# VLSI Implementations of the Cryptographic Hash Functions MD6 and ïrRUPT

L. Henzen*, F. Carbognani*, J.-Ph. Aumasson†, S. O'Neil‡, and Wolfgang Fichtner*

*IIS ETH Zurich, Switzerland
henzen@iis.ee.ethz.ch

†FHNW Windisch, Switzerland
jeanphilippe.aumasson@gmail.com

‡VEST Corporation, France
s.oneil@vest.fr

***Abstract*— A public competition organized by the NIST recently started, with the aim of identifying a new standard for cryptographic hashing (SHA-3). Besides a high security level, candidate algorithms should show good performance on various platforms. While an average performance on high-end processors is generally not critical, implementability and flexibility in hardware is crucial, because the new standard will be implemented in a variety of lightweight devices. This paper investigates VLSI architectures of the SHA-3 candidates MD6 and ïrRUPT. The fastest circuit is the $16\times$parallel MD6 core, reaching 16.3 Gbps at a complexity of 69.8 k gate equivalents (GE) on ASIC and 8.4 Gbps using 4465 Slices on FPGA. However, large memory requirements preclude the application of MD6 to resource-constrained systems. The most flexible and efficient circuit turns out to be our 2-ïrRUPT64x2-256/8 core, which achieves a throughput of 5.0 Gbps at 12.7 kGE on ASIC and 1.7 Gbps using 613 Slices on FPGA.**

## I. Introduction

Cryptographic hash functions are ubiquitous algorithm used in numerous schemes like digital signatures, public-key encryption, or MAC's. Hash functions process an arbitrary-length message to produce a small fixed-length digital fingerprint, and should satisfy a variety of security properties (preimage resistance, collision resistance, pseudorandomness, etc.). In the last years, a wide range of attacks have been applied to the previous standards MD5 and SHA-1, to break their collision resistance [1], [2]. Although only collisions in reduced versions of the current standard SHA-2 are known [3]–[5], researchers are skeptical about its long-term security. As a response, the U.S. National Institute of Standards and Technologies (NIST) recently launched a call for candidate functions for a new cryptographic hash algorithm (SHA-3) family [6]. The hash functions MD6 [7] (by the author of MD5) and ïrRUPT [8] have been accepted as Round 1 candidates.

Besides a high security, the new hash standard should be suitable for implementations on a wide range of applications. In particular, hardware efficiency will be crucial to determine the future SHA-3, because hardware resources are often limited, whereas on high-end PC's it does not matter much in general; indeed, even the slowest hash function has acceptable performance on a PC. Furthermore, hash function designers seldom study the hardware performance. It is thus necessary to independently study implementations of future candidates on ASIC and FPGA, and determine their suitability for resource-limited environments.

*Contributions:* This paper presents five hardware architectures for the hash functions MD6 and ïrRUPT. Particular attention has been drawn in the analysis of the round process to exploit parallelism, to maximize the circuit speed. A comparison between the analyzed circuits and other cryptographic hash primitives is provided. To the best of the authors' knowledge this is the first paper that investigates hardware implementations of SHA-3 candidates.

*Outline:* The remainder of this paper is organized as follows. Sec. II defines the algorithm specification of MD6 and ïrRUPT. Sec. III describes our methodologies and the parallelization process used in our implementations. Sec. IV reports our results along with a comparison to previous hash function, and Sec. V draws the conclusions.

## II. Algorithm Specification

The hash functions considered produce a $n$-bit hash value by processing fixed-length blocks of the variable-length message. In some cases, before starting to hash the message, the final block needs to be completed, in order to reach the required length. This procedure, referred to as message padding, is a common preprocessing operation in several hash functions.

### A. The MD6 Hash Function

Messages processed by the MD6 hash function are parsed in words of $w$=64 bits. The first distinctive feature of MD6 is its mode of operation. The standard operation mode builds a quadtree structure (in the spirit of Merkle trees), which allows the parallel computation of large messages. Every leaf node of the tree computes an intermediate chaining value, which is passed to a parent node with other three chaining values, coming from other leaf nodes. The final digest is computed in the root node of the tree. However, MD6 has also been specified to work in a sequential mode (where the level of the tree is $L$=0) similar to the classical iterated hash construction.

The full tree-based and the alternative reduced tree-based modes allow multi-processor computations. However, this feature concerns more the software implementation of MD6 rather than the hardware. From a pure hardware point of view, the main component turns out to be the internal compression function $f_c()$. This defines the speed of the hashing core and mainly contributes to the circuit size. Furthermore, by replicating the compression core and switching from the

**Algorithm 1** MD6 compression function $f_c()$

```
input $A = (a_0, a_1, \dots, a_{88})$
          $= (Q, K, U, V, C^{(-1)}, B^{(0)}, B^{(1)}, B^{(2)})$
set $s =$ 0x0123456789abcdef
    $\text{mask} =$ 0x7311c2812425cfa0
for $i = 89, 90, \dots, 16r + 88$ do
    $x = s \oplus a_{i-89} \oplus a_{i-17}$
    $x = x \oplus (a_{i-18} \wedge a_{i-21}) \oplus (a_{i-31} \wedge a_{i-67})$
    $x = x \oplus (x \gg r_{i-89})$
    $a_i = x \oplus (x \ll l_{i-89})$
    if $(i - 89) \bmod 16 = 15$
        $s = (s \lll 1) \oplus (s \wedge \text{mask})$
return $C = (a_{16r+73}, a_{16r+74}, \dots, a_{16r+88})$
```

sequential to a tree-based mode, the overall throughput of the MD6 circuit can simply be increased.

In the sequential $L$=0 operation mode, the function $f_c()$ compresses four blocks of 16 64-bit words to a 16-word block $C = (c_0, \dots, c_{15})$. The first input block corresponds to the previous computed compression value $C^{(-1)}$ and the last three are message blocks $B^{(j)} = (b_0^{(i)}, \dots, b_{15}^{(i)})$. Before the computation, the 64-word input is prefixed by a constant vector $Q$ of 15 words, a 8-word key $K$, and two additional control words $U$ (unique node ID) and $V$ (parameters). The resulting 89-word input vector $A$ is then compressed to the 16-word $C$ through a sequence of $16r$ steps (see Alg. 1). If the computed compression is the last node (root), its final $C$ value is truncated to the desired $n$-bit digest length. The number of rounds is set as $r = \lfloor n/4 \rfloor + 40$, i.e. MD6-256 makes 104 rounds and MD6-512 168 rounds. The 16-word shift operators $r_i$ and $l_i$ increase diffusion between the input words, while the constants $s_i$, defined by a simple recurrence, reduce self-similarity.

## *B. The ïrRUPT Stream Hashing Mode*

ïrRUPT is the stream hashing mode of the polymorphic cryptographic primitive EnRUPT. The compressed hash value $C$ is computed by processing a $S$-bit message one word at a time, with $w$=64 bits. The EnRUPT stream hash functions have different ïrRUPT$w$x$P$-$n$/$s$ variants, where $P$ and $s$ defines the parallelization end security degree. After collisions were found in the first presented version with $P$=2 and $s$=4, the authors now recommend to use $s$=8 for 256-bit hashing (ïrRUPT64x2-256/8), and $s$=16 for 512-bit hashing (ïrRUPT64x2-512/16).

**Algorithm 2** ïrRUPT stream hashing mode

```
input $M = (m_0, m_1, \dots, m_{\lfloor S/w \rfloor})$
set $H = s, x_{0 \dots H-1} = d_0 = d_1 = r = 0$
for $i = 0, 1, \dots, \lfloor S/w \rfloor$ do    $ïr2s(m_i)$
$ïr2s(n)$
for $i = 0, 1, \dots, H-1$ do    $ïr2s(0)$
for $i = 0, 1, \dots, n/w - 1$ do    $c_i$=$ïr2s(0)$
return $C = (c_o, c_1, \dots, c_{n/w-1})$
```

Alg. 2 depicts the structure of ïrRUPT. The stream construction schedules a first process phase, where the $H$-word internal state $X$ is updated using message words and the round number $r$. The middle stage seals the state, while the output stage sorts a word $c_i$ of the digest every iteration. The *ïr2s(p)* function performs $2s$=8 non-invertible EnRUPT round computations *ïr1* over $X$ and the delta accumulator $D = (d_0, d_1)$. At the end, it returns the second word of $D$ updated and combined with the input word, i.e., $d_1 = d_1 \oplus p$. The basic component of the algorithm is the *ïr1* module, defined as

$$\begin{cases} f & = & [(2x_{r\oplus 1} \oplus x_{r+4} \oplus d_{r\wedge 1} \oplus r) \ggg w/4] \times 9 \\ x_{r+2} & = & x_{r+2} \oplus f \\ d_{r\wedge 1} & = & d_{r\wedge 1} \oplus f \oplus x_r \\ r & = & r + 1 \end{cases} \tag{1}$$

This structure only uses exclusive-ORs (XOR's), word-wise rotations, and modulo $2^w$ additions. Multiplication by nine is turned into the addition of the word with its '$\ll 3$' part.

## III. Hardware Implementation

The basic description of MD6 and ïrRUPT in hardware is based on the implementation of a memory to store the internal state variable ($A$ and $X$, respectively) with in addition one or more combinational units to execute the round function.

## *A. MD6 Architecture*

The MD6 compression function needs at least 89 64-bit registers to store $A$ inside a shift register memory unit. Every cycle the words of $A$ are shifted toward the lower index, erasing the lowest index term and adding the new computed word at the top of the memory. The area estimate of such a 712-byte memory is about 30 kGE. This large size suggests the impossibility of low-area architectures (specifically on ASIC).

Our main goal was to achieve a hardware-efficient high-speed compression core. Analyzing the structure of the loop inside Alg. 1, one observes that 17 new $a_i$ words can be computed in parallel (the lowest tap is 17). Nevertheless, a 16×parallel architectures was designed, where only 16 new words are computed within a clock cycle. This choice is motivated by the generation of the $s$ constants. The same constant is indeed used for 16 steps, after which is updated. Fig. 1 depicts and overview of the implemented architecture.

Memory is implemented as a 89-word 16×shift register. A compression thus requires $r$ iterations, instead of $16r$.

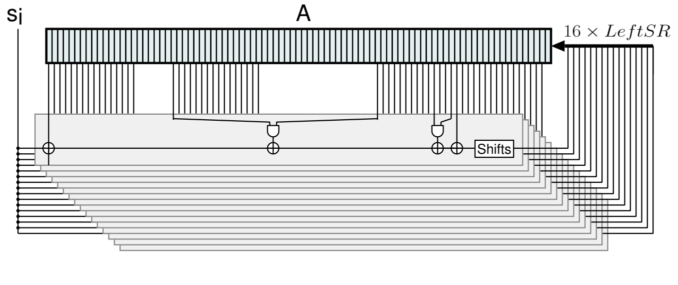


Fig. 1. Block diagram of the 16×parallel MD6 compression function. All connections are 64-bit wide.

Every round module uses simple 64-bit AND, XOR, and shift operators. Furthermore, MD6-256 and MD6-512 are based on the same described architecture, where essentially only the round number $r$ differs.

## *B. ïrRUPT Architectures*

The first investigated ïrRUPT architecture, i.e., $2s$-ïrRUPT, is isomorphic to the ïrRUPT algorithm (see Fig. 2(A)). Since the $P$ parameter is equal to 2, a single double-*ïr1* is composed by two parrallel *ïr1* round functions. Every cycle, the updated $X$ state and the new delta accumulator words $d_i$ are computed and stored inside the $H$+2 $w$-bit memory. The aim of $2s$-ïrRUPT is to achieve a high throughput, by maintaining the computational depth of the algorithm.

The second architecture is based on the iterative decomposition of the *ïr2s*() function (see Fig. 2(B)). A single double-*ïr1* unit has been implemented in combination with the memory. In 2-ïrRUPT, after $s$ clock cycles a complete *ïr2s*() is executed. The final digest words $c_i$ are generated every four cycles, during the output phase.

In both architectures, the double-*ïr1* unit consists in a progressive word-shift of $X$, combined with two parallel *ïr1* blocks (see Fig. 3). As for the implementation of MD6, the *ïr1* modules take as inputs always the same $x_i$ words. Hence, according to the algorithm, every word is shifted by two every cycle. This final word shift operation is equivalent to the increment of the index variable $r$ by two.

## IV. Results and Performance Comparison

The speed of a hash circuit is the result of the operational frequency $f$ multiplied by the mean number of processed message bits per clock cycle. This last term is the quotient of the length of the input message $S$ and the number of cycles $n_{lat}$ needed to compute the digest. In standard block hashing schemes, e.g., SHA-2, MD5, $n_{lat}$ corresponds to the product between the latency in cycles of a single compression process and the number of executed compressions. The throughput in our MD6$_{L=0}$ architecture is, therefore, described by

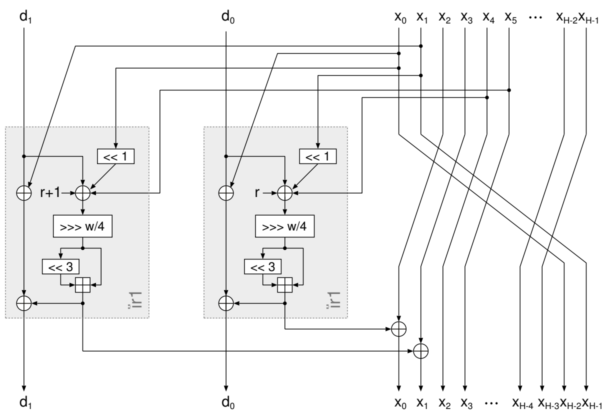
$$T = f\frac{S}{n_{lat}} = f\frac{S}{r\left\lceil \frac{S}{3\cdot 16w}\right\rceil}. \tag{2}$$

Conversely, the ïrRUPT stream structure is divided into three functional stages, from which the process stage persists until the whole message is injected, while the other two are defined by $H$ and $n$. This leads to a throughput formula of

$$T = f\frac{S}{n_{lat}} = f\frac{S}{\frac{1}{k}\left(\left\lfloor \frac{S}{w}\right\rfloor + 2 + H + \frac{n}{w}\right)}, \tag{3}$$

where $k$ defines the iterative degree of the architecture. In $2s$-ïrRUPT, $k$ is equal to one, while in 2-ïrRUPT, $k$=1/$s$. Fig. 4 shows the throughput variation of the implemented architectures with increasing message length. Using long sizes, the speed of the ïrRUPT cores converges to $f \cdot k \cdot w$, while in the MD6$_{L=0}$ core, to $f \cdot (3 \cdot 16w)/r$.

The throughput results with the area requirements of the circuits are summarized in Tab. I and Tab. II. The hashing cores have been coded in functional VHDL and synthesized with



d1
d0
x0 x1 x2 x3 x4 x5 ··· xH-2 xH-1
<< 1
r+1
>>> w/4
<< 3
ïr1
<< 1
r
>>> w/4
<< 3
ïr1
d1
d0
x0 x1 x2 x3 ··· xH-4 xH-3 xH-2 xH-1

Fig. 3. Word-shift structure of the double-*ïr1* module. All connections are $w$-bit wide.

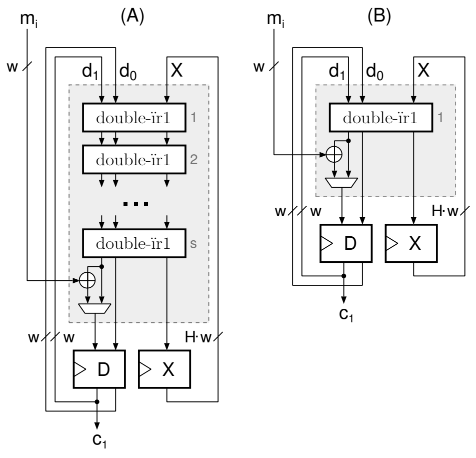


Fig. 2. Block diagrams of 16/32-ïrRUPT (A), and 2-ïrRUPT (B) architectures.

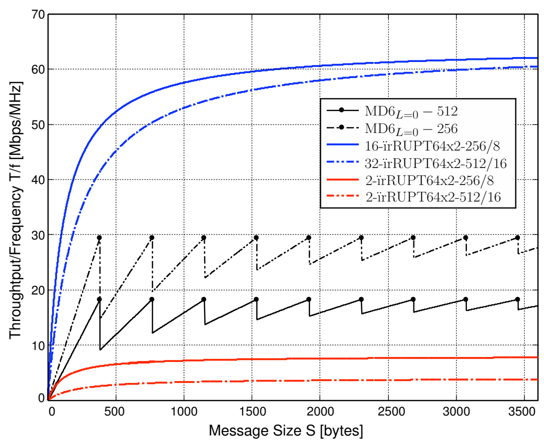


Fig. 4. Throughput/message size trade-off of the implemented architectures. The black points corresponds to ideal message sizes, i.e., that are multiple of the block length (padding is not required).

TABLE I

ASIC IMPLEMENTATION RESULTS FOR A 0.18 μM CMOS TECHNOLOGY.

| Ref. | Function | Area [kGE] | | Frequency [MHz] | $n_{lat}$ [Cycles] | Throughput [Gbps] | | HW-eff. [Kbps/GE] | |
|---|---|---|---|---|---|---|---|---|---|
| | | Memory (ratio) | Total | | | 1 kbit $S$ | Long $S$ | 1 kbit $S$ | Long $S$ |
| Ours | MD6$_{L=0}$-256 | 33.13 (47 %) | 69.78 | 552 | $104\lceil S/(48w)\rceil$ | 5.440 | 16.320 | 78.0 | 233.9 |
| Ours | MD6$_{L=0}$-512 | 33.13 (47 %) | 69.78 | 552 | $168\lceil S/(48w)\rceil$ | 3.368 | 10.103 | 48.3 | 144.8 |
| Ours | 16-ïrRUPT64x2-256/8 | 4.22 (7 %) | 57.86 | 99 | $\lfloor S/w\rfloor + 14$ | 3.363 | 6.305 | 58.1 | 109.0 |
| Ours | 2-ïrRUPT64x2-256/8 | 4.30 (34 %) | 12.73 | 621 | $8(\lfloor S/w\rfloor + 14)$ | 2.650 | 4.969 | 208.1 | 390.2 |
| Ours | 32-ïrRUPT64x2-512/16 | 7.65 (7 %) | 117.56 | 50 | $\lfloor S/w\rfloor + 26$ | 1.213 | 3.184 | 10.3 | 27.1 |
| Ours | 2-ïrRUPT64x2-512/16 | 7.21 (44 %) | 16.41 | 621 | $16(\lfloor S/w\rfloor + 26)$ | 0.946 | 2.484 | 57.7 | 151.4 |
| [9] | SHA-256 | - | 15.10 | 190 | 72 | 1.349 | | 87.6 | |
| [9] | SHA-512 | - | 30.75 | 170 | 88 | 1.969 | | 64.0 | |
| [9] | Whirlpool | - | 52.79 | 262 | 21 | 6.382 | | 120.9 | |
| [10] | Grøstl-256 | - | 130.64 | 86 | - | 4.379 | | 33.5 | |
| [10] | Grøstl-512 | - | 340.50 | 85 | - | 6.225 | | 18.3 | |
| [11] | BLAKE-32 | 4.79 (12 %) | 41.31 | 170 | 21 | 4.153 | | 100.5 | |
| [11] | BLAKE-64 | 6.13 (7 %) | 82.73 | 136 | 29 | 4.810 | | 58.1 | |

the Synopsys Compiler, using a 0.18 μm CMOS technology. The hardware analysis also includes the evaluation on a Xilinx Virtex-4 LX100 FPGA device.

The architecture for MD6$_{L=0}$-256 and MD6$_{L=0}$-512 is the fastest design. It takes advantage of the low combinational logic depth of a step, which translates into high frequency values. A single compression core indeed reaches 16.3 Gbps on ASIC and 8.4 Gbps on FPGA. However, the large memory requirements reduce the adaptability of MD6 to resource-constrained environments, where the available chip area for security features can be assumed between 1 and 10 kGE.

Much more versatile is the ïrRUPT function. The isomorphic 16-ïrRUPT64x2-256/8 core achieves 6.3 Gbps on ASIC and 2.1 Gbps on FPGA, hashing long messages. The best speed/size trade-off is obtained with the 2-ïrRUPT64x2-256/8 architecture, which shows the most compact area 12.7 kGE on ASIC and 613 Slices on FPGA, and the best hardware efficiency.

## V. CONCLUSION

This paper investigated five architectures of the cryptographic hash functions MD6 and ïrRUPT. A close comparison to competitor algorithms stresses an overall increase of the speed performances of the proposed cores. Although the fastest speed values have been obtained with the MD6 circuit, the algorithm construction showed a scarce adaptability to hardware environments, caused by its large memory requirements. From a pure hardware point of view, the ïrRUPT hash function results the suitable algorithm for a wide range of VLSI applications. High-speed systems can integrate the proposed $2s$-ïrRUPT architecture, while for minimum-size environments the 2-ïrRUPT turns out to be the best choice. These statements are confirmed by the computed hardware efficiency ratios of the circuits.

TABLE II

FPGA PERFORMANCE COMPARISON OF THE HASH FUNCTIONS.

| Ref. | Function | Area [Sl. -BRAM] | Freq. [MHz] | Throughput [Gbps] | FPGA Type |
|---|---|---|---|---|---|
| | | | | 1 kbit - Long | |
| Ours | MD6$_{L=0}$-256 | 4,465-0 | 286 | 2.813 - 8.440 | XC4VLX100 |
| Ours | MD6$_{L=0}$-512 | 4,515-0 | 286 | 1.741 - 5.254 | XC4VLX100 |
| Ours | 16-ïrRUPT | 2,957-0 | 32 | 1.101 - 2.065 | XC4VLX100 |
| Ours | 2-ïrRUPT | 613-0 | 213 | 0.908 - 1.702 | XC4VLX100 |
| Ours | 32-ïrRUPT | 6,300-0 | 14 | 0.384 - 0.914 | XC4VLX100 |
| Ours | 2-ïrRUPT | 876-0 | 213 | 0.324 - 0.851 | XC4VLX100 |
| [7] | MD6-512 | 7,529-0 | 142 | 1.894 | XC2VP30 |
| [12] | SHA-256 | 755-1 | 174 | 1.370 | XC2VP |
| [12] | SHA-512 | 1,667-1 | 141 | 1.780 | XC2VP |
| [13] | Whirlpool | 2,118-32 | 220 | 5.380 | XC4V |
| [10] | Grøstl-256 | 6,582- | 130 | 4.439 | XC3S5000 |
| [10] | Grøstl-512 | 20,233- | 340 | 5.901 | XC3S5000 |
| [11] | BLAKE-32 | 2,754-0 | 70 | 2.235 | XC4VLX100 |
| [11] | BLAKE-64 | 6,054-0 | 40 | 1.413 | XC4VLX100 |

## REFERENCES

[1] V. Klima, "Finding MD5 collisions - a toy for a notebook," Cryptology ePrint Archive, Report 2005/075, 2005, http://eprint.iacr.org/.

[2] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *CRYPTO*, ser. LNCS, vol. 3621. Springer, 2005, pp. 17–36.

[3] S. K. Sanadhya and P. Sarkar, "New collision attacks against up to 24-step SHA-2," Cryptology ePrint Archive, Report 2008/270, 2008, http://eprint.iacr.org/.

[4] I. Nikolic and A. Biryukov, "Collisions for step-reduced SHA-256," in *FSE*, ser. LNCS, vol. 5086. Springer, 2008, pp. 1–15.

[5] S. Indesteege, F. Mendel, B. Preneel, and C. Rechberger, "Collisions and other non-random properties for step-reduced SHA-256," in *SAC*, ser. LNCS. Springer, 2008, to appear.

[6] NIST, "Call for a new cryptographic hash algorithm (SHA-3) family," Federal Register, Vol.72, No.212, 2007, http://www.nist.gov/hash-competition.

[7] R. L. Rivest, "The MD6 hash function A proposal to NIST for SHA-3," Submission to NIST, 2008, http://groups.csail.mit.edu/cis/md6/.

[8] S. O'Neil, K. Nohl, and L. Henzen, "EnRUPT hash function specification," Submission to NIST, 2008, http://www.enrupt.com/.

[9] A. Satoh, "ASIC hardware implementations for 512-bit hash function whirlpool," in *Proc. of the Int. Conf. on Circuits and Systems*, May 2008, pp. 2917–2920.

[10] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, "Grøstl - a SHA-3 candidate," Submission to NIST, 2008, http://www.groestl.info.

[11] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "SHA-3 proposal BLAKE," Submission to NIST, 2008, http://131002.net/blake/.

[12] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-efficient SHA hardware accelerators," *IEEE Trans. VLSI Syst.*, vol. 16, no. 8, pp. 999–1008, Aug. 2008.

[13] ——, "Merged computation for whirlpool hashing," in *Proc. of the Conf. on Design, Automation and Test in Europe*, Munich, Germany, 2008, pp. 272–275.