FPGA Implementation of Shabal: Our First Results* (1/15/2010)

Romain Feron and Julien Francq

EADS Defence & Security, Cyber Security Customer Solutions Center (CSCSC)

Abstract. In this short note, we describe a new hardware implementation of `Shabal`. Our results outperform the state-of-the-art. In particular, `Shabal` can achieve a high throughput, and can also be implemented with very low area.
Key words: SHA-3 Contest, `Shabal`, Hardware Implementation, FPGA.

1 Preliminaries

In this section, we summarize the algorithm specifications of `Shabal`.
It uses a sequential iterative hash construction to process messages in blocks of 512 bits (see Figure 1). `Shabal` operates with 32-bit words, so each 512-bit input block M_i is partitioned into 16 words. Its internal state consists of the three components A (384 bits), B and C (both 512 bits). W stores a 64-bit message-block counter. In Figure 1, the boxed plus sign ($\boxplus$) represents a 32-bit addition (modulo 2^{32}), and the boxed minus sign ($\boxminus$) represents a 32-bit subtraction (modulo 2^{32}). Circled plus sign ($\oplus$) represents a bit-by-bit XOR operation.

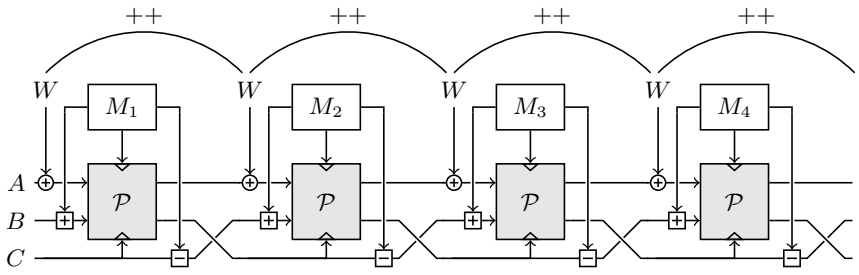

Fig. 1. `Shabal`'s Mode of Operation.

`Shabal` construction makes use of a keyed permutation $\mathcal{P}$ depicted in Figure 2. $\mathcal{P}$ consists of 48 steps, each updating one 32-bit word of A and

* This work was partially supported by the French Agence Nationale de la Recherche through the SAPHIR2 project under Contract ANR-08-VERS-014.

B. Each step (except the first one) involves the result of the previous one, so it can be considered that $\mathcal{P}$ is based on a Non-Linear Feedback Shift Register (NLFSR) construction. It features two multiplications called $\mathcal{U}$ and $\mathcal{V}$ with small constants (resp. 3 and 5) modulo 2^{32}, rotations of words by 1 (denoted $\lll 1$) or 15 bits ($\lll 15$) to the left[1], an AND operation, and additions modulo 2^{32} in order to update A words at the last step of a permutation (not depicted in Figure 2).

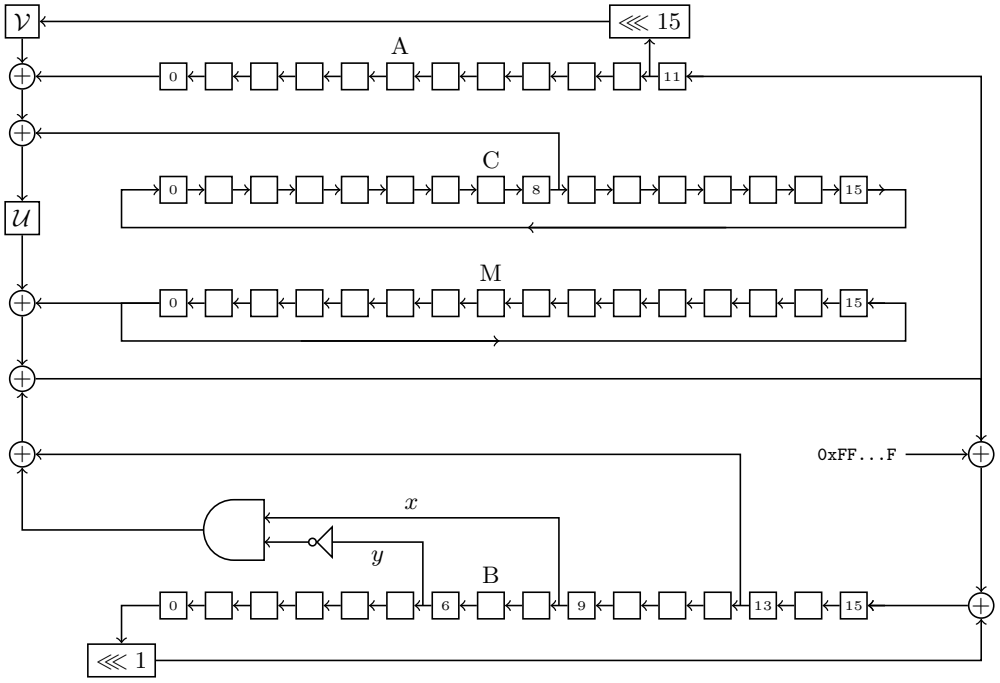

Fig. 2. Permutation $\mathcal{P}$.

2 First Remarks

We want to get a high-speed implementation of `Shabal`. To do so, we have to compute as many calculations as possible in parallel. In that way, at the beginning of the permutation, we can compute in one clock cycle:

- the incrementation of the counter W (see Figure 1),
- $B \leftarrow B + M_w$ (see Figure 1),

[1] A rotation of B words by 17 bits to the left is also needed at the first step of a permutation, but it is not depicted in Figure 2.

- $A \leftarrow A \oplus W$ (see Figure 1),
- $B \leftarrow B \lll 17$ (for each B word).

At the end of a permutation, it is also possible to compute in parallel (and in one clock cycle) all A words which are results of the operation $A \leftarrow A + C$. Finally, the subtraction of the message block $C \leftarrow C - M$ and the swappping of B and C can be computed likewise.
At first sight, it appears that $\mathcal{P}$ has a restricted parallelizability, so further investigations are needed to compute it in less than 48 cycles.

3 Proposed Hardware Architecture

In our high-speed implementation of `Shabal`, we take into account all the remarks given in Section 2. Moreover, we apply the shift-register approach proposed in [BCCM$^+$08] (page 105), and we implement all the operations of $\mathcal{P}$ ($\mathcal{U}$, $\mathcal{V}$, rotations, XORs, additions) in parallel.
The multiplications modulo 2^{32} with 3 and 5 are efficiently implemented using the shift-then-add method.
In this way, our VLSI implementation of `Shabal`'s mode of operation includes:

- $8 \times 32 \times$ 2-input XOR gates,
- 54 32-bit adders[2],
- 1 32-bit subtractor,
- 2×32 NOT gates[3],
- $1 \times 32 \times$ 2-input AND gates,
- 2 32-bit incrementers for W,
- simple wiring for rotation operations.

4 Results and Discussion

We choose to design `Shabal` on a FPGA platform rather than in an ASIC because of its low cost relative to this latter, its greater flexibility, and its easier prototyping process. In order to ease comparisons between our hardware implementation of `Shabal` with the state-of-the-art [BBH$^+$09] [KIM$^+$10], we choose to give results (area and throughput) on the same platform, which is Virtex-5 FPGA [X06].
The post-place and route results for throughput (resp. area-) optimized design presented in this paper is plotted in Table 1 (Table 2).
Table 1 shows that we globally outperform the state-of-the-art: our design takes the least amount of area (i.e. it occupies the least number of FPGA slices), it produces the best throughput (denoted TP), and consequently, the best throughput per slice (which is considered as a hardware efficiency factor).
Except for the throughput, Table 2 exhibits the same results.
Shortly, our `Shabal` implementation currently gives the best overall balance between throughput and area of the state-of-the-art, when implemented on a Virtex-5.

[2] 16 for computing $B \leftarrow B + M_w$ (see Figure 1), 2 for computing $\mathcal{U}$ and $\mathcal{V}$ (see Figure 2) and 36 for computing $A \leftarrow A + C$ at the end of a permutation.

[3] Note that we implement the $\oplus$ module having `0xFF`$\cdots$`F` as one of its inputs (see Figure 2) with 32 NOT gates.

Architecture	**Nb. Slices**	**TP** (Mbps)	**TP/Nb. Slices** (Kbps/slice)
[BBH+09]	2768	1450	523
[KIM+10]	1251	1739	1390
Our Work	**1171**	**2588**	**2210**

Table 1. High-Speed Implementation Results.

Architecture	**Nb. Slices**	**TP** (Mbps)	**TP/Nb. Slices** (Kbps/slice)
[BBH+09]	2307	**1330**	576
Our Work	**596**	1142	**1916**

Table 2. Low-Area Implementation Results.

5 Conclusion

In this short note, we described our first hardware implementation of `Shabal`. In one hand, we have shown that `Shabal` can achieve a high throughput, and can be implemented with very low area in other hand. In a funny way, we can say that, contrary to the French Rugby player `Chabal` (see Figure 3), its homonym `Shabal` can be fast and lightweight.

Fig. 3. `Chabal`: 1.92 m, 115 kg (credits: http://www.flickr.com/photos/sam_herd/2620280308/).

References

[BBH+09] B. Baldwin, A. Byrne, M. Hamilton, N. Hanley, R. P. McEvoy, W. Pan, and W. P. Marnane. FPGA Implemen-

tations of SHA-3 Candidates: CubeHash, Grøstl, LANE, Shabal and Spectral Hash. In *Cryptology ePrint Archive*, number 342, 2009.

[BCCM+08] E. Bresson, A. Canteaut, B. Chevallier-Mames, C. Clavier, T. Fuhr, A. Gouget, T. Icart, J.-F. Misarsky, M. Naya-Plasencia, P. Paillier, T. Pornin, J.-R. Reinhard, C. Thuillet, and M. Videau. Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition Initiated by the Saphir project. 2008.

[KIM+10] K. Kobayashi, J. Ikegami, S. Matsuo, K. Sakiyama, and K. Ohta. Evaluation of Hardware Performance for the SHA-3 Candidates Using SASEBO-GII. In *Cryptology ePrint Archive*, number 010, 2010.

[X06] Production Product Specification of Virtex-E Field Programmable Gate Arrays, Xilinx. 2006.