Silicon Implementation of SHA-3 Finalists: BLAKE, Grøstl, JH, Keccak and Skein

Xu Guo, Meeta Srivastav, Sinan Huang, Dinesh Ganta, Michael B. Henry, Leyla Nazhandali and Patrick Schaumont

Center for Embedded Systems for Critical Applications (CESCA)
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061, USA
{xuguo,meeta,shuang86,diganta,mbh,leyla,schaum}@vt.edu

Abstract. Hardware implementation quality is an important factor in selecting the NIST SHA-3 competition finalists. However, a comprehensive methodology to benchmark five final round SHA-3 candidates in ASIC is challenging. Many factors need to be considered, including application scenarios, target technologies and optimization goals. This work describes detailed steps in the silicon implementation of a SHA-3 ASIC. The plan of ASIC prototyping with all the SHA-3 finalists, as an integral part of our SHA-3 ASIC evaluation project, is motivated by our previously proposed methodology, which defines a consistent and systematic approach to move a SHA-3 hardware benchmark process from FPGA prototyping to ASIC implementation. We have designed the remaining five SHA-3 candidates in 0.13 μm IBM process using standard-cell CMOS technology. The fabricated chip is due from foundry in May 2011. In this submission, we discuss our proposed methodology for SHA-3 ASIC evaluation and report the latest results based on pre-silicon post-layout simulation of the five SHA-3 finalists with Round 3 tweaks.

1 Introduction

The SHA-3 competition organized by NIST aims to select, in three phases, a successor for the mainstream SHA-2 hash algorithms in use today. By the completion of Phase II in December 2010, 5 out of the 14 Round 2 candidates were identified for further evaluation as SHA-3 finalists. For the final round of the competition, NIST is looking for additional cryptanalytic results, as well as for performance evaluation data on hardware platforms.

eBACS is a well known benchmarking environment, including a scripting environment and a performance database, for the evaluation of crypto-software [5]. Compared to such a benchmarking environment, benchmarking crypto-hardware is ad-hoc. There are several reasons why the same progress is not seen in the hardware design community. This is due, in part, to the large heterogeneity of the hardware design space, to the absence of standard metrics for cost and performance, and to the absence of standard interfaces. To address the above issues we designed a SHA-3 ASIC by following a fair SHA-3 hardware evaluation

methodology. We started by defining a standard interface, and optimized the designs with a single metric in mind, *Throughput-per-Area*. Next, we developed an FPGA prototype that can provide a seamless transition into ASIC implementation. Finally, we designed an ASIC chip with all the five finalists with the latest round 3 tweaks. This ASIC is the focus of this paper.

2 Related Work

The hardware evaluation of SHA-3 candidates has started shortly after the specifications and reference software implementations of 51 algorithms submitted to the contest became available. The majority of initial comparisons were limited to less than five candidates. More comprehensive efforts became feasible only after NIST's announcement of 14 candidates qualified to the second round of the competition in July 2009. Since then, several comprehensive studies in SHA-3 ASIC implementations have been reported [19,18,14,10,11,17,13]. Guo et al. [10] used a consistent and systematic approach to move the SHA-3 hardware benchmark process from the FPGA prototyping by [16] to ASIC implementations based 130nm CMOS standard cell technology. Tillich et al. [18] presented the first ASIC post-synthesis results using 180nm CMOS standard cell technology with high throughput as the optimization goal and further provided post-layout results [19]. Henzen et al. [14] implemented several architectures in a 90nm CMOS standard cell technology, targeting high- and moderate-speed constraints separately, and presented a complete benchmark of post-layout results.

Table 1 compares these benchmarking efforts, and demonstrates that a comparison between different ASIC benchmarks is hard because of several reasons. First, most groups do not share the same source codes. Second, the ASIC benchmarks do not use a common hardware interface. Third, the reported metrics and optimization targets are different.

Table 1. Compare the related SHA-3 hardware benchmarking work in ASICs

	Tillich [19,18]	Guo [10]	Henzen [14]
Technology Choices	180nm CMOS Standard Cell	130nm CMOS Standard Cell	90nm CMOS Standard Cell
Hardware Interface	Assume infinite bandwidth interface	Defined standard 'handshake' interface	Assume infinite bandwidth interface
Chosen Metrics	Area, Throughput,	Area, Throughput, Power, Energy	Area, Throughput, Energy
Design Flow	Post-layout/synthesis simulation	Post-layout simulation	Post-layout simulation

3 Methodology

In this section, we describe the overall design flow that combines FPGA prototyping with ASIC design, and next elaborate the efforts to automate and standardize the ASIC implementation process.

3.1 Overview

Figure 1 illustrates the hardware evaluation flow for our SHA-3 ASIC benchmarking process. We developed the register transfer level (RTL) designs for the 5 Final Round candidates for the Round 3 tweaks. These hardware descriptions are next mapped to FPGA technology or ASIC technology. We use the same RTL descriptions for both types of design flow. Our objective is to use the FPGA as a prototyping technology for the ASIC, rather than a direct technology target. Hence, dedicated FPGA optimizations, such as the use of specialized multipliers or memory cells, are not used.

The ASIC and FPGA design flows look very similar, and cover the same two technology mapping steps. The first step is synthesis and maps the RTL code (in Verilog or VHDL) to a netlist of technology primitives. The second step is place and route, and this step decides the spatial relationships of technology primitives in a layout. Both of these steps can be automated using scripts. The results of technology mapping are performance estimates such as circuit area and circuit delay. The performance delays obtained after place-and-route are more accurate than those obtained after synthesis. With respect to the circuit area, place-and-route will reveal the precise dimensions of the ASIC design. With respect to the circuit delay, place-and-route reveals implementation effects (annotated as parasitics in Fig. 1) which characterize delay effects caused by the interconnections.

In the case of ASIC prototyping, we have taped out an ASIC for the 5 final round candidates. During Phase II, we performed prototyping on FPGA only and

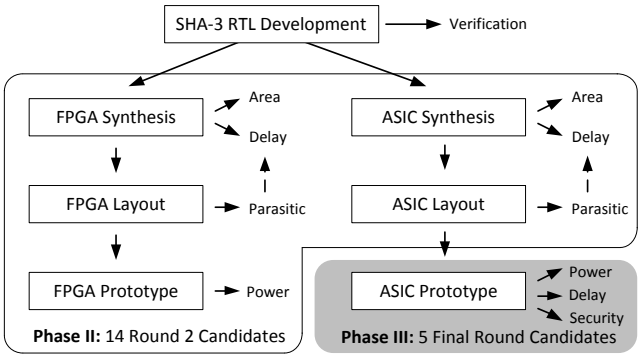

Fig. 1. An overview of the SHA-3 ASIC evaluation project.

reported ASIC post-layout numbers for individual candidates. The FPGA and ASIC prototyping are very useful to accurately evaluate the power consumption, such as the FPGA power measurement results discussed in the paper [16], which is especially important for further side-channel analysis. In the next subsection, we discuss the implementation details of the prototype design.

3.2 Platform for integrated FPGA prototyping and ASIC performance evaluation

The experimental environment for FPGA prototyping contains a PC, a SASEBO-GII [2] board and an oscilloscope as shown in Fig. 2. A SASEBO-GII board contains two FPGAs: a control FPGA, which supports the interfacing activities with a PC, and a cryptographic FPGA containing the hashing candidate. During the ASIC prototyping phase, the cryptographic FPGA is replaced by the SHA-3 ASIC. A board from the SASEBO-R [2] series will be used for this purpose.

The SASEBO board was originally developed for side-channel analysis. Hence, a potential research area for the future ASIC prototype is side-channel analysis of SHA-3 candidates. In our experiments, we used the SASEBO series board for a more obvious application, namely the measurement of power dissipation of the SHA-3 candidates mapped to ASICs.

The interface of the SASEBO board on the PC side is a software driver that can read test vectors and that can send messages to the SHA-3 FPGA through USB. The Control FPGA manages the data flow of the messages and generates control signals according to the timing requirements of the extended hash interface based on [7]. After SHA-3 FPGA finishes hash operations, the digest is returned to the PC through the Control FPGA.

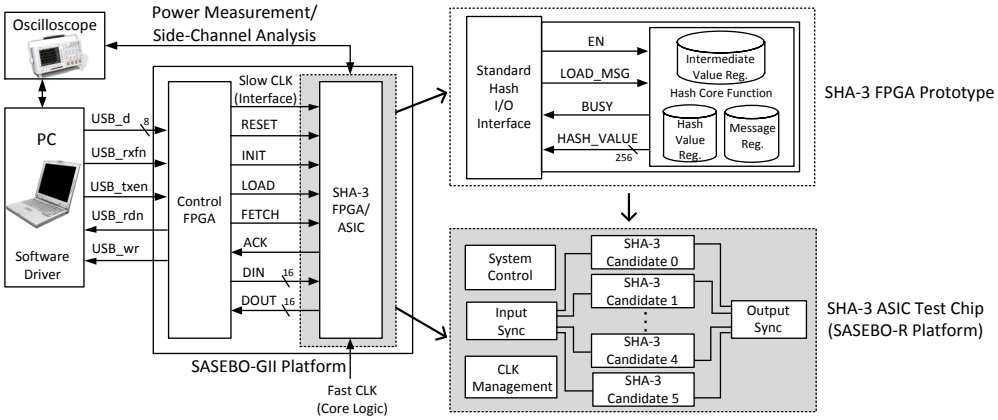

Fig. 2. Experimental environment for FPGA prototyping and final ASIC testing.

3.3 Standard Interface

So far, several research groups have proposed standard hardware interfaces with well supported design flows, including proposals from [9,14,7,3]. A more detailed discussion on hash interface issues can be found in [16]. The key issue for a fair comparison is to use a common interface for all candidates. Therefore, we selected the interface proposal of Chen et al. [7] (with a data I/O width of 16 bit) and extended it to add dual-clock support as shown in Fig. 8.

3.4 Design Strategy

There are three types of strategies used for SHA-3 hardware evaluation, namely fully autonomous, external memory and core functionality. More details of the differences between these three design strategies can be found at [8].

In this work we designed all the 5 finalists with a fully autonomous architecture. In this architecture, one transfers message data to a hash function over multiple clock cycles until a complete message block is provided. The hash module buffers a complete message block locally, before initiating the hash operation. Therefore, this architecture can work autonomously, and the resulting hash module is well suited for a hardware IP for system-on-chip integration.

3.5 Optimization Target

The use of *Throughput-to-Area Ratio* as the optimization target for SHA-3 hardware evaluation was first proposed by Gaj et al. [9], and later appeared in NIST Status Report on the Second Round of the SHA-3 Competition as a hardware evaluation criterion. One of the obvious advantages of choosing this target rather than *Throughput* alone is that it can avoid highly unrolled hash designs with small throughput benefits but significant circuit area overhead.

3.6 Evaluation Metrics

In this work we have used *Throughput-to-Area Ratio* as a primary metric and also reported other common metrics, including area, maximum frequency, maximum throughput, and power/energy efficiency.

Area In this evaluation we will use the circuit area of each SHA-3 candidate with both the interface and hash core after layout. The area will be reported in kilo gate equivalents (kGE), where a gate equivalent corresponds to the area of a standard NAND2 gate in the standard cell library. We divided the EDA tools reported layout area with unit in μm^2 by the area of an NAND2 gate for conversion from the absolute circuit area to kGE.

Throughput In general the time required to hash a message consists three parts: the latency for loading one block of message, L_{in} , the hash core latency, L_{core}, the latency for finalization step, L_{final}, and the latency for outputting the message digest, L_{out}. For short message hashing, all these four latencies are important performance factors. And the metric of *Latency* is frequently used to characterize the short message hashing speed instead of *Throughput*. In the case of hashing a long message, L_{final} and L_{out} can be neglected. Since L_{in} is dependent on the system I/O throughput which may vary in different contexts, here we report the *Throughput* results of the hash core function:

$$Tp_{core} = \frac{W_B \times f_{max}}{L_{core}}, \tag{1}$$

where W_B is the block size of the hash and f_{max} is the maximum frequency the circuit can run at.

Throughput-to-Area The *Throughput-to-Area Ratio* is an effective metric to measure the hardware efficiency, where the *Throughput* is the above defined Tp_{core} and the *Area* is for the layout circuit area expressed in terms of kGE.

Power/Energy The power numbers are measured with fixed achievable clock frequencies based on the average power of hashing long messages, and the capture period is only for the core hashing operations (e.g. round function for each block of input message). The energy metric is expressed as energy per bit of the input message block compressed by the hash core function.

4 VLSI Implementation of SHA-3 Hash Functions

In this section we briefly review the implementations of each SHA-3 finalist. For details on the SHA-3 candidates please refer to the related specification documents on NIST SHA-3 web sites. For hardware architectures, we have looked into several public available reference implementations [1,12,6] and optimized them for our system architecture.

4.1 BLAKE

BLAKE follows a HAIFA iteration mode and operates on an inner state that can be represented as a four by four matrix of words. The inner state is initialized using an Initial Value (IV), a salt and a counter. The state is updated using the **G** function, which is based on the ChaCha stream cipher [4]. The **G** function mainly contains modular addition, XOR, and rotate operations.

As shown in Fig. 3, before the message block enters into the round function it will first go through a permutation layer and XOR with predefined constants. One stage of pipeline is inserted inside the permutation layer for higher throughput. Four parallel **G** functions are implemented for one round. Eight **G** functions or even fully unrolled structures are also possible high performance solutions if there is no area constraint. Each **G** function instantiates two carry-save adders.

4.2 Grøstl

Grøtl is a wide-pipe Merkle-Dåmgard hash algorithm with an output transformation. The compression function is a novel construction, using two fixed $2n$-bit permutations together. The output transformation processes the final chaining state, and discards half the bits of the result, yielding an n-bit hash output. The underlying fixed permutations are themselves based closely on the structure of AES, reusing the S-box, but expanding the size of the block to 512 bits for Grøtl-256 in a straightforward way.

Grøstl can be implemented with parallel computation of **P** and **Q** permutations as well as a single permutation with iterations. As shown in Fig. 6, we implement the parallel **P** and **Q** structure, which enables better *Throughput-to-Area* ratio in our case since the hash core throughput can be doubled without doubling the overall area when considering the hardware interface overhead. Within the parallel **P** and **Q** structure, 128 AES SBoxes are used and all of them are implemented in Galois field operations also for better hardware efficiency.

4.3 JH

The compression function of JH is constructed from a large block cipher with constant key as fixed permutation. The large block cipher is based on a generalized AES design methodology and can be constructed with small components. There are three components of the underlying permutation **E8** : 4 bit SBoxes, an 8 bit L-permutation, and a P-permutation.

As shown in Fig. 5, two similar round operations, **R8** and **R6**, are used for compression and round constant generation, respectively. **R8** is used to update the 1024 bit hash state, and **R6** generates the round constant on-the-fly. As there are only two 4 bit SBoxes are used, we simply used LUT-based implementations.

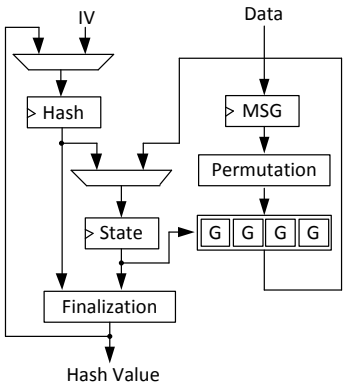

Fig. 3. Structure of the BLAKE-256.

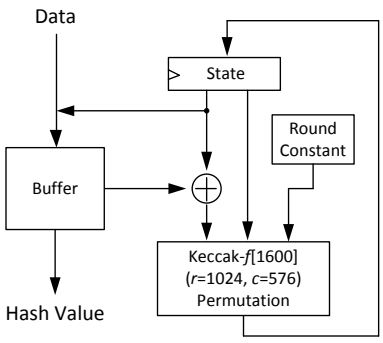

Fig. 4. Structure of the Keccak-256.

4.4 Keccak

Keccak follows the sponge construction model. The permutation can be considered as a substitution-permutation network with 5 bit wide SBoxes, or as a combination of a linear mixing operation and a very simple nonlinear mixing operation. The building block permutation is from a set of 7 permutations, indicated by Keccak-$f[b]$ (b is the width of Keccak-f with default value 1600).

We implement the Keccak-$f[1600]$ with rate $r = 1024$, capacity $c = 576$, and output digest size of 256 bits. The Keccak core design are based on the authors' provided reference high speed core designs [6].

4.5 Skein

Skein is an iterative hash algorithm built on a tweakable block cipher C Threefish. Threefish is used to build the compression function of Skein using a modified Mateas-Meyer-Oseas construction, which is then iterated in a chaining mode similar to HAIFA. The designers refer to the whole construction as a Unique Block Iteration (UBI) mode. Threefish is a 72-round substitution-permutation network using a 128-bit MIX function consisting of a 64-bit addition, rotate and XOR operations.

As shown in Fig. 7, we implemented the Skein512-256 version and each Threefish-512 round out of 72 rounds consists of four parallel MIX operation and a permutation. Four rounds are unrolled and chained together in hardware.

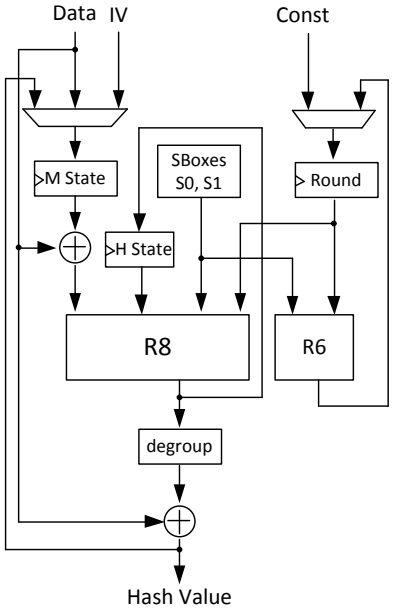

Fig. 5. Structure of the JH-256.

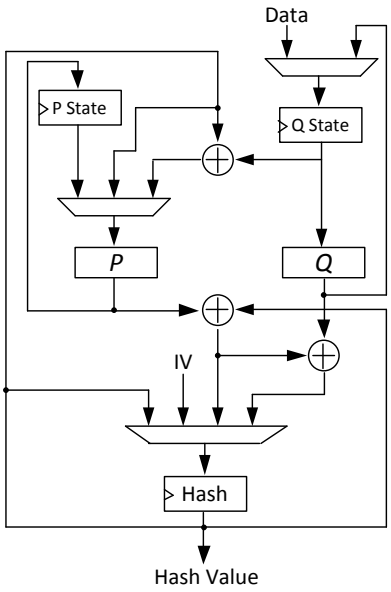

Fig. 6. Structure of the Grøstl-256.

4.6 Summary

Table 2 summarizes the major implementation aspects of each SHA-3 finalist. The design decisions are made to achieve the primary optimization for *Throughput-to-Area*.

Table 2. The summary of design specifications of SHA-3 finalists

Algorithm	Implementation Descriptions
Blake-256	4 parallel **G** functions; 1-stage pipeline in permutation
Grøstl-256	Parallel **P** and **Q** with 128 GF-based AES SBoxes
JH-256	SBoxes **S0** and **S1** are implemented in LUT
Keccak-256	One clock cycle per round
Skein512-256	Unrolled 4 Threefish rounds

5 ASIC Realization

Compared to other published work on SHA3 chip design [14], our chip has the following benefits:

– *Updated to Round-3 specifications.* By the end of January 2011, all of the SHA-3 final round candidate authors released their Round 3 specifications

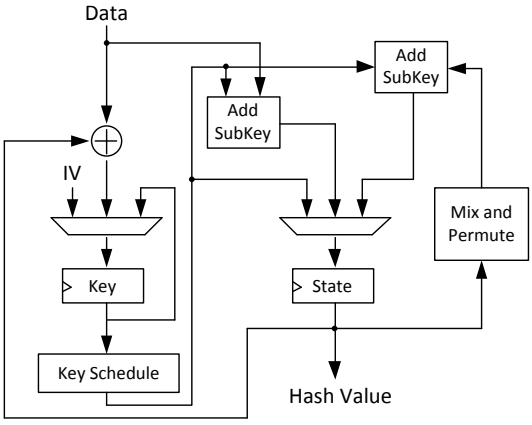

Fig. 7. Structure of the Skein512-256.

reflecting the feedbacks from the extensive security and performance evaluation during the SHA-3 Round 2 competition. A summary of Round 3 tweaks can be found in Table. 3. Some tradeoffs between security and performance were made in several candidates and caused changes in the performance profile (e.g. for BLAKE-256, the round number is increased from 10 to 14; for JH, the round number is changed from 35.5 to 42). Depending on the way how designers implement these algorithm (e.g. whether the round function is unrolled), some of the changes including changing the round number may not only affect the control logic but also the hash core logic.

– *Single technology and consistent methodology.* As discussed in [10], the impact of technologies in benchmarking SHA-3 ASIC implementations when shifting from 0.13 μm to 90 nm is non-trivial. Even under the same technology feature size, the impact of different standard-cell libraries, synthesis/layout constraints, and tool flows can lead to biased comparison results. We fabricated all the five finalists on the same die with uniform tool flow and environmental settings, and this may enable a more fair comparison in ASIC performance between each candidate.

Table 3. The summary of SHA-3 Round 3 tweaks

Algorithm	Round 3 tweaks for 256 bit digest version
Blake-256	Number of rounds is changed from **10** to **14**
Grøstl-256	Change shift values in **Q**;
	Use 'bigger' round constants in **P** and **Q**
JH-256	Number of rounds is changed from 35.5 to 42
Keccak-256	Simplified and shortened padding rule
Skein512-256	Change the key schedule parity constant

For ASIC evaluation in general, the best way to follow is always to fabricate an actual chip, measure it and explore its true potential. However, as mentioned earlier the SHA-3 ASIC evaluation process is much more difficult due to the lack of concrete application scenarios and requirements. In addition, our ASIC design has constraints, too, including the area budget, the total number of available IOs for the chip socket on the testing platform, and the capabilities of the testing infrastructure. In the following sections, we will discuss about how we designed the SHA-3 ASIC to tackle all the above issues and how the final area and performance results will be affected by those design decisions.

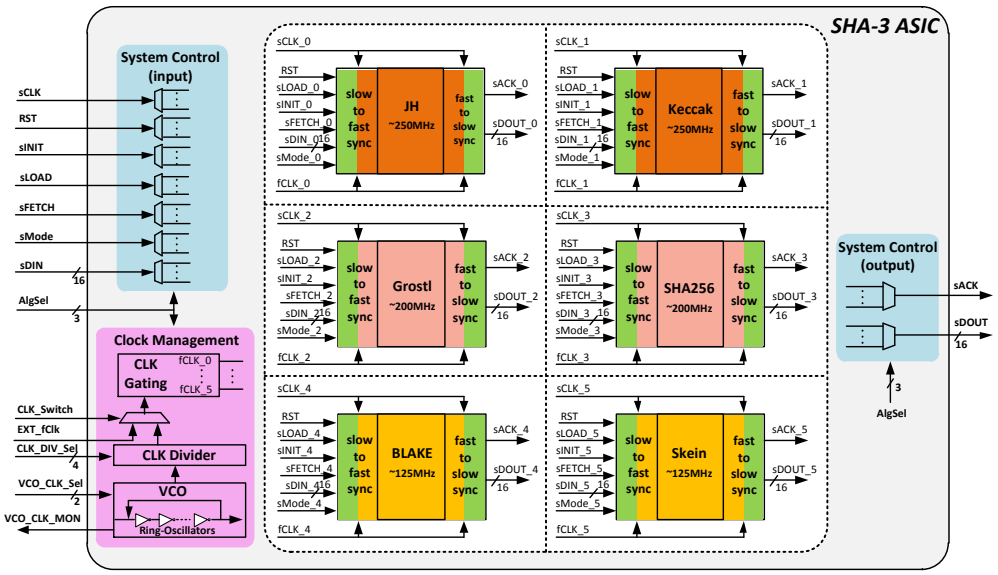

Fig. 8. The chip diagram of SHA-3 ASIC.

5.1 Chip Architecture

As shown in Fig. 8, besides the SHA-3 candidates modules our SHA-3 chip contains an on-chip clock management module, a system control module, and a synchronization module. All these features are integrated in order to fulfill the needs of fitting the SHA-3 ASIC chip into the SASEBO-R platform and meet our final chip testing requirements.

- *Clock Management Module.* This module has two functions: clock gating and clock generation. The clock gating is used to isolate the impact of the other candidates when measuring the power consumption of one candidate at a time. As shown in Fig. 2, all the inputs to the SHA-3 ASIC are generated from the control FPGA on board which connects to the SHA-3 ASIC. By default the control FPGA runs at a relatively low frequency (e.g. 24 *MHz*). It is not recommended to transmit high frequency signals through on board connections. For high frequency testing purpose, we used the custom-cell design approach to integrate a ring oscillator based voltage-controlled oscillator (VCO) into the chip together with a standard-cell implemented clock divider, which can offer a good range of clock frequencies on chip. The on-chip generated clocks are also MUXed with the external clock input and can be configured through dedicated ports.
- *System Control Module.* The system control module is in charge of decoding the control signals for algorithm selection, clock gating, clock division ratio and hash mode from ports. The SHA-3 ASIC can support operation modes.

The normal hashing mode loads 16 bit input data for each transfer until it buffers a whole block of data to start compression. This I/O bottleneck makes it very difficult for full speed testing since there is a relatively long message loading period between two consecutive hashing. Therefore, we defined a second mode of operation: full speed testing. Under this hash mode, the input buffer of each candidate only stores 16 bit of input data per block, and replicates this input data to fill an entire block.

– *Cross-Clock Domain Synchronizer.* There are two clock domains in our chip: the slow one is for the interfacing logic and the fast one is for hash modules. In order to avoid complex synchronizer designs based on asynchronous FIFOs or feedback synchronization and alleviate the burden of the backend process to deal with the two clock domains, we simplified the synchronizer design by making a reasonable assumption that the internal hash clock working frequency is always at least two times faster than the slow interface clock. As a result, the slow interface can be deemed as a normal control signal and the whole chip only has one single fast clock either input from external clock or internal VCO generated clock. Within this approach we extended the standard hash interface [7] of each candidate to integrate this synchronizer, and the final reported layout area for each candidate will also include the overhead of this extended hash interface.

5.2 Constraints

The timing constraints are selected to optimize *Throughput-to-Area Ratio*, using the methodology described in [11]. Based on this analysis, we categorized the five SHA-3 candidates plus the SHA256 based on their achievable frequencies after layout into three groups: 250 *MHz* (JH and Keccak), 200 *MHz* (Grøstl and SHA256), and 125 *MHz* (BLAKE and Skein).

With these timing constraints for each group of hash modules, we are able to finish the layout of the SHA-3 ASIC with the chip core of 1.656 $mm \times 1.656$ mm and overall chip area fitting in the die size of 5 mm^2 including pad cells.

5.3 Design Environment

We used the Synopsys Design Compiler (C-2009.06-SP3) to map the SHA-3 RTL codes to IBM MOSIS 0.13 μm CMOS Technology with CMR8SF-RVT standard cell library. We use the typical case condition characterization of the standard cell libraries. The 130nm technology uses 7 metal layers. In general, more metal layers allow for a denser interconnect, and hence a more optimal use of die area.

The Synopsys IC Compiler (C-2009.06-SP5) is used for the back-end process. The overall chip core area utilization ratio after layout is 73%. The utilization is defined as the die area devoted to active components (standard cells) as compared to the total die area. After place-and-route, design flow errors such as timing and Design Rule Check (DRC) violations may occur. In some cases, the initial utilization must be lowered in order to relax the constraints to the place-and-route process.

The timing results can be obtained from the post-synthesis and post-layout steps. First, the Synopsys IC Compiler is used to extract the post-layout parasitic and generate an SDF file containing the delays of all the interconnections and instances. Second, Synopsys VCS can be used to do the post-simulation and generate the VCD file that records all the switching activities of the netlist. Finally, Synopsys Prime Time (C-2009.06-SP3) reads the final netlist, VCD file and .spef parasitic file and does the power estimation.

5.4 ASIC Results and Analysis

From the post-layout results shown in Table 4, all the five SHA-3 candidates meet the target maximum frequency and have a larger area but higher Throughput than the reference SHA256; the maximum throughput of Grøstl and Keccak can almost reach 10 *Gbps*; Keccak is the best in hardware and energy efficiency; closely followed by BLAKE and Keccak, JH is the most power efficient SHA-3 but still less efficient than SHA256.

Although we have tried our best to ensure a fair methodology to guide our ASIC benchmark process, there are still many factors that may affect the results.

– *The limitations of absolute results numbers.* We often see many papers cited other papers' area results (unit: *kGE*) for direct comparisons. However, we want to point out that the gate counts are strongly dependent on the standard-cell libraries and tool settings even with the same technology node. The comparison of absolute numbers is considered as fair *ONLY* if the same library, same tools, and same settings are used. This is also one important reason of why we intend to open-source all of our RTL designs and scripts.

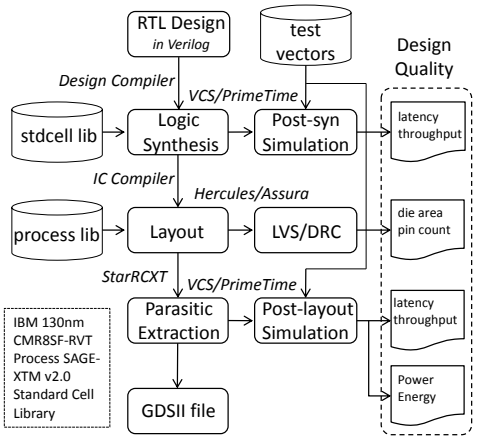

Fig. 9. The overview of our ASIC design flow.

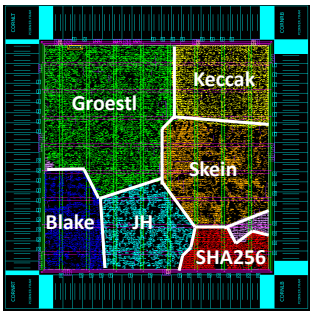

Fig. 10. Die layout of the SHA-3 ASIC chip in 0.13 μm IBM process using standard cell CMOS technology.

– *Further optimizations of hash design.* Due to the short period between the releasing the Round 3 specifications of each SHA-3 candidate and deadline for the chip tape-out run, we believed that fine grained optimizations are possible to better address the *Throughput-to-Area Ratio* optimization goals [15]. Since we have already optimized the designs based on some open-source SHA-3 projects [1,6,12], the designs and results described in this paper can be served as a good basis for future work.

6 Conclusions

In this paper we presented the methodology of evaluating SHA-3 ASIC implementations and reported the latest pre-silicon results for the SHA-3 final round candidates. Our SHA-3 ASIC is very likely the first chip implementing five SHA-3 finalists based on the Round-3 specifications published online in January, 2011.

Table 4. ASIC Characterization of the SHA-3 ASIC chip in IBM MOSIS 0.13 μm CMOS Technology with CMR8SF-RVT standard cell library, the Gate Equivalent count is calculated by dividing the post-layout die area by the area of a NAND2XLTF (5.76 μm^2)

	Block Size [*bits*]	Core Latency [*cycles*]	Area [*kGEs*]	Max Freq. [*MHz*]
BLAKE-256	512	30	34.15	125
Grøstl-256	512	11	124.34	200
JH-256	512	42	49.29	250
Keccak-256	1024	24	42.49	250
Skein512-256	512	21	66.36	125
SHA256	512	68	21.67	200

	Throughput [*Gbps*]	Throughput-to-Area [*kbps/GE*]	Power [*mW*]	Energy [*mJ/Gbits*]
BLAKE-256	2.13	62.47	15.65	36.67
Grøstl-256	9.31	74.87	99.59	85.58
JH-256	3.05	61.83	10.39	34.08
Keccak-256	10.67	251.05	15.68	14.70
Skein512-256	3.05	45.93	39.71	65.15
SHA256	1.51	69.54	3.72	19.75

*: NIST SHA-3 Round 3 Specifications by January, 2011.
**: The above numbers are from post-layout simulation with slow chip interface clock at 10MHz and fast hash core clock at 25MHz.
***: Throughput and Power/Energy numbers are measured for hash core operations.

Despite the accuracy in characterization of an actual chip (e.g. power measurement), we have also planed to use this SHA-3 chip for future security evaluation of SCA attacks based on SASEBO-R board. We also intend to open-source the RTL versions of the SHA-3 designs that we evaluated and the EDA tool scripts we used for the backend process.

Acknowledgments. The effort reported in this paper was supported by a NIST Measurement, Science and Engineering Grant ("Environment for Fair and Comprehensive Performance Evaluation of Cryptographic Hardware and Software"). The authors would like to thank National Institute of Advanced Industrial Science and Technology (AIST) of Japan for their hardware support.

References

1. AIST-RCIS. SHA-3 hardware project, May 2011. `http://www.rcis.aist.go.jp/special/SASEBO/SHA3-en.html`.
2. AIST-RCIS. Side-channel attack standard evaluation board, May 2011. `http://staff.aist.go.jp/akashi.satoh/SASEBO/en/index.html`.
3. Brian Baldwin, Neil Hanley, Mark Hamilton, Liang Lu, Andrew Byrne, Maire ONeill, and William P. Marnane. FPGA Implementations of the Round Two SHA-3 Candidates. In *The Second SHA-3 Candidate Conference*, August 2010.
4. D. Bernstein. ChaCha, a variant of Salsa20, January 2008. `http://cr.yp.to/chacha/chacha-20080128.pdf`.
5. D. Bernstein and T. Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems, May 2011. `http://bench.cr.yp.to`.
6. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak sponge function family – Updated VHDL package, May 2011. `http://keccak.noekeon.org/VHDL_3.0.html`.
7. Zhimin Chen, Sergey Morozov, and Patrick Schaumont. A Hardware Interface for Hashing Algorithms. Cryptology ePrint Archive, Report 2008/529, 2008. `http://eprint.iacr.org/2008/529`.
8. ECRYPT. The SHA-3 Zoo, May 2011. `http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations`.
9. Kris Gaj, Ekawat Homsirikamol, and Marcin Rogawski. Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates Using FPGAs. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *LNCS*, pages 264–278. Springer Berlin / Heidelberg, 2010.
10. Xu Guo, Sinan Huang, Leyla Nazhandali, and Patrick Schaumont. Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations. In *The Second SHA-3 Candidate Conference*, August 2010.
11. Xu Guo, Sinan Huang, Leyla Nazhandali, and Patrick Schaumont. On The Impact of Target Technology in SHA-3 Hardware Benchmark Rankings. Cryptology ePrint Archive, Report 2010/536, 2010. `http://eprint.iacr.org/2010/536`.
12. Xu Guo, Sinan Huang, Meeta Srivastava, Leyla Nazhandali, and Patrick Schaumont. Performance Evaluation of Cryptographic Hardware and Software – Performance Evaluation of SHA-3 Candidates in ASIC and FPGA, May 2011. `http://rijndael.ece.vt.edu/sha3/`.

13. L. Henzen, J.-P. Aumasson, W. Meier, and R. C.-W. Phan. VLSI Characterization of the Cryptographic Hash Function BLAKE. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2010.
14. Luca Henzen, Pietro Gendotti, Patrice Guillet, Enrico Pargaetzi, Martin Zoller, and Frank Gürkaynak. Developing a Hardware Evaluation Method for SHA-3 Candidates. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 248–263. Springer Berlin / Heidelberg, 2010.
15. Ekawat Homsirikamol, Marcin Rogawski, and Kris Gaj. Comparing hardware performance of fourteen round two sha-3 candidates using fpgas. Cryptology ePrint Archive, Report 2010/445, 2010. `http://eprint.iacr.org/`.
16. K. Kobayashi, J. Ikegami, M. Knezevic, E.X. Guo, S. Matsuo, Sinan Huang, L. Nazhandali, U. Kocabas, Junfeng Fan, A. Satoh, I. Verbauwhede, K. Sakiyama, and K. Ohta. Prototyping platform for performance evaluation of SHA-3 candidates. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 60 –63, 2010.
17. A.H. Namin and M.A. Hasan. Hardware implementation of the compression function for selected SHA-3 candidates. CACR 2009-28, July 2009.
18. Stefan Tillich, Martin Feldhofer, Mario Kirschbaum, Thomas Plos, Jörn-Marc Schmidt, and Alexander Szekely. High-Speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. Cryptology ePrint Archive, Report 2009/510, 2009. `http://eprint.iacr.org/2009/510`.
19. Stefan Tillich, Martin Feldhofer, Mario Kirschbaum, Thomas Plos, Jörn-Marc Schmidt, and Alexander Szekely. Uniform Evaluation of Hardware Implementations of the Round-Two SHA-3 Candidates. In *The Second SHA-3 Candidate Conference*, August 2010.