Does gate count matter? Hardware efficiency of logic-minimization techniques for cryptographic primitives

Shashank Raghuraman and Leyla Nazhandali

Abstract

Logical metrics such as gate count have been extensively used in estimating the hardware quality of cryptographic functions. Mapping a logical representation onto hardware is a trade-off driven process that depends on the standard cell technology and desired performance, among other things. This work aims to investigate the effectiveness of logical metrics in predicting hardware efficiency of cryptographic primitives. We will compare circuits optimized by a new class of logic minimization techniques that aim at reducing gate count with circuits of the same functionality that have not optimized for gate count. We provide a comprehensive evaluation of these designs in terms of area and power consumption over a wide range of frequencies at multiple levels of abstraction and system integration. Our goal is to identify different regions in the design space where such logic minimization techniques are effective. Our observations indicate that the logic-minimized circuits are much smaller than the reference designs only at low speeds. Moreover, we observe that in most cases, the logical compactness of these circuits does not translate into power-efficiency.

I. INTRODUCTION

One of the advantages of representing cryptographic functions as Boolean expressions is that such a representation provides an estimate of the complexity of the circuit by means of the number of logic operations required to express it. Furthermore, such a representation facilitates logic-minimization through Boolean algebraic simplifications such as factoring out sub-expressions. Due to the lack of an accurate estimate of the size of a logical representation on hardware, it makes sense for optimization techniques to focus on reducing a circuit's complexity by expressing the function using as few logic gates as possible. Understandably, there has been significant research on lightweight cryptographic hardware that has made extensive use of logic gate count as a metric to quantify the compactness of new designs and to compare them with existing solutions [1], [2], [3], [4], [5], [6]. Moreover, optimization tools have been developed for different classes of functions, driven primarily by gate count and/or logical depth as their cost functions [7], [8], [9], [10], [11]. Few works discuss the expected circuit speed by means of its logical depth before hardware synthesis [12], [13], [14], [15], [16], or as an estimate obtained from a library, depending on logical complexity [17].

While such logical metrics provide a preliminary estimate of the circuit's size on hardware, they do not account for the fact that converting a Boolean expression onto hardware is not a trivial task. It involves mapping a logical representation to a set of physical "standard cells" provided by a technology vendor. This logical-to-physical mapping is not straightforward due to the diversity in the size and functionality of standard cells. Commercial tools for this logic mapping and synthesis are governed by trade-offs between area, power, and performance of circuits. What this means is that a given Boolean function can be realized using many different hardware representations, and synthesis tools leverage the flexibility offered by standard cells to achieve a trade-off between area, performance, and power of the circuit, even if it entails logic modification.

The aforementioned dependence on standard cell technology necessitates an assessment of logic-minimized circuits that captures different corners of the design space. Techniques that reduce gate count might result in greater difficulty to optimize the circuit for speed, or consume more power. This eventually brings us to the question of whether the estimate of hardware efficiency provided by logical metrics remains accurate over a range of constraints. Many existing optimizations of circuits [18], [19], [9], [20] include synthesis results obtained for a particular frequency to validate their compactness. While this establishes their area efficiency at that particular frequency, we believe that a comprehensive analysis of the area, delay, and power of a more diverse group of circuits minimized by similar techniques would go a long way in providing designers a clearer picture of how they are transformed along the hardware implementation flow.

*This work was supported by NIST

In this work, we systematically evaluate the hardware quality of cryptographic primitives reduced by a new class of record-setting circuit-minimization techniques optimized for reducing gate-count [7], [8], [21]. This Low Gate-Count (LGC) tool reduces multiplicative complexity, minimizes the number of XOR operations, and is also capable of reducing the depth of combinatorial circuits. These techniques have generated circuits of the least known gate count [1], [2]. Our aim is to perform a comprehensive hardware efficiency analysis of these circuits covering a range of constraints on the design trajectory. Since these tools have been optimized for a large class of combinatorial cryptographic circuits, we believe this analysis provides significant insight into the overall hardware efficiency of such methodologies, and helps identify specific regions in the design space where these circuits are efficient. Specifically, we attempt to address the following points:

- **Trade-off regions**: The conflicting nature of hardware quality metrics makes it conceivable that synthesis methods that are superior in one metric are inferior in another. Identifying these regions of the solution space provides a sound assessment of when LGC tools are preferable over other alternatives.
- **Suitability towards wide range of functions**: It is possible that one synthesis method outperforms another for a particular class of logic functions, and not so for a different class. Structural properties of functions determine how they are affected by hardware optimization strategies. Since the LGC tool is applicable to a wide range of circuits, we analyze the consistency of hardware efficiency over different logic functions.
- **Scaling of hardware metrics**: Logic synthesis being a constraint-driven process, it is possible that a circuit that is better at one operating frequency can be worse at a higher frequency. We wish to observe how area and power scale with design constraints and complexity.

The rest of the paper is organized as follows. Section II briefly provides the required background on the aforementioned logic minimization techniques. Section III presents the analysis methodology adopted in our evaluation. This is followed by a discussion of important results of hardware synthesis, impact of physical design, and an integrated design example in Section IV. Section V concludes the paper.

II. BACKGROUND

A. *Digital Logic synthesis*

As there is no unique mapping of a logical description of function to a standard cell netlist, selecting the best hardware implementation is driven by trade-offs between technology cost factors. One of them is the delay of a cell, which simply refers to the time taken for a change in its inputs to be reflected at its output. Another property of a standard cell is its ability to drive logic at its output, referred to as its "drive strength". A cell of higher drive strength is naturally faster, but also bigger in size. This behavior is instrumental in an important fundamental trade-off between the area and performance of combinatorial circuits after synthesis.

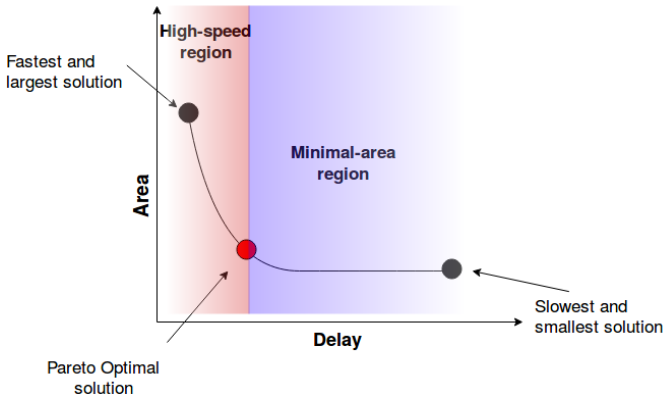
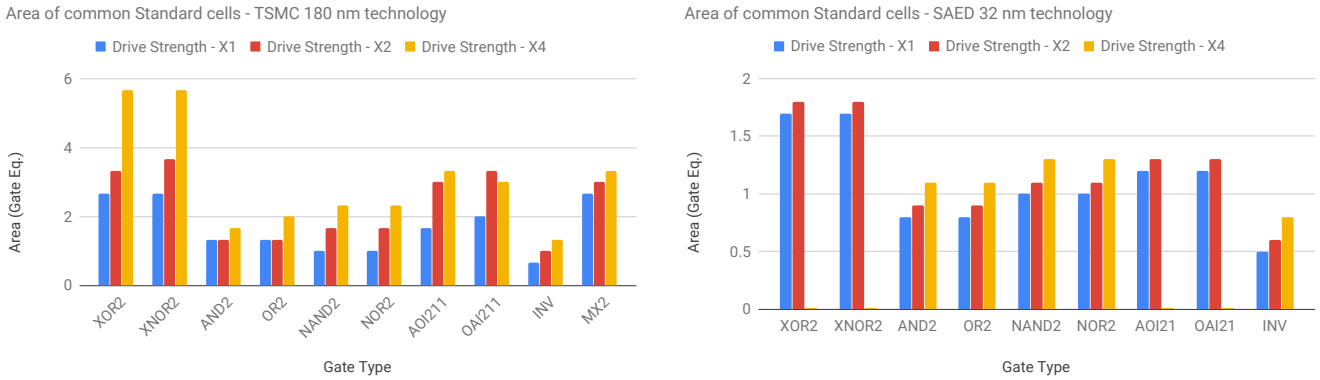


Fig. 1: A typical area-delay curve depicting trade-off points.

Figure 1 shows a typical area-delay curve obtained after hardware synthesis. The figure shows two regions in the plot. At low speeds (large circuit delays), the lack of tight performance constraints lets standard cells be weak

and slow, and consequently as small as possible. This is referred to as the "Minimal-Area" region. As the speed increases, standard cells in the circuit need to operate faster, and hence stronger cells are used. As a result, the circuit now incurs a sharp increase in its area in this region, referred to as the "High-Speed" region. The final solution on hardware is not always guaranteed to be one that optimizes both area and delay equally. Rather, it is one that minimizes area for given speed constraints.

Impact of standard cells: One of the challenges to logic synthesis tools is to find a sweet spot between the designer's requirement in terms of area, delay, and power, and what the technology library offers along with its design rules. Synthesis cost functions include all these constraints, and tools constantly evaluate trade-offs between them. An important point that needs mention is that there are variations in standard cells with respect to their hardware properties that cannot be overlooked. For example, Figure 2 shows a simple example of the area of commonly used standard cells from two different libraries, normalized to that of a 2-input NAND gate of the same technology. It is easy to see that XOR and XNOR gates are significantly bigger than other cells of an equivalent drive strength. Similar observations can be made for delay and power consumption - they are different for different cells, and depend on input signal transition and output load.



(a) (b)

Fig. 2: Area comparison of common 2-input standard cells from (a) TSMC 180 nm, and (b) Synopsys SAED 32/28 nm standard cell libraries.

This highlights the fact that a cryptographic LGC circuit that is generally dominated by XOR gates cannot be directly assumed to occupy smaller area on hardware, just by virtue of having fewer gates. The differences on hardware depend on heuristics used by synthesis tools to find an optimal mapping and sizing of cells to meet timing. While the logical starting point could be the smallest possible representation of a circuit, it is conceivable that the tool replaces certain logic gates with more complex cells in the library that are faster or have a higher ratio of drive strength to area. These effects become pronounced only when evaluation covers a range of speeds, which is the focus of this work.

B. Low gate-count (LGC) logic minimization techniques

This section discusses the important properties of circuit minimization techniques proposed by Peralta et al. [7], [8], [21]. Cryptographic logic primitives are optimized for low gate-count by partitioning the circuit into its linear (XOR) and non-linear (AND) parts. The non-linear portion is first reduced by techniques such as automatic theorem proving, resulting in a representation with fewer AND gates than the original. The linear portion of the circuit is now reduced using a greedy algorithm factoring out commonly used sub-expressions. The set of variables required to represent the function is initially filled with all the input variables, and gradually "grows" as it is filled in with sub-expressions that minimize the total number of XOR gates required. This process is performed repeatedly with random combinations of variables from the set, until a target number of XOR gates or a predefined

maximum time is reached. This technique was used with the addition of greedy depth-minimization heuristics to obtain a very compact circuit for AES SBox [2]. These algorithms have also been used to obtain some of the smallest known circuits for Galois Field arithmetic [1] and polynomial multiplication [22].

III. Analysis Methodology

A. Area and Power Analysis

To evaluate the LGC tool, we compare the quality of designs it creates, against those produced by commercial tools for other representations of the same logic functions. These comparisons are performed at different levels of abstraction in the implementation flow of an Application Specific Integrated Circuit (ASIC). In addition to evaluating the quality of combinatorial primitives as standalone blocks, we include analysis of an overall system design incorporating these primitives. This is aimed at demonstrating their suitability in a practical setting. The overall evaluation flow adopted is shown in Fig. 3.

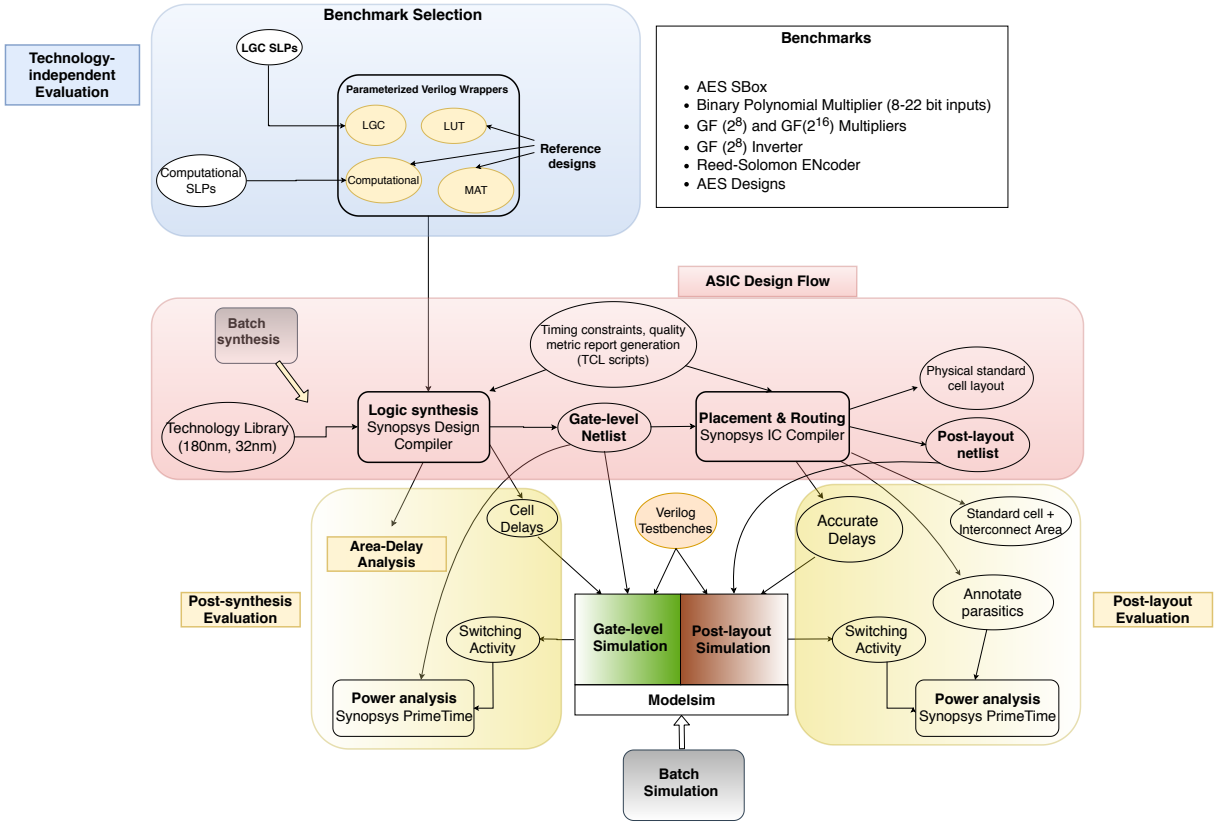


Fig. 3: Analysis methodology for evaluation of LGC circuits.

Logic synthesis of each design is performed at multiple frequencies using Synopsys Design Compiler (DC). This is continued till the point where the design fails to meet timing. Area analysis makes use of elaborate reports generated by DC. Moving further down the ASIC design flow, the effects of physical design are observed after placement and routing of these circuits using Synopsys IC Compiler. Power analysis is performed after both synthesis and layout, by first running a gate-level simulation of the netlists obtained at different frequencies, along with delays annotated through a Standard Delay Format (SDF) file. We feed 2^{16} random inputs to each of the design alternatives and obtain the switching activity in a Value Change Dump (VCD) file from ModelSim. For combinatorial blocks with 8-bit inputs such as the SBox and $GF(2^8)$ inverter, the test set is created in such a way that it covers all 2^{16} possible 8-bit transitions. The VCD files generated are then provided to Synopsys PrimeTime, which computes the power consumption of the circuits averaged over the simulation duration.

B. Cryptographic Benchmarks

We specifically focus on cryptographic circuits that are used as building blocks in bigger designs. To evaluate the effectiveness of optimization on different types of representations, we choose two types of benchmark designs where possible - *(i)* an abstract representation of the input-output relation with minimal external logic reduction, and *(ii)* a design that has been minimized by exploiting the computational properties of the circuit. In this section, we discuss the benchmark designs for two of the functions shown in Fig. 3 - AES SBox and Binary polynomial multiplier - as they highlight key shortcomings of using logical metrics to indicate hardware efficiency. The complete list of benchmarks can be found in [23]. The LGC tool provides minimized circuits in SLP format. To seamlessly insert these designs into a standard synthesis flow, these SLPs are first converted to dataflow Verilog that can be input to DC for logic synthesis. These Verilog designs are parameterized for each benchmark design, and for the multipliers, they are additionally parameterized for each input size. We obtained some of the LGC SLPs from [22], and the rest were provided to us by the designers.

1) AES SBox: The AES SBox has been extensively studied and several implementations have been proposed in literature [24], [25], [26], [2], [20] targeting various metrics for hardware efficiency.

- The AES SBox at its highest level is an $8X8$ look-up table whose gate-level realization is left completely to the logic synthesis tool. This reference design is denoted as *sbox_lut*.
- The computational properties of the SBox have been exploited to produce very compact designs in literature. The SBox by Wolkerstorfer et al. [24] decomposes elements in $GF(2^8)$ into two-term polynomials with coefficients in the sub-field $GF(2^4)$, owing to its simpler hardware implementation. Canright's design [25] further reduces gate-count by using a representation over the composite field $GF(((2^2)^2)^2)$, and the introduction of normal bases. These designs are denoted as *sbox_wolkerstorfer* and *sbox_canright* respectively. They are implemented in dataflow Verilog from the expressions used in their construction [24], [27].
- Another way of describing an SBox is using a Sum-of-Products or a Product-of-Sums form derived from its truth table. This gives a single-stage Positive Polarity Reed-Muller (PPRM) representation [28], denoted here by *sbox_pprm1*. Further, Morioka and Satoh proposed an architecture [29] which restricts the PPRM representations to three different stages of the SBox, leveraging both the PPRM structure and composite field representation (denoted by *sbox_pprm3*). Verilog models of these designs were obtained from [30].
- The LGC version used here is the low gate-count SBox proposed by Peralta et al. [2], denoted as *sbox_lgc*. This circuit was minimized by the LGC and depth-reduction techniques discussed in [7], [2].

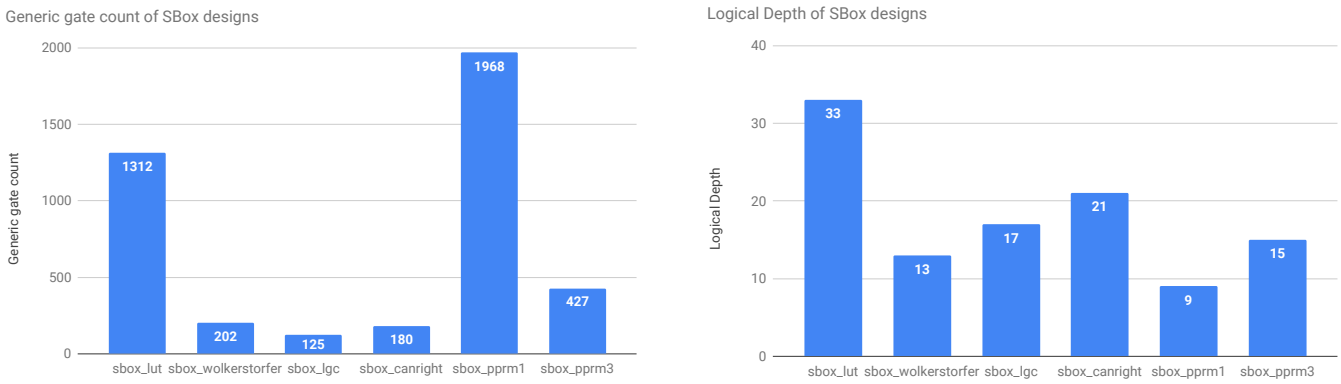
2) Binary Polynomial Multiplication: This can be viewed as multiplication of two polynomials of degree n over $GF(2)$. A polynomial $a(x) = x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a1 \cdot x + a_0$ is represented as an $n-$bit vector whose bits are the coefficients of $a(x)$. Polynomial multiplication is generally performed as the first step of field multiplication, and is followed by polynomial reduction. For multiplication in a field F_{2^n}, the arithmetic complexity of reduction is $O(n)$, while that of multiplication is $O(n^\omega)$, where $1 < \omega \leq 2$ [3]. It is therefore worthwhile to look at circuits for polynomial multiplication alone, which has been an old and much-studied problem. The benchmarks used are listed below. Since the complexity of binary multiplication grows quadratically with n, we perform comparison for a range of widths from 8 to 22 bits to evaluate how the efficiency of these designs scales with design complexity.

- The first benchmark is a *bit-parallel* matrix-based multiplier as described in [16]. It is referred to as *polymult_mat*, and is realized entirely as combinatorial logic employing $GF(2)$ addition and multiplication.
- The LGC versions of polynomial multipliers, denoted by *polymult_lgc*, are available at [22]. Many of them are designs that use the aforementioned computational versions as starting points for further logic reduction.

IV. Evaluation of the hardware implementation of logic-minimized circuits

As mentioned in the previous section, we present and discuss important results of AES SBox and polynomial multiplier, in order to highlight different properties of LGC designs that can affect their hardware quality. Results for the complete set of benchmarks can be found in [23].

A. *Experimental Results - AES SBox*

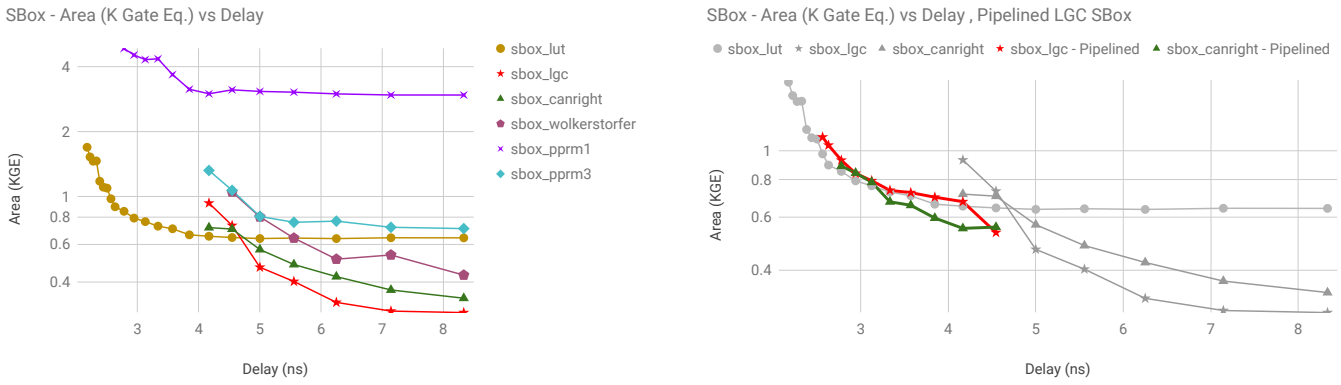


(a) Logical Gate count

(b) Logical Depth

Fig. 4: Technology-independent comparison of SBox designs

At first, a technology-independent comparison of the generic gate count and logical depth of the different SBox alternatives is shown in Fig. 4. From this figure, the *logical complexity* of *sbox_lut* appears to be extremely high, with over 10× more gates and 16 extra levels of logic as compared to *sbox_lgc*. A comparison of the expected hardware efficiency at this point would automatically declare *sbox_lut* to be not just bigger, but also significantly slower than *sbox_lgc* owing to all the additional levels of gates. However, as will become clear in the rest of this section, without more comprehensive evaluation, this estimate does not present the complete picture.
Fig. 5 shows the area (in K Gate Equivalents) of different SBox circuits plotted against the circuit delay, after logic synthesis using TSMC 180 nm technology library. The first point to be noted is that the compactness properties of *sbox_lgc* holds at large delays (10 ns), where it is upto 50% smaller than *sbox_lut*. The reason for this is that in the minimal-area region, there is little or no requirement for cell sizing and logic modification of *sbox_lgc*. Also, it becomes clear that commercial synthesis tools do not perform the type of rigorous logic reduction that the LGC tools do, which keeps the area of *sbox_lut* significantly larger than that of *sbox_lgc*.



(a) Area of SBox designs

(b) Pipelined versions of *sbox_lgc* and *sbox_canright*

Fig. 5: Area-delay comparison of SBox designs, using TSMC 180 nm technology.

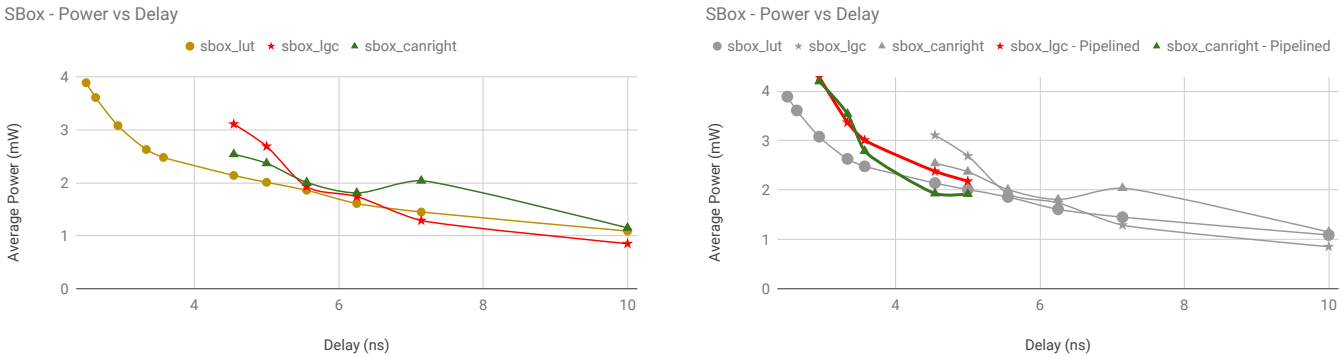
It is also clear from Fig. 5 ref that with increase in speed, the logic-minimized designs incur a sharp increase in area to the point where *sbox_lgc* becomes about 40% larger than *sbox_lut* (at 5-6 ns delay). Furthermore, in this delay-range, the area plot of *sbox_lut* remains largely flat, indicating greater ease to meet delay requirements. The reason for this is that *sbox_lut* offers greater flexibility for optimization with a particular target technology

library [31]. Owing to its abstract high-level representation, it is easily collapsed from 33 levels of gates before synthesis (Fig. 4(b)) to as few as 14 after synthesis. This is in sharp contrast to *sbox_lgc*, which is more restricted in its representation and hence does not allow such a reduction in depth - in fact, synthesis increases its depth from 17 to 18-19 levels of cells. Consequently, the critical path of *sbox_lgc* comprises more cells, each of which needs to be of higher strength than those of *sbox_lut* to meet delay requirements.

A second reason for the large area of *sbox_lgc* is that it is dominated by XOR gates, which is a natural result of its Boolean representation. In case of *sbox_lut*, its flexibility for optimization by mixing and matching different cells in the library results in zero XOR cells after synthesis, as opposed to over 80 XOR cells in *sbox_lgc* after synthesis. As was seen in Fig. 2, an XOR cell is much larger than other common cells of similar drive strength. This point, combined with the first observation of higher drive strength of cells in *sbox_lgc*, indicates an important property - in spite of *sbox_lgc* consisting of fewer cells overall than *sbox_lut*, a majority of these cells are both XOR and of a higher drive strength, making them 4-5$\times$ bigger than those of *sbox_lut*.

As a result of its ability to be collapsed onto fewer levels of cells, *sbox_lut* is naturally capable of reaching much higher speeds, as seen from Fig. 5. An optimization strategy to enable *sbox_lgc* to attain similar speeds, involves inserting a pipeline stage. This shortens the critical paths, and hence it is reasonable to expect the fewer cells to meet timing even in spite of being smaller and slower. Pipeline registers were therefore added at the inputs of logic-minimized SBoxes, and automatic retiming by DC was enabled, to push these registers through the combinatorial logic. The area-delay curve after pipelining is shown in Fig. 5(b). It is evident that this keeps the area-increase of *sbox_lgc* in check and enables it to achieve smaller delays, while occupying an area that is within $\pm$ 15% of the area of *sbox_lut*.

We now evaluate the power consumption of SBox designs. Fig. 6(a) shows that although *sbox_lgc* is upto 50% smaller in the minimal-area region, similar improvements are not seen for power. It consumes about 15-20% less power at very high delays (8-10 ns), but for speeds higher than that, power consumed by *sbox_lut* stays lower. Contrary to the observations noted in case of area, pipelining does not improve the power of *sbox_lgc* (shown in Fig. 6(b)), in spite of reducing cell sizes.



(a) Non-pipelined designs

(b) Pipelined versions of *sbox_lgc* and *sbox_canright*

Fig. 6: Power consumption of SBox designs, using TSMC 180 nm technology

One of the reasons for the high power-efficiency of *sbox_lut* is that by virtue of its ROM-structure, it has separate paths to each output from its inputs. As a result, not all of its cells are active for each combination of input bits. On the other hand, LGC designs involve greater signal activity for each SBox computation due to their algebraic re-computation of the outputs for every bit-flip at the input [31]. In addition, XOR gates propagate dynamic hazards with a probability of 1 [29]. Hence, the high XOR-dominance of *sbox_lgc* is another reason results in high switching. In spite of comprising fewer cells, *sbox_lgc* involves almost the same number of toggles as *sbox_lut*, with each toggle being more expensive due to the high drive strengths of cells in *sbox_lgc*. This makes it abundantly clear that fewer logic gates alone do not automatically imply power efficiency of LGC designs.

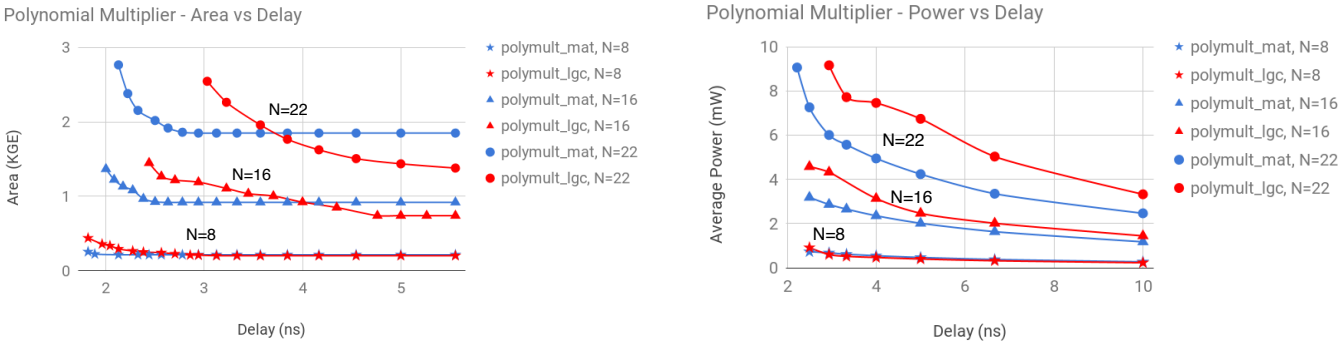
We conclude this analysis with Table I, where of *sbox_lgc* is compared with the two best benchmark designs. In the table, **-** indicates smaller area (or lower power), while **+** indicates higher area/power of *sbox_lgc* over its alternatives. The compactness of *sbox_lgc* is well-reflected in hardware at low speeds. Achieving higher speeds comes at the cost of an increase in both area and power over an abstract LUT-based design.

Benchmark Design	Region	Area	Power
Comparison of *sbox_lgc* with *sbox_lut*	Min-Area	- 54%	- 11-20%
	High-Speed	+ 2-13%	+12-40%
Comparison of *sbox_lgc* with *sbox_canright*	Min-Area	- 17-24%	- 4-36%
	High-Speed	+ 4-22%	+ 3-23%

TABLE I: Summary of analysis results for *sbox_lgc* with TSMC 180 nm technology library.

B. Experimental Results - Polynomial Multiplier

For the polynomial multiplier designs, we point out some of the key differences in properties from the SBox observations discussed in Section IV-A. In order to best observe a trend in the area and power, we present values for three different multiplier sizes. Fig. 7(a) shows the comparison in area between *polymult_lgc* and *polymult_mat* for input widths of 8, 16, and 22 bits. It is evident that in the minimal-area region, *polymult_lgc* gets better and better than *polymult_mat* with increase in multiplier width - from being only 6% smaller for an 8×8 multiplier, to being 25% smaller for a 22×22 multiplier. In the high-speed region, however, *polymult_lgc* gets more and more inferior to *polymult_mat*, to the point of being over 40% bigger for a 22×22 multiplier.



(a) Area of Polynomial Multipliers

(b) Power of Polynomial Multipliers

Fig. 7: Power consumption of Polynomial Multipliers, using TSMC 180 nm technology

In case of power, Fig. 7(b) shows that a small improvement of 14% due to *polymult_lgc* is seen only for an 8×8 multiplier in the minimal-area region. Everywhere else, we see *polymult_mat* to be more power-efficient. Moreover, this discrepancy widens with the increase in both speed and input width. From these area and power results, it is clear that a matrix-based multiplier certainly "scales" better with speed and input size than the LGC designs. Although the matrix-based design is not as abstract and high-level as an LUT-based BOX, it is still very high-level and symmetrical which makes it conducive to various optimizations by the synthesis tool resulting in better area efficiency in high speeds when compared to LGC design. The power efficiency of *polymult_mat* is also due to its balanced structure. In spite of its XOR-dominance, majority of the XOR gates have their inputs coming from gates at the same depth from the inputs. As a result, there is a far less likelihood of their input delays being mismatched [29]. In case of *polymult_lgc*, such a balance is much harder to achieve due to its minimization by aggressive removal of redundancies. As a result, it involves higher toggling due to dynamic hazard propagation and consequently higher power consumption.

In summary, logical compactness is susceptible to be lost at high speeds due to the impact of logic synthesis and even in low speeds it may not translate into power efficiency.

C. Impact of physical design

The physical design stage in the ASIC implementation flow involves placement and routing on a die area. These steps can further modify cell sizes due to the effects of the physical locations and interconnects between cells of the circuit. Hence, it is important to observe if this significantly changes the post-synthesis results. Taking the AES SBox as an example, our results show that the effects of physical design are minimal for circuits that have a large difference in their logical gate count. For example, *sbox_lgc* has about 10× fewer logic gates than *sbox_lut*. As a result, its area-delay curve after placement and routing follows a similar pattern as the post-synthesis values, and *sbox_lgc* remains 40% smaller than *sbox_lut* in the minimal-area region. However, for circuits whose logical representations differ by few tens of gates or less, such as *sbox_lgc* and *sbox_canright*, or 8×8 polynomial multipliers, the area advantages in post-synthesis might not extend to post-layout. In Fig. 8, it is clearly seen that while *sbox_lgc* is 20% smaller than *sbox_canright* after synthesis, it gets 20% bigger after layout. This underlines the point that minute differences in gate count are overshadowed by the heuristics of physical design and the impact of cells' locations, making them susceptible to variations in post-synthesis and post-layout size.

(a) SBox Area after synthesis

(b) SBox Area after layout

Fig. 8: Effect of physical design on SBox area

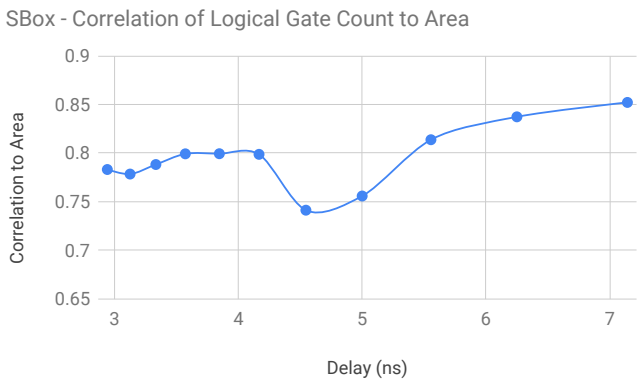
D. Relation between logical and technology-dependent quality metrics

Having looked at results and comparisons for specific benchmark circuits, we take a broader view of how well abstract logical quality metrics relate to hardware quality metrics. In this regard, we analyze the correlation of hardware area and power to these circuits' logical gate count. Owing to the large differences in properties of designs, such an analysis needs to be performed separately for each circuit. Fig. 9 shows the correlation of gate count to the area of both the SBox and polynomial multipliers.

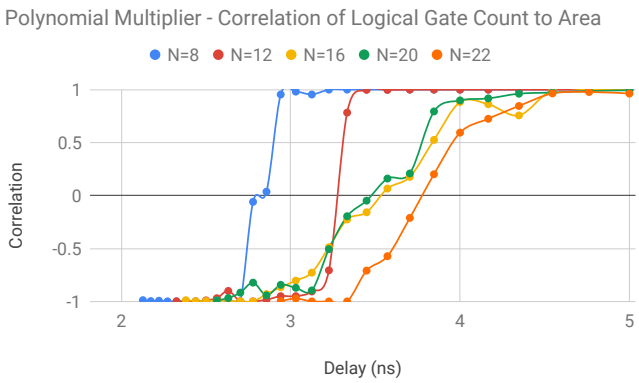
Logical gate count has a high correlation of over 80% in the minimal-area region, and falls to lower values with increase in speed. In case of polynomial multipliers, this drop is faster for bigger multipliers due to their increased complexity. These figures corroborate the utility of gate count as a good predictor of hardware size only at low speeds. The greater complexity involved in power consumption is depicted by a lack of meaningful correlation of gate count to the power of SBox and multipliers in Fig. 10. Except for very small multipliers, the correlation of gate count stays less than 60%, indicating its inability to provide a useful reflection of power-efficiency.

E. Integrated Design Example

The analyses performed so far considered the combinatorial blocks as standalone primitives. The primary reason for having chosen these benchmarks is that they have practical utility in bigger designs. We believe that it is important to assess the effectiveness of combinatorial optimization towards minimizing the area or power

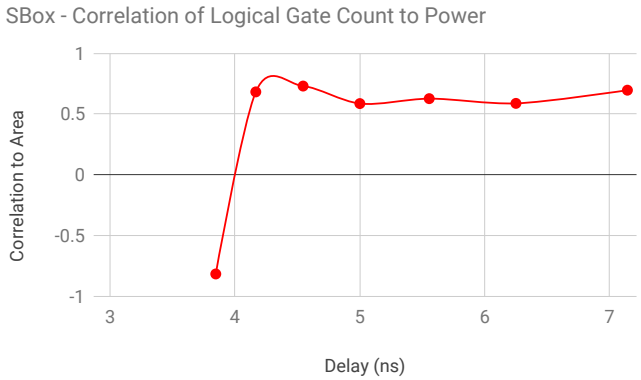


(a) Gate Count correlation to SBox Area

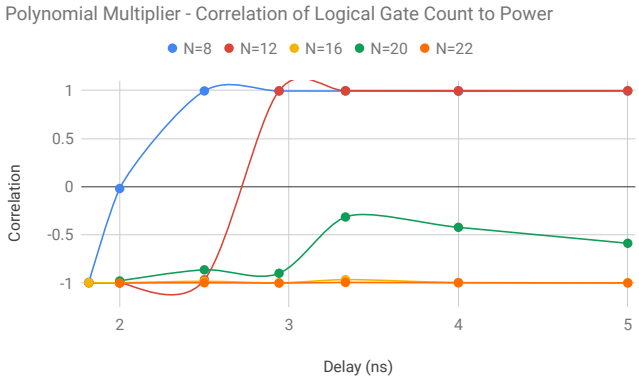


(b) Gate Count correlation to Multiplier Area

Fig. 9: Correlation of gate count to area.



(a) Gate Count correlation to SBox Power



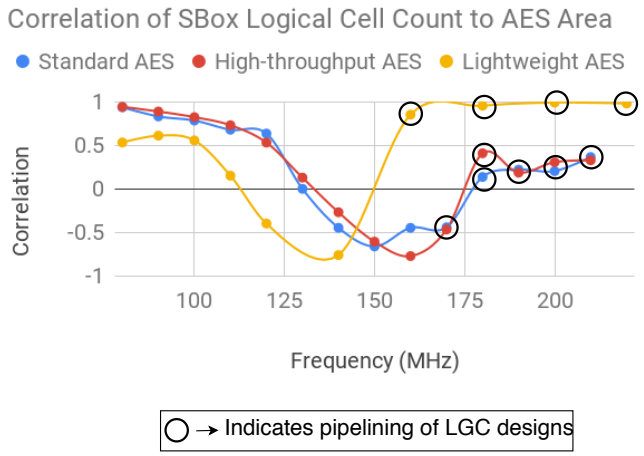
(b) Gate Count correlation to Multiplier Power

Fig. 10: Correlation of gate count to power.

of the complete system they are a part of. In this sub-section, we analyze the logic synthesis results of AES designs making use of different SBox circuits. The aim of this study is to highlight the fact that the total impact on a system is affected by the contribution of the primitive to the bigger design's area, and also the optimization performed by combining these primitives with external logic that is part of the design. Three different AES designs are analyzed here:

- *Standard AES*: An AES design with one SBox per each byte of the state and the key. There are 20 SBox circuits in total, and each encryption operation is completed in 10 clock cycles, i.e. one round per cycle.
- *High-throughput AES*: Computes two AES rounds in one cycle. There are 40 SBox circuits in total.
- *Lightweight AES*: Resource-shared design, with 4 SBoxes in total, with multiplexed inputs and outputs.

Fig. 11(a) shows the correlation of the SBox gate count to that of the total AES area. It can be clearly seen that while this correlation is high for the standard and high-throughput designs, it is around 50% for lightweight AES at low frequencies. The correlation increases at higher frequencies due to the effects of pipelining. The improvement in AES area obtained by using LGC SBox is shown in Fig. 11(b). It is evident that benefits of a smaller SBox are diminished when integrated with a lightweight AES design.



(a) Correlation of SBox Gate count to AES area

Region	AES Type	LGC *vs* LUT	LGC *vs* Canright
Minimal-Area	Standard	12-32% smaller	7-13% smaller
	High-Throughput	18-33% smaller	5-13% smaller
	Lightweight	8% smaller	4-8% smaller
High-Speed	Standard	9-16% smaller	6-14% smaller
	High-Throughput	11-19% smaller	1-7% smaller
	Lightweight	9% smaller	± 5%

(b) Improvement in AES area obtained by using *sbox_lgc*

Fig. 11: Analysis of SBox integrated into AES designs.

V. CONCLUSION

The analysis in this work has made it clear that conversion of a logical circuit representation to hardware is not trivial. Area and power efficiency of a combinatorial circuit are determined by standard cell library, how conducive the circuit structure is for optimization, delay requirements, and surrounding logic when it is part of a bigger design. Efficiency with respect to one metric does not imply that with respect to another. This reiterates the point that while logical metrics provide a reasonable initial estimate, comparing logical designs solely based on minute differences in gate count or logical depth is not an accurate comparison of their expected hardware quality. The more prudent question is under what conditions logical metrics and hardware quality have high correlations. In this regard, we conclude this paper by showing Table II. This table summarizes the correlation analysis for the benchmark designs studied in this paper and in [23]. In summary, the gate count is only a good representative of the design area in low speeds (low performance) and it is never a proper indicator for power consumption. When it comes to logical depth, the results are mixed and there is no clear pattern on its usability for predicting area or power. The future work of this research includes establishing methodologies and metrics that can outperform gate count and logical depth in predicting the post-synthesis quality of a circuit design.

Logical Metric	Design		Min-area Region		High-speed Region	
			Area	Power	Area	Power
Gate Count	SBox		H	M	M	L
	Polynomial Multiplier	N≤14	H	H	L	L
		N>14	H	L	L	L
	GF Multiplier		M	M	L	L
	GF Inverter		M	M	L	L
	AES	Standard	H	L	L→M	H→L
		High-throughput	H	L	L→M	M
		Lightweight	M	L	L→H	H
Logical Depth	SBox		M	L	L	L
	Polynomial Multiplier	N≤14	L	L	H	H
		N>14	L	H	H	H
	GF Multiplier		H	H	H	H
	GF Inverter		L	L	M	L→M
	AES	Standard	H	L	L→M	H →L
		High-throughput	H	L	L→M	M
		Lightweight	M	L	L→H	H

TABLE II: Correlation of abstract metrics to hardware quality metrics. H-High (Correlation $>$ 0.8), L-Low (Correlation $<$ 0.5), M-Moderate (0.5 $\leq$ Correlation $\leq$ 0.8).

REFERENCES

[1] J. Boyar, M. Find, and R. Peralta, "Small low-depth circuits for cryptographic applications," *Cryptography and Communications*, vol. 11, pp. 109–127, 2018.

[2] J. Boyar and R. Peralta, "A Small Depth-16 Circuit for the AES S-Box." in *Gritzalis D., Furnell S., Theoharidou M. (eds) Information Security and Privacy Research. SEC 2012. IFIP Advances in Information and Communication Technology*. Springer, Berlin, Heidelberg, 2012.

[3] M. Cenk and M. A. Hasan, "Some new results on binary polynomial multiplication," Cryptology ePrint Archive, Report 2015/094, 2015, https://eprint.iacr.org/2015/094.

[4] J. L. Imaña, R. Hermida, and F. Tirado, "Low complexity bit-parallel multipliers based on a class of irreducible pentanomials," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 1388–1393, 2006.

[5] M. Elia, M. Leone, and C. Visentin, "Low complexity bit-parallel multipliers for GF(2^m) with generator polynomial $x^m + x^k + 1$," *Electronics Letters*, vol. 35, pp. 551 – 552, 05 1999.

[6] C. Beierle, T. Kranz, and G. Leander, "Lightweight multiplication in GF(2^n) with applications to MDS Matrices," in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 625–653.

[7] J. Boyar and R. Peralta, "A New Combinational Logic Minimization Technique with Applications to Cryptology." in *Festa P. (eds) Experimental Algorithms. SEA 2010. Lecture Notes in Computer Science*, vol. 6049. Springer, Berlin, Heidelberg, 2010.

[8] J. Boyar, P. Matthews, and R. Peralta, "Logic minimization techniques with applications to cryptology," *J. Cryptol.*, vol. 26, no. 2, pp. 280–312, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1007/s00145-012-9124-7

[9] C. A. Wood, "Large substitution boxes with efficient combinational implementations," *Master's Thesis, B. Thomas Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, New York*, 2013. [Online]. Available: https://scholarworks.rit.edu/cgi/viewcontent.cgi?referer=https://www.google.com/httpsredir=1article=6531context=theses

[10] N. Courtois, D. Hulme, and T. Mourouzis, "Solving circuit optimisation problems in cryptography and cryptanalysis." *IACR Cryptology ePrint Archive*, vol. 2011, p. 475, 01 2011.

[11] C. Fuhs and P. Schneider-Kamp, "Synthesizing shortest Linear Straight-Line Programs over $gf(2)$ using SAT," in *Theory and Applications of Satisfiability Testing – SAT 2010*, O. Strichman and S. Szeider, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 71–84.

[12] A. Cilardo, "Fast parallel GF(2^m) polynomial multiplication for all degrees," *IEEE Transactions on Computers*, vol. 62, pp. 929–943, 2013.

[13] Y. Nogami, K. Nekado, T. Toyota, N. Hongo, and Y. Morikawa, "Mixed bases for efficient inversion in $f_{((2^2)^2)^2}$ and conversion matrices of SubBytes of AES," in *CHES*, 2010.

[14] J. Jean, T. Peyrin, and S. M. Sim, "Optimizing implementations of lightweight building blocks," *IACR Trans. Symmetric Cryptol.*, vol. 2017, pp. 130–168, 2017.

[15] A. Halbutogullari and C. Koc, "Mastrovito multiplier for general irreducible polynomials." *IEEE Trans. Computers*, vol. 49, pp. 503–518, 01 2000.

[16] C. Paar, "A new architecture for a parallel finite field multiplier with low complexity based on composite fields," *Computers, IEEE Transactions on*, vol. 45, pp. 856 – 861, 08 1996.

[17] S. S. Kumar, T. J. Wollinger, and C. Paar, "Optimum digit serial $gf(2^m)$ multipliers for curve-based cryptography," *IEEE Transactions on Computers*, vol. 55, pp. 1306–1311, 2006.

[18] L. Song and K. K. Parhi, "Low-complexity modified mastrovito multipliers over finite fields GF(2^m)," 1999.

[19] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "A systematic evaluation of compact hardware implementations for the Rijndael S-Box," vol. 3376, 02 2005, pp. 323–333.

[20] R. Ueno, N. Homma, Y. Sugawara, Y. Nogami, and T. Aoki, "Highly efficient GF(2^8) inversion circuit based on redundant GF arithmetic and its application to AES design," in *IACR Cryptology ePrint Archive*, 2015.

[21] R. Peralta and J. Boyar, "Method of optimizing combinational circuits," Apr. 22 2014, uS Patent 8,707,224 B2.

[22] J. Boyar, M. Dworkin, R. Peralta, M. Turan, C. Calik, , and L. Brandao, "Circuit minimization work," http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html. Past collaborators include: Michael Bartock, Ramon Collazo, Magnus Find, Michael Fischer, Murat Cenk, Christopher Wood, Andrea Visconti, Chiara Schiavo, Holman Gao, Bruce Strackbein, Larry Bassham.

[23] S. Raghuraman, "Efficiency of Logic Minimization Techniques for Cryptographic Hardware Implementation," *Masters Thesis, Virginia Polytechnic Institute and State University*, 2019.

[24] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC implementation of the AES SBoxes," in *CT-RSA*, 2002.

[25] D. Canright, "A very compact s-box for aes," in *CHES*, 2005.

[26] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact rijndael hardware architecture with S-Box optimization," in *ASIACRYPT*, 2001.

[27] D. Canright, "A very compact Rijndael S-box," 2004. [Online]. Available: https://calhoun.nps.edu/handle/10945/791

[28] T. Sasao, "AND-EXOR expressions and their optimization," 01 1993.

[29] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," in *CHES*, 2002.

[30] "Tohoku university: Cryptographic Hardware Project," May 2015. [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/crypto/.

[31] S. Tillich, M. Feldhofer, and J. Großschädl, "Area, delay, and power characteristics of standard-cell implementations of the AES S-Box," in *SAMOS*, 2006.