INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 3, PP. 1-7
Manuscript received January 31, 2025; revised July 2025. doi: 10.24425/ijet.2025.153622

# Reducing Power of BLAKE3 implementations with dedicated FPGA resources

Jarosław Sugier

***Abstract*—The BLAKE cryptographic hash functions are efficiently expressed in software; however, their hardware implementations do not match the speed and power efficiency of alternative methods. This paper assesses a possible method of decreasing power consumption in BLAKE3 FPGA implementations by application of dedicated DSP resources for binary summations in place of standard adders realized in logic cells within the programmable array. The analysis considers various viable configurations of cipher realization: from the standard iterative architecture (featuring one round instance in hardware), to organizations with 2, 4, and 6-stage pipelining employed for high processing efficiency. The power results are generated by simulating operation of the designs after their full implementation in a Spartan-7 device. Substituting the standard adders configured in programmable fabric with 7 series DSP48E1 elements can significantly decrease the high dynamic power consumption that adversely affected the standard non-pipelined BLAKE3 implementation, but can also bring some disadvantages with regard to hardware size or speed. Moreover, it does not offer any improvement in highly pipelined architectures. In addition to exploring one approach for reducing power consumption of this particular cipher, the paper can also serve as another case study on improving FPGA implementation by leveraging specialized resources that would otherwise remain unused but are available in the used device.**

***Keywords*—cryptographic hash function; dynamic dissipated power; DSP slice**

## I. Introduction

MINIMIZING power consumption is a vital benefit in hardware implementations of a cryptographic algorithm. The BLAKE3 hash function – initially proposed for the SHA-3 contest ([1]-[3]) and still of active interest ([4]-[5]) – can be efficiently executed in software, yet its hardware implementations are neither as fast nor as power-efficient as those of alternative hashing methods. This study examines an approach for lowering power losses in an FPGA implementation of this algorithm which consists in utilizing dedicated DSP resources for binary additions intensively employed in the cipher transformations, in place of logic cells within the programmable array. In this practical study the BLAKE3 compression function is implemented in hardware using various architectures: starting from the standard iterative organization requiring one cipher round instantiated in hardware, high-throughput derivative cases are created by application of pipelining from 2 through 4 up to 6 stages. Such collection of architectures is implemented using DSP resources rather than conventional binary adders, and the resulting size, speed and power metrics are compared with similar studies of standard implementations referenced in [6] and [7]. The results are derived from simulations: once the designs are implemented in a Spartan-7 device, power estimates are generated by the implementation tools based on signal activity traces produced during timing simulation. The findings suggest that using 7 series DSP48E1 resources in place of the "in-array" adders is an effective method for decreasing high dynamic power which was a serious problem of the standard BLAKE3 implementations. Nevertheless, this approach can have adverse influence on other implementation parameters and yields little to no power improvement in highly pipelined designs.

The set of tested BLAKE3 architectures was adopted from [6], where iterative and pipelined organizations were assessed for their size and speed efficiency and demonstrated substantial improvements achieved by the third version of the cipher compared to its predecessors. In [7], the evaluation was broadened to include power analysis, revealing severe issues caused by excessively intense signal glitches generated by cipher transformations; it was shown that pipelining could significantly mitigate them, thereby reducing power losses. Further analysis of signal behavior and glitch origins was conducted in [8]. This paper, being an extended version of [9], explores another method to address this issue: substituting conventional carry-propagation adders in the programmable array cells (identified as a potential source and multiplier of the glitches) with specialized DSP resources positioned outside the array.

Organization of the paper is as follows. Chapter 2 provides a concise overview of the BLAKE3 algorithm and explains how dedicated DSP resources, available in the Spartan-7 device but often underutilized in cryptographic computations, can be effectively employed to replace the conventional in-array resources. Chapter 3 presents the implementation results of the redesigned BLAKE3 architectures, highlighting key differences in their speed and size characteristics. In Chapter 4, a detailed power analysis is conducted to evaluate the efficiency of the proposed solution. The findings are summarized in the last chapter, along with a discussion of the contributions made by this research.

Jarosław Sugier is with Wrocław University of Science and Technology, Poland, Faculty of Information and Telecommunication Technology, Department of Computer Engineering (e-mail: jaroslaw.sugier@pwr.edu.pl).

## II. The Algorithm and Possible Adaptation of DSP Logic

### A. The BLAKE Compression Function

Similar to many round-based ciphers, BLAKE3 processes data through a series of repetitive round transformations applied to a set of state words (Fig. 1). The BLAKE3 state **V** comprises 16 32-bit words ($v_0$ to $v_{15}$), totaling 512 bits. These words undergo $n_R = 7$ identical rounds ([3]). The core of the hardware implementation lies in the realization of the round transformation, as the surrounding logic primarily handles multiplexing and has minimal impact on overall performance and power consumption.

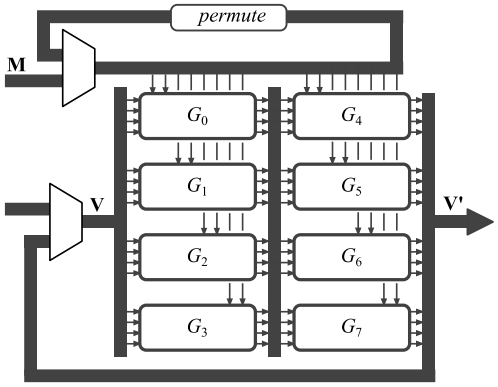


Fig. 1. Iterative organization of the BLAKE3 compression function with internals of round processing.

Within each round, the 16 state words undergo a sequence of two transformations using so called *G* function. Each *G* instance operates on four words (plus two 32-bit words $m_i$ from the message being hashed), necessitating four parallel instances for complete state processing. A round transformation comprises two cascades of four parallel *G* modules ($G_{0\text{-}3}$ and $G_{4\text{-}7}$). All BLAKE3 architectures considered in this paper require eight *G* instances. While internally identical, each instance distinguishes itself through a specific selection of state/input words ($v_i/m_i$); all state words are read by 4 parallel *G* modules while all message ones – by 8 such modules, i.e. in the whole round.

### B. Introducing the DSP Slices as 32-bit Adders

As identified in [8], the increased power consumption of BLAKE3 cores in FPGA implementations is primarily due to the excessive number of transient signal switching. These glitches, especially in the long propagation lines of the routing, lead to significant power losses. With an average of over 1000 glitches per clock cycle in the most unstable signals of the standard iterative architecture, this parameter remains substantially unchanged compared to BLAKE2, despite reduced size and faster speed. Moreover, such value significantly exceeds number of glitches observed in equivalent architectures of Keccak. A potential solution to this problem, as demonstrated in [7], is pipelining. By dividing the round processing into multiple stages (the study considered 2, 4, and 6 stages), glitches are prevented from propagating through the pipeline registers, effectively blocking their avalanche effect at the stage boundaries. While this approach often leads to a reduction in power consumption as a side effect besides improved overall throughput, as discussed in the literature (e.g., [10]-[12]), this work proposes a more direct method. We focus on reducing glitches at their source: the resources responsible for implementing the summation operations that appear among the cipher transformations.

Fig. 2 provides a detailed illustration of the internal operations within the *G* module. These operations encompass bitwise XOR, constant rotations (implemented efficiently through dedicated routing), and 32-bit arithmetic additions. Locations of 2- and 3-input adders in datapaths of the state words can be seen in the figure. Since the other elementary transformations, like bitwise XOR, do not generate transients, the carry-propagation adders were identified as the primary source of glitches in previous BLAKE3 implementations analyzed in [7]-[8].

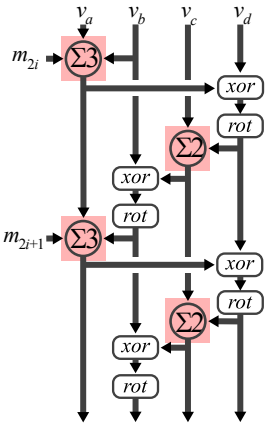


Fig. 2. Internal transformations of the *G* function with 2- and 3-input adders.

In FPGAs, apart from a look-up table (LUT – a rudimentary Boolean function generator), summations additionally employ two dedicated, simple primitives within each logic cell: a 2:1 multiplexer and a 2-input XOR gate. All these primitives together create a 1-bit full adder (one per logic cell) which, thanks to hardwired routing, feature a very fast carry path. In the context of the BLAKE3 algorithm, a cascade of 32 such adders are required for one 2-input summation. While these adders guarantee sufficient speed, the serial nature of 32-level carry propagation can introduces numerous transients on their outputs before settling the final result. The situation gets even worse if the input arguments are unstable themselves because in this case their glitches can be multiplicated in an avalanche effect. In a standard, unpipelined iterative architecture a state word goes through a combinatorial path including a cascade of two *G* modules so in the worst case passes by 8 adders in a single clock period ($v_a$ path in Fig. 2 with two 3-input adders, each constructed from two 2-input ones).

Thus the primary objective of this study is to supplant the error-prone, in-array adders with auxiliary resources positioned outside of the configurable array. Commencing with the Xilinx's Virtex-II family of FPGA which debuted over 20 years ago, these devices incorporate dedicated hardwired cores – known as DSP slices ([13]) – optimized for "multiply and accumulate" computations which are abundant in digital signal processing. Each slice, as illustrated in the upper portion of Fig. 3, can be configured to employ a pre-adder, a full 25×18 multiplier, and a 48-bit ALU executing addition, subtraction,

accumulation, or logical operations. The resources are located in the device in vertical columns, are interconnected within each column by rapid dedicated routing segments (PCOUT to PCIN) and can be connected to logic cells in the surrounding programmable fabric. These cores are very proficient in calculations of e.g. complex FIR filters but the application considered here needs only a small portion of their capabilities: bypassing all initial preprocessing, the required summation can be performed within the ALU. As depicted in Fig. 2, every instance of the *G* function applies two 2-input adders (each in a separate DSP slice) plus two 3-input adders (each would occupy two contiguous DSP slices, as shown in the lower portion of Fig. 3). Consequently, one *G* function calls for 6 DSP slices, and the entire round instance needs a total of 48. This amount is valid for all architectures considered in this paper, whether pipelined or not.

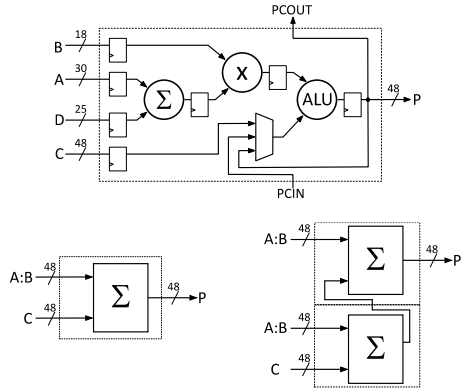


Fig. 3. DSP48E1 slice structure (top) and its configuration as BLAKE3 adders (bottom).

## III. IMPLEMENTATIONS

### A. The Architectures and Their FPGA Implementations

To maintain consistency with the preceding studies, the application of DSP resources was tested in six architectures of the BLAKE3 compression function:

- standard iterative organization (X1): a single, unpipelined round instance, producing output after $n_R = 7$ clock cycles;
- two-stage pipelined organization (P2): separates processing of $G_{0\div3}$ and $G_{4\div7}$ modules, generating two hashes in 14 clock cycles;
- four-stage pipelined organization (P4): introduces additional stage boundary within the *G* instances located in front of the second 3-input adder in the $v_a$ path (Fig. 2), and produces 4 hashes every 28 clock cycles;
- six-stage pipelined organizations (P6a÷c) computing a group of six hashes in 42 clock cycles.

The three P6*x* variants were replicated subsequent to their initial introduction in [6]. The first variant, P6a, evenly distributes the six adders across the three pipeline stages (2+2+2), which results in the division of the second 3-input adder in the $v_a$ path. To circumvent this division, the other two variants retain the 3-input adder in the middle pipeline segment and apply alternative distributions in the other two ones: 2+3+1 (P6b) or 3+2+1 (P6c). For further details concerning these designs, please refer to [6].

All six architectures required a single round logic instance and utilized a total of 48 DSP slices to accommodate the 32-bit state word summations. Among the Spartan-7 family ([14]), the XC7S25, in the csga225 package and -2 speed grade, was selected as the smallest device with sufficient DSP resources to meet this requirement. To maintain consistency with previous methodologies (and to make power simulation of the designs realistic), basic serial-in, parallel-out circuitry was added to all six cipher modules. This approach was necessary to ensure a reasonable I/O pin count because a completely-ported hashing unit would have required 1152 pins. In this way complete and functional designs were created and these were implemented using Vivado 2023.2 tools.

TABLE I
SIZE AND SPEED CHARACTERISTICS OF THE IMPLEMENTATIONS

| | X1 | P2 | P4 | P6a | P6b | P6c |
|---|---|---|---|---|---|---|
| Slices | 827 | 831 | 1107 | 1227 | 1234 | 1263 |
| LUTs | 1693 | 1397 | 1717 | 2306 | 2280 | 2264 |
| Registers | 2185 | 3206 | 4743 | 5854 | 5768 | 5768 |
| DSP48E1 | 48 | 48 | 48 | 48 | 48 | 48 |
| Max $F_{clk}$ [MHz] | 29.2 | 57.8 | 110.8 | 145.0 | 124.2 | 125.2 |
| Target $F_{clk}$ [MHz] | 28.0 | 55.0 | 100.0 | 133.0 | 115.0 | 115.0 |
| Actual $F_{clk}$ [MHz] | 28.1 | 51.4 | 104.7 | 135.2 | 116.3 | 118.9 |
| Route delay | 39% | 42% | 42% | 45% | 35% | 34% |
| Logic levels | 21 | 11 | 5 | 3 | 4 | 4 |
| (incl. DSP) | (12) | (6) | (3) | (2) | (3) | (3) |

Following the same approach as in [7], two types of studies were conducted. First, the maximum operational frequency of each architecture was determined, and only then each design was implemented with a target frequency reduced by a certain margin in order to guarantee stable operation. Such implementations were then subjected to power analysis. Table I provides results of these implementations, including:

- resource utilization: number of occupied logic slices, LUTs, and registers;
- number of occupied DSP elements (just confirming that 48 of them were used across all cases);
- speed efficiency: maximum operational frequency, requested and achieved frequencies for power analysis, and longest path parameters (routing delay percentage and count of logic levels, including DSP slices).

### B. Evaluation of Speed and Size Efficiency

Before delving into power analysis, it's worth mentioning the notable changes in implementation efficiency which resulted from the relocation of summation logic to the DSP resources. Fig. 4 graphically presents the modifications observed in speed and size in comparison to [7], i.e. the implementations based on conventional adders.

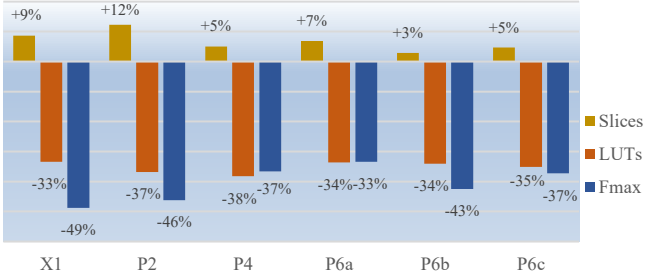


Fig. 4. Changes in slice, LUT counts, and maximum frequency due to DSP-based summations.

Considering the size characteristics (LUT occupancy and slice utilization), it is evident that the modification primarily affected logic density. Although the LUT amount was reduced remarkably (by 33-38%), slice occupancy remained almost unchanged or slightly increased. This resulted in a significant fall in logic density measured as an average number of LUT in one slice – this ratio decreased from 2.95 to 1.80. The reduced logic density in the programmable array further indicate elongated routing channels, adversely impacting power and speed efficiency. The consequences of this observation to power consumption will be examined later, here – as for speed efficiency – one should note in Fig. 4 severe degradation in maximum operating frequencies which declined by 33-49% across all the architectures.

To comprehend this phenomenon, let's examine Fig. 5 which depicts the geometric placement of occupied resources over the FPGA array in, taken as an example, the iterative architecture. The novel implementation, illustrated on the left, necessitates the utilization of 48 of 60 available DSP elements. Because these elements are arranged in two columns within the array as marked in the figure, the placement tool endeavored to distribute the whole design equitably along both of them, and this resulted in a substantial vertical footprint, spanning 60% of the matrix height. In comparison to a more concentrated standard design (shown in the right part) which did not have to be placed close to the DSP slices, this dispersion contributed to a much roomier distribution thus longer interconnection lines. It should be also noted that this wider layout was observed in designs which used significantly less resources in the configurable blocks, after transferring all the summation logic out of them to the DSP resources.

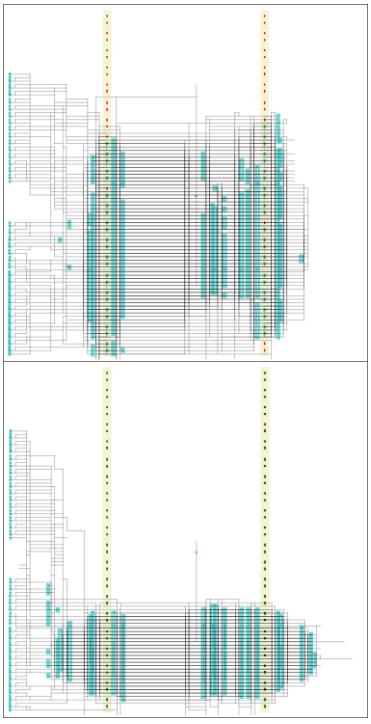

Fig. 5. The programmable array with DSP-based X1 core (above) vs. standard implementation (below).

## IV. Power Calculations

### A. Method of Computation

To ensure the accuracy of power estimates, post-route simulations were conducted on the fully implemented designs to capture activity of the internal signal. The resulting data was recorded in SAIF files for subsequent power estimation procedures ([15]). Given the limitations of the tools which could only measure an average power consumption over particular period of simulation time, dedicated test scenarios and input stimula were developed to maintain the cipher cores in a fully active state throughout the simulation period. These scenarios guaranteed continuous data processing, avoiding idle cycles and ensuring a consistent workload for all pipeline streams running concurrently in the cores ([8]).

Table II presents the calculated power consumption for all six architectures operating at their nominal frequencies. The analysis focuses exclusively on dynamic power which – apart from clock frequency – is determined only by size and complexity of the circuit in silicone. Static power, dependent on factors such as idle matrix characteristics and external conditions (package thermal resistivity, cooling efficiency, fluctuations of supply voltages, etc.), is excluded from this analysis[1]. The table provides both the overall (dynamic) power and its breakdown by resource type: clock network, slice logic, DSP resources, propagation segments, and I/O. The last two rows include additional synthetic parameters calculated from the above.

TABLE II
DYNAMIC POWER ESTIMATED FOR MODULES OPERATING AT THEIR NOMINAL FREQUENCIES

| | X1 | P2 | P4 | P6a | P6b | P6c |
|---|---|---|---|---|---|---|
| $F_{CLK}$ [MHz] | 28 | 55 | 100 | 133 | 115 | 115 |
| Dynamic power [mW] | 579 | 334 | 286 | 331 | 287 | 296 |
| Clocks | 2 | 7 | 18 | 27 | 23 | 24 |
| Slice Logic, incl.: | 92 | 42 | 34 | 39 | 33 | 30 |
| LUT | 92 | 41 | 31 | 33 | 29 | 24 |
| Register | 0 | 1 | 3 | 6 | 4 | 4 |
| DSP | 167 | 109 | 96 | 112 | 104 | 103 |
| Routing | 317 | 173 | 133 | 149 | 121 | 134 |
| I/O | 1 | 2 | 4 | 4 | 4 | 4 |
| $P_{MHz}$ [mW/MHz] | 20.7 | 6.1 | 2.86 | 2.49 | 2.49 | 2.57 |
| $E_h$ [nJ/hash] | 144.7 | 42.6 | 20.0 | 17.4 | 17.4 | 18.0 |

After the implementation, the power analyses of all architectures were conducted also for various clock frequencies. The results included in [9] proved that, like in the standard implementations of [7], the total dynamic power was a linear function of $F_{CLK}$ – which allowed to describe this function with a value of proportionality factor $P_{MHz}$, given in Table II in mW per MHz. Consequently, this also made possible to express energy consumed by a single hash calculation independently from particular clock speed; this parameter is denoted as $E_h$ and given in nJ per hash in the last row. The value of $E_h$ can also be understood as the power dissipated per unit hashing speed, with nJ/hash corresponding to Watt per million of hashes per second (Mhps). This parameter is the optimal, cumulative characteristic which

[1] While this approach is valid for comparison of power requirements of various BLAKE3 implementations – which is the purpose of this work – a simple practical task to calculate the device's operating temperature would have to consider the static component as well.

describes the effective power requirement of a hashing hardware under consideration, and is best for comparisons.

### B. Evaluation of Signal Activity

The SAIF files, which were a foundation for estimation of the power losses, can furthermore be examined in order to get an insight into level of signal activity in the FPGA fabric. They include all signals operating in the array after its configuration, and for each one provide, among others, total number of transitions over the whole simulation period. From this parameter, knowing the clock frequency appearing in the relevant stimulus and overall simulation scenario (e.g. lengths of initialization vs. activity periods), one can calculate an average number of signal switches per one clock tick during hash calculation. Fig. 6 shows a mean value of these numbers in 5% of the most active signals in each architecture which will be denoted as $g_{CK}$; this parameter is the best illustration of signal glitches across different organizations of the hardware.

In general, the graph confirms that pipelining is an effective way – apart from its own purposes – of glitch reduction also in the new BLAKE3 implementations. Like it was analyzed in [7] and [8], this results from two factors. Firstly, division of long combinatorial paths into shorter pipeline segments reduces levels of logic the signals must traverse, hence it is less likely that propagation times of any two of them arriving at one LUT generator will be different (which is a central mechanism of glitch formation). Secondly, any glitches already generated do not propagate through pipeline registers until a stable value settles at the end of the clock period, and this prevents their avalanche multiplication.

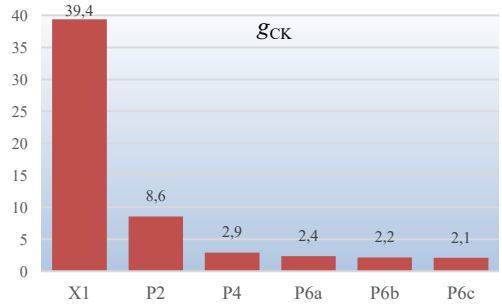


Fig. 6. Average number of transitions in one clock period in 5% of the most active signals ($g_{CLK}$ parameter).

Nevertheless, now this reduction is notably less intense than it was measured in standard implementations of [8]: while previously splitting X1 paths into two pipeline stages resulted in $g_{CK}$ param divided by as much as 29, this time it is only by 4.6. Correspondingly, transition from P2 to P4 now reduces $g_{CK}$ by 2.9 instead of 6.6. On the other hand, it is also worth noting the overall small difference brought by transition to P6*x* architectures: the reduction of P4's value is only by 1.3 which is a rather minor improvement.

Finally, let's examine changes in the $g_{CK}$ parameter after introduction of DSP slices within each architecture, which is presented in Fig. 7. The foremost observation which should be made here is as follows: while the dedicated adders indeed improve signal stability – in the extreme case of the X1 architecture the glitch per clock value is reduced to a mere 3% – this positive effect diminishes with increasing length of the pipeline. Although the trend looks similar to the one seen in Fig. 6, these are actually two independent phenomena.

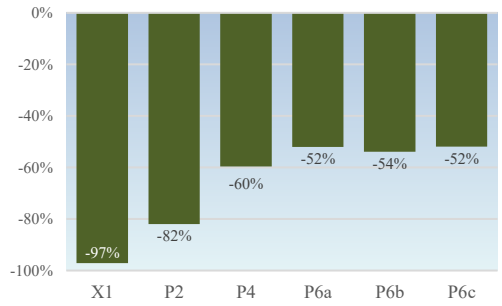


Fig. 7. Reductions in the values of $g_{CK}$ param after application of DSP adders.

The extraordinary improvement seen in the X1 case is reduced nearly by half in the P6*x* cases and is proportionally smaller in the P2 and P4 organizations. Nevertheless, looking at this sole parameter, replacing standard in-array adders with dedicated DSP slices in the X1 architecture improves signal stability as much as application of 2-stage pipelining described in [8].

### C. Analysis of Power Losses

Fig. 8 illustrates the power consumption split into various types of the FPGA components. In contrast to previous findings, with DSP elements now replacing carry logic, they account for 29-38% of the total power. By offloading adder operations to DSP, LUT stress was substantially reduced, with their share decreasing from 28÷32% to 9÷16%. This underscores the significant reliance on LUTs in standard adder implementations of prior designs and, generally, the arithmetically intensive nature of BLAKE processing. While routing remains the primary single generator of losses, for the first time combined logic (LUT+DSP) emerges as the dominant power-consuming constituent in most architectures (with the only exception of X1). This emphasizes the critical role which the DSP adders has now taken in power distribution.

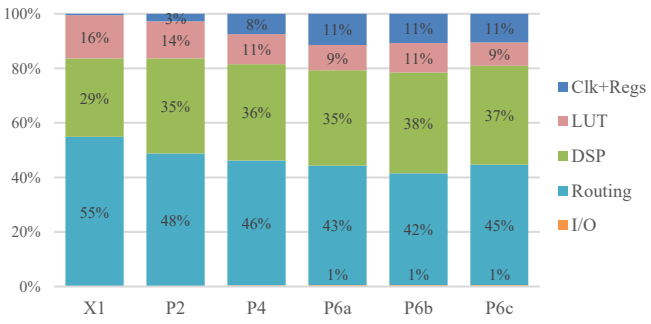


Fig. 8. Breakdown of power losses by FPGA resource.

Fig. 9 depicts variations in the $E_h$ metrics across the architectures. While X1 remains the most power-intensive, now the gap between X1 and P2 has narrowed significantly compared to the standard implementations: the current ratio is 7:2, down from approximately 20:1. This reduction indicates that introduction of DSP elements indeed can improve power predictability for the X1 architecture, even without application of pipelining.

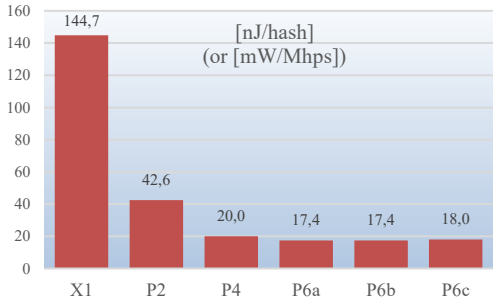


Fig. 9. Energy per hash $E_h$ across all the architectures implementing DSP additions.

While pipelining can still provide advantages in terms of power reduction, the gains are diminishing as the number of pipeline stages increases. This is especially evident when comparing the P4 and P6*x* architectures: while switching from P2 to P4 reduces $E_h$ by more than a half, the next step to P6 brings decrease by approx. 1/10. This drop is even smaller than the one observed in the $g_{CK}$ parameter. The saturation point for the power benefits of pipelining seems to be reached sooner than in the standard implementations: now it is worth considering whether the extra power savings brought by going from 4 to 6 pipeline stages justify the increased latency in producing the results and more complex organization of data flow in 6 parallel streams.

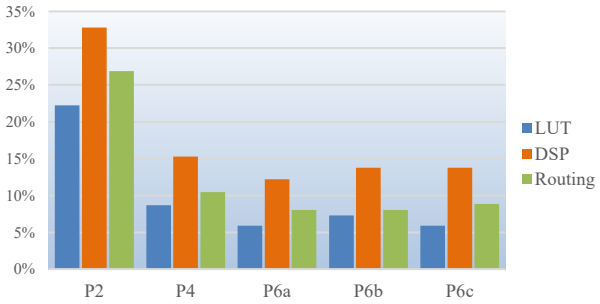


Fig. 10. Power losses in the main types of components in the pipelined architectures as percentages of values form the X1 case.

To explain further effects of pipelining, Fig. 10 illustrates the dynamic power consumption of the primary FPGA components – LUTs, DSPs, and routing – as a percentage of the X1 baseline. The data demonstrate that the power savings associated with increasing pipeline length is marginally more pronounced in the routing as compared to the logic elements. When examining the individual components within each architecture, the most significant power reductions are observed in the LUT generators, while the DSP adders exhibit the least drop. This suggests that although now summations do not constitute the primary source of power dissipation, their inherent internal processing also results in intensive signal switching. Consequently, pipelining offers limited benefits in mitigating their power consumption. The remaining BLAKE logic which is implemented in LUT generators, along with the routing fabric, exhibit more substantial power reductions due to the suppression of glitches at the pipeline registers.

### *D. Comparison with the Conventional Approach*

Fig. 11 assesses the impact of DSP utilization on the synthetic $E_h$ metric. The results demonstrate excellent effectiveness of this modification in the X1 organization, where this parameter is lowered by tenfold, i.e. to 9%. This result positively verifies the hypothesis which was the starting point of this work: employing dedicated summation resources in place of standard in-array adders indeed can enhance signal stability and significantly reduce power consumption. However, results of the other architectures indicate that this positive effect is offset by additional factors that become more pronounced with longer pipelines: with the P2 organization still accomplishing a significant reduction by 41%, the extreme condition is observed in the P6*x* cases which experience an increase in the $E_h$ metric instead of decrease. The P4 architecture, on the other hand, maintains the original result, reflecting a near-perfect balance between the positive and negative effects of the modification.

To elucidate this counter-effect, Fig. 12 presents a comparative analysis of changes in individual $E_h$ components, categorized into three primary groups: combinational logic (LUT generators plus either DSP or carry elements), routing, and miscellaneous resources (I/O blocks, registers, and clock networks). Despite the reduction in glitches, the power consumption of logic components has not decreased as significantly with pipelining as in previous implementations. This suggests that DSP sites, with their intricate 48-bit internal architecture and substantial computational capacity, are less power-efficient. Looking at Fig. 12, reductions in power dissipated by logic components (hence also by DSP elements) are observed only in the X1 and P2 cases; the P2 experienced a growth (instead of decrease) by 27%, while the P6*x* cases – by 51% to 69%, all compared to conventional implementations. Such notable increases instead of expected reductions explain the growths seen in $E_h$ values of Fig. 11.

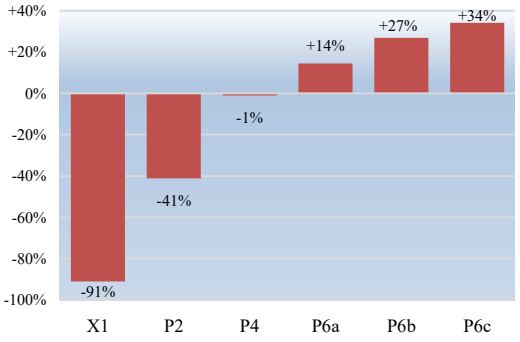


Fig. 11. Changes in the unit hash energy ($E_h$) after introducing DSP adders.

While the routing component also exhibits a similar trend, the impact is less pronounced, with relatively minor increases of 3% and 18% observed only in the final two P6*x* modules. This is despite the significant increase in connection length in the new designs, as discussed in relation to Fig. 5. This confirms that the primary benefit of introducing the DSP adders is in elimination of transient signal instabilities which leads to reduced power losses despite much longer transmission lines. However, the widespread design placement depicted in Fig. 5 has led to increased power losses in long clock distribution lines. The "Other" component, whose value primarily encompasses clocking losses, is elevated in all six architectures, regardless of the pipelining level.

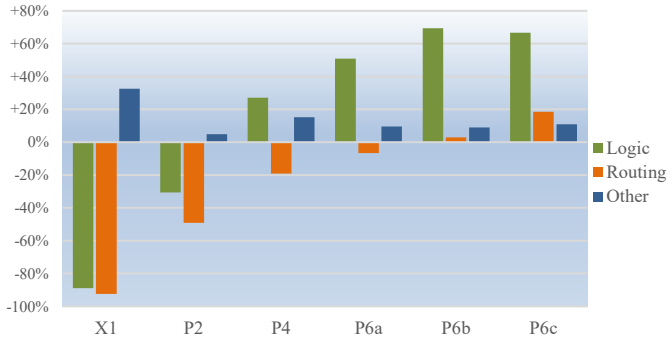


Fig. 12. Changes in $E_h$ components triggered by application of DSP adders.

## CONCLUSIONS

This research offered a novel approach to optimizing power consumption within FPGA implementations of the BLAKE3 cryptographic unit. The proposed methodology involves replacing 32-bit adders implemented in logic cells of the configurable array with dedicated summation units residing in DSP slices. Our findings demonstrated that this approach can significantly enhance power efficiency, particularly in the standard iterative implementation which was characterized by excessive transient glitches arising from sequential nature of carry calculation in traditional adders. Looking just at signal stability, indeed this approach offers an improvement which previously was achieved only by application of 2-stage pipelining.

While DSP elements are optimized for computational performance, they do not exhibit the same level of power efficiency. Consequently, the advantages they offer can be offset in pipelined architectures where glitches are already effectively mitigated by stage registers. The results indicated that the overall energy consumed per one hash calculation, a standardized metric for comparing architectures independently from operating frequencies, is not improved in designs with four pipeline stages and deteriorates in designs with six stages.

The study also indicated that the DSP elements were internally too intricate for the needs of this specific problem and did not align with the significantly higher density of the remaining logic. While the BLAKE3 round necessitates 48 adders, this amount of DSP slices alone occupied 60% of the total matrix height, leading to extended transmission lines whereas the entire cipher circuitry could be substantially more compact. The elongated design layout resulting from this approach hindered achievable speed, as evidenced by a reduction in maximum clock frequency by approximately one-third. Still, on the other hand, this method demonstrated remarkable and unquestionable power advantage in iterative or two-stage pipelined architectures.

While this study offers a practical method for optimizing power consumption in BLAKE3 hardware implementations, from a broader perspective it can also be treated as an illustration of how an FPGA project can be enhanced by effectively leveraging specialized components, such as DSP slices, that are often underutilized. By carefully considering specific requirements of a given application and capabilities of all resources available in the selected FPGA device, designers can achieve noteworthy improvements in performance, power efficiency, or overall system organization.

## REFERENCES

[1] J.P. Aumasson, L. Henzen, W. Meier, R.C.-W. Phan “SHA-3 proposal BLAKE, version 1.3”, https://www.aumasson.jp/blake/blake.pdf, 2010, last accessed Oct. 2024.

[2] J.P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, C. Winnerlein “BLAKE2: Simpler, Smaller, Fast as MD5”, in M. Jacobson, M. Locasto, P. Mohassel, R. Safavi-Naini (eds) Applied Cryptography and Network Security 2013, Lecture Notes in Computer Science, vol. 7954, pp. 119-135, Springer, Berlin-Heidelberg, 2013. https://doi.org/10.1007/978-3-642-38980-1_8

[3] J. O’Connor, J.P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, “BLAKE3: one function, fast everywhere”, Real World Crypto 2020 (lightning talk), available at https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf, last accessed Oct. 2024.

[4] I.T. Ciocan, E.A. Kelesidis, D. Maimuț, L. Morogan, “A Modified Argon2i Using a Tweaked Variant of Blake3”, in Proc. 2021 26th IEEE Asia-Pacific Conference on Communications (APCC), Kuala Lumpur, pp. 271-274, IEEE Xplore, 2021. https://doi.org/10.1109/APCC49754.2021. 9609933

[5] S. Sinha, S. Anand, K.P. K, “Improving Smart Contract Transaction Performance in Hyperledger Fabric”, in Proc. 2021 Emerging Trends in Industry 4.0 (ETI 4.0), pp. 1-6, IEEE Xplore, 2021. https://doi.org/10.1109/ETI4.051663.2021.9619202

[6] J. Sugier, “FPGA Implementations of BLAKE3 Compression Function with Intra-Round Pipelining”, in W. Zamojski et al. (eds) New Advances in Dependability of Networks and Systems, Lecture Notes in Networks and Systems, vol. 484, pp. 319-330, Springer, Cham, 2022. https://doi.org/10.1007/978-3-031-06746-4_31

[7] J. Sugier, “Power Analysis of BLAKE3 Pipelined Implementations in FPGA Devices”, in W. Zamojski et al. (eds) Dependable Computer Systems and Networks, Lecture Notes in Networks and Systems, vol. 737, pp. 295-308, Springer, Cham, 2023. https://doi.org/10.1007/978-3-031-37720-4_27

[8] J. Sugier, “Comparison of power consumption in pipelined implementations of the BLAKE3 cipher in FPGA devices”, Int. J. of Electronics and Telecommunications, vol. 70, no. 1, pp. 23-30, 2017. https://doi.org/10.24425/ijet.2023.147710

[9] J. Sugier. “Dedicated FPGA Resources in Improving Power Efficiency of Implementations of BLAKE3 Hash Function”, in W. Zamojski et al. (eds) System Dependability – Theory and Applications, Lecture Notes in Networks and Systems, vol. 1026, pp. 283–295, Springer, Cham, 2024. https://doi.org/10.1007/978-3-031-61857-4_28

[10] E. Boemo, J. Oliver, G. Caffarena, “Tracking the pipelining-power rule along the FPGA technical literature”, in Proc. 10th FPGAworld Conference, FPGAworld, 2013. https://doi.org/10.1145/2513683.2513692

[11] N. Grover, M.K. Soni, “Reduction of Power Consumption in FPGAs - An Overview”, International Journal of Information Engineering and Electronic Business, vol. 4, no. 5, pp. 50-69, 2012. https://doi.org/10.5815/ijieeb.2012.05.07

[12] S.J.E. Wilton, S.S. Ang, W. Luk, “The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays”, in J. Becker, M. Platzner, S. Vernalde (eds) Field Programmable Logic and Application 2004, Lecture Notes in Computer Science, vol 3203. Springer, Berlin-Heidelberg, 2004. https://doi.org/10.1007/978-3-540-30117-2_73

[13] Advanced Micro Devices, Inc., “7 Series DSP481 Slice: User Guide”, UG479.PDF available at www.amd.com, last accessed Sept. 2024.

[14] Advanced Micro Devices, Inc., “7 Series FPGAs Data Sheet: Overview”, DS180.PDF available at www.amd.com, last accessed Sept. 2024.

[15] Advanced Micro Devices, Inc., “Vivado Design Suite User Guide: Power Analysis and Optimization”, UG907.PDF available at www.amd.com, last accessed Sept. 2024.