Figure 6b, we can see that the DORCIS optimized circuit has depth 32 and T-depth 4. This is a



(a) Used in [17] (by LIGHTER-R)

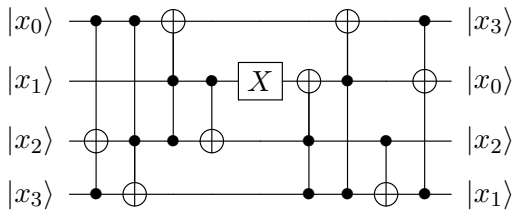


(b) Depth-optimized by DORCIS

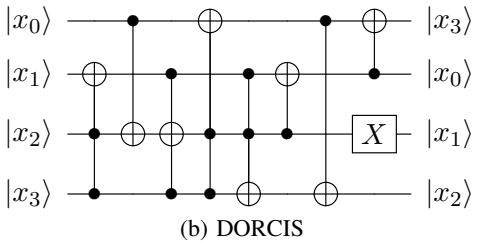
Fig. 6: PRESENT S-box implementation

step-up compared to LIGHTER-R, and a step-up in depth optimization from all previously known implementations of PRESENT.

Recent work has involved the development of a block cipher, DEFAULT, resistant to differential fault attacks. The DEFAULT block cipher consists of layers, with an outer layer (DEFAULT-LAYER) being applied, an inner layer (DEFAULT-CORE), then a second round of DEFAULT-LAYER [4, Chapters 7, 8] (and also in [3, Chapters 7, 8]). Since DEFAULT is a relatively new proposal, we will demonstrate the utility of DORCIS for analysis of lesser known protocols by examining the stark differences in implementations given by [4, Chapters 7,8] and DORCIS.



(a) LIGHTER-R (from [14])



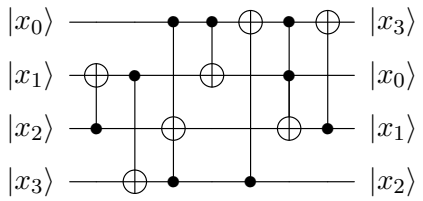
(b) DORCIS

Fig. 7: DEFAULT-CORE S-box implementation

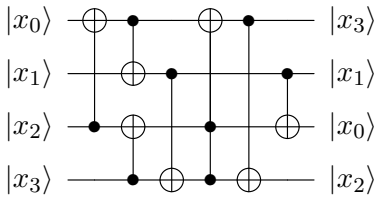
The provided DEFAULT-CORE S-box implementation has depth 45, T-Depth 6. It was generated by LIGHTER-

R. DORCIS provided an implementation with depth 31, and T-depth 4, for a total reduction of 14 in the depth metric, and 2 in the T-depth metric.

The provided DEFAULT-LAYER implementation has depth 19, T-depth 2. The implementation found by a DORCIS search has depth 11, T-depth 1.



(a) LIGHTER-R (from [14])



(b) DORCIS

Fig. 8: DEFAULT-LAYER S-box implementation

In Table II, we display the results of several sample S-boxes and their best depth-optimized decomposition. To find the reversible/quantum circuit, we used LIGHTER-R and weighted each of the gate costs the same as their depth weights for DORCIS. Hence, the LIGHTER-R-produced circuit aims only to minimize the number of total gates used. Notably, in all of our tests, we never found an S-box for which the most depth optimized circuit is different from the most T-Depth optimized circuit, nor did we find an S-box where the LIGHTER-R-produced circuit has a greater T-depth. This could be because non-Clifford gates are costly, so both algorithms minimize their usage at all costs. Thus, it is possible that when more non-Clifford gates are needed, an improvement on T-depth will also be seen in our software compared to the LIGHTER-R-produced circuit.

V. Conclusion & Future Work

From our demonstrated examples, we can see an improvement on depth and T-depth optimization on the currently known best results. Because depth and T-depth can be major indicators of circuit complexity, we believe DORCIS will be a compelling tool for finding 3-bit and 4-bit quantum S-box implementations.

DORCIS can be extended further from 3-bit and 4-bit S-boxes to 5-bit S-boxes and beyond. In our tests for determining 5-bit compatibility, we observed that the search-space was far too large to find implementations in. However, progress has already been made to extend such software into use in higher bit contexts [20]. Such methods may also be applied to DORCIS, which could see extended functionality given a similar approach. Apart from this, a follow-up work of the Kwan's algorithm [10], [11], [19] would be appreciated (as LIGHTER's search strategy is inferior) in the classical (irreversible) computing paradigm.

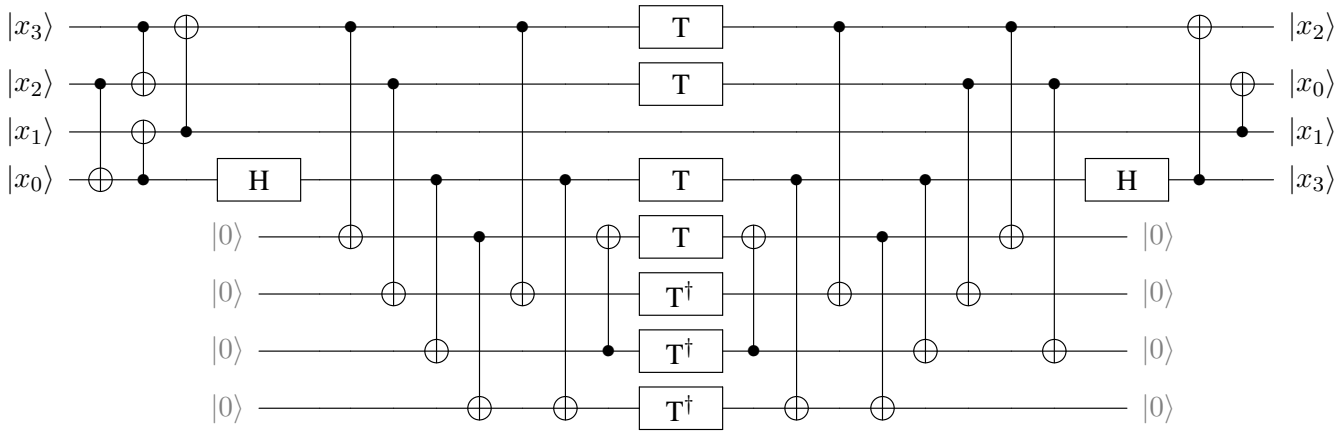


Fig. 9: Decomposition of DORCIS DEFAULT-LAYER S-box

TABLE II: DORCIS result summary

(a) 3-bit S-box

SBox	LUT	DORCIS	
		Depth	T-Depth
LOWMC [1]	01367452	23	3
PYJAMASK-3 [13]	13652470	23	3
SEA [23]	05674312	23	3

(b) 4-bit S-box

SBox	LUT	LIGHTER-R		DORCIS	
		Depth	T-Depth	Depth	T-Depth
ELEPHANT [8]	EDB0214F7A859C36	35	4	33	4
PICCOLO [22]	E4B238091A7F6C5D	32	4	30	4
PYJAMASK-4 [13]	2D397BA6E0F4851C	32	4	31	4
RECTANGLE [25]	65CA1E79B03D8F42	33	4	32	4
GIFT [6]	1A4C6F392DB7508E	32	4	31	4
PRESENT [9]	C56B90AD3EF84712	33	4	32	4
DEFAULT-CORE [4]	196F7C82AED043B5	45	6	31	4
DEFAULT-LAYER [4]	037ED4A9CF18B265	32	4	31	4

VI. Decomposed Quantum Gates (Qiskit Codes)

In Code 4 and in Code 5, we respectively present the Qiskit implementation of the DEFAULT-LAYER and the LOWMC S-boxes (as generated by DORCIS).

Code 4: DEFAULT-LAYER S-box (decomposition level)

```
from qiskit import QuantumCircuit
import matplotlib.pyplot as plt
circuit = QuantumCircuit(8)
circuit.cx((0),(1))
circuit.cx((3),(0))
circuit.cx((1),(2))
circuit.cx((2),(3))
circuit.h(1)
circuit.cx(3,4)
circuit.cx(0,5)
circuit.cx(1,6)
circuit.cx(4,7)
circuit.cx(3,5)
circuit.cx(1,7)
circuit.cx(6,4)
circuit.t(3)
circuit.t(0)
circuit.t(1)
circuit.t(4)
circuit.tdg(5)
circuit.tdg(6)
circuit.tdg(7)
circuit.cx(6,4)
circuit.cx(1,7)
circuit.cx(3,5)
circuit.cx(4,7)
circuit.cx(1,6)
circuit.cx(0,5)
circuit.cx(3,4)
circuit.cx(0,6)
circuit.h(1)
circuit.cx((1),(3))
circuit.cx((2),(0))
print(circuit.depth())
circuit.draw(output="mpl")
plt.show()
```

Code 5: LOWMC S-box (decomposition level)

```
from qiskit import QuantumCircuit
import matplotlib.pyplot as plt
circuit = QuantumCircuit(15)
circuit.h(0)
circuit.cx(2,3)
circuit.cx(1,4)
circuit.cx(0,5)
circuit.cx(3,6)
circuit.cx(2,4)
circuit.cx(0,6)
circuit.cx(5,3)
circuit.t(2)
circuit.t(1)
circuit.t(0)
circuit.t(3)
circuit.tdg(4)
circuit.tdg(5)
circuit.tdg(6)
circuit.cx(5,3)
circuit.cx(0,6)
circuit.cx(2,4)
circuit.cx(3,6)
circuit.cx(0,5)
circuit.cx(1,4)
circuit.cx(2,3)
circuit.cx(1,5)
circuit.h(0)
circuit.h(1)
circuit.cx(2,7)
circuit.cx(0,8)
circuit.cx(1,9)
circuit.cx(7,10)
circuit.cx(2,8)
circuit.cx(1,10)
circuit.cx(9,7)
circuit.t(2)
circuit.t(0)
circuit.t(1)
circuit.t(7)
circuit.tdg(8)
circuit.tdg(9)
circuit.tdg(10)
circuit.cx(9,7)
circuit.cx(1,10)
circuit.cx(2,8)
circuit.cx(7,10)
circuit.cx(1,9)
circuit.cx(0,8)
circuit.cx(2,7)
circuit.cx(0,9)
circuit.h(1)
circuit.h(2)
circuit.cx(1,11)
circuit.cx(0,12)
circuit.cx(2,13)
circuit.cx(11,14)
circuit.cx(1,12)
circuit.cx(2,14)
circuit.cx(13,11)
circuit.t(1)
circuit.t(0)
circuit.t(2)
circuit.t(11)
circuit.tdg(12)
circuit.tdg(13)
circuit.tdg(14)
circuit.cx(13,11)
circuit.cx(2,14)
circuit.cx(1,12)
circuit.cx(11,14)
circuit.cx(2,13)
circuit.cx(0,12)
circuit.cx(1,11)
circuit.cx(0,13)
circuit.h(2)
circuit.cx((0),(1))
circuit.cx((1),(2))
print(circuit.depth())
circuit.draw(output="mpl")
plt.show()
```

REFERENCES

[1] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[2] M. Amy, D. Maslov, M. Mosca, and M. Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, jun 2013.

[3] Anubhab Baksi. *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*. PhD thesis, School of Computer Science & Engineering, Nanyang Technological University, Singapore, 2021. https://dr.ntu.edu.sg/handle/10356/152003.

[4] Anubhab Baksi. *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*. Springer, Singapore, 2022. https://doi.org/10.1007/978-981-16-6522-6.

[5] Anubhab Baksi, Kyungbae Jang, Gyeongju Song, Hwajeong Seo, and Zejun Xiang. Quantum implementation and resource estimates for rectangle and knot. *Quantum Information Processing*, 20(12), dec 2021.

[6] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345, 2017.

[7] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.

[8] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant v2, 2021.

[9] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *CHES*, volume 4727, pages 450–466. Springer, 2007.

[10] Marcus Dansarie. *Cryptanalysis of the SoDark family of cipher algorithms*. PhD thesis, Naval Postgraduate School, Dudley Knox Library, 2017. https://calhoun.nps.edu/handle/10945/56118.

[11] Marcus Dansarie. sboxgates: A program for finding low gate count implementations of S-boxes. *Journal of Open Source Software*, 6(62):2946, 2021.

[12] Vishnu Asutosh Dasu, Anubhab Baksi, Sumanta Sarkar, and Anupam Chattopadhyay. Lighter-r: Optimized reversible circuit implementation for sboxes. *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, pages 260–265, 2019.

[13] Dahmun Goudarzi, Jérémy Jean, Stefan Kölbl, Thomas Peyrin, Matthieu Rivain, Yu Sasaki, and Siang Meng Sim. Pyjamask v1.0. 2019.

[14] Kyungbae Jang, Anubhab Baksi, Jakub Breier, Hwajeong Seo, and Anupam Chattopadhyay. Quantum implementation and analysis of default. Cryptology ePrint Archive, Paper 2022/647, 2022.

[15] Kyungbae Jang, Anubhab Baksi, Hyunji Kim, Hwajeong Seo, and Anupam Chattopadhyay. Improved quantum analysis of SPECK and lowmc. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, volume 13774 of *Lecture*

Notes in Computer Science, pages 517–540. Springer, 2022.

[16] Kyungbae Jang, Anubhab Baksi, Gyeongju Song, Hyunji Kim, Hwajeong Seo, and Anupam Chattopadhyay. Quantum analysis of AES. *IACR Cryptol. ePrint Arch.*, page 683, 2022.

[17] Kyungbae Jang, Gyeongju Song, Hyunjun Kim, Hyeokdong Kwon, Hyunji Kim, and Hwajeong Seo. Efficient implementation of present and gift on quantum computers. *Applied Sciences*, 11(11), 2021.

[18] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Transactions on Symmetric Cryptology*, 2017:130–168, Dec. 2017.

[19] Matthew Kwan. Reducing the gate count of bitslice des. Cryptology ePrint Archive, Paper 2000/051, 2000. https://eprint.iacr.org/2000/051.

[20] Zhenqiang Li, Fei Gao, Sujuan Qin, and Qiaoyan Wen. New record in the number of qubits for a quantum implementation of aes. Cryptology ePrint Archive, Paper 2023/018, 2023. https://eprint.iacr.org/2023/018.

[21] Peter Selinger. Quantum circuits of t-depth one. *Physical Review A*, 87(4), apr 2013.

[22] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 342–357, 2011.

[23] François-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. Sea: A scalable encryption algorithm for small embedded applications. In Josep Domingo-Ferrer, Joachim Posegga, and Daniel Schreckling, editors, *Smart Card Research and Advanced Applications*, pages 222–236, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[24] Ko Stoffelen. Optimizing s-box implementations for several criteria using sat solvers. In Thomas Peyrin, editor, *Fast Software Encryption*, pages 140–160, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[25] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Sci. China Inf. Sci.*, 58(12):1–15, 2015.