Hardware Implementation of the SHA-3 Candidate Skein

Stefan Tillich

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
Stefan.Tillich@iaik.tugraz.at

Abstract. Skein is a submission to the NIST SHA-3 hash function competition which has been optimized towards implementation in modern 64-bit processor architectures. This paper investigates the performance characteristics of a high-speed hardware implementation of Skein with a 0.18 µm standard-cell library and on different modern FPGAs. The results allow a first comparison of the hardware performance figures of full Skein with other SHA-3 candidates.

Keywords: SHA-3, Skein, high-speed, hardware, standard-cell library, FPGA.

1 Introduction

The Skein hash function family has been conceived by Ferguson et al. [3] and submitted as a candidate for the SHA-3 competition of NIST [5]. The designers of Skein have optimized the hash function towards performance on 64-bit processors. The first hardware performance figures for Skein have been made available by Long for the core UBI functionality of an iterated implementation on FPGAs [4]. In this paper we report the performance results of our full implementation of all three Skein variants for standard cells and FPGAs.

2 Skein in a Nutshell

Skein is a family of hash functions based on the tweakable block cipher Threefish. The block and key size of Threefish are equal and can be set to either 256, 512, or 1,024 bits (designated as Threefish-256, Threefish-512, and Threefish-1024, respectively). Threefish is used in Matyas-Meyer-Oseas mode to construct the Skein compression function. Together with the format specification of the tweak and a padding scheme, this defines the so-called Unique Block Iteration (UBI) chaining mode. An example of the UBI mode is shown in Figure 1.

In Figure 1, the block TF denotes a Threefish encryption. The message M consists of three message blocks (M_0 to M_2). The first Threefish encryption key (UBI_IN) is supplied to UBI, the tweak values depend on the position and

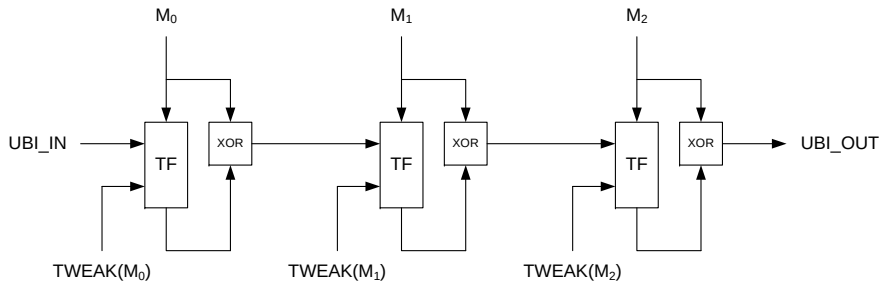

Fig. 1. UBI mode.

bit length of the respective message block. UBI is used in Skein for message compression and the output transformation, but also for IV generation and other optional operation modes (e.g., tree hashing, keyed hashing).

For each block size, the message digest can be set to a more or less arbitrary length. The Skein variant with a block size of X bits and a message digest size of Y bits is designated as Skein-X-Y. As the message digest size is a minimal tweak to an hardware implementation, it is sufficient to investigate the performance characteristics of the three different block sizes X.

The Threefish block cipher is based on three simple operations: Addition modulo 2^{64}, XOR, and bit permutation. These operations are defined on the intermediate state organized in 64-bit words. The MIX operation depicted in Figure 2 transforms two of these 64-bit words and is common to all Threefish variants. The rotation distance (rot_dist) depends on the Threefish block size, the round index and the position of the two 64-bit words in the Threefish state.

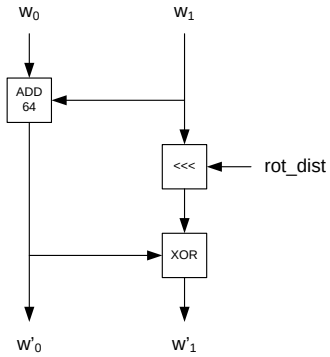

Fig. 2. The Threefish MIX operation.

Threefish is structured in a number of similar rounds which are applied to the input block. A number of subkeys are derived from the cipher key and tweak

in a simple key schedule and are added to the input block and the intermediate state in each fourth round. The total number of rounds depends on the state size: 72 rounds for state sizes of 256 and 512 bits, 80 rounds for Threefish-1024.

Apart from the different rotation distances, the rounds of Threefish are similar, consisting of a layer of MIX operations and a subsequent permutation of the state words (this permutation is fixed for a specific Threefish variant). Each fourth round differs slightly as it also includes the addition of a subkey (using addition modulo 2^{64}). Four consecutive rounds of Threefish-256 are depicted in Figure 3.

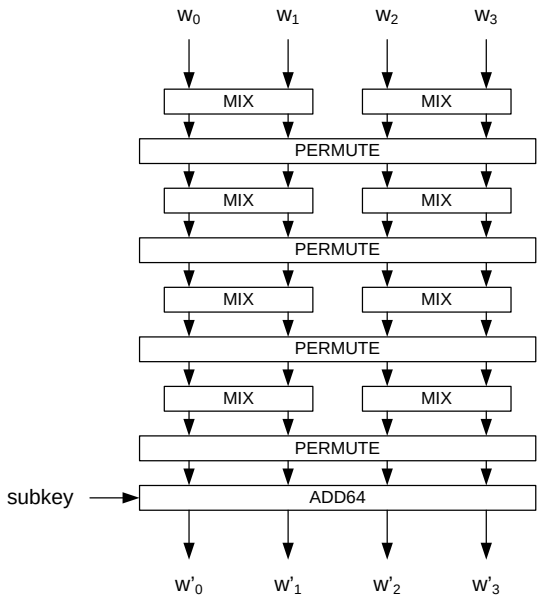

Fig. 3. Four rounds of Threefish-256.

The rotation constants for the MIX operation repeat after eight rounds. Therefore, the output of a hardware block implementing eight consecutive Threefish rounds depends only on the intermediate input block and the according two subkeys. A "natural" design option is thus to unroll a multiple of eight Threefish rounds [3]. Such a design prevents the need for variable-size rotation in hardware, which decreases the hardware cost and increases the speed of the implementation.

3 Description of the Hardware Module

We have implemented the main variants of Skein from scratch in a configurable hardware module. This module can realize either Skein-256-256, Skein-512-512, or Skein-1024-1024.

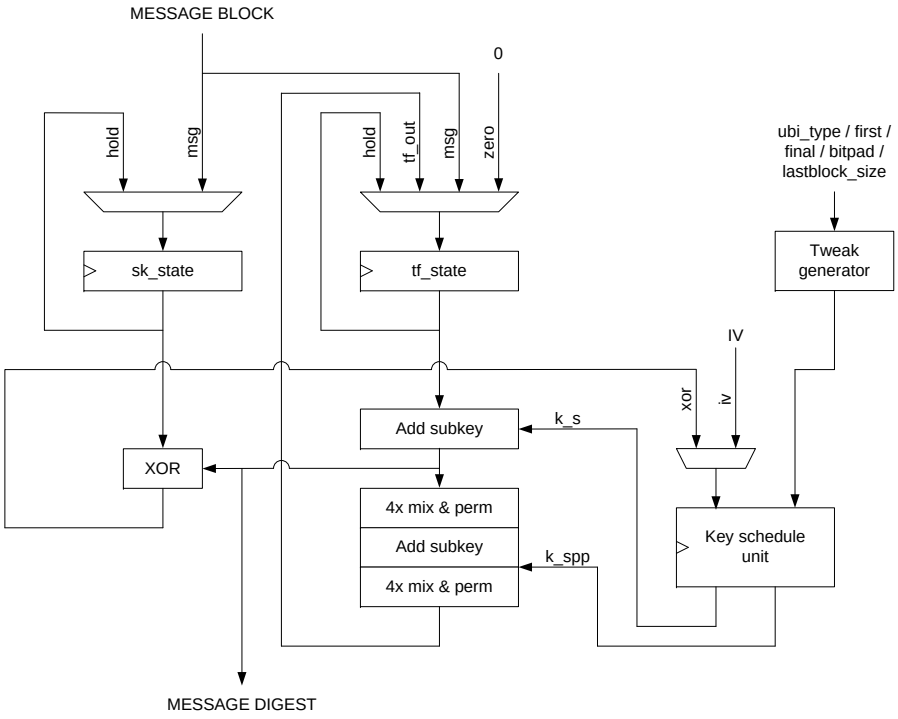

Fig. 4. Datapath of the Skein hardware module with eight unrolled Threefish rounds.

The complete datapath of the Skein module is depicted in Figure 4. The core of the datapath consists of eight unrolled rounds of Threefish and a key schedule unit which can supply two consecutive subkeys at a time. The advantage of this architecture is that the Threefish rounds have fixed rotation distances for their MIX layer, which allows for a compact implementation. Thus, the output of the Threefish unit only depends on the input block and the two subkeys. As the critical path of the whole design runs through the Threefish unit, this design choice helps to improve the overall speed of the design.

The key schedule unit is loaded with an input key (either the IV or the XOR of the previous message block and its Threefish-encrypted equivalent) and input tweak at the beginning of each Threefish encryption. Upon the load, both key and tweak are expanded by a 64-bit word into the extended key and extended tweak, which are stored in registers. Two subsequent subkeys are derived from these registers through a number of 64-bit adders. The contents of the registers can be rotated by a distance of two in order to allow generation of the next pair of subkeys.

Additionally, the datapath contains two registers of the size of a Threefish block. One of these registers saves the current message block while the other is used to hold intermediate values of the Threefish encryption. The XOR of two

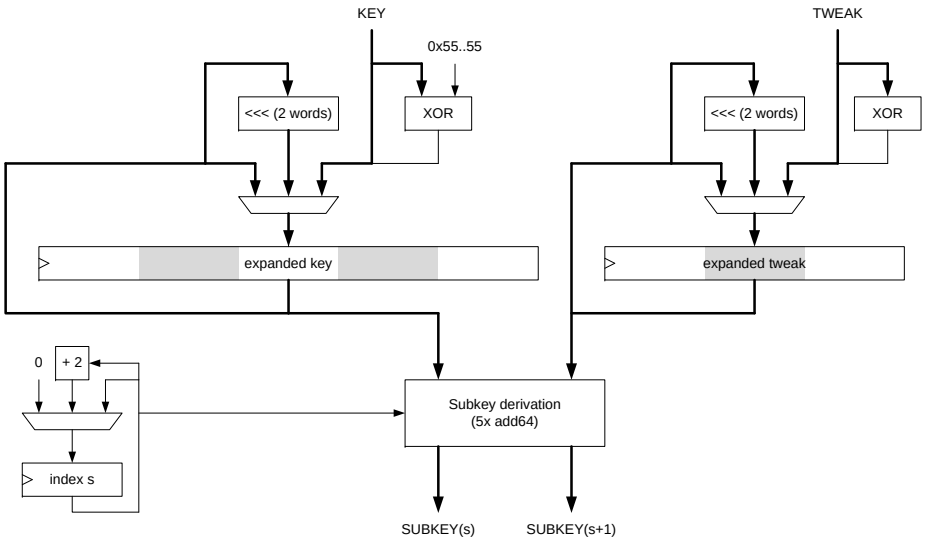

Fig. 5. Key schedule unit for Threefish-256.

Threefish block-sized values—required for UBI chaining—can be loaded into the key schedule unit. A simple tweak generator unit delivers the appropriate tweak for the key schedule unit.

The datapath is controlled by a state machine, which iterates the message block the necessary number of times through the Threefish unit, performs the UBI chaining, loads in new message blocks, keeps track of the number of processed bytes, delivers necessary input signals to the tweak generator, and produces some status signals for the module's external interface.

The external interface of the Skein hardware module has been kept very simple. The message block input and message digest output are implemented in full size in order to prevent any possible performance bottlenecks by a narrow data interface. An interface module could be easily wrapped around the Skein module in order to cater for different data bus widths of the overall system.

The Skein module is able to perform the complete hash function with the exception of the message padding. This functionality could be easily implemented either in software or the interface module, depending of the capabilities of the overall system.

4 Results

The Skein hardware module has been implemented in VHDL. The 64-bit adders have been implemented in a generic fashion (using the "+" operator of the IEEE.std_logic_unsigned package) in order to allow the synthesizer the greatest

flexibility for optimization[1]. For Skein-X-X, the processing of an X-bit message block requires 10 clock cycles for Skein-256-256 and Skein-512-512 and 11 clock cycles for Skein-1024-1024[2].

The functionality for all three supported variants has been successfully verified against the complete set of official short message and long message known-answer tests (KATs) through HDL simulation with Cadence ncsim. We have performed synthesis targeting a 0.18 µm standard-cell library and two different FPGA architectures (Xilinx Virtex 5 and Spartan 3).

For ASIC synthesis, we have used the 0.18 µm standard-cell library (FSA0A_C) from Faraday [2]. Synthesis has been performed with the Cadence PKS-Shell (v05.16) [1]. Different target clock frequencies for worst-case conditions have been used for different synthesis runs. Only the results of successful runs (i.e., where the set target was met) are reported. The results are given in Table 1[3]. For each Skein variant, we report two different synthesis results:

- Small: Relaxed constraints for the critical path delay to minimize silicon area.
- Fast: Tight timing constraints for maximum throughput.

Table 1. Implementation results for 0.18 µm standard-cell technology.

	Area	**Clock freq.**	**Throughput**
Implementation	GE	MHz	Gbit/s
Skein-256-256 (small)	44,287	34.46	0.882
Skein-256-256 (fast)	53,871	68.82	1.762
Skein-512-512 (small)	80,870	34.19	1.751
Skein-512-512 (fast)	102,346	48.85	2.501
Skein-1024-1024 (small)	178,793	24.46	2.277
Skein-1024-1024 (fast)	195,789	33.56	3.124

For synthesis targeting the Xilinx FPGAs, the ISE tools (v10.1.03) have been used. The target devices were a Xilinx Virtex 5 LX100, speed grade 3, package FF1760 (xc5vlx110-3ff1760) and a Xilinx Spartan 3 5000, speed grade 5, package FG676 (xc3s5000-5fg676). The device resource usage and clock frequency estimation after synthesis are reported in Table 2.

For both FPGA architectures, the number of used Configurable Logic Block (CLB) slices is reported. For Virtex 5, a CLB slice contains four 6-input look-up tables (LUTs) and four configurable flip-flops [7], while for Spartan 3, a slice

[1] Hand-optimized adder implementations could be dropped in to try to increase the speed of the module.

[2] The latency equals the number of rounds divided by eight plus an extra cycle for the loading of the block.

[3] The reported clock frequency and throughput are for typical operating conditions at 25 °C.

consists of two 4-input LUTs and two flip-flops [6]. No other functional blocks (e.g., Block RAMs) have been used by our implementations.

Table 2. Implementation results for FPGAs.

		Area	**Clock freq.**	**Throughput**
Implementation	**FPGA**	Slices	MHz	Gbit/s
Skein-256-256	Xilinx Spartan 3	2,421	26.14	0.669
Skein-256-256	Xilinx Virtex 5	937	68.40	1.751
Skein-512-512	Xilinx Spartan 3	4,273	26.66	1.365
Skein-512-512	Xilinx Virtex 5	1,632	69.04	3.535
Skein-1024-1024	Xilinx Spartan 3	8,198	27.33	2.482
Skein-1024-1024	Xilinx Virtex 5	2,994	68.90	6.414

5 Conclusions

In this work we presented the design of our high-speed hardware implementation of the hash function Skein. We also reported the performance figures of our implementation for all three block sizes both for 0.18 µm standard-cell implementation and two modern FPGA architectures.

References

1. Cadence Design Systems. The Cadence Design Systems Website. `http://www.cadence.com/`.
2. Faraday Technology Corporation. Faraday FSA0A_C 0.18 μm ASIC Standard Cell Library, 2004. Details available online at `http://www.faraday-tech.com`.
3. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker. The Skein Hash Function Family. Available online at `http://www.skein-hash.info/sites/default/files/skein1.1.pdf`, November 2008.
4. M. Long. Implementing Skein Hash Function on Xilinx Virtex-5 FPGA Platform. Available online at `http://www.skein-hash.info/sites/default/files/skein_fpga.pdf`, February 2009.
5. National Institute of Standards and Technology (NIST). Cryptographic Hash Algorithm Competition Website. `http://csrc.nist.gov/groups/ST/hash/sha-3`.
6. Xilinx, Inc. Spartan-3 FPGA Family Data Sheet, June 2008. Available online at `http://www.xilinx.com`.
7. Xilinx, Inc. Virtex-5 FPGA User Guide, March 2009. Available online at `http://www.xilinx.com`.