INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 1, PP. 23-30
Manuscript received November 12, 2023; revised March, 2024. doi: 10.24425/ijet.2023.147710

Comparison of power consumption in pipelined implementations of the BLAKE3 cipher in FPGA devices

Jarosław Sugier

***Abstract*—This article analyzes the dynamic power losses generated by various hardware implementations of the BLAKE3 hash function. Estimations of the parameters were based on the results of post-route simulations of designs implemented in Xilinx Spartan-7 FPGAs. The algorithm was tested in various hardware organizations: based on a standard iterative architecture with one round instance in the programmable array, various derived versions with pipeline processing were elaborated, which ultimately led to a set of 6 architectural variants of the cipher, from the iterative case (without pipeline) to one with maximum of 6 pipeline stages. Moreover, the results obtained for the iterative architecture were compared with analogous implementations of the BLAKE2 (direct predecessor) and KECCAK (the foundation of the current SHA-3 standard) algorithms. This case study illustrates the differences (or lack thereof) in the power requirements of these three hash functions when they are implemented on an FPGA platform, and illustrate the significant savings that can be achieved by introducing pipeline to the processing of the BLAKE round.**

***Keywords*—cryptographic hash function; KECCAK; SHA-3; dynamic dissipated power**

I. INTRODUCTION

CRYPTOGRAPHIC hash functions are frequently used in today's IT security infrastructure for tasks as diverse as key generation, digital signatures and authentication schemes, or the construction of stream ciphers. In 2007, considering the latest advances in cryptanalysis of that time, the National Institute of Standards and Technology called for the development of a new standard for SHA hash functions, with a completely fresh approach to construction of the algorithm. In the announced open competition, a total of 14 proposals were accepted for analysis, and many currently used solutions (including all three algorithms considered in this article) originated from this group. During evaluation of the candidates, effective – i.e. fast or fast relatively to the size – hardware implementation was an important analysis aspect and was one of the main evaluation criteria (see, e.g., [1]). However, the analysis of power consumption, being a characteristic which depends on many mutable and subjectively selected factors (hardware platform, technology process of the implementation device, working environment, etc.), was not easy in objective assessment and not always was discussed in detail.

This paper presents a case study in which various hardware implementations of the BLAKE3 cipher – the latest variant of one of the main finalists of the NIST competition – were implemented in an FPGA device, optionally with different pipelined organizations. The estimation of power losses was carried out by implementation tools using the most accurate method based on a precise analysis of the activity of internal signals, obtained by simulating the final system, fully placed and routed in the programmable array. For the case study commercial devices from the Sprtan-7 family from Xilinx were selected as the hardware platform. The test range consisted of a total of 6 variants of the BLAKE3 algorithm – from standard iterative organization to pipelined architectures with a maximum of 6 pipeline stages – which were additionally supplemented with iterative implementations of the BLAKE2 and Keccak algorithms. The final test set of 8 complete systems was implemented on the same hardware platform and with the same tools. The obtained results show the fundamental impact that the application of pipelining has on power losses in the BLAKE3 algorithm, and also compare its iterative version with its direct predecessor (BLAKE2) and the core of the SHA-3 standard (KECCAK).

A similar set of algorithms and their hardware implementations was presented in [2], where the subject of the analysis was their processing speed and the maximum throughput. In [3], these results were extended to include the aspect of energy efficiency, and this paper is a follow-up of that work with an elaborated discussion of the significant power reductions that are achieved in the BLAKE algorithm by using intra-round pipeline processing.

The text is organized as follows. The structures of the three algorithms and specific challenges of their implementation in hardware are the subject of Chapter 2. The next chapter – No. 3 – describes the research methodology and numerical results obtained for all 8 projects under analysis, both in terms of processing efficiency and – what is the main focus of this work – power consumption. The analysis of the results is the subject of Chapter 4 and includes a discussion of the effects of using the pipeline in various variants of BLAKE3 architectures, as well as a comparison of this algorithm with its predecessor and the SHA-3 core.

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

II. Hardware Implementations of the Three Algorithms

A. The Ciphers

The BLAKE hash function [4] was submitted as one of the candidates in the NIST competition and, after initial evaluation, was qualified for the second round along with four other proposals. Although it was not ultimately chosen as the basis for the new SHA standard losing to the KECCAK algorithm, it was highly rated during the evaluation due to its high cryptographic strength with great potential for fast software implementations. In particular, its design made very good use of the possibilities offered by the dedicated assembly instructions of contemporary microprocessors. Because of these advantages, and also because of the relatively slow operation of the new SHA-3 standard in software, it quickly attracted attention of the cryptographic community which was always looking for a new, strong and fast algorithm to implement in cryptographic libraries. Still during the competition, in 2012 the authors amended their original proposal by announcing the second revision of the algorithm in [5] and this version – called BLAKE2 – found applications in cryptographic data protection, e.g. in the RAR archive format, in the Linux kernel, and in the specialized key derivation systems. Due to its popularity, the authors did not lose their interest in the algorithm and in 2020 announced the BLAKE3 specification in [6]. The latest modification further increased hashing speed by simplifying the processing but without reductions in its cryptographic efficiency. This version began to be implemented quite quickly as one of the available hashing methods, e.g. in blockchain transaction protocols ([7]) or in automatic key generation ([8]).

Both versions of the BLAKE algorithm which are considered in this article use a very similar internal round organization, as illustrated in Fig. 1 which shows the complete data flow in a standard iterative architecture taken as an example. In general, the task of the round is to transform the 512-bit state V, organized in 16 words $v_{0\div15}$, with the additional use of words of the encoded message M, $m_{0\div15}$ (all words are 32-bit). The actual computations are performed in 8 instances of the so-called *quarter rounds* $G_0 \div G_7$, each calculating new values of four words v using also two words m. The set of elementary transformations applied to the words include arithmetic addition, bitwise XOR and rotations. In each round, state V goes through a two-stage cascade of 4 G instances, which, working in parallel, process a complete set of 16 words v.

For the analysis of this article, the most important are two differences that BLAKE3 introduced to its predecessor: (a) the number of round repetitions N_R was reduced from 10 to 7, and (b) a new method was proposed to select words m which are passed to individual G modules. The first of the above modifications obviously reduced the calculation time and increased the effective throughput of the hardware (thus also decreased the total energy needed to complete the calculations), but did not have a noticeable impact on the organization of the implementation in the programmable array (apart from trivial modifications to the control system which counts the execution of rounds). The second modification, presented quite briefly by the authors in the specification [6], had more far-reaching consequences for the organization of the hardware.

Whereas in BLAKE2 a set of 10 permutations σ_r decided which m words were to be used by each G module (and a different one had to be applied in subsequent iterations of the round), in the third version of the algorithm the assignment of m words to each G instance is fixed, while between rounds the words are permuted among themselves according to one and the same scheme. The consequences of this change in hardware are significant: while in version 2 it was necessary to implement expensive 16:1 multiplexers with a width of 32 bits (two for each G instance) to execute a specific permutation depending on the round number, in BLAKE3 only one constant permutation of the words remained and, in order to implement it between the rounds, it can be hard-coded in routing which reload m registers, i.e. without involving any logical resources. The hardware implementation of 16 multiplexers, each with 16 inputs and 32-bit wide paths, created problems already discussed in [9] (where it was considered to replace them with RAM blocks storing the m words), and, as the results in [2] show, their elimination can lead to over 40% reduction in the FPGA array occupancy.

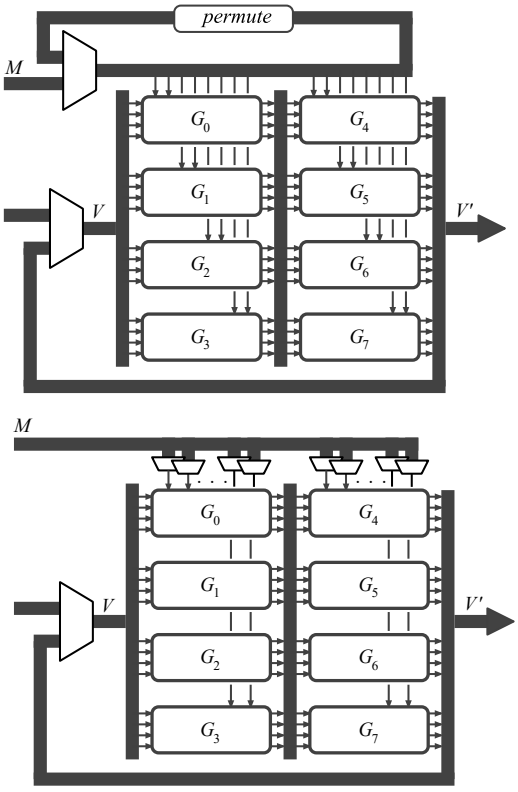

Fig. 1. Datapaths in BLAKE3 (above) and BLAKE2 (below) iterative processing

Compared to these nuances, the internal organization of the KECCAK compression function ([10],[11]) is relatively simple: its large, 1600-bit state is organized as a 5×5 array of words, each 64-bit, and the operations of the round apply 5 specific elementary transformations N_R = 24 times. Because the transformations transcode different fragments of the state (columns, rows or planes inside a 5×5×64-bit cuboid), the data flow cannot be illustrated with a diagram as simple as the ones in Fig. 1. A detailed analysis of the proposed implementation of the algorithm on the FPGA platform is described in [12], but it is worth noting one difference: since in BLAKE the message words m remain active as input parameters to the G modules in

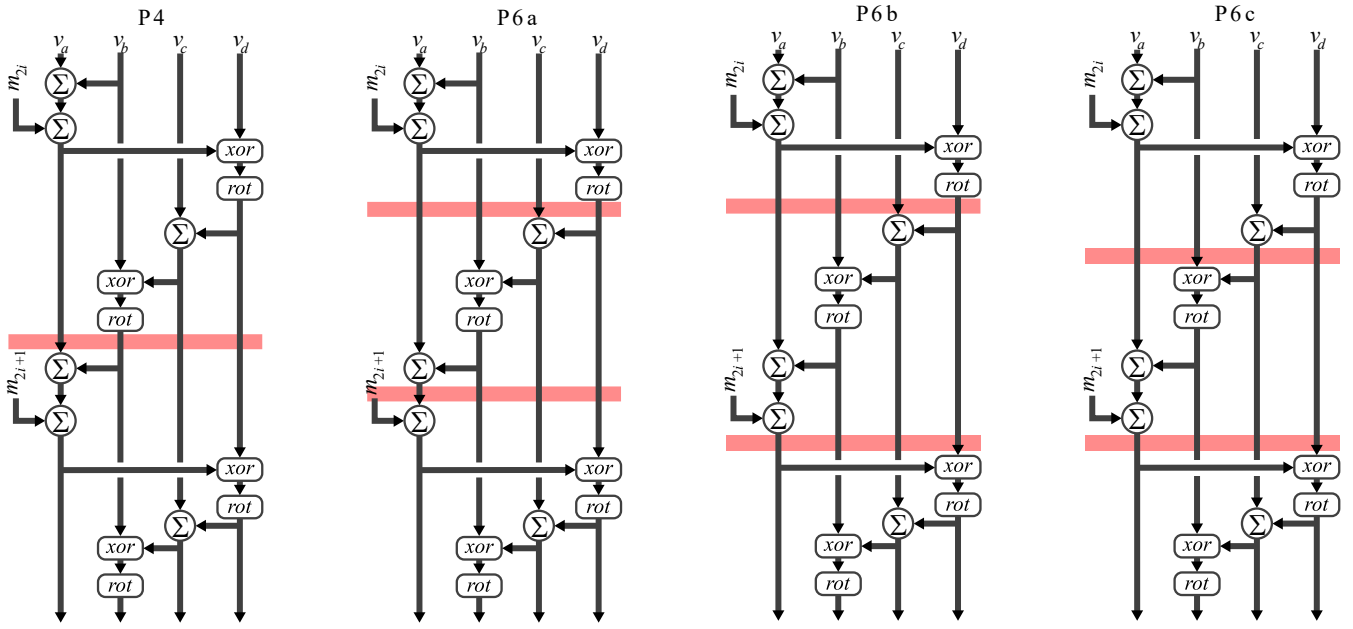

Fig. 2. Boundaries of pipeline stages inside the *G* modules

all rounds, it is necessary to store and distribute them through a separate, 512-bit path running parallel to the state (compare Fig. 1), while in the KECCAK the message bits are loaded only as the initial value of the state before the first round, after which their storage and distribution to the transformation logic are not needed.

B. Hardware Implementations

The analysis in this paper includes a total of 6 different hardware implementations of the BLAKE3 algorithm. The starting point was the standard iterative organization typical to any algorithm with repetitive round processing: the programmable array contains logic that performs operations of one round, and the state bits are passed through it N_R times. Derived variants were obtained by dividing the round logic into pipeline stages which process multiple data in parallel. The basic iterative architecture (without pipelining) will be denoted as X1, and the pipelined variants will be denoted as P*k*, where *k* is the number of pipeline stages within the round, and therefore also the number of parallel data streams that are concurrently hashed. The set of pipelined variants consisted of P2, P4, and three possible versions of P6. Next to them, the X1 implementations of all three algorithms (BLAKE versions 3 and 2 and KECCAK) were analyzed, which gave a total of 8 complete projects.

In particular, developing pipelined architectures for the BLAKE algorithm required decisions about the location of the pipeline stage boundaries within a round. It was natural to adapt as such a boundary the signals connecting the two halves of the 4+4 cascade of *G* modules (Fig. 1), which was all needed to create the P2 variant. Variants P4 and P6 required modifications in the inner structure of the *G* function, i.e. divisions of its logic into two or three stages. A discussion of possible options in this regard is included in [2], and the selected choices are presented in Fig. 2. While the division of *G* processing into two parts (arch. P4) did not raise any major doubts, the division into 3 levels (arch. P6) was not so obvious if one wanted to maintain an even distribution of elementary operations in the stages. In particular, if the goal was to divide the 6 adders defining the critical path in the proportions 2+2+2 (arch. P6a), it was necessary to separate their two instances appearing immediately one after another in the path of word v_a (see Fig. 2). This separation could make optimization of the two adders (merging their resources) impossible, so the test set included also variants P6b and P6c, in which the 2+3+1 and 3+2+1 division was used in order to keep adders within one pipeline stage.

III. CALCULATING THE POWER

A. Methodology

For each of the 8 projects, the evaluation of its power consumption consisted of three steps: development of a complete encryption unit and its implementation in the selected FPGA device, post-route simulation of the resulting system with a test vector forcing continuous operation of the unit with maximum load, and the final calculation of power parameters based on the results obtained in simulation from tracking the internal signals of the systems.

1) Implementation

All hardware modules representing the considered ciphers are defined as modules with wide parallel I/O ports whose total size may exceed a thousand bits. For example, the BLAKE3 unit requires reading 512 message bits plus 256 bits of chain hash value along with 128 bits with other parameters, and outputs the result as a 256-bit value – i.e. it requires transferring a total of 1152 bits through its ports. For power evaluation, it was crucial to implement each unit as a complete design, fully distributed in the FPGA array, with a reasonable number of I/O pins. To reduce the necessary number of outputs, the cryptographic cores were supplemented with a simple buffering system whose task was to iteratively load the input data (Serial-In Parallel-Out registers) and output the result (Parallel-In Serial-Out regs), both in 32-bit chunks. These buffers reduced pin requirements so that standard IC packages could be used, and although they consumed some registers they had a negligible impact on the

TABLE I
PARAMETERS OF THE CIPHERS AFTER IMPLEMENTATION

	X1	P2	P4	P6a	P6b	P6c	BLAKE2 (X1)	KECCAK (X1)
Slices	761	892	1054	1150	1199	1207	1247	1397
LUTs	2541	2607	2779	3473	3458	3491	4362	4623
Registers	2184	3206	4743	5804	5774	5783	2437	4815
MUXes	0	0	0	0	0	0	1536	0
Target F_{CLK} [MHz]	40.0	90.0	140.0	190.0	190.0	170.0	40.0	200.0
Actual F_{CLK} [MHz]	44.4	91.9	146.6	194.2	197.2	172.0	40.6	215.5
Route delay	56%	49%	51%	42%	54%	52%	56%	88%
Logic levels	45	23	14	12	10	11	45	1
(inlc. Carry)	(33)	(18)	(10)	(9)	(7)	(8)	(30)	(0)

operation and power losses of the system, with cipher cores being oriented primarily towards combinatorial logic.

Once the complete units were developed, their designs were synthesized and implemented by Xilinx Vivado tools in devices from the Spartan-7 family ([13]). All variants of the BLAKE3 cipher were fit in the smallest xc7s6cpga196-2 chip, while the iterative BLAKE2 and KECCAK modules had to be implemented in the second chip of the family, xc7s15cpga196-2, due to their too large size. The budget family of 7 Series and the smallest possible devices were chosen deliberately to get the results typical for cost-oriented, mass projects. During implementation, it was necessary to provide a target operating frequency for the tools; the chosen values were close to the architectures' maximums as determined in the analysis [2], with some safety margin ensuring stable operation of the array.

2) Simulation

Since the power calculations performed by the Vivado tools provide an average result for the entire simulation period (transient analysis is unavailable), it was necessary to develop special test runs that would correspond to the continuous operation of the hashing engine with full load (i.e. subsequent calculations would start immediately after the previous ones ended). The method of ensuring continuous load was not obvious for the prepared implementations, in which transferring a new set of data in 32b fragments would require many clock ticks, e.g. for BLAKE3 – 28 ticks for 896b of the input and 8 for 256b of the output. For this reason, a simplified approach was adopted, based on two solutions. First, the initial test vector was assigned as the initialization value of the input SIPO register, so that the simulation could start immediately with hashing, bypassing the data loading phase. Secondly, since each pipeline unit P*k* in the first *k* ticks must be loaded with a new set of data for concurrent processing, subsequent vectors were generated from the initial one by shifting a random pattern from the input pins into the SIPO register, one per each clock cycle. Strictly speaking, this did not ensure completely independent data for each stage of the pipeline (which should be observed for a correct simulation of signal switching, being the basis for estimating the consumed power), but – thanks to the avalanche effect of ciphers – state bits in neighboring stages after just a few clocks became sufficiently different to assume their approximately independent distribution.

Having such test vectors, post-route timing simulation of the fully implemented system was performed with an extra request to generate a SAIF file describing the switching activity of all internal signals found in the array ([14]). Each of the 8 implementations was tested with a set of different operating frequencies, which required preparing a distinct stimulus file for each frequency and generating a separate SAIF file.

3) Calculation of Power Consumption

The SAIF file contains switching characteristics of every signal in the FPGA array as they were traced during the simulation, and this data was the basis for the power estimations performed by the Vivado Power Analysis tool. The calculations were carried out assuming standard values of supply voltages but with maximum process settings, i.e. for the most unfavorable parameters of power losses in the semiconductor structures of the chip. The results analyzed in this article are based on reports obtained this way which included, among others, estimated total power losses in the device, their dynamic and device static parts, junction operating temperature, etc. ([14]).

The analysis and conclusions presented in the rest of the paper take into account only the reported dynamic power as the sole parameter related to the operation of the implemented project and determined by the nature of the cipher algorithm. The static component has been omitted because it is connected to losses caused by leakage currents and varies with, among others, the size and construction of the FPGA array, ambient temperature, power supply parameters, etc.. Being generally dependent on conditions of heat transport (package thermal resistivity, cooling efficiency, air flow, mounting on the board, etc.), has no relation to the operation of the programmed encryption core. Limiting the analysis only to the dynamic power allows for an assessment of power efficiency for particular cipher and eliminates factors unrelated to the algorithm, but it presents only a part of the picture of general power analysis (for example, the components ignored here would be necessary to estimate the operating temperature of the device working in any particular setup, etc.).

B. FPGA Implementations

Final parameters of the 8 encryption modules after implementation in FPGA devices are summarized in Table 1, which repeats the results published in [2]. The size of the designs (occupancy of the programmable array) is described by the first 4 lines which give the number of used LUT logic generators, slices, registers and dedicated multiplexers. One can note that the last type of element was needed in the BLAKE2 implementation only, where it was necessary for multiplexing message words *m* between *G* modules.

The dynamic characteristics of the implementations are given in the second part of the table. Listed are the requested (in implementation) and actually obtained maximum operating frequencies, as well as the parameters describing the longest

TABLE II
DYNAMIC POWER LOSSES IN IMPLEMENTATIONS RUNNING WITH MAXIMAL FREQUENCIES, IN WATTS

	X1 40MHz	P2 90MHz	P4 140MHz	P6a 190MHz	P6b 190MHz	P6c 170MHz	BLAKE2 40MHz	KECCAK 200MHz
Total	9.219	0.931	0.401	0.413	0.373	0.326	9.214	1.801
Clocks	0.002	0.010	0.020	0.033	0.034	0.031	0.003	0.030
Slice Logic	3.330	0.357	0.144	0.146	0.137	0.121	3.394	0.674
LUT	2.969	0.303	0.118	0.112	0.107	0.091	3.029	0.671
Carry	0.361	0.051	0.022	0.026	0.023	0.022	0.366	0.000
Register	0.001	0.001	0.004	0.008	0.008	0.007	0.001	0.003
MUXes	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
Routing	5.884	0.560	0.231	0.228	0.194	0.168	5.816	1.092
I/O	0.001	0.003	0.007	0.007	0.007	0.007	0.001	0.004
Efficiency metrics:								
P_{MHz} (mW/MHz)	230	10.3	2.87	2.18	1.96	1.92	230	9.01
E_h (µJ/hash)	1.61	0.072	0.020	0.015	0.014	0.013	2.30	0.216

propagation path that determined minimum clock period: the percentage of delay coming from routing (and not from logic) and the number of logic levels, including dedicated carry propagation primitives.

The evaluation of the implementation efficiency of these configurations was the subject of publication [2]. For the purposes of the power analysis which will follow, the conclusions can be summarized in the next points. 1) The change in the BLAKE3 algorithm processing scheme, which eliminated the multiplexing of *m* words at the *G* module inputs, resulted in a very significant reduction in array occupancy (up to 61% in the number of slices and 58% of LUT generators), but contributed little to speeding up the operation; this size reduction is diminished to some extent by pipelining, in which the size increases with the number of stages. 2) The very large size of the KECCAK implementation should be attributed to its much larger state; even if we consider path *M* as the state extension in BLAKE*x* algorithms, their combined width of 1024b is still less than 1600b. 3) The implementation efficiency of KECCAK transformations is much better than all other variants: in its critical path, the whole processing was fit in only 1 level of logic, while in the two iterative implementations of BLAKE*x* – in 45. The modifications introduced in the third version of the algorithm did not change anything in this regard, and the improvement can come only after dividing the round logic into pipeline stages in P*k* architectures. 4) The 32-bit adders present in both versions of BLAKE*x,* due to the method of their implementation in FPGA, do not allow to exploit the full capabilities of the wide, 6-input LUT generators available in the Spartan-7 family, as effectively as the KECCAK implementation does.

C. Power Estimation

The dynamic power parameters calculated by the tools are summarized in Table 2. The values were determined for the operation with the target (requested during implementation), maximal clock frequency. The total losses are additionally broken down into components generated by various resources of the FPGA array: clock distribution networks, slice logic (LUT generators, elements of carry propagation, registers and dedicated multiplexers), routing and I/O blocks.

The calculations were repeated for each system also with reduced frequencies and the estimated total power as functions of F_{CLK} are presented in Fig. 3. The graph allows for a collective comparison of the operating conditions of all systems in different frequency ranges and, additionally, confirms the general rule for estimating power losses in a digital system, showing the linear relationship between them and the system's frequency.

The last two rows of Table 2 contain additional metrics which characterize the power efficiency; it was possible to define them thanks to the linear dependency between the losses and the clock frequency. The first one – P_{MHz} – expresses the ratio between the dissipated power and F_{CLK} (in mW/MHz units). The second metric – E_h – is the average energy needed to calculate one hash value (in µJ/hash). Alternatively, its value is equivalent to the dynamic power required to achieve a specific hashing rate (number of hashes per second, hps); a value in µJ/hash corresponds to W/Mhps. From definitions of the metrics, $E_h = P_{MHz} \cdot N_R$, hence their values have identical relative distribution among all 6 implementations of the BLAKE3 algorithm and the differences in BLAKE2 and Keccak come only from different number of rounds.

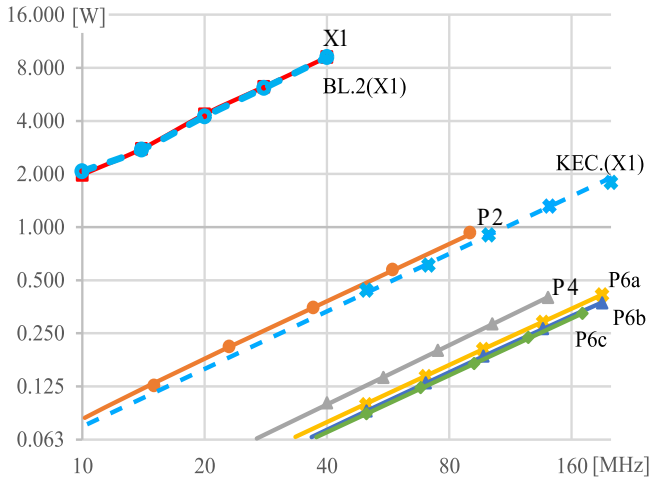

Fig. 3. Dynamic power losses of the implementations as functions of F_{CLK} (note the logarithmic scales)

IV. EVALUATION OF POWER EFFICIENCY

A. Components of the Losses

Evaluation of the results should be started by examining the structure of losses in individual projects, i.e. by analyzing power consumption by various resources in the FPGA array. Figure 5 presents the data from Table 2 as percentages. For readability, the two smallest components – I/O blocks and dedicated multiplexers – have been merged as the "Other" component.

While the low values in I/O blocks are completely expected and only validate correctness of the prepared simulation scenarios (where operation of I/O shift registers was deliberately reduced to the necessary minimum), very low losses in MUX elements in BLAKE2 is the first factor indicating that their massive impact on the size of the project (see comments in section III.B) does not result in an equally significant effect on power consumption.

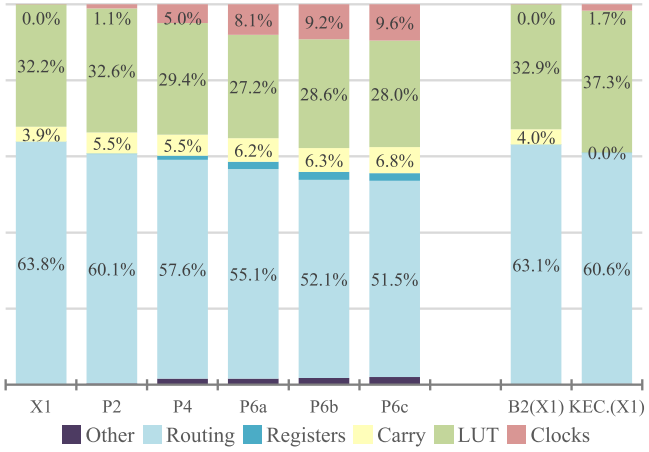

Fig. 4. Power consumption by different components of the array

Clock networks can also be omitted as a factor of little importance in this analysis because only in the P6*x* architectures, where they serve over 5,000 flip-flops with a clock frequency close to 200MHz, their share varies around 9%. Another negligible component are losses generated by registers, which become noticeable only in cases P4 and P6*x*, but even then their part reaches a maximum of 2%. Despite of high frequencies in these designs, operation of registers turns out to be very power-efficient.

One of the characteristics which distinguishes the BLAKE*x* elementary transformations from the KECCAK ones is the use of numerous adders in the operations of the *G* function. Nevertheless, the calculated results do not imply their high power requirements since carry propagation resources contribute to 4÷7% of overall losses in both versions of the cipher. Thus, it indicates that the 32b adders, implemented – similarly to the multiplexers of the *m* words – with dedicated elements and connections within the slices, keep the energy demand relatively low.

This brings the attention to the last two resources that actually determine the level of dissipated power, generating together from 80% (P6*x*) up to 96% (X1) of the losses: the LUT generators and the connection resources,. It is understandable that in any hardware realization of a cryptographic algorithm the overwhelming majority of processing consists in combinatorial transformations, hence the high load on LUT elements is not surprising, but it is worth emphasizing that the power losses are even two times higher in the routing resources. Analogous to the contribution to critical path delays in the performance parameters (Table 1), configurable interconnects with their buffering consume the most power and with a share of 51 to 64% dominate over other components.

B. Pipelining in BLAKE3

When comparing the absolute values presented in Table 2, the most significant observation is the very large reduction in power demands achieved in BLAKE3 implementations thanks to the pipelining. Already dividing the X1 round logic into two pipeline stages resulted in a reduction of the E_h value by as much as 95% (from 1.61 to 0.072 µJ/hash), and further divisions with increasing number of stages brought a reduction of, respectively, 62% (transition from P2 to P4) and approx. 30% (P4 vs. P6*x*). This effect leads to an unusual situation clearly visible in Fig. 3, when the pipelined cases not only are able to achieve much higher operating frequencies than a purely iterative one, but also at these frequencies they consume significantly less power. The P_{MHz} factor, which reaches as much as 230 mW/MHz in the X1 organization, drops to 10.3 (P2), 2.87 (P4) and 1.92 ÷2.18 in P6*x* cases.

It turns out that the hardware implementation of BLAKE*x* core is a profound example of the effect known and described in the literature ([15]-[17]) when reductions in power consumption are caused by reorganization of the digital system with pipeline processing. The mechanism standing behind this effect is rooted in elimination of transient signal disturbances, so called *glitches* – short-lived intermediate states which the signal temporarily assumes before reaching a stable and correct new value, which can occur after changes at the inputs of a combinatorial circuit. In the general case, such disturbances can occur when two or more inputs of a logic gate generating the signal change non-simultaneously due to (a) different delays in their propagation paths or (b) different numbers of logic levels through which they pass. The first factor is unavoidable in FPGA implementations, where transmission paths are constructed with different segments of configurable connections and ensuring their balanced delays is difficult and expensive in practice; the second factor often occurs in complex combinatorial logic designs. In this case, both factors converge and intensify each other: the BLAKE round, as the combinatorial module with 1024 input bits and a deliberately complex and irregular internal structure, is particularly susceptible to the glitch generation, and the resulting parasitic switching of signals on long transmission lines with capacitive loads generate large power losses. Pipelining eliminates these problems by introducing two changes: (a) after dividing long propagation paths into pipeline sections, the number of logical levels is reduced – as it can be seen in Table 1, from the initial 45 in the X1 architecture to 10÷12 in the P6*x* – which decreases probability that any two signals reaching the same LUT element would traverse different number of logic levels, (b) the flip-flops which latch the signals at the pipeline boundaries eliminate the avalanche reproduction of glitches by pausing their propagation until the signal reaches a stable state.

The improvement in signal stability introduced in the pipelined architectures becomes evident after examination of the SAIF files. This format records, among others, the number of value changes (including transient glitches) for each internal signal in the system. Figure 5 shows this number averaged per one clock cycle; to restrict the examination only to signals that actually belong to the encryption core, the averages are limited to the 5% most frequently switching signals in each design. The

data was read from the same SAIF files as the ones used for power calculations. The results illustrate the huge differences between the X1 and the pipelined BLAKE*x* architectures, providing also the KECCAK behavior for comparison. The relationship between the cases in this graph is essentially a repetition of the one seen in the power comparison: if in the P6*x* cores the most active signals switched on average 5 times in each clock cycle (i.e. they went through 4 transient states before reaching the final value), in iterative architectures this number was as high as 1380 or 1268. It is assumed that in a cryptographic transformation ideally a half of the state bits, randomly selected, should change after each round, hence 0.5 would be the desired, ideal value of this parameter.

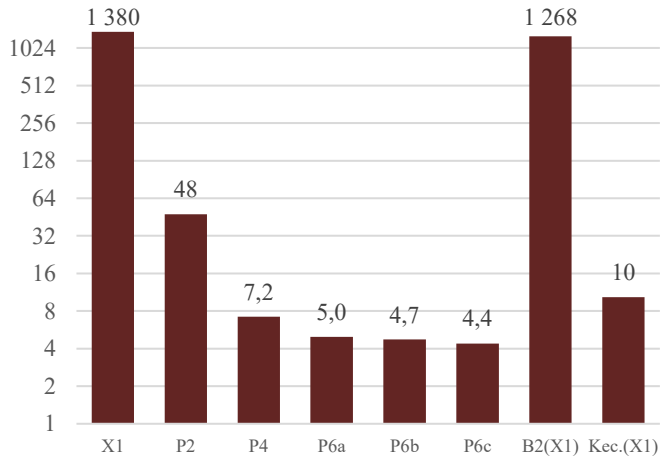

Fig. 5. Average number of value changes per one clock cycle for 5% most active signals (log scale)

We can also see that this problem is particularly intensive in the BLAKE algorithm: even in the iterative architecture of KECCAK the number of switches was 10, which is approximately what BLAKE achieves only after introducing 4-stages pipeline. This weakness of the algorithm can be attributed to the presence of six 32-bit adders in the combinatorial paths of each *G* module. In FPGA, the adders are implemented in a (simplest) ripple-carry propagation configuration which does not require complex logic, but its sequential nature is very susceptible to the generation of intermittent transient states on the outputs (being a reflection of carry values propagated sequentially across the bits). This organization is fast enough and itself does not dissipate much power thanks to the use of dedicated connections inside the slices, but the additional glitches it generates are replicated by the complex combinatorial logic and cause very large power losses in the propagation paths.

Leaving for the moment the absolute values of power losses, the results presented here allow also for a better comparison of the three variants of the six-stage pipeline architectures of the BLAKE3 algorithm. In [2] this comparison was based on performance metrics and did not clearly indicate the best option. Better differentiation now comes from the power analysis: the E_h parameter for the P6c variant reaches the best value, being lower by a moderate 3% than in the P6b case but by a significant 14% than in the P6a. Thus, the most natural division of the *G* processing paths, which maintained an even division of 6 adders into three pipeline stages in the P6a variant, turned out to be the least energy-efficient solution, and the problem of separating two adjacent adders indeed creates noticeable implementation problems but they manifest themselves in the power parameters rather than in the performance. Optimizing the two adjacent adders in the cases P6b and P6c led to such a reduction in the glitch generation that it outweighed the problems resulting from unbalanced logic within the stages. The P6c case was the least efficient in the performance analysis (9% lower operation speed compared to the fastest P6a), but it proved to be the most energy efficient.

C. *Comparison of the Three Ciphers*

Particular attention should be paid to the fact that the power losses in both variants of the BLAKE cipher turned out to be practically identical: the difference in E_h metrics is a purely arithmetic effect resulting from the reduction in the number of rounds, and the proportionality factor of power consumption as a function of the frequency P_{MHz} is identical. It is significant that this parity is observed in two projects of such different sizes, with the BLAKE3 implementation being as much as approx. 40% smaller both in the number of occupied slices and LUT generators. In the performance analysis of [2] such a reduction in resource consumption was considered to be an obvious benefit, but now it has not translated into an advantage in power consumption.

The size difference, as noted in Chapter 2, comes from the elimination of the 16:1 multiplexers that switch 32-bit words *m* in the message path. The obtained results indicate that such a large amount of logic did not dissipate significant power, which can be related to the following two factors. (a) Thanks to the use of internal routing and dedicated MUX primitives, in the Spartan 7 architecture 16-input multiplexers can be created without LUTs and general-purpose connections, leading to a construction with predictable and repeatable timing parameters. (b) In general, any multiplexer is a combinatorial circuit with a perfectly regular structure of a balanced binary tree, so it does not suffer from the problem of different numbers of logic levels on the paths from input to output. As long as this regularity (including propagation times in the transmission lines) can be preserved in the FPGA realization – which in this case was possible by meeting criterion (a), even for a 16:1 mux size – a glitch-free combinatorial circuit can be created. As a result, both versions of the BLAKE*x* algorithm showed a virtually identical value of the total dynamic power, and its individual components in Table 2 differ by no more than 2%.

The lack of improvement in the power parameters of BLAKE3 confirms its inferiority to the KECCAK algorithm, which is not so exposed to power losses caused by glitches. Its transformations do not contain arithmetic adders and, despite their complicated structure extending down to the single bit level, are very well implemented with 6-input LUT generators. As a result, the critical path in the iterative implementation of this algorithm contains only a single LUT element and the signals are not so burdened with noise. With the measured average number of switches of the 5% most active signals equal to 10, the total power consumption of the complete module was much lower: 7 times lower energy per hash and even 25 times lower power per megahertz – all achieved in an implementation with about 80% higher resource occupancy.

CONCLUSIONS

This work analyzed the power consumption of BLAKE3, BLAKE2 and KECCAK hash functions implemented in various

hardware architectures in an FPGA device. The same design and implementation methodology ensured uniform test conditions and comparability of the results received for 8 projects, allowing to show the impact of pipelining the BLAKE3 core in the FPGA array and, additionally, to compare the power efficiency of the three ciphers. The results indicated the primary impact that signal stability has on the level of power consumption, being a factor as important as the size and amount of occupied logic resources.

The study indicated an exceptionally high level of glitches in the (non-pipelined) round of the BLAKE*x* algorithms. This fact increased power requirements to levels so high that the standard iterative organization of these ciphers becomes prohibitively expensive in operation, at least in applications requiring high clock frequencies. In such cases, even if the associated improvement in throughput is not required, pipelining may be considered as a way to reduce power losses, if the increased latency remains acceptable. It is worth noting, however, that the presented analysis focused on the extreme case of continuous operation of the hashing module under full load, i.e. with the pipeline filled in 100% and generating a new output on average every N_R clock ticks. The decision criteria may be different if the working environment does not provide enough data for continuous hashing and the power budget of the entire project is able to accommodate the increased losses. Nevertheless, the discussed problems add another aspect (apart from speed efficiency) that confirms the advantage of the KECCAK algorithm over BLAKE in FPGA implementations.

A separate issue is the trustworthiness of power estimations at such high levels of signal glitches. The most accurate method used by the tools is based on SAIF data generated by the simulator, which evaluates activity of each signal by giving only the total number of its switches, without distinguishing between transient disturbances and stable states. In the case of the BLAKE*x* algorithms, these numbers reached over a thousand in a single clock cycle, which translates to frequencies too high for even fastest logic gates to switch; in such conditions standard models that estimate current pulses from charging capacitive loads may need to be amended.

REFERENCES

[1] K. Gaj, J.P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, B.Y. Brewster, "ATHENa – Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware Using FPGAs", in Proc. 20th International Conference on Field Programmable Logic and Applications, Milano, pp. 414-421, 2010. doi:10.1109/FPL.2010.86

[2] 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. doi:10.1007/978-3-031-06746-4_31

[3] 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. doi:10.1007/978-3-031-37720-4_27

[4] 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 Sept. 2023.

[5] 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. doi:10.1007/978-3-642-38980-1_8

[6] 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 Sept. 2023.

[7] 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. doi:10.1109/ETI4.051663.2021.9619202

[8] 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. doi:10.1109/APCC49754.2021. 9609933

[9] J. Sugier, "Simplifying FPGA Implementations of BLAKE Hash Algorithm with Block Memory Resources", Procedia Engineering, vol. 178, pp. 33–41, 2017. doi:10.1016/j.proeng.2017.01.057

[10] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, "The Keccak reference", SHA-3 competition (round 3), available at https://keccak.team/papers.html, 2011, last accessed Sept. 2023.

[11] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS pub. 202, https://csrc.nist.gov/publications/detail/fips/202/final, last accessed Sept. 2023. doi:10.6028/NIST.FIPS.202

[12] J. Sugier, "Low cost FPGA devices in high speed implementations of Keccak-f hash algorithm", in W. Zamojski et al. (eds) Proc. Ninth International Conference on Dependability and Complex Systems, Advances in Intelligent Systems and Computing, vol. 286, pp. 319-330, Springer, Cham, 2014. doi:10.1007/978-3-319-07013-1_42

[13] Xilinx, Inc., "7 Series FPGAs Data Sheet: Overview", DS180.PDF available at www.xilinx.com, last accessed Sept. 2023.

[14] Xilinx, Inc., "Vivado Design Suite User Guide: Power Analysis and Optimization", UG907.PDF available at www.xilinx.com, last accessed Sept. 2023.

[15] E. Boemo, J. Oliver, G. Caffarena, "Tracking the pipelining-power rule along the FPGA technical literature", in Proc. 10th FPGAworld Conference, FPGAworld, 2013. doi:10.1145/2513683. 2513692

[16] 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. doi: 10.5815/ijieeb.2012.05.07

[17] 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. doi:10.1007/978-3-540-30117-2_73