Comparison of Multi-Purpose Cores of Keccak and AES

Panasayya Yalla, Ekawat Homsirikamol, Jens-Peter Kaps
Department of Electrical and Computer Engineering, George Mason University, Fairfax,Virginia 22030, U.S.A
{pyalla, ehomsiri, jkaps}@gmu.edu http://cryptography.gmu.edu

***Abstract*—Most widely used security protocols, Internet Protocol Security (IPSec), Secure Socket Layer (SSL), and Transport Layer Security (TLS), provide several cryptographic services which in turn require multiple dedicated cryptographic algorithms. A single cryptographic primitive for all secret key functions utilizing different mode of operations can overcome this constraint. This paper investigates the possibility of using AES and Keccak as the underlying primitives for high-speed and resource constrained applications. Even though a plain AES implementation is typically much smaller and has a better throughput to area ratio than a plain Keccak, adding additional cryptographic services changes the results dramatically. Our multi-purpose Keccak outperforms our multi-purpose AES by a factor of 4 for throughput over area on average. This underlines the flexibility of the Keccak Sponge and Duplex functions. Our multi-purpose Keccak achieves a throughput of 23.2 Gbps in AE-mode (Keyak) on a Xilinx Virtex-7 and 28.7 Gbps on a Altera Stratix-IV. In order to study this further we also implemented two versions of a dedicated Keyak and dedicated AES-GCM. Our dedicated Keyak implementation outperforms our dedicated AES-GCM on average by a factor 6 in terms of throughput over area reaching a throughput of 28.9 Gbps and 4.1 Gbps respectively on a Xilinx Virtex-7.**

I. Introduction and Motivation

IPSec [1], SSL [2] and TLS [3] provide the following services: **Integrity** is provided through a **Hash Function** which takes a variable-length input message M and produces a fixed length output which is called hash H. The length of the message is denoted by $|M|$. **Authentication** and integrity can be provided by a **Message Authentication Code** (MAC). It takes the same inputs as a hash function and additionally a secret key K. The output of the MAC is an authentication tag T. **Confidentiality**, integrity, and authentication can simultaneously be provided by **Authenticated Encryption** (AE). AE schemes have the same inputs as MACs and generate, in addition to T, the encrypted message called cipher text C. AE schemes, that also support authentication of data that is associated with the message AD, are called Authenticated Encryption with Associated Data (AEAD) schemes. Some cryptographic algorithms require an initialization vector IV as an additional input. **Pseudo Random Number Generators** are used by these protocols for providing secret keys and nonces. PRNGs use a random seed S as input and generate a random string R.

Several of these services could be provided by a single secret key algorithm such as AES [4] through application of several modes of operation. While AES is a natural choice for an all-in-one implementation due to its popularity, Keccak [5], specifically Keccak's f-permutation, is also a very interesting option. Keccak is the winner of the competition for the next Secure Hash Algorithm (SHA-3). Its versatile f-permutation [6] allows it to operate in multiple modes to support various cryptographic services needs. Furthermore, the Keccak f-function is also the basis of two candidates of the cryptographic Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) namely Ketje and Keyak.

TABLE I: AES / Rijndael* and Keccak Modes
(Rd. = Number of rounds)

	Operation	Mode	Block	Key	Rd.	ρ	Inputs	Outputs
AES	Hash*	AES-Hash	256	N/A	14		$\|M\|$, M	H
	MAC	CMAC	128	128	10		$\|M\|$, M, K, IV	T
	AEAD	GCM	128	128	10		$\|M\|$, M, K, IV, $\|AD\|$,AD	T, C
	PRNG	Fortuna	128	N/A	14		S	R
Keccak	Hash	Sponge	1600	N/A	24	1088	$\|M\|$, M	H
	MAC	Sponge	1600	128	24	1088	$\|M\|$, M, K, IV	T
	AEAD	Duplex	1600	128	12	1344	$\|M\|$, M, K, IV, $\|AD\|$,AD	T, C
	PRNG	Duplex	1600	N/A	12	1344	S	R

II. Modes of Operation

AES: We chose AES-Hash [7] which has been proposed to NIST as a mode of operation for hashing. It is a variant of Davies-Meyer [8] and uses Rijndael with a block size of 256-bit and a 256-bit key. The Cipher based Message Authentication Code (CMAC) [9] is a NIST recommended mode of operation for authentication and is equivalent to OMAC1, a variation of One-Key CBC-MAC (OMAC). For AE schemes, NIST recommends Galois/Counter Mode (GCM) [10]. Fortuna [11] was developed as a cryptographically secure PRNG mode.

Keccak: is a family of cryptographic hash functions which maps a variable-length input to variable-length output using a fixed length permutation called f-permutation. All the modes we present in this paper for Keccak are based on Sponge construction for Hash and MAC and Duplex construction for PRNG and AEAD. We chose Keccak as our hashing mode. In MAC mode, Key and IV are processed as message to generate the initial state. Then the message is hashed to produce the MAC. The AE mode, Keyak, produces a key stream after processing Key and IV. The key stream is XORed with the message to produce the cipher text C. This process is repeated until the last block of the message. The output after processing the last block is T. In PRNG mode, a nonce is used as an initial seed. It then produces pseudo random bits up to a maximum number determined by the bit rate (r). Additional random bits can be generated through repeated calls to the f-function with an empty block as input. Table I shows the parameters for each mode of AES and Keccak.

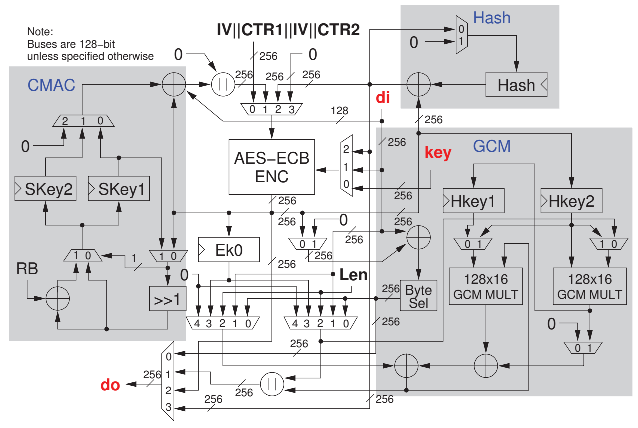

Fig. 1: High-Speed Datapath of AES

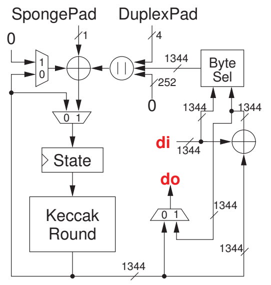

Fig. 2: High-Speed Datapath of Keccak

III. Design Decisions

We designed one high-speed (HS) and one low-area (LA) architecture for each Keccak and AES. These two architectures support all modes for Hash, MAC, AEAD, and PRNG. Analyzing the results of these implementations led us to additionally designing two dedicated implementations of each Keyak and AES-GCM. The HS architectures use the native datapath width of the respective algorithm, i.e. 1600 bits for Keccak and 128 bits for AES. The datapath width of the LA architecture for AES is 32 bits as this is the width of the largest single operation: MixColumn. For Keccak we used a datapath width of 64 bits which is the width of a word in Keccak. All architectures have the secondary design goal of high throughput to area ratio. We assume that all input data must be a multiple of a byte. Padding of messages is performed in hardware. Input data is assumed to be zero padded if not a multiple of the I/O width. For more details about padding, we refer to [5], and [12]. Keccak-r is chosen to be 1088-bit, which is the recommended bit rate that would provide the same level of security as SHA-256. To keep the design simple, the size of key, IV and seed are fixed to 128 bits.We consider that 128-bit key and IV provide adequate security margin. The interface used is based on [13] and [14]. The data input (DI) and outputs (D0) are designed to operate with FIFOs. The width of these interfaces w is set to 16-bit and 128-bit for low-area and high-speed architectures, respectively.

IV. Implementations of AES

High-speed architecture: Our base design contains two AES-128 encryption only cores that can support key sizes of 128- and 256-bit. The two AES-128 cores also have the capability to form a single Rijndael-256 core for the AES-Hash. All operations are the same for AES-128 and Rijndael-256 with the exception of ShiftRow. The top level datapath of multi-AES is shown in Fig. 1. The non-shaded region represents the shared path amongst different modes of operation. The shaded region on the left is dedicated for CMAC which does not support parallel processing. CMAC requires pre-computation of SubKey1 from AES round. Then SubKey2 is derived from SubKey1. The right hand side top shaded region represents the resources required for AES-Hash and the bottom shaded region for AES-GCM. As we are processing two blocks of data at a time, two 128x16 Galois field multipliers are used with Hkey1 calculated by doubling Hkey2.

Low Area Architecture: The LA AES datapath has two dual-port 32-bit wide RAMs with dedicated read and write ports are used to store various inputs and state variables. These RAMs are actually a combination of four 8-bit wide RAMs which allows splitting of 32-bit words into four individual bytes. This way of storing the state allows to perform the shift-row operation by addressing. It takes four clock cycles to perform one round of AES. In case of AES-Hash, it takes eight clock cycles for one-round due to the 256-bit block size. The original key (K), round key (K_{rnd}), and two subkeys used in AES-CMAC are stored in an additional RAM. The multiplication in AES-GCM is performed using a 128x2 multiplier and two 128-bit registers. It takes 64 clock cycles to perform one 128x128 multiplication. The dedicated AES-GCM design is a reduced version of multi-purpose core without additional hardware to support other modes.

V. Implementations of Keccak

High-Speed Architecture: The design shown in Fig. 2 is based on the single round iterative architecture from [15]. The input data block is extended to 1344 bits in order to support the duplex mode which was not available in our base architecture. An XOR-unit is added to provide the encryption/decryption capability and a multiplexer to select which part of the datapath provides the output. The new byte selection unit (*Byte Sel*) enables the selection of input data or encrypted data on a byte basis. A DuplexPad signal is combined with the output from the byte selection unit and injected into the state via an XOR. At this point, SpongePad signal can be activated to provide proper padding. It must be noted that the difficulties in adding support for multiple modes of operation for Keccak are the support for non-uniform block size coupled with the need for an internal padding unit. These difficulties cause the controller for the input loading module of Keccak to be more complex than for AES, even though Keccak has a simpler datapath. Due to the reduced round requirement of Keccak in Duplex mode used in AEAD and PRNG operations, Keccak requires at least 128-bit I/O width in order to achieve its maximum theoretical throughput.

Low-Area Architecture: Two dual-port distributed RAMs with dedicated read and write ports are used to store the state matrix along with all the other state variables. The state variables C and D in the θ step of f-function are computed using a register and a couple of multiplexers. It takes 14 clock cycles to compute the 5 state variables C_0 to C_4 and 6 clock cycles to compute other 5 state variables D_0 to D_4. These 10 state variables are stored in both RAMs as they are required for both even and odd state words. The π step is performed by means of addressing the words. The 25 different cyclic rotations in ρ step are performed using three pipelined 4x1 multiplexers and the χ step using three registers. All together, these three steps are computed in 39 clock cycles. To conserve resources only 6-bits of each of the 24 round-constants in ι step are stored in memory as the remaining bits are all zeros. No additional clock cycles are required for ι step. In total it takes 58 clock cycles to perform one round operation. Through scheduling the operations of two consecutive rounds, the total clock cycles for 24 round is reduced to 1323 from the expected 1392 clock cycles. An additional single-port RAM is used to store the Key, Seed and IV. Since the sizes of Key, Seed and IV are fixed, padding for them requires minimal additional logic. Hence it is included in the core. The dedicated Keyak design is derived from the multi-purpose Keccak with minimal change.

VI. Results

All of our results are after place-and-route and were generated using Automated Tool for Hardware EvaluatioN (ATHENa) [16] with Xilinx ISE 14.7 and Quartus II 13.1. Throughput (TP) is calculated for long messages i.e the number of clock cycles required for initialization, preprocessing of Key and IV are assumed to be zero. None of our designs utilize embedded resources for ease of comparison. Using embedded resources improves the performance of AES, however it would degrade the performance of Keccak as shown in a previous study [17]. Some results reported by others, especially the ones from the industry, may include them as not all information is provided. We implemented our designs on Xilinx Virtex-5, Spartan-6, Virtex-6, Artix-7, and Virtex-7 and Altera Cyclone-IV and Stratix-IV FPGAs. Detailed results for high-speed implementations on Xilinx Virtex-7 and for low-area implementations on Xilinx Artix-7 are shown in Table II. Due to space constraints, results for other FPGAs are not shown in detail. All performance comparisons are made with respect to TP/Area.

Figure 3 shows the performance of our multi-Keccak implementations relative to the multi-AES implementations for all modes of operation on all devices. In almost every case, the performance of multi-Keccak is much better, up to 14 times, than of multi-AES. In terms of throughput, this is not surprising as the width of the Keccak datapath in high-speed designs is 12.5 times wider than AES and 2 times for low-area designs while the number of rounds is similar. However, this increase in datapath width does not come with an increase in area, leading to much better TP/Area results. This is due to the fact that AES modes of operation have vastly different underlying characteristics. As a result, resource sharing is not possible. On the other hand, the primary difference between Keccak modes is how the input blocks are formatted. Hence, Keccak requires minimal additional resources. In case of low-area designs the performance of Keccak in AEAD mode stands out. The reason is the number of clock cycles required for the AES-GCM multiplier.

Our dedicated implementations of Keyak outperform our AES-GCM implementations in a similar way as multi-Keccak outperforms multi-AES (Fig. 4). For low-area implementations the relative performance of Keyak is higher than multi-Keccak in AEAD mode. However, for high-speed designs the opposite is true. That is due to the fact that multi-AES employs a dual-core AES while AES-GCM has only a single AES core.

Comparison with results from literature, which are on Xilinx Virtex-5, are reported in Table III. As there are no reported results of a design which can be operated in all the four modes of operations, we compare our results with designs that utilize block ciphers operating in a specific mode. Our high-speed multi-Keccak has a reduced performance in terms of TP/Area compared to the dedicated hash core in [15]. A huge contribution to this is the increase in area due to the additional modes of operation. The slower frequency is due to the additional padding unit used in our designs. For MAC, our high-speed multi-AES still trails behind the design for GMAC [18] while multi-Keccak can compete against the previous

TABLE II: Results of AES and Keccak Implementations

Mode	Design	Area (Slices)	Freq. (MHz)	TP (Gbps)	TP/Area (Mbps/ Slices)
High-Speed Designs on Xilinx Virtex-7 FPGA					
Hash	Multi-AES	3061	188.18	3.212	1.049
	Multi-Keccak	2495	206.70	9.370	3.756
MAC	Multi-AES	3061	188.18	2.190	0.715
	Multi-Keccak	2495	206.70	9.370	3.756
AEAD	Multi-AES	3061	188.18	4.380	1.431
	Multi-Keccak	2495	206.70	**23.150**	9.279
PRNG	AES-PRNG	3061	188.18	3.212	1.049
	Multi-Keccak	2495	206.70	23.150	9.279
Dedicated AEAD	AES-GCM	1455	352.98	4.107	2.823
	Keyak	2444	258.40	**28.941**	11.841
Low-Area Designs on Xilinx Artix-7 FPGA					
Hash	Multi-AES	629	82.83	0.166	0.263
	Multi-Keccak	264	152.23	0.125	0.474
MAC	Multi-AES	629	82.83	0.189	0.301
	Multi-Keccak	264	152.23	0.119	0.451
AEAD	Multi-AES	629	82.83	0.074	0.117
	Multi-Keccak	264	152.23	0.274	1.037
PRNG	Multi-AES	629	82.83	0.379	0.602
	Multi-Keccak	264	152.23	0.280	1.060
Dedicated AEAD	AES-GCM	548	71.09	0.630	0.115
	Keyak	260	177.87	0.136	1.231

TABLE III: Comparison of our designs with other implementations on Xilinx Virtex-5
(TW = This Work)

	Mode	Design	Area (Slices)	Freq. (MHz)	TP (Gbps)	TP/Area (Mbps/ Slices)
High-Speed	Hash	Multi-AES [TW]	2871	203.29	3.470	1.208
		Multi-Keccak [TW]	2805	163.92	7.431	2.649
		Keccak [15]	1395	281.84	12.777	9.16
	MAC	Multi-AES [TW]	2871	203.29	2.366	0.824
		Multi-Keccak [TW]	2805	163.92	7.431	2.649
		GMAC [18]	9405	120.17	15.382	1.636
	Dedicated AEAD	AES-GCM [TW]	1089	283.53	3.299	3.030
		AES-GCM [19]	678	335.00	2.250	3.319
		AES-CCM [20]	490	274.00	1.525	3.112
		Grøestl/AES [21]	3102	233.00	3.848	1.240
		Keyak [TW]	2357	243.96	27.324	11.593
Low-Area	Hash	Multi-AES [TW]	478	131.23	0.262	0.549
		Multi-Keccak [TW]	318	257.00	0.211	0.665
		Keccak [22]	275	251.25	0.118	0.430
		Keccak [23]	393	159.0	0.864	2.198
	Dedicated AEAD	AES-GCM [TW]	351	130.87	0.116	0.331
		AES-GCM [19]	247	393.00	0.230	0.931
		AES-CCM [20]	214	272.00	0.363	1.696
		Keyak [TW]	259	281.29	0.506	1.954

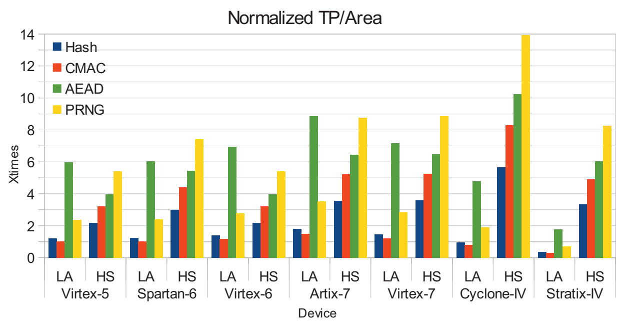

Fig. 3: Performance improvement of multi-Keccak over multi-AES for specific modes of operation

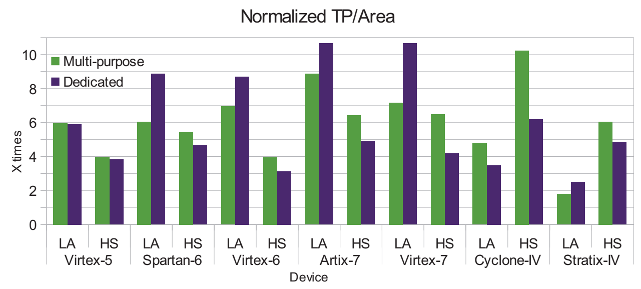

Fig. 4: Performance improvement of dedicated and multi-purpose Keccak over corresponding AES cores for AEAD

work relatively well. The performance of our dedicated high-speed AES-GCM closely matches those of single message AES-based AEAD designs in literature in terms of TP/Area. This validates our claim that adding additional modes to AES-GCM significant impacts its performance. Keyak, on the other hand, outperforms them by as much as a factor of 9. This is largely due to the larger block size with similar number of rounds as compared to other algorithms. When comparing our low-area designs in Hash mode, our Multi-Keccak performs better than [22] but not against [23]. We believe that the reduction of performance is due to support for other modes. In case of dedicated modes, our Keyak performs better that [19], [20].Unfortunately, our dedicated AES-GCM does not perform as well as [19] and [20]. However, their implementation details are not known.

VII. Conclusion

Overall, our Multi-Keccak design has a much better TP/Area than our Multi-AES design by about a factor of 4 across all functions and FPGAs as can be seen in Fig. 3. The reason is that even though the AES core can be implemented much more compactly than Keccak f-permutation for high speed design, the addition of modes is much more complex for AES which can be seen in Figs. 1, 2. Also, the throughput of Keccak exceeds AES's on most FPGAs. The maximum throughput for Multi-Keccak AEAD is **23.2 Gbps** on Virtex-7 and **28.7 Gbps** on Stratix-IV. Multi-AES in GCM mode achieves **4.4 Gbps** and **5.6 Gbps** on the same devices respectively. In case of dedicated cores, the maximum throughput for Keyak and AES-GCM are **28.9 Gbps** and **4.1 Gbps** on Virtex-7 respectively. All in all, this clearly shows that Keccak is more suitable than AES as a basis for multi-service functions.

References

[1] S. Frankel, K. Kent, R. Lewkowski, A. D. Oerbaugh, R. W. Ritchey, and S. S. R., *Guide to IPsec VPNs*, NIST, SP 800-77, Dec 2005.

[2] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (SSL) protocol," IETF, RFC 6101, Aug 2011.

[3] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," Network Working Group, RFC 5246, Aug 2008.

[4] *Advanced Encryption Standard (AES)*, NIST, FIPS Publication 197, Nov 2001.

[5] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The Keccak SHA-3 submission," Submission to NIST (Round 3), 2011, http://keccak.noekeon.org/Keccak-submission-3.pdf.

[6] ——, "Cryptographic sponge function," http://sponge.noekeon.org/CSF-0.1.pdf, Jan 2011.

[7] B. Cohen and B. Laurie, *AES-Hash*, Submission to NIST, May 2001, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf.

[8] S. Matyas, C. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," IBM Tech. Disclosure Bulletin, 5658–5659, Tech. Rep. 27, 1985.

[9] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST, SP 800-38B, Mar 2005.

[10] ——, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication*, NIST, SP 800-38D, Apr 2006, draft.

[11] R. McEvoy, J. Curran, P. Cotter, and C. Murphy, "Fortuna:cryptographically secure pseudo-random number generation in software and hardware," in *Irish Signals and Systems Conference, 2006 IET*, June 2006, pp. 457–462.

[12] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. V. Keer, "Caesar submission: Keyak v1," http://competitions.cr.yp.to/round1/keyakv1.pdf, Mar 2014.

[13] *Hardware Interface of a Secure Hash Algorithm (SHA)*, v. 1.4 ed., Cryptographic Engineering Research Group, George Mason University, Jan 2010.

[14] K. Gaj, E. Homsirikamol, and M. Rogawski, "Fair and comprehensive methodology for comparing hardware performance of fourteen round two SHA-3 candidates using FPGA," in *CHES*, ser. LNCS, S. Mangard and F.-X. Standaert, Eds., vol. 6225. Springer, 2010, pp. 264–278.

[15] E. Homsirikamol, M. Rogawski, and K. Gaj, "Throughput vs area trade-offs architectures of five round 3 SHA-3 candidates implemented using Xilinx and Altera FPGAs," in *CHES*, ser. LNCS, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, Sep 2011, pp. 491–506.

[16] K. Gaj, J.-P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, and B. Y. Brewster, "ATHENa – automated tool for hardware evaluation: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs," in *FPL*. IEEE, 2010, pp. 414–421.

[17] M. U. Sharif, R. Shahid, M. Rogawski, and K. Gaj, "Use of embedded FPGA resources in implementations of five round three SHA-3 candidates," ECRYPT II Hash Workshop 2011, May 2011.

[18] Y. Lu, G. Shou, Y. Hu, and Z. Guo, "The research and efficient fpga implementation of ghash core for gmac," in *E-Business and Information System Security, 2009. EBISS '09*, 2009, pp. 1–5.

[19] *AES-GCM Core family for Xilinx FPGA*, Helion Technology, Fulbourn, Cambridge CB21 5DQ, England, 2011, http://www.heliontech.com/downloads/aes_gcm_8bit_xilinx_datasheet.pdf.

[20] *AES-CCM Core family for Xilinx FPGA*, Helion Technology Limited, Fulbourn, Cambridge CB21 5DQ, England, 2011, http://www.heliontech.com/downloads/Helion_PB_-_AES-CCM_8-bit_FPGA.pdf.

[21] M. Rogawski, "Development and Benchmarking of New Hardware Architectures for Emerging Cryptographic Transformations," Ph.D. dissertation, George Mason University, July 2013.

[22] J.-P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, and S. Gurung, "Lightweight implementations of SHA-3 finalists on FPGAs," Mar 2012, third SHA-3 Candidate Conference.

[23] B. Jungk and J. Apfelbeck, "Area-efficient FPGA implementations of the SHA-3 finalists," in *International Conference on ReConfigurable Computing and FPGAs*. IEEE: ReConfig'11, DEC 2011.