INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 3, PP. 1-7
Manuscript received March 11, 2026; revised June, 2026. doi: 10.24425/ijet.2026.1716

Different generations of FPGA devices in implementation of hash functions – A case study

Jarosław Sugier

***Abstract*—Cryptographic algorithms pose a significant challenge for hardware designers, primarily due to their complex and intentionally irregular internal transformations. These characteristics lead to large, multilayered combinational structures that are difficult to efficiently map onto FPGA architectures. This paper analyzes FPGA implementations of two hash functions: KECCAK — the core component of the SHA-3 algorithm — and BLAKE3. Their performance is evaluated across three generations of AMD devices: 7 Series, UltraScale, and UltraScale+. The study covers both iterative and pipelined variants of the two ciphers, yielding a total of seven distinct hardware realizations. Each module is examined in terms of resource utilization, achievable operating frequency, and dynamic power dissipation across three devices from Kintex families. The use of a unified implementation and evaluation methodology enables a reliable comparison of results, allowing not only to capture the performance and energy-efficiency benefits offered by newer technologies, but also to identify areas where these advantages become less pronounced for such demanding cryptographic applications.**

***Keywords*—cryptographic hash function; KECCAK; BLAKE3; FPGA; semiconductor manufacturing technology**

I. INTRODUCTION

FOLLOWING AMD's announcement ([1]) of long-term support for two older FPGA families — at least until 2040 for 7 Series devices and until 2045 for UltraScale — designers can now treat these mature platforms as stable and reliable options even for new projects, alongside the latest UltraScale+ devices. In this work, we evaluate the efficiency of implementing two modern cryptographic constructions: the KECCAK permutation and the BLAKE3 hash function, comparing them across three generations of AMD (formerly Xilinx) programmable gate arrays. In total, seven designs were analyzed — two iterative architectures and five pipelined variants (two for KECCAK and three for BLAKE3). Each of them was implemented in three Kintex devices: 7 Series, UltraScale, and UltraScale+, yielding 21 designs included in the study.

Modern cryptographic algorithms belong to a class of constructions that are exceptionally demanding from a hardware-implementation perspective. Their internal structure is complex and intentionally irregular, which hinders cryptanalysis but simultaneously results in large, difficult-to-optimize combinational transformations. Such operations pose a significant challenge for FPGA designers, who must contend with architectural constraints as well as the timing and energy costs associated with implementing these functions. In this context, the goal of this study is to perform a comprehensive evaluation and comparison of the operating speed, hardware resource usage, and dynamic power consumption of seven cryptographic designs realized in comparable devices from three generations of the same FPGA family.

The same set of KECCAK and BLAKE3 algorithm organizations, implemented on the Spartan-7 platform, was thoroughly analyzed in our earlier publications [2]–[6]. Those works identified several important challenges related to both performance and energy efficiency in iterative and pipelined implementations of the two ciphers, highlighting their specific features that either hinder or facilitate their realization in a programmable fabric. This paper, being an extended version of [7], broadens those results by assessing the same issues in the context of two newer technologies — UltraScale and UltraScale+. Since the Spartan family has limited availability (it was not included in the UltraScale generation, and its UltraScale+ variant was introduced only recently, after the core research for this study had been completed), the Kintex platform was selected as the simplest device line available across all three generations — from 7 Series to UltraScale+.

The structure of this paper is organized to gradually introduce the reader to successive stages of the analysis. Section II discusses the fundamental characteristics of the devices representing the three FPGA technologies and provides a concise description of the seven hardware architectures of the evaluated ciphers — both iterative variants and designs employing intra-round pipelining. Section III focuses on the size and speed characteristics of the obtained implementations, presenting their parameters in a form that enables direct comparison. Section IV analyzes aspects related to power consumption and energy efficiency, indicating some peculiarities of the devices manufactured in the new technologies. Finally, Section V summarizes the key observations and formulates the conclusions drawn from the study.

II. HARDWARE PLATFORMS AND THE CIPHERS

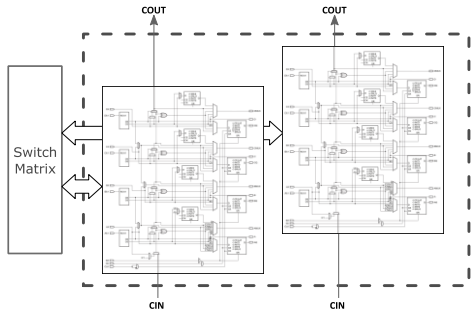
A. Three Generations of AMD FPGA Devices

Contemporary FPGA designers have access to three device generations which—despite technological differences—retain an almost identical logical structure. This marks a departure from earlier years, when each new family introduced not only higher speed, capacity, and energy efficiency, but also substantial functional changes clearly signaling the intention to replace its predecessors. The observed stabilization of the

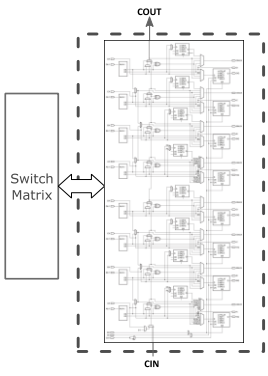
Jarosław Sugier is with Wrocław University of Science and Technology, Poland, Faculty of Information and Telecommunication Technology, Department of Computer Engineering, Poland (e-mail: jaroslaw.sugier@pwr.edu.pl, ORCID: https://orcid.org/0000-0003-1452-3067).

logic-cell architecture is the result of two key milestones in the evolution of Xilinx (now AMD) devices. First, the Virtex-5 family introduced full 6-input LUT generators, replacing earlier 4-input tables. In addition to offering four times greater capacity, the second output port allowed them to optionally generate two different functions of the same 5-bit arguments. Later, the Virtex-6 family added a flip-flop to the second LUT output as well. These two steps shaped the logic-cell structure that has remained unchanged in subsequent generations, including the three families analyzed in this work: each cell contains a LUT generator capable of implementing any Boolean function of six variables on the O6 output, an optional O5 output enabling the generation of a 2-bit function of five variables, and two flip-flops associated with both outputs ([8]-[9]).

The organization of these logic cells inside a node of the programmable array – called a Configurable Logic Block, CLB – represents a second, closely related architectural aspect. In the 7 Series architecture, four logic cells sharing a common carry-propagation path formed an intermediate construct denoted as a slice, and two such slices — placed side by side and equipped with parallel carry chains — constituted a single CLB block, as illustrated in Fig. 1a. In the newer UltraScale and UltraScale+ generations, the logic-cell structure remained unchanged, but the CLB organization was redesigned. Instead of two separate slices, their cells were merged into a single eight-element stack, which became the only logical unit within a CLB and eliminated the earlier subdivision into slices, as seen in Fig. 1b.



(a)



(b)

Fig. 1. Configurable Logic Blocks in devices considered in this study: (a) two slices of 7 Series vs. (b) single-column block in UltraScale.

This modification did not affect the overall resource density or functionality because each CLB still comprised 8 identical logic cells, but it had one important consequence for the present analysis: the length of the arithmetic carry-propagation path within a single block increased from 4 to 8 bits. This means that a longer portion of an adder can be handled within a single CLB, influencing placement and routing strategies for designs that heavily rely on addition — which is the case in the second algorithm considered in this study.

Since the three FPGA generations share essentially the same logical structure and comparable resource sets, the differences between them arise primarily from technological parameters — namely, achievable operating speed and energy efficiency. 7 Series devices, manufactured in a 28-nm process, were promoted as offering a very favorable balance of cost, performance, and power consumption, with up to 50% lower energy usage compared to the earlier 45-nm technology. The next generation, UltraScale, uses a 20-nm process, which according to the manufacturer should yield an additional ~40% reduction in power consumption, particularly in components responsible for switching activity. The latest UltraScale+ family, based on a 16-nm FinFET process, is expected to deliver even greater improvements: up to 60% lower power consumption relative to 7 Series, while maintaining a favorable performance-per-watt ratio.

B. Implementations of the Ciphers

Both KECCAK [10] and BLAKE [11] originate from the multi-year selection process conducted as part of the NIST SHA-3 competition. The process concluded in 2012, when KECCAK was selected to be the foundation of the new hash standard SHA-3 [12]. Although the BLAKE proposal did not win the final round of the contest, it consistently ranked highly throughout the evaluation stages, receiving recognition for its strong cryptographic properties and excellent performance in software implementations. As a result, the algorithm has since established itself as one of the most respected hashing functions in the cryptographic community. In this study, we analyze its most recent, modernized version published in 2020 [13].

In most modern cryptographic algorithms — including KECCAK and BLAKE3 — data processing is organized as a sequence of identical rounds that iteratively modify the algorithm's internal state. In KECCAK, this state consists of 1600 bits (den. A), while in BLAKE3 it includes 512 bits of state (v_i words) together with an additional 512 bits of input data (m_i). The number of round repetitions, denoted N_R, is 24 in KECCAK and 7 in BLAKE3. In the latter case, this value was reduced from the original 12 rounds proposed in the primary 2010 submission. Although newer variants of KECCAK developed after the competition also employ a reduced number of rounds, this work adheres to the initial N_R parameter because it is still binding in the SHA-3 standard.

In hardware implementations of such processing, the repetitive structure of rounds can be mapped to hardware in several ways, each representing a different trade-off between resource usage and throughput. The simplest approach — mirroring software execution and typically used as a reference point — is the iterative architecture, in which the entire logic of a single round is placed in one combinational module, and a dedicated controller repeatedly processes the internal state through this module in N_R cycles of the system clock. In our analysis, these variants are denoted as K X1 and B3 X1.

To achieve higher throughput, the iterative modules can be further modified by introducing intra-round pipelining. This technique divides the long combinational paths of a round into

several shorter segments separated by registers, which shortens the clock period but also enables multiple data blocks to be processed simultaneously. Widely used in hardware implementations of any class of data processing, pipelining increases overall throughput, although it typically introduces a slight increase in latency — the time between providing input data and obtaining the result.

As shown in earlier studies [2]-[3], the KECCAK round can be divided into two or three pipeline stages. In the case of BLAKE3, whose round logic is deeper and contains more combinational levels, several granularities of pipelining are possible: 2, 4, and 6 stages. In total, with iterative approach, this produces seven distinct designs, each of which was implemented and evaluated across the three Kintex generations.

C. *Options for Pipelined Architectures*

A KECCAK round consists of five transformations — θ, ρ, π, χ, and ι — applied to the bits of the state *A*, together with a 64-bit round constant RC:

$$A' = \text{Round}(A, RC) = \iota(\chi(\pi(\rho(\theta(A))))), RC)$$

From a hardware-resource perspective, the round function forms an essentially linear dataflow, in which successive XOR operations gradually modify the state *A* (see Fig. 2). The only exception is the θ step, which requires computing two 320-bit intermediate values, C and D, before the transformation can be applied. The following ρ and π stages do not involve any combinational logic — they consist solely of word rotations, which in hardware are implemented through routing connections, even though in software they require multiple processor instructions.

In [2], several ways of dividing this data path into pipeline segments were analyzed to shorten the critical path and increase the clock frequency. Based on those results, representative two variants were selected for this study:

- K P2 — registers inserted after the π operation,
- K P3 — additional registers inserted for both the D values and the output of the π stage.

These two variants achieve different trade-offs between pipeline depth and implementation complexity; location of their pipeline registers are also marked in Fig. 2.

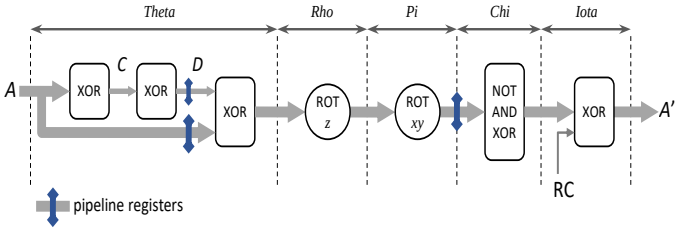
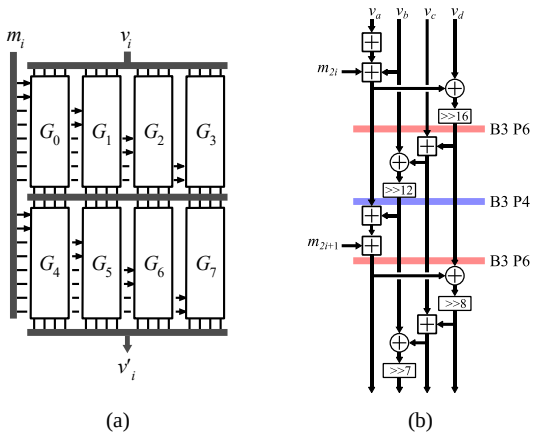


Fig. 2. Structure of KECCAK round with location of pipeline registers

The round structure of BLAKE3, illustrated in Fig. 3a, is built around a set of G functions operating in parallel on four independent portions of the state. Each round uses these functions twice, and their behavior includes not only rotations and XOR operations — as in KECCAK — but also 32-bit arithmetic additions. This last element has significant hardware implications: in FPGA devices, adders are implemented as vertical chains of logic cells using dedicated fast-carry paths, forming long structures that span multiple logic blocks. These structures complicate placement and routing, making the topology more demanding than in KECCAK.



(a) (b)

Fig. 3. Construction of BLAKE round with *G* functions (a) and their internal processing with boundaries of pipeline stages (b)

In [3], several approaches to pipelining this logic were evaluated. In addition to the natural two-stage variant, where the pipeline boundary lies between two consecutive G-function instances – denoted here as B3 P2 – finer-grained divisions of the G function into two or three segments were also considered. Based on results of that evaluation, here the two most effective variants were selected:

- B3 P4 — the round logic divided into two pipeline stages,
- B3 P6 — a three-stage pipeline.

Boundaries of pipeline stages in both variants are highlighted in Fig. 3b.

III. IMPLEMENTATIONS

A. *Hardware Size*

The seven cipher-module variants described in the previous section were extended with simple serial-to-parallel and parallel-to-serial interfaces, allowing them to function as complete, standalone designs suitable for FPGA implementation. Each module was then mapped onto three different devices from the Kintex family, resulting in a total of 21 implementations included in the analysis.

For the UltraScale and UltraScale+ families, the smallest available models were selected as test platforms — xcku035fbva676-1c and xcku3pffva676-1e, respectively — offering approximately 318k and 356k logic cells. Since the smallest Kintex device in the 7 Series family provides significantly fewer resources, the second-smallest model in that lineup, xc7k160tfbg484-1, was chosen to maintain a comparable resource margin. Even in this device, logic utilization did not exceed 6% of available slices, and the usage of registers and other resources was even lower. This ensures that none of the designs were constrained by device capacity, and the results obtained across all three platforms can be considered equivalent.

All devices were also selected in their baseline, lowest speed grade, ensuring uniform evaluation conditions.

Table I summarizes the results of all 21 implementations, presenting in each cell the parameters obtained in Kintex, from top to bottom, 7 Series, UltraScale, and UltraScale+. The upper part of the table lists resource-utilization metrics — the number of LUTs, flip-flops, and CLB blocks used. The lower part

focuses on timing parameters: the maximum operating frequency reported by the implementation tools for fully placed-and-routed designs, the number of logic levels in the critical path, and the share of routing delay within that path (the remainder being logic delay).

TABLE I
IMPLEMENTATION RESULTS IN KINTEX DEVICES

		K X1	K P2	K P3	B3 X1	B3 P2	B3 P4	B3 P6
LUT	7S	5183	4776	4779	2669	2759	2714	3444
	US	5406	4772	4781	2669	2749	2690	3412
	US+	5337	5402	4778	2663	2692	2681	3439
FilpFlops	7S	4817	6432	8353	2184	3206	4821	5842
	US	4815	6432	8353	2184	3206	4833	5876
	US+	4814	6432	8353	2184	3206	4859	5894
CLB	7S	705	768	808	418	508	540	644
	US	801	741	849	428	524	560	628
	US+	792	899	843	429	555	608	644
F_{max} [MHz]	7S	311	445	472	73	139	231	305
	US	408	609	612	82	157	270	327
	US+	518	777	747	122	232	364	488
Levels of logic	7S	3	1	1	39	22	12	10
	US	3	2	1	28	14	9	8
	US+	3	1	1	29	18	10	5
Routing delay	7S	85%	77%	81%	55%	51%	53%	43%
	US	72%	71%	79%	51%	51%	45%	41%
	US+	79%	78%	83%	52%	57%	55%	62%

A clear and unsurprising conclusion from the first two rows of the table — number of LUTs and registers — is the near-perfect consistency across the three FPGA generations. Differences in resource usage are minimal: in no case does the deviation from the mean exceed 6.2% (which is observed in the K P2 design), and in most implementations it remains below 0.5%, especially for registers. This confirms that, given the identical logic-cell structure across all three FPGA families, newer technologies offer no inherent advantage in mapping the cipher logic — at least in terms of basic LUT and FF usage.

Among the analyzed parameters, the only noticeable — though still relatively small — difference between the three generations is the density of logic placement across the occupied fabric. This can be calculated as the average number of LUTs per CLB block and is illustrated in Fig. 4. The highest density appear in non-pipelined variants of both KECCAK and BLAKE3, which is expected: the absence of intermediate registers encourages more compact combinational structures, taking up less LUT generators.

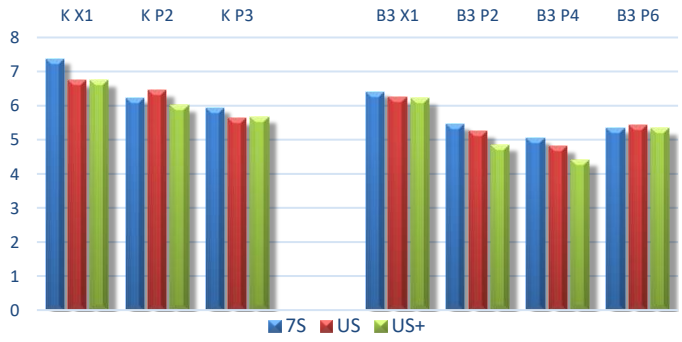
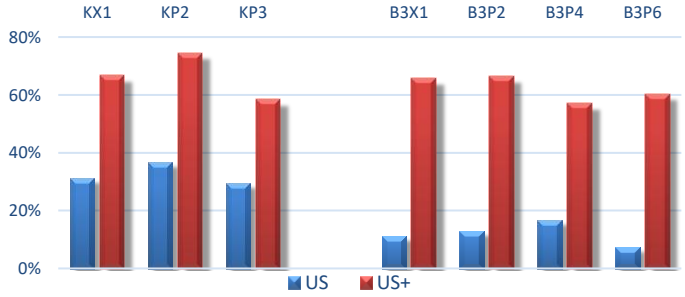


Fig. 4. Average number of LUT per CLB

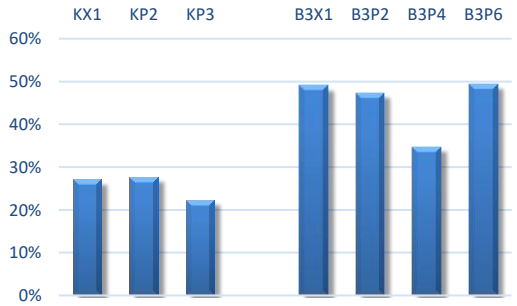
In the newer FPGA generations, however, a slight decrease in density is observed. This trend appears in most designs and is particularly pronounced in the B3 P2 and B3 P4 implementations. This suggests that, despite the identical logic-cell structure, changes in CLB organization and implementation-tool behavior can influence how logic is packed in more complex, pipelined BLAKE3 variants.

B. *Speed*

The most pronounced benefits of newer process technologies appear in the timing parameters. The comparison shown in Fig. 5a illustrates how significantly the maximum operating frequency increases when moving from the 7 Series family to UltraScale and UltraScale+. The data clearly indicate a substantial performance gain: UltraScale devices achieve, on average, a 21% higher F_{max}, although the improvement is more pronounced in KECCAK implementations (approximately 33%) than in BLAKE3 (around 12%). Even greater gains are observed in the UltraScale+ technology, where the average frequency increase reaches 64% — specifically 67% for KECCAK and 62% for BLAKE3. The difference between the two algorithms in the two newest technologies is noteworthy and is discussed separately later in the paper.



(a)



(b)

Fig. 5. Increases in maximum operational frequencies F_{max}: (a) US and US+ vs. 7 Series (b) US+ compared to US

Importantly, in the latest generation the improvement is not only larger but also more uniform across both algorithms, suggesting that the UltraScale+ architecture handles diverse logic structures more effectively. The additional comparison shown in Fig. 5b highlights only the difference between UltraScale and UltraScale+, emphasizing the substantial performance boost provided by the 16-nm process.

The mechanism behind these frequency improvements can be partially explained by analyzing the number of logic levels in the critical paths — this comparison is presented in Fig. 6. For KECCAK, these values already approach the achievable minimum in the 7 Series family, leaving very little room for further optimization, so they are omitted in the graph. The situation is different for BLAKE3: in this case, the newer

UltraScale and UltraScale+ technologies allow for a noticeable reduction in the depth of combinational paths, effectively simplifying the logic and directly contributing to higher maximum clock frequencies. The difference between them, though, is not always in favor of US+.

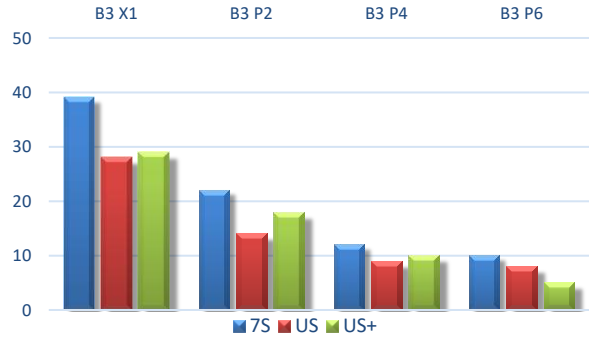


Fig. 6. Fluctuations in number of logic levels in the longest paths of BLAKE3 designs

Moreover, this relationship does not appear in the analysis of the final parameter from Table 1 — the share of routing delay in the total delay of the critical path. Although the average values differ significantly between the two algorithms (approximately 78% in KECCAK and 51% in BLAKE3, which harmonizes with earlier observations in [5]), no consistent trend is visible across the three technologies within individual designs. The fluctuations are irregular and do not support any general rule regarding the influence of process technology on the routing-delay contribution.

IV. POWER EVALUATION

A. Methodology

The energy parameters of all 21 implementations were determined using the most accurate estimation method available in the implementation environment. As in the analyses presented in [4]-[5], the calculations were based on SAIF files describing signal-switching activity. To obtain these files, it was first necessary to perform timing simulations of the final, fully routed designs, and that in turn called for creating a specially prepared set of stimuli.

After each design was implemented — at an operating frequency intentionally reduced relative to F_{max} by a safety margin — simulation scenarios were developed that forced continuous delivery of input data. This ensured that the cipher modules remained fully loaded, which is particularly demanding for pipelined constructions. The switching-activity traces recorded during these simulations were saved in SAIF files and then passed to the implementation tools, which used them to perform accurate power-consumption estimation [14].

To focus exclusively on the influence of algorithm construction on energy demand in FPGA devices, the analysis considers only the dynamic component of the total dissipated power. The static component was omitted because it depends on numerous external factors — such as cooling method, package type, and environmental conditions — and does not directly reflect algorithm-dependent implementation characteristics. It is worth noting, however, that static power is strongly tied to the manufacturing process, and therefore in practical applications it may influence the choice between the three FPGA families, even though it is not included in the detailed comparisons which follow in this work.

Table II summarizes the power-estimation results. Since the dynamic component of power increases proportionally with clock frequency, the absolute values were appropriately normalized, and comparisons between designs are based on the parameter P_d, expressed in mW/MHz, which represents dynamic power losses per unit of operating frequency. The first row of the table lists its total value, which is then broken down into five main categories: power consumed by logic resources (primarily LUTs, although in BLAKE3 implementations the carry-chain elements also contribute noticeably), flip-flops, routing, clock networks, and I/O blocks.

The final row contains a synthetic metric E_h, representing the energy required to perform a single hash-function computation. This parameter can be treated as the ultimate measure of energy efficiency for a given implementation. However, because E_h is directly derived from dynamic power, its variation across the three FPGA families mirrors the same relationships as the ones shown in the P_d values.

TABLE II
DYNAMIC DISSIPATED POWER AND ITS COMPONENTS

	K X1	K P2	K P3	B3 X1	B3 P2	B3 P4	B3 P6
P_d [mW/MHz]	8.17	2.19	1.53	215.7	9.88	2.46	1.73
	5.56	1.50	1.18	35.5	5.18	1.77	1.38
	4.89	1.54	1.26	46.2	5.96	2.05	1.36
components:	3.395	1.090	0.580	75.1	3.617	0.895	0.620
- comb. logic	2.185	0.733	0.433	11.1	1.917	0.625	0.480
	2.147	0.787	0.453	16.5	2.555	0.810	0.515
	0.015	0.030	0.050	0.008	0.017	0.025	0.040
- registers	0.020	0.040	0.060	0.008	0.017	0.030	0.045
	0.017	0.033	0.060	0.006	0.015	0.030	0.040
	4.605	0.860	0.660	140.4	6.083	1.330	0.850
- routing	3.190	0.547	0.473	24.2	3.083	0.945	0.675
	2.570	0.453	0.433	29.4	3.170	0.975	0.560
	0.135	0.190	0.210	0.083	0.117	0.165	0.190
- clocks	0.130	0.140	0.173	0.083	0.100	0.125	0.130
	0.107	0.140	0.173	0.063	0.100	0.120	0.125
	0.015	0.010	0.020	0.050	0.042	0.040	0.030
- I/O	0.030	0.040	0.040	0.117	0.067	0.050	0.045
	0.053	0.120	0.147	0.188	0.115	0.115	0.115
E_h [nJ]	196	53	37	1510	69	17	12
	133	36	28	248	36	12	10
	117	37	30	323	42	14	9

B. Comparison with 7 Series

A detailed discussion of the variations of the P_d parameter across various architectures of both ciphers — particularly the large differences between iterative and pipelined implementations — was already presented in [5]-[6] for 7 Series devices, and therefore is not repeated here. Instead, this work focuses on the effects of transitioning to the two newer technologies. The comparison shown in Fig. 7 illustrates the reduction in total dynamic power P_d (the same fluctuations would be seen in E_h values) achieved in UltraScale and UltraScale+ relative to 7 Series. In most cases, a clear — and sometimes very substantial — improvement is observed, although the results deserves some interpretation and several noteworthy nuances can be highlighted.

The most striking change is the significant improvement in the energy efficiency of the iterative BLAKE3 variant — the reduction in power reaches 16–21%. It is worth recalling that

earlier analyses on the Spartan-7 platform in [5] indicated that the X1 variant exhibits an exceptionally unfavorable energy profile. Those studies suggested that pipelining should be considered the strongly recommended approach for this algorithm — not only because of the increase in throughput, but also due to the substantial improvement in energy efficiency that appears as an additional benefit [15].

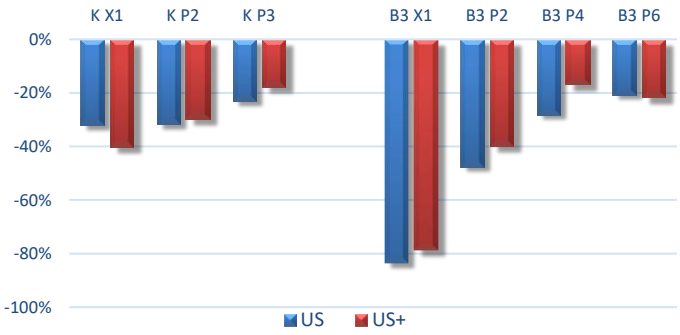


Fig. 7. Dynamic dissipated power: changes in UltraScale and UltraScale+ devices as compared to 7 Series

In the newer technologies, however, the magnitude of this improvement is smaller. While in the 7 Series device the transition from B3 X1 to B3 P2 reduced P_d by nearly a factor of 22 ([4]), in UltraScale the reduction is only 6.8×, and in UltraScale+ — 7.8×. A similar trend is observed for deeper pipelines: increasing the number of stages still lowers dynamic power, but the rate of reduction is noticeably slower in UltraScale and UltraScale+ than in Kintex-7. In other words, the newer technologies improve the energy efficiency of the iterative variant so effectively that the additional benefits of pipelining become less dramatic than in the older generation.

C. *UltraScale vs. UltraScale+*

A closer look at the relationship between the energy efficiency of the two newest FPGA families reveals that — with only two exceptions in the K X1 and B3 P6 designs — UltraScale+ devices do not offer any improvement over UltraScale. In most implementations, the dynamic power consumption in UltraScale+ is actually higher. This trend is illustrated in Fig. 8, which shows not only the overall change in P_d (or E_h) but also its two key components: losses in combinational logic and losses in routing.

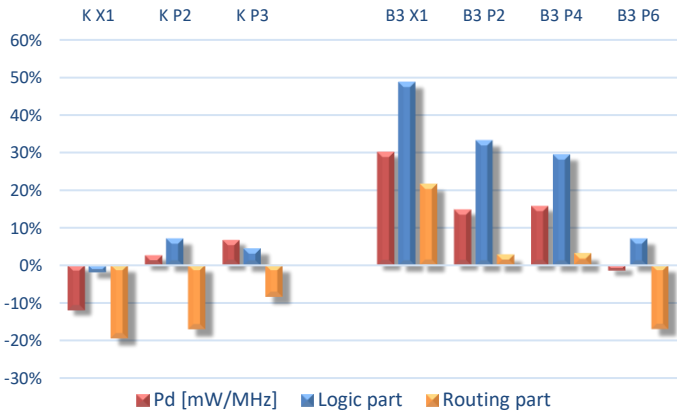


Fig. 8. UltraScale vs. UltraScale+: changes in the total dynamic power and in its two dominant constituents

The plot indicates that the only case in which UltraScale+ provides a clearly noticeable improvement in energy efficiency is the K X1 design, where the reduction in power reaches approximately 12%. In all other implementations — especially in BLAKE3 variants from X1 through P4 — an increase in power consumption is observed, reaching up to 30%. Moreover, the differences between KECCAK and BLAKE3 are very pronounced: in KECCAK, adding more pipeline stages systematically worsens the energy efficiency of UltraScale+, whereas in BLAKE3 the trend is reversed. The initial ~30% increase in power in the X1 variant gradually decreases as the pipeline deepens, eventually turning into a small, roughly 1% improvement in the P6 design.

A more detailed analysis of the two main contributors to dynamic power P_d helps formulate hypotheses about the sources of these differences. In all implementations — except K X1 — the increase of losses in logic is significant and consistently larger than the changes observed in routing. This indicates that logic elements are the primary factor degrading the energy efficiency of UltraScale+. Routing, on the other hand, behaves more favorably: its contribution to power grows more slowly and partially compensates for the increases generated by logic.

Notably, the two analyzed algorithms exhibit opposite tendencies:

- in KECCAK, the largest routing-related energy savings in UltraScale+ appear in the iterative variant and then gradually diminish as more pipeline stages are added;
- in BLAKE3, the situation is reversed: routing in the non-pipelined version is the least energy-efficient, and improvements appear only with pipelining.

The conclusions are therefore twofold. First, routing in UltraScale+ is generally more energy-efficient than in UltraScale, except in implementations with very long propagation paths and many logic levels — namely, BLAKE3 variants from X1 to P4 — where the older technology performs better. Second, logic resources in UltraScale+ almost always consume more energy than in UltraScale, making them the dominant factor responsible for the overall worse energy efficiency of UltraScale+ in most designs.

CONCLUSION

The results presented in this work make it possible to assess the actual benefits offered by the two newest FPGA manufacturing technologies in various implementations of the two ciphers. Since the logical structure of the fabric has remained essentially unchanged, the newer generations do not provide noticeable savings in resource utilization. Their advantages instead manifest in two other areas: operating speed and energy characteristics, where the 20-nm and 16-nm processes clearly outperform the older 28-nm technology.

With respect to achievable clock frequency, both UltraScale and UltraScale+ deliver a substantial improvement over 7 Series. Moreover, UltraScale+ proves to be not only faster but also more predictable: all implementations show a consistent increase in maximum frequency, regardless of algorithm or pipeline depth. This uniformity suggests that the architecture of UltraScale+ handles diverse logic structures more effectively than its predecessor.

The situation is more nuanced in the domain of dynamic power consumption. While both newer families generally reduce energy usage relative to 7 Series, the magnitude of improvement varies significantly across designs. In particular, the iterative BLAKE3 variant benefits the most from the transition to newer devices, whereas the gains from pipelining become less dramatic than in the older generation. A detailed comparison between UltraScale and UltraScale+ reveals that the latter does not universally improve energy efficiency; in fact, in most implementations UltraScale+ consumes more dynamic power. The primary source of this effect appears to be the increased energy usage of logic resources, partially offset by more efficient routing.

Overall, the study demonstrates that although UltraScale and UltraScale+ offer clear performance advantages, their energy-efficiency benefits are dependent on the structure of the implemented algorithm. For cryptographic constructions with deep combinational logic — such as BLAKE3 — the improvements are less uniform and require some consideration when selecting the target FPGA family.

When a designer has access to three FPGA devices from the same family but manufactured in different process technologies, the need arises to define criteria for making an informed choice, and the material collected in this publication provides certain data relevant to that task. It must be emphasized, however, that comparing implementation results across different device families and different algorithmic characteristics always requires caution — differences in tools, fabric structure, or technology characteristics can easily lead to unwarranted generalizations. Nevertheless, the study presented here, based on a uniform set of implementations and a consistent evaluation process, makes it possible to formulate several practical observations. These observations may help assess the three FPGA generations in terms of resource utilization, achievable performance, and energy efficiency, at least in the context of complex, highly combinational applications — such as contemporary cryptographic algorithms.

REFRERENCES

[1] AMD, Inc., “AMD Supports New, Long Lifecycle FPGA Designs through 2040, 2045, and Beyond”, 2025. [Online]. Available: https://community.amd.com/t5/adaptive-computing/ amd-supports-new-long-lifecycle-fpga-designs-through-2040-2045/ba-p/702533. Accessed: Dec. 2025.

[2] J. Sugier, “Intra-round pipelining of KECCAK permutation function in FPGA implementations”, in *Proc. DepCoS-RELCOMEX*, Cham, pp. 606–615, 2020. doi: 10.1007/978-3-030-48256-5_59

[3] J. Sugier, “FPGA implementations of BLAKE3 compression function with intra-round pipelining”, in *Proc. DepCoS-RELCOMEX*, Cham, pp. 319–330, 2022. doi: 10.1007/978-3-031-06746-4_31

[4] J. Sugier, “Power analysis of BLAKE3 pipelined implementations in FPGA devices”, in *Proc. DepCoS-RELCOMEX*, Cham, pp. 295–308, 2023. doi: 10.1007/978-3-031-37720-4_27

[5] J. Sugier, “Comparison of power consumption in pipelined implementations of the BLAKE3 cipher in FPGA devices”, *Int. J. Electron. Telecommun.*, vol. 70, no. 1, pp. 23–30, 2024. doi: 10.24425/ijet.2023.147710

[6] J. Sugier, “Reducing Power of BLAKE3 Implementations with Dedicated FPGA Resources”, *Int. J. Electron. Telecommun.*, vol. 71, no. 3, pp. 1–7, 2025. doi: 10.24425/ijet.2025.153622

[7] J. Sugier, “Implementing cryptographic algorithms across various generations of FPGA devices – a case study”, in *Proc. DepCoS-RELCOMEX*, Cham, pp. 209–220, 2024. doi: 10.1007/ 978-3-031-92734-8_21

[8] AMD, Inc., “7 Series FPGAs Configurable Logic Block User Guide”, UG474.PDF, 2025. Available: www.amd.com. Accessed: Dec. 2025.

[9] AMD, Inc., “UltraScale Architecture Configurable Logic Block User Guide”, UG574.PDF, 2025. [Online]. Available: www.amd.com. Accessed: Dec. 2025.

[10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “The KECCAK reference”, 2011. [Online]. Available: https://keccak.team/files/Keccak-reference-3.0.pdf. Accessed: Dec. 2025.

[11] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, “SHA-3 proposal BLAKE, version 1.3”, 2010. [Online]. Available: https://www.aumasson.jp/blake/blake.pdf. Accessed: Dec. 2025.

[12] National Institute of Standards and Technology, “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions”, FIPS 202, 2015. doi: 10.6028/NIST .FIPS.202

[13] J. O’Connor, J.-P. Aumasson, S. Neves, and Z. Wilcox-O’Hearn, “BLAKE3: one function, fast everywhere”, presented at Real World Crypto, 2020. [Online]. Available: https://github.com/BLAKE3-team/BLAKE3-specs/blob/ master/blake3.pdf. Accessed: Dec. 2025.

[14] AMD, Inc., “Vivado Design Suite User Guide: Power Analysis and Optimization”, Tech. Rep. UG907, 2025. [Online]. Available: www.amd.com. Accessed: Dec. 2025.

[15] E. Boemo, J. Oliver, and G. Caffarena, “Tracking the pipelining-power rule along the FPGA technical literature”, *FPGAworld Conference – Academic Proceedings*, pp. 1–6, 2013. doi: 10.1145/2513683. 2513692