High-speed SABER Key Encapsulation Mechanism in 65nm CMOS

Malik Imran[1*], Felipe Almeida[1], Andrea Basso[2], Sujoy Sinha Roy[3] and Samuel Pagliarini[1]

[1*] Department of Computer Systems, Tallinn University of Technology, Estonia.
[2] School of Computer Science, University of Birmingham, United Kingdom.
[3] IAIK, Graz University of Technology, Austria.

*Corresponding author(s). E-mail(s): malik.imran@taltech.ee;
Contributing authors: felipe.almeida@taltech.ee; a.basso@pgr.bham.ac.uk;
sujoy.sinharoy@iaik.tugraz.at; samuel.pagliarini@taltech.ee;

Abstract

Quantum computers will break cryptographic primitives that are based on integer factorization and discrete logarithm problems. SABER is a key agreement scheme based on the Learning With Rounding problem that is quantum-safe, i.e., resistant to quantum computer attacks. This article presents a high-speed silicon implementation of SABER in a 65nm technology as an Application Specific Integrated Circuit. The chip measures $1\boldsymbol{mm^2}$ in size and can operate at a maximum frequency of $715\boldsymbol{MHz}$ at a nominal supply voltage of 1.2V. Our chip takes $10\boldsymbol{\mu s}$, $9.9\boldsymbol{\mu s}$ and $13\boldsymbol{\mu s}$ for the computation of key generation, encapsulation, and decapsulation operations of SABER. The average power consumption of the chip is $153.6\boldsymbol{mW}$. Physical measurements reveal that our design is 8.96x (for key generation), 11.80x (for encapsulation), and 11.23x (for decapsulation) faster than the best known silicon-proven SABER implementation.

Keywords: ASIC, Post-quantum, Crypto accelerator, Silicon-proven, SABER

1 Introduction

Using Shor's quantum factoring algorithm [1], the most prevalent public-key cryptographic primitives such as RSA [2], Diffie-Hellman [3], and Elliptic Curve Diffie-Hellman protocols [4] are vulnerable to attacks by sufficiently powerful quantum computers [5, 6]. These protocols are extensively used, in practice, to protect secure web pages, encrypt emails, and other sensitive information. Therefore, breaking these systems would have significant consequences for digital security and privacy. To secure future information and communication systems, researchers and developers are constructing new reliable quantum-resistant cryptographic protocols. In this context, in 2017, the National Institute of Standards and Technology (NIST) initiated a new competition process to standardize post-quantum public-key algorithms. Currently, the competition is in its third round [7] and SABER remains one of the competing quantum-resistant key encapsulation mechanisms. A silicon demonstration of SABER is the central piece of this work.

The design characteristics of SABER have been investigated over different implementation platforms. Field-programmable gate array (FPGA) accelerators are described in [8–10]. Performance-oriented implementations of SABER on a RISC-V processor and on a GPU are shown in [11] and [12], respectively. Some resource-constrained implementations on ARM platforms are given in [13–15]. Side-channel protected software implementations are presented in [16, 17]. Similarly, a side-channel protected hardware implementation of SABER is described in [18]. An efficient implementation of SABER on an embedded microcontroller is presented in [19].

Additionally, SABER has been considered and demonstrated as an application-specific integrated circuit (ASIC) in [20–23]. ASICs provide a platform that is specialized to compute a specific cryptographic operation and thus yield superior performance than other platforms. In turn, the effort and cost to produce an ASIC is much higher. For this reason, the authors of [20, 23] report implementation results from simulations instead of physical measurements.

An energy-efficient crypto processor architecture for supporting all variants of SABER is described in [20]; the authors perform energy optimizations by employing a hierarchical Karatsuba framework for multiplying polynomial coefficients. Moreover, they have presented a layout implementation of SABER on 40nm technology with an area of $0.38mm^2$ and a maximum frequency of $400MHz$. Recently, in [21], the authors present a low-area and low-power silicon-verified SABER design on 65nm commercial technology. For SABER 256-degree polynomial multiplications, their chip incorporates a Toom-Cook multiplier with a striding of 4.

For several hard mathematical problems, i.e., lattice, code, and multivariate, a flexible crypto processor design is fabricated on a 28nm process technology in [22]. The supported cryptographic algorithms are SABER, NTRU, Dilithium, Rainbow, Kyber and McEliece. At a 0.9V supply voltage, their design can operate up to a maximum frequency of $500MHz$ and displays a chip size of $3.6mm^2$.

Earlier in [23], we have performed a design space exploration of SABER on a 65nm commercial technology with the goal to maximize performance. Our pre-silicon results indicated that a $1GHz$ clock frequency could be achieved with the careful use of pipelining, some notion of resource sharing, and different memory arrangements. This work expands on these initial results.

In this article, we have chosen the most promising architectures from our design space exploration of [23] in order to execute a silicon validation on a commercial 65nm technology[1]. Therefore, the contributions of this article include: (i) physical synthesis of the SABER core on a targeted 65nm technology (i.e., chip design); (ii) more realistic results for area, timing, and power characteristics after physical measurements on the fabricated ASIC; and (iii) a fair comparison against other works that also perform measurements instead of simulations.

The key findings after physical measurements revealed that the fabricated chip can operate in the range of 0.6V to 1.4V. At nominal 1.2V, a frequency of $715MHz$ is achieved. The average power consumption and chip size are $153.66mW$ and $1mm^2$, respectively. For security equivalent to AES-192, the processing time for one key-generation, encapsulation, and decapsulation operation are $10\mu s$, $9.98\mu s$, and $13.28\mu s$, respectively.

The structure of this paper is organized as follows: Section 2 describes the related background. Our chip's architecture is presented in Section 3. Chip measurements and results are shown in Section 4. The comparison to state-of-the-art SABER implementations is given in Section 5. Finally, the critical findings of this work are discussed in Section 6.

2 Related Background

2.1 Notations

This section presents the symbols used throughout the paper. Let p and q are moduli powers of 2. Set of integers is presented with $\mathbb{Z}$. The rings of integers modulo p and q is $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively. Similarly, the ring of polynomials for an integer N is presented with $\mathcal{R}_p = \mathbb{Z}_p[x]/\langle x^N + 1\rangle$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^N + 1\rangle$ where N is a fixed power of 2. Vectors are shown in bold and lower case font (e.g., $\boldsymbol{a}$).

[1]The Verilog HDL code is already available in our saber-chip repository on GitHub [24].

2.2 Security hardness

The security strength of SABER depends on the hardness of the module Learning With Rounding (Mod-LWR) problem [9]. A Mod-LWR sample is given as follows:

$$\left(\boldsymbol{a}, b = \left\lfloor \frac{p}{q}(\boldsymbol{a}^T \boldsymbol{s}) \right\rceil\right) \in \mathcal{R}_q^{l\times 1} \times \mathcal{R}_p \qquad (1)$$

Where $\boldsymbol{a}$ is a vector of uniformly random polynomials in $\mathcal{R}_q$, $\boldsymbol{s}$ is a secret vector of polynomials with coefficients from an error distribution, and the modulus $p < q$. The result of the vector-vector multiplication $\boldsymbol{a}^T.\boldsymbol{s}$ is a polynomial in $\mathcal{R}_q$. It is then rounded using the scaling by $\frac{p}{q}$ where $p < q$ to produce b in $\mathcal{R}_p$. The rounding operation introduces a noise to system. The decision mod-LWR problem asks to distinguish between mod-LWR samples generated using Eq. 1 for a fixed secret, and uniformly random samples in $\mathcal{R}_q^{l\times 1} \times \mathcal{R}_p$. The Mod-LWR problem is presumed to be computationally infeasible to solve using classical as well as quantum computers.

2.3 Supported operations

SABER is a Chosen-Ciphertext Attack, i.e., IND-CCA, resistant key encapsulation mechanism (KEM) built on module lattices. Moreover, it uses the Mod-LWR problem with both p and q power-of-two to construct a Chosen Plaintext Attack, i.e., IND-CPA, secure public-key encryption (PKE) scheme. The related cryptographic operations for PKE are the generation of a pair of public and private keys (PKE.KeyGen), encryption (PKE.ENC) and decryption (PKE.DEC) and the corresponding algorithms are 1, 2, 3. Similar to PKE operations, the supported KEM operations are a generation of pairs of public and private keys (KEM.KeyGen), encapsulation (KEM.ENCAPS) and decapsulation (KEM.DECAPS) and the respective algorithms are 4, 5, 6. Some minor details for associated PKE and KEM operations are presented in the following.

Algorithm 1 SABER.PKE.KeyGen() [9]

```
Require: SABER Parameter Lengths
Ensure: $pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b}), sk \Leftarrow (\boldsymbol{s})$
1: $seed_{\boldsymbol{A}} \Leftarrow \mathcal{U}(\{0,1\}^{256})$
2: $\boldsymbol{A} \Leftarrow gen(seed_{\boldsymbol{A}}) \in \mathcal{R}_q^{l\times l}$
3: $\boldsymbol{r} \Leftarrow \mathcal{U}(\{0,1\}^{256})$
4: $\boldsymbol{s} \Leftarrow \beta_\mu(\mathcal{R}_q^{l\times l};\ \boldsymbol{r})$
5: $\boldsymbol{b} \Leftarrow ((\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{h})\ mod\ q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}_p^{l\times l}$
```

Algorithm 2 SABER.PKE.ENC() [9]

```
Require: $pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b}), m \in R_2;\ r)$
Ensure: $c \Leftarrow (c_m, \boldsymbol{b}')$
1: $\boldsymbol{A} \Leftarrow gen(seed_{\boldsymbol{A}}) \in \mathcal{R}_q^{l\times l}$
2: if $r$ is not specified then
3:     $r \Leftarrow \mathcal{U}(\{0,1\}^{256})$
4: end if
5: $\boldsymbol{s}' \Leftarrow \beta_\mu(\mathcal{R}_q^{l\times l};\ \boldsymbol{r})$
6: $\boldsymbol{b}' \Leftarrow ((\boldsymbol{A}\boldsymbol{s}' + h)\ mod\ q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}_p^{l\times 1}$
7: $v' \Leftarrow \boldsymbol{b}^T(\boldsymbol{s}'\ mod\ p) \in R_p$
8: $c_m \Leftarrow (v' + h_1 - 2^{\epsilon_p - 1}m\ mod\ p) \gg (\epsilon_p - \epsilon_T) \in \mathcal{R}_T$
```

Algorithm 3 SABER.PKE.DEC() [9]

```
Require: $sk \Leftarrow \boldsymbol{s}, c \Leftarrow (c_m, \boldsymbol{b}')$
Ensure: $m'$
1: $v \Leftarrow \boldsymbol{b}'^T(\boldsymbol{s}\ mod\ p) \in \mathcal{R}_p$
2: $m' \Leftarrow ((v - 2^{\epsilon_p - \epsilon_T}c_m + h_2)\ mod\ p) \gg (\epsilon_p - 1) \in \mathcal{R}_2$
```

KeyGen. PKE.KeyGen begins by randomly generating a seed that defines an $l \times l$ matrix $\boldsymbol{A}$ comprising l^2 polynomials in $\mathcal{R}_q$.A function *gen* of Algorithm 1 is used to generate a matrix from the seed based on SHAKE-128. A secret vector $\boldsymbol{s}$ of polynomials is also generated. These polynomials are sampled from a centered binomial distribution. The generated public key contains a matrix seed and rounded product $\boldsymbol{A}^T\boldsymbol{s}$, while the secret key contains a secret vector $\boldsymbol{s}$. KEM.KeyGen follows the same acts as used for the PKE.KeyGen, except that it appends a secret key with a hash of the public key and a randomly generated string z.

ENC and ENCAPS. The PKE.Enc operation consists of generating a new secret $\boldsymbol{s}'$ and adding message to the inner product between the public key and the new secret $\boldsymbol{s}'$. This forms

Algorithm 4 SABER.KEM.KeyGen() [9]

```
Require: SABER.PKE.KeyGen()
Ensure: $pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b}), sk \Leftarrow (\boldsymbol{s}, z, pkh)$
1: $pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b})$
2: $pkh \Leftarrow \mathcal{F}(pk)$
3: $z \Leftarrow \mathcal{U}(\{0,1\}^{256})$
```

Algorithm 5 SABER.KEM.ENCAPS() [9]

```
Require: $pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b})$
Ensure: $c, K$
1: $m \Leftarrow \mathcal{U}(\{0,1\}^{256})$
2: $(\hat{K}, r) \Leftarrow \mathcal{G}(\mathcal{F}(pk), m)$
3: $c \Leftarrow$ SABER.PKE.ENC$(pk, m;\ r)$
4: $K \Leftarrow \mathcal{F}(\hat{K}, c)$
```

Algorithm 6 SABER.KEM.DECAPS() [9]

```
Require: $sk \Leftarrow (\boldsymbol{s}, z, pkh), pk \Leftarrow (seed_{\boldsymbol{A}}, \boldsymbol{b}), c$
Ensure: $K$
1: $m' \Leftarrow$ SABER.PKE.DEC$(\boldsymbol{s}, c)$
2: $(\hat{K}', r') \Leftarrow \mathcal{G}(pkh, m')$
3: $c' \Leftarrow$ SABER.PKE.ENC$(pk, m';\ r')$
4: if $c = c'$ then
5:     $K \Leftarrow \mathcal{H}(\hat{K}', c)$
6: else
7:     $K \Leftarrow \mathcal{H}(z, c)$
8: end if
```

the first part of the ciphertext while the second part contains the rounded product $\boldsymbol{A}\boldsymbol{s}'$. The KEM.Encaps operation starts by randomly generating a message m and obtaining from that the public key. The ciphertext c contains the encrypted message and a value achieved from the message and public key.

DEC and DECAPS. PKE.Dec requires the secret key $\boldsymbol{s}$ to extract original message from the inner product between the public and secret keys. PKE.Dec is the counterpart to PKE.Enc. KEM.Decaps re-encrypts the obtained message with the randomness associated with it and checks whether the ciphertext corresponds to the one received.

2.4 Supported variants and their parameters

For security equivalent to AES-128, AES-192, and AES-256, SABER supports three different variants: LightSABER, SABER, and FireSABER. All three variants use polynomial degree $N = 256$ and moduli $q = 2^{13}$ & $p = 2^{10}$. They differ only in the module dimension, binomial distribution parameter (μ), and the message space. For additional details about security parameters, PKE and KEM operations, we refer readers to the SABER specification [25].

3 Architecture of our SABER Design

First, we clarify that in order to demonstrate a chip that implements the SABER protocol, our design has to be augmented with appropriate interfaces for control purposes and debug purposes. At the center of our chip lies a coprocessor-styled crypto core and its many specialized blocks for SABER. The entire architecture of our SABER accelerator design is shown in Fig. 1. At the highest level, it consists of a wrapper, a serial-in/out interface, and the SABER crypto core itself.

The wrapper acts as a controller to operate the required cryptographic operations. As the name implies, serial-in/out bears inputs serially from outside to the chip and also results in a serialized output. The SABER crypto core is responsible for the computations of corresponding operations such as KeyGen, ENCAPS and DECAPS. Moreover, it comprises a data memory, a routing network, a pipeline register, a shared shift buffer, several specific building blocks, and an FSM-based dedicated controller. The building blocks of SABER read input operands from the data memory and, after computations, write the result back onto the same memory. The used data memory is of 8KB size such that all SABER variants (LightSABER, SABER, and FireSABER) can be operated. An essential design parameter is the word size of inferred memory. We use the same word size of 64-bit as utilized in the SABER accelerators described in [9, 23]. All the blocks of Fig. 1 support 64-bit data for reading/writing operations.

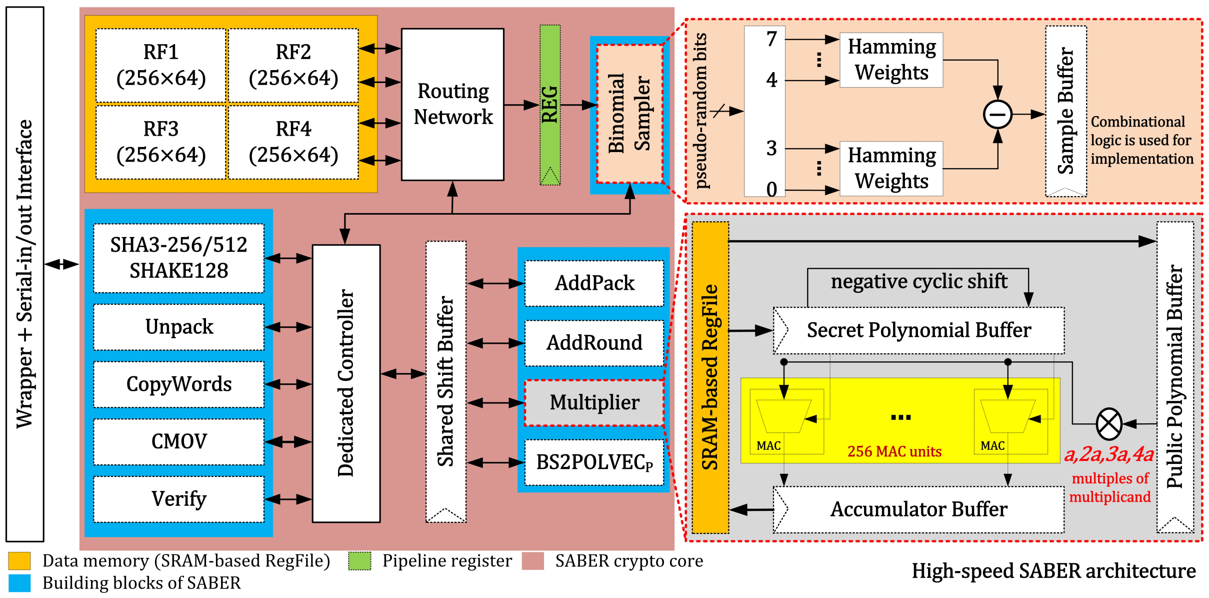

Fig. 1 Block diagram of our fabricated SABER chip. The I/O pins are not shown for clarity.

More insights and details of several blocks of our SABER chip architecture are described in the following subsections.

3.1 Wrapper

The wrapper of our chip contains 16 single-bit I/O pins, not shown in Fig. 1 for the sake of clarity. The input pins are $clk1$, $clk2$, rst, $start$, we, $cont$, $addr$, $addr\_ready$, din, $lad1$, $lad2$, $crypto\_op\_1$, $crypto\_op\_2$ and $crypto\_op\_3$. Similarly, the output pins are $dout$ and $done$.

The $clk1$ pin drives a slower clock that feeds the serial I/O interface of the chip. Similarly, $clk2$ drives the faster clock that is connected to the inner SABER crypto core. The names of various other I/O pins are intuitive: rst is a reset signal, $start$ is a trigger signal for starting cryptographic operations, we is a write-enable, din is data in, $dout$ is data out, $addr$ is address. The pins $addr\_ready$ and $done$ inform when operations are finalized, either loading an address or an entire crypto operation.

The purpose of the use of a $cont$ pin is to measure the power consumption of our chip when the KeyGen, ENCAPS and DECAPS operations are executed continuously (i.e., in an infinite loop). By doing so, we make sure that the power measurement is not affected by I/O limitations.

The combined use of $lad1$ and $lad2$ allow us to drive four possible combinations: (i) 2'b00 means "no-operation", (ii) 2'b01 means load read/write address on the chip using $addr$, (iii) 2'b10 means load input data vector from outside on the chip using din, and (iv) 2'b11 means reading data back from the chip on $dout$. The $crypto\_op\_1$, $crypto\_op\_2$, and $crypto\_op\_3$ signals are used to select the crypto operation, either KeyGen, ENCAPS, or DECAPS.

The wrapper of our chip is an FSM-based dedicated controller. It is responsible to execute the KeyGen, ENCAPS and DECAPS operations by properly orchestrating the sequential use of the SABER blocks. The chip remains in an IDLE mode until the $start$ signal is asserted. Next, based on the values of $crypto\_op\_1$, $crypto\_op\_2$, and $crypto\_op\_3$, the FSM begins to execute the corresponding sequence of instructions for computation of KeyGen, ENCAPS and DECAPS operations. When the required KEM operation completes its execution, the FSM switches back the chip into an IDLE mode (if $cont$ is 0, otherwise the operation is continuously executed non-stop when $cont$ is 1).

3.2 Serial-in/out interface

The purpose of the serial interface is to load or read data serially. The same interface is used for loading user input and for debugging purposes.

For this reason, the serial interface gives access to the entire 8KB memory addressing space. To achieve this, the interface accumulates incoming bits into desired vector lengths (i.e., 10-bits for read/write addresses, and 64-bits for read/write data). To accumulate a 10-bit address or a 64-bit input/output data, the serial-in/out interface employs three shift registers: (i) one for read/write address, (ii) one for data input and (iii) one for data output. More precisely, the *addr*, *din*, and *dout* pins of our fabricated chip are connected to the corresponding read/write address, data input, and data output shift registers. The corresponding values on $lad1$ and $lad2$ are utilized to mux the corresponding shift register to the right pins.

3.3 SABER crypto core

The intention of the SABER crypto core is to perform the related cryptographic operations, i.e., KeyGen, ENCAPS, and DECAPS, of the SABER protocol. As shown in Fig. 1, it consists of several blocks, i.e., a data memory, routing network, a pipeline register, a shared shift buffer, building blocks and a dedicated controller. We provide the relevant details of these blocks in the following subsections.

3.3.1 Data memory

In [9], a BRAM-based dual-port data memory of size 1024×64 is utilized in a coprocessor architecture. Recently, we have optimized this FPGA-targeted coprocessor architecture of [9] for evaluation as an ASIC on a commercial 65nm technology [23]. We replace the BRAM with an SRAM with the same overall size of (1024×64) but different implementation. The SRAM is generated by using a commercial memory compiler of a partner foundry. Moreover, a smart memory synthesis process is (also) explored to minimize the critical path and eventually to maximize the clock frequency. The concept of smart synthesis determines that smaller and distributed memories in an ASIC design could be more advantageous as the smaller memories require simpler address decoder units (which are faster and leads to performance improvements with area and power overheads). Based on this observation, five optimized architectures of SABER with different memory configurations are presented in [23]. The highest clock frequency is achieved when using four instances of a single-port SRAM-based RegFile. The RegFile is not an array of flip-flops. It is a "high-speed" variant of SRAM according to its vendor.

Based on the aforesaid concept, and as also highlighted in Fig. 1, our architecture utilizes four instances of 256× 4 size of a single-port SRAM-based RegFile as a data memory to retain initial, intermediate, and final results for the execution of required cryptographic operations. The total size of our four memory instances is $(256\times\ 4) \times 4 =$ 65Kbits.

3.3.2 Routing network

The proposed SABER chip splits the memory address space in multiple memory blocks. However, each memory requires a unique write enable, read/write address and input/output data signals. Thus, a unified routing network is necessary for several building blocks of the SABER to communicate with the corresponding memory instance(s) transparently. Consequently, the routing network of our proposed architecture consists of several multiplexers to deal with the corresponding memory instances for reading and writing operations.

3.3.3 Pipeline register

The use of different memory configurations results in a change in the critical path of the SABER design, as presented in [23]. In our architecture, the critical path becomes from the output of a memory instance to *dout* (a chip output) through the binomial sampler. Therefore, to shorten the critical path and to eventually improve the clock frequency, we have placed a pipeline register between the routing network and the binomial sampler, as shown in Fig. 1.

3.3.4 Shared shift buffer

The several building blocks of SABER, i.e., AddRound, AddPack, BS2POLVEC$_\mathrm{p}$, and multiplier, require shift registers with different lengths to acquire data from many memory addresses and then accumulate into local registers. For example, a 320-bit long register is required in AddPack and BS2POLVEC$_\mathrm{p}$ while a 64 and 676-bit register is required in AddPack and multiplier, respectively. Similar to our previous work, published in [23], we have shared a single 676-bit register across

AddRound, AddPack, BS2POLVEC$_\mathrm{p}$, and multiplier in our SABER chip architecture. This saves area at no cost in performance since the critical path lies elsewhere.

3.3.5 SABER building blocks

As shown in Fig. 1, the required building blocks are, polynomial multiplier wrapper, variants of secure hash algorithms, (i.e., SHA3-256, SHA3-512, and SHAKE-128), a binomial sampler, AddRound, AddPack, Unpack, Constant-time Move (CMOV), CopyWords, BS2POLVEC$_\mathrm{p}$ and Verify. Some more insight details for these building blocks are given as follows.

The multiplier wrapper incorporates a centralized multiplier architecture based on a schoolbook multiplication for multiplying polynomial coefficients. The idea for the centralized multiplier architecture is precomputation of several multiples of multiplicand at once and then forwarding of these multiples to the parallel MAC (multiply-and-accumulate) units. Next, the employed MAC instances select their right multiple of a multiplicand depending on their corresponding bits of the multiplier and then add to the accumulator. It is important to note that the proposed SABER chip architecture utilizes the centralized multiplication architecture of [26]. Therefore, we direct readers to [26] for complete algorithmic and architectural details of centralized schoolbook-based polynomial multiplication.

SABER requires variants of hash functions (SHA3-256/512) that were standardized in [27]. Moreover, to generate pseudorandom numbers, an extendable output function SHAKE-128 is also required and is standardized in [27]. Since all of these functions utilize the Keccak sponge function [27], we operate the SHA3-256, SHA3-512 and SHAKE-128 like a wrapper in our SABER chip architecture as implemented in [9]. For the detailed unified architecture of SHA3-256, SHA3-512 and SHAKE128, we redirect readers to [9].

A sampler is mandated to compute the sample from a pseudo-random input string for all (supported) PKE and KEM operations. Similar to [9], the binomial sampler in our proposed SABER chip architecture is a combinational block that directly maps pseudo-random bits from an input buffer to a sample value.

The verify block of the SABER crypto core is only required during the decapsulation operation of KEM. It is responsible to provide a word-by-word comparison between the received ciphertext and re-encrypted ciphertext. The result of verify block is stored in a register that is used by CMOV to either copy the decrypted session key or a pseudo-random string at a specified memory address. The AddPack performs coefficient-wise addition with a constant followed by the generated message, and it packs the resultant bits into a byte string. Similar to the Addpack block, AddRound computes coefficient-wise addition of a constant followed by coefficient-wise rounding. The conversions from byte into a bit string are the responsibility of unpack unit. The BS2POLVEC$_\mathrm{p}$ block converts the byte string into a polynomial vector.

For more insight into the details and architectures of the building blocks of SABER, we refer readers to [9, 23].

3.3.6 Controller

Based on the instructions from the wrapper for the computation of KeyGen, ENCAPS and DECAPS, the controller generates the corresponding control signals to the inner SABER core. Moreover, it control the use of the shared shift buffer and the routing network. Since the binomial sampler is connected through a pipeline reg (highlighted with green color in Fig. 1), this creates execution bubbles. The controller also handles the synchronization effort between blocks.

4 Results and Chip Measurements

The silicon demonstration of our proposed SABER architecture is carried out in a 65nm CMOS technology. For RTL (register-transfer level) description and verification, the proposed SABER architecture is implemented in Verilog. Next, the top-level design was synthesized using Cadence Genus and a foundry-provided 65nm standard cell library. After that, the generated netlist was loaded for physical implementation in Cadence Innovus. For physical verification (DRC and LVS), we have used Calibre from Mentor Graphics. Later on, the GDSII file was submitted to the foundry for fabrication. The design implementation was completed in August 2021, the chip underwent fabrication in the September–November time frame, fabricated parts were delivered in December 2021 and finally, we finished with the testing and measurements in February 2022. A total of one hundred chips was fabricated, but only twenty-five were packaged in a Dual-In-Line-28 (DIP-28) form factor.

In Fig. 2, we show the layout of our chip in which the four memory instances are highlighted around the corners across the core. We used M2 to M7 for signal routing purposes. M7 is also utilized for creating a power ring around the core. Moreover, the power is distributed across the core using stripes in M8 and M9. The die size is $960\mu m \times 960\mu m$.The SABER design barely fits in this size. The placement density of the core area is 93.4%, with the remaining 6.6% occupied by decap and filler cells. This incredibly high density made the design very challenging for timing closure. Note the high density zone shown by the yellow dotted lines in Fig. 2. The image inset shows the few empty spaces in orange. For the sake of visibility, all signal routing layers were excluded. Moreover, the I/O pins (seven on each side of the chip) and power stripes routed across the entire chip, horizontally and vertically, are visible. Similarly, we show a die shot of an unpackaged chip taken with the aid of a microscope in Fig. 3. It is possible to recognize the same power routing stripes and IOs as in the layout. In figures 2 and 3, pins of the lower right corner of the chip are highlighted for orientation.

The testing setup utilized to bring-up our chip is shown in Fig. 4. A custom PCB was fabricated to facilitate the test and enable measurements. The packaged chip is placed on the PCB on a DIP-28 socket. Two power sources are connected to the PCB via BNC connectors. Then, the PCB distributes power to the core logic (1.2V) and IO cells (2.5V). On the PCB, small decoupling capacitors are mounted manually for both VDDs. The STM32F446RE [28] microcontroller is integrated with the PCB to drive all the input signals except the faster clock (i.e., *clk*2). The microcontroller also collects the outputs of the chip. To generate the fast *clk*2, we have used a high frequency generator (shown between the two power sources in Fig. 4). Our chip does not contain an internal clock generator.

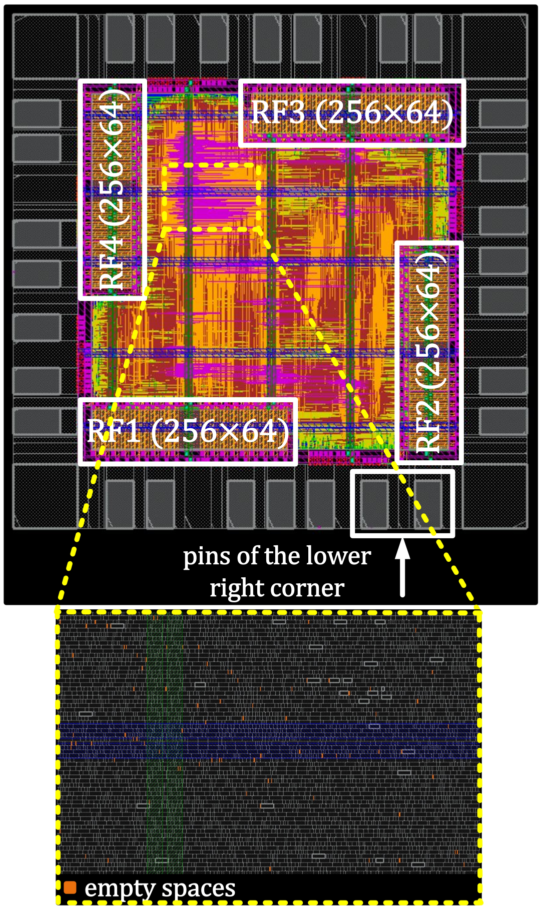

Fig. 2 Screenshot of the chip layout from Cadence Innovus.

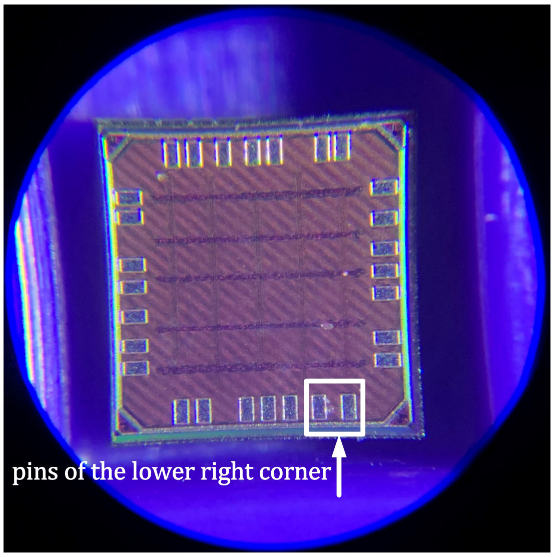

Fig. 3 Microscope view of an unpackaged die where we can identify the IOs (7 on each side) and horizontal & vertical power stripes on the top metal layers.

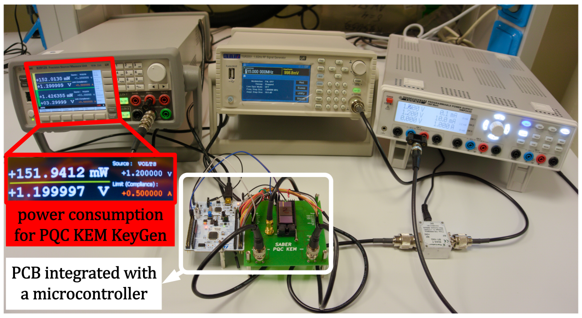
+151.9412 mW
+1.199997 V
+1.200000 V
+0.500000 A
power consumption for PQC KEM KeyGen
PCB integrated with a microcontroller

Fig. 4 Testing setup used to validate our fabricated SABER chip.

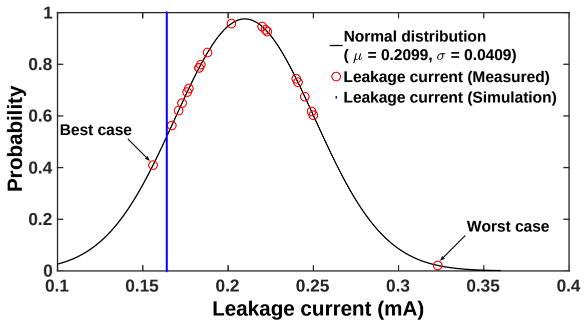

Fig. 5 Average leakage current measurement plotted as a normal distribution. Each red circle corresponds to a single sample or chip. Best and worst values are highlighted.

4.1 Leakage current measurement

In Fig. 5, we plot a normal distribution of the average leakage current measurements of the twenty five packaged chips. We remind the reader that leakage (or state-off) current is the level of current that flows through a device even when the device is not actively computing. The average leakage current is $0.2099mA$ and the standard deviation is 0.0409. The measured data points are plotted as red circles over the normal distribution (black line). The pre-silicon leakage current results (obtained from Innovus) for three different corners, i.e., typical, worst, and best, are $0.164mA$ (on 1.2V), $0.450mA$ (on 1.08V) and $3.20mA$ (on 1.32V) and these values are relative to temperatures of $25°C$, $125°C$ and $0°C$, respectively. The blue vertical line in Fig. 5 shows the pre-silicon leakage current value for typical corner. It appears that the measurement results are a bit more pessimistic than the simulated value predicted, but within the expected range. The best and the worst measured data points are also highlighted in Fig. 5.

Table 1 Timing results for CCA-secure KEM SABER after physical measurements at $715MHz$, nominal 1.2V. The detailed clock cycles description is available in our earlier work [23].

Operation	KeyGen	ENCAPS	DECAPS
Clock cyles	7154	7136	9359
Latency[1] (in μs)	10.00	9.98	13.08

[1]The latency values are calculated using $\frac{clock\ cycles}{715MHz}$.

Table 2 Top level area breakdown of our SABER chip.

Design unit(s)	Utilized area (mm^2)
Pads and I/O ring	0.350
Wrapper + Serial interface	0.041
SABER core	0.232
Memories	0.104

4.2 Area, timing and power results

To identify the highest possible frequency of operation and the corresponding power consumption, we place the sample identified as ‘best case’ on the PCB. At 1.2V, the KEM-supported operations, i.e., KeyGen, ENCAPS, and DECAPS, of SABER can be executed on 770, 715 and $840MHz$. On identical operational conditions, corresponding power values for KeyGen, ENCAPS and DECAPS operations are 151, 158 and $157mW$. Therefore, we have determined that $715MHz$ is the optimal clock frequency where KEM-associated KeyGen, ENCAPS and DECAPS operations perform correctly.

On 1.2V @ $715MHz$, the consumed power of our SABER chip is $151mW$ (for KeyGen), $158mW$ (for ENCAPS) and $152mW$ (for DECAPS). Therefore, the average power consumption is $153.6mW$. The timing results in terms of clock cycles and latency for KEM supported operations are provided in Table 1. Similarly, the top-level area breakdown of our fabricated SABER design is shown in Table 2 where column one provides the design units and column two shows the utilized area.

Table 2 shows that the I/O placement, serial-in/out interface, SABER crypto core, and four instances of small memories utilize 0.350, 0.041, 0.232 and $0.104mm^2$ area out of the total $1mm^2$ chip size. If we calculate the sum of the areas of these blocks, the net area becomes $0.727mm^2$.

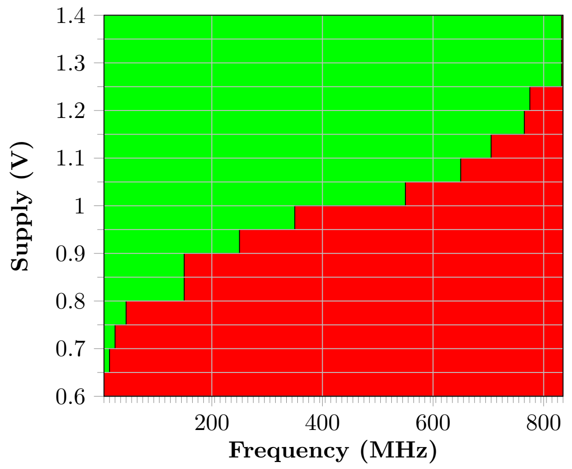

Fig. 6 Graphical representation of the entire range of operation that the chip supports (Shmoo plot).

Most of the remaining area is wasted with mandatory empty spaces between the IO cells and the seal ring, the IO cells and the core, and power rings.

The graphical representation of the complete range of operation that our fabricated SABER chip supports is illustrated in Fig. 6. The horizontal axis is the frequency of operation (in MHz), where each tick represents an increment of $10MHz$. The supplied voltage (in V) is shown on the vertical axis in steps of 0.05V.

Figure 6 demonstrates that the chip is fully operational at a very small clock frequency of $10MHz$ with a supplied voltage of 0.65V. The increase in VDD (from 0.65 to 1.4) results in an increase in the operational frequency (from $10MHz$ to a bit more than $800MHz$).

4.3 Limitations of our chip

As illustrated in section 3.3.1, we employ four smaller memories. The addressing ranges of the memories are [0-255], [256-511], [512-767], and [768-1023]. Unfortunately, due to a logic bug, the first address of memories 2, 3, and 4 is incorrectly decoded and data is overwritten. This minor logical error results in a few flipped bits on the output of the chip when compared with the expected results. This issue could be bypassed for lightSABER by avoiding these memory addresses, but the SABER and fireSABER variants would still encounter this limitation. In either case, the computational blocks of SABER are not affected, nor is the number of memory accesses changed. For this reason, we are confident that the the power values reported in this manuscript are representative.

5 Comparison and Discussion

The comparison of area, timing and power to existing ASIC implementations of SABER is shown in Table 3. The reference design (Ref) is presented in column one. Column two provides the targeted implementation technology (Tech). The area utilization (in mm^2) is shown in column three. Columns four and five present the clock cycles and frequency (Freq. in MHz) values, respectively. The latency (Lat. in μs) values are given in column six. Finally, the last column shows the total consumed power (Pow. in mW).

Fully parallelized architecture of [20]. On a more recent 40nm technology, the implementation results reported in [20] are after logic synthesis. The comparison shows that our fabricated SABER design on 65nm technology utilizes 2.63 times higher hardware resources because we have presented a real chip while in [20] appears to be a block design (i.e., no I/Os). Additionally, we have a serialized infrastructure for communication and debug purposes that also requires a small amount of area.

The clock cycles, reported in [20], for KeyGen, ENCAPS and DECAPS operations are 6.87, 4.95 and 5.57 times lower than our fabricated SABER design. Let us explore the reasons for the clock cycles utilization. For multiplying two 256-degree polynomials in SABER, a centralized schoolbook multiplier architecture of [26] is used in our SABER design. It takes 256 clock cycles to perform one polynomial multiplication. On the other hand, in [20], the use of an 8-level Karatsuba multiplier for the same polynomial length requires 81 clock cycles rather than 256. Despite the different polynomial multiplier, another reason is the use of a high-speed Keccak module comprising two parallel sponge functions (Keccak-f) in [20]. It performs two Keccak-f[1600] computations in each clock cycle and each round of Keccak is performed every 12 clock cycles. A single sponge function in a serial fashion is incorporated in our SABER chip architecture which requires 28 clock cycles to generate 1,344 bits of a pseudo-random string. The lower clock cycles utilization in [20] results in lower

Table 3 Comparison of our SABER accelerator with existing ASIC implementations. All implementation results are for security equivalent to AES-192. Clock cycles and latency values are for KeyGen/ENCAPS/DECAPS. The area of SABER crypto core is reported for [20] and [23] while chip size is reported for [21] and [22]. For this work (TW), chip size is also reported as area.

Ref	Tech/Fab?	Area (mm^2)	Clock cyles	Freq. (MHz)	Lat. (in μs)	Pow. (mW)
[20]	40nm/No	0.38	1040/1440/1680	400	2.6/3.6/4.2	–
[23]	65nm/No	0.31	7154/7136/9359	1000	7.1/7.1/9.3	185.9
[21]	65nm/Yes	1.6	14336/18704/23376	160 @ 1.1V	89.6/116.9/146.1	–
[21]	65nm/Yes	1.6	–/–/–	10 @ 0.7V	–/–/–	0.334
[22]	28nm/Yes	3.6	–/–/–	500 @ 0.9V	–/–/–	39–368
TW	65nm/Yes	1	7154/7136/9359	160 @ 1.2V	44.7/44.6/58.4	43.5
TW	65nm/Yes	1	7154/7136/9359	10 @ 0.7V	715.4/713.6/935.9	0.855
TW	65nm/Yes	1	7154/7136/9359	715 @ 1.2V	10/9.9/13	153.6

Note: The **Fab?** entry determines if the reported results are from simulation/synthesis or from measurement of fabricated chips. For [21], we calculated clock cycles by multiplying the corresponding latency values with 160MHz clock frequency.

latency values (shown in column six of Table 3). Naturally, these numbers have to analyzed with the immense caveat that we are comparing pre-silicon data in 40nm to silicon measurements in 65nm.

High-speed SABER architecture of [23]. Our SABER crypto core employs the same architecture of [23]. The difference is the real chip that we have presented in this work where it contains the real I/Os, serial-in/out interface and three shift registers for accumulating inputs/outputs to/from the fabricated chip. Then, as expected, the utilized hardware resources in [23] is comparatively 3.22 times lower than our fabricated chip. As shown in column four of Table 3, the number of clock cycles is the same between this work and [23].

Concerning comparison to the clock frequency, the value obtained after logic synthesis in [23] is 1GHz which is comparatively 1.39 times higher than our fabricated SABER chip architecture (where we achieved 715MHz). This drop in frequency is somewhat expected since logic synthesis can be too optimistic, specially for a very dense floorplan like the one in our chip. A portion of the drop can also be attributed to process variation, which is also expected. Due to the reduction in clock frequency, the total average power of our fabricated SABER chip is 1.21 times lower as compared to the value obtained after logic synthesis in [23]. In summary, in this work, the presented results for area, timing, and power are more realistic than the synthesis results reported in [23].

SABER design fabricated in [21]. To provide a realistic comparison on an equivalent 65nm technology, as shown in Table 3, we have also used the same conditions (160MHz @ nominal 1.2V and 10MHz @ 0.7V) for measurement results as used in [21]. The comparison is given below.

F=160MHz, VDD=1.2V. For the computation of KeyGen, ENCAPS and DECAPS operations of SABER, our chip is 2, 2.62 and 2.50 times faster in terms of clock cycles and computational time (latency). Because, for multiplying two 256-degree polynomials in SABER, we used a centralized schoolbook multiplier of [26] which requires 256 clock cycles to perform one polynomial multiplication. On the other hand, in [21], the use of Toom-cook with striding of 4 is utilized to reduce the memory requirement to half but with an excess of clock cycles (i.e., 1298 for one polynomial multiplication). As investigated in [29], the Toom-Cook multiplier is inherently more expensive in the hardware area as compared to the schoolbook multiplier. Then, the use of a schoolbook multiplier and a shared shift buffer across various building blocks of SABER results in 1mm^2 chip size which is comparatively 1.6 times lower as compared to [21]. The power comparison is not possible as the relevant information is not reported.

F=10MHz, VDD=0.7V. The comparison to clock cycles and latency parameters is not possible as the corresponding information is not available in [21]. Only the comparison to power is feasible. Comparatively, our fabricated chip consumes 2.55

times more power. The reason is that we fabricated the SABER chip with aid to obtain higher clock frequency while the objective in [21] was low area and power reduction.

If we provide a comparison of $715MHz$ @ 1.2V with the highest obtained clock frequency (i.e., $160MHz$ @ 1.1V) of [21], our fabricated chip is 8.96, 11.80 and 11.23 times faster for the computation of KeyGen, ENCAPS and DECAPS operations, respectively.

Flexible design fabricated in [22]. As shown in Table 3, a realistic and reasonable comparison to area, timing, and power parameters is not possible as the implementation technologies are different (we use 65nm while a modern 28nm is used in [22]). Moreover, our proposed design is specific to SABER while a flexible design for several cryptographic primitives (SABER, NTRU, Dilithium, Rainbow, Kyber and McEliece) is demonstrated in [22]. Depending on the execution of a specific cryptographic protocol, the power values are in the range of 39–$368mW$. Therefore, this comparison is also not possible to provide.

6 Conclusions

This article has presented a fabricated design of the SABER protocol on 65nm technology. The main features of the design are a centralized schoolbook multiplier for 256-degree polynomial multiplications, a shared buffer across several building blocks, pipelining, and distributed memories. Overall, the chip size is relatively small at $1mm^2$, but the achieved frequency is the highest among the considered works. This confirms that our smart memory strategy and pipelining decisions appear to be very beneficial. As future work, we believe there remain many optimizations possible, including better using the distributed memories for improved throughput.

Declarations

Funding

This work was partially supported by the EC through the European Social Fund in the context of the project “ICT programme”. It was also partially supported by European Union’s Horizon 2020 research and innovation programme under grant agreement No 952252 (SAFEST).

Conflict of interest

The authors declare that they have no conflict of interest.

Code availability

The codes generated during and/or implemented during the current study are available in the saber-chip repository, https://github.com/Centre-for-Hardware-Security/saber-chip

Data availability

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

References

[1] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172

[2] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342

[3] W. Diffie, M. Hellman, New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638

[4] R.C. Merkle, Secure communications over insecure channels. Commun. ACM **21**(4), 294–299 (1978). https://doi.org/10.1145/359460.359473

[5] U.S. NSA. Commercial national security algorithm suite and quantum computing faq (last accessed on March 17, 2022). Available at: https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf

[6] E. Yeniaras, M. Cenk, Faster characteristic three polynomial multiplication and its application to ntru prime decapsulation. Journal

of Cryptographic Engineering (2022). https://doi.org/10.1007/s13389-021-00282-7

[7] NIST. Round 3 finalists: Public-key encryption and key-establishment algorithms (last accessed on March 11, 2022). Available at: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[8] A. Basso, F. Aydin, D. Dinu, J. Friel, A. Varna, M. Sastry, S. Ghosh. Where star wars meets star trek: Saber and dilithium on the same polynomial multiplier. Cryptology ePrint Archive, Report 2021/1697 (2021). https://ia.cr/2021/1697

[9] S. Sinha Roy, A. Basso, High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**, 443–466 (2020). https://doi.org/10.13154/tches.v2020.i4.443-466

[10] J. Maria Bermudo Mera, F. Turan, A. Karmakar, S. Sinha Roy, I. Verbauwhede. Compact domain-specific co-processor for accelerating module lattice-based kem (2020). Paper presented at the 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, p. 1–6, July 20–24 2020.

[11] T. Fritzmann, G. Sigl, J. Sepúlveda. Risq-v: Tightly coupled risc-v accelerators for post-quantum cryptography. Cryptology ePrint Archive, Report 2020/446 (2020). https://ia.cr/2020/446

[12] W.K. Lee, H. Seo, S.O. Hwang, A. Karmakar, J.M.B. Mera, R. Achar. Dpcrypto: Acceleration of post-quantum cryptographic algorithms using dot-product instruction on gpus. Cryptology ePrint Archive, Report 2021/1389 (2021). https://ia.cr/2021/1389

[13] H. Becker, J.M. Bermudo Mera, A. Karmakar, J. Yiu, I. Verbauwhede, Polynomial multiplication on embedded vector architectures. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**, 482–505 (2021). https://doi.org/10.46586/tches.v2022.i1.482-505

[14] A. Abdulrahman, J.P. Chen, Y.J. Chen, V. Hwang, M.J. Kannwischer, B.Y. Yang. Multi-moduli ntts for saber on cortex-m3 and cortex-m4. Cryptology ePrint Archive, Report 2021/995 (2021). https://ia.cr/2021/995

[15] A. Karmakar, J.M.B. Mera, S.S. Roy, I. Verbauwhede. Saber on arm cca-secure module lattice-based key encapsulation on arm. Cryptology ePrint Archive, Report 2018/682 (2018). https://ia.cr/2018/682

[16] M.V. Beirendonck, J.P. D'anvers, A. Karmakar, J. Balasch, I. Verbauwhede, A side-channel-resistant implementation of saber. J. Emerg. Technol. Comput. Syst. **17**(2), 1–26 (2021). https://doi.org/10.1145/3429983

[17] T. Fritzmann, M. Van Beirendonck, D. Basu Roy, P. Karl, T. Schamberger, I. Verbauwhede, G. Sigl, Masked accelerators and instruction set extensions for post-quantum cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**, 414–460 (2021). https://doi.org/10.46586/tches.v2022.i1.414-460

[18] A. Abdulgadir, K. Mohajerani, V.B. Dang, J.P. Kaps, K. Gaj. A lightweight implementation of saber resistant against side-channel attacks (2021). In: Adhikari, A., Küsters, R., Preneel, B. (eds) Progress in Cryptology – INDOCRYPT 2021. INDOCRYPT 2021. Lecture Notes in Computer Science(), vol 13143. Springer, Cham. https://doi.org/10.1007/978-3-030-92518-5_11

[19] B. Wang, X. Gu, Y. Yang. Saber on esp32. Cryptology ePrint Archive, Report 2019/1453 (2019). https://ia.cr/2019/1453

[20] Y. Zhu, M. Zhu, B. Yang, W. Zhu, C. Deng, C. Chen, S. Wei, L. Liu, Lwrpro: An energy-efficient configurable crypto-processor for module-lwr. IEEE Transactions on Circuits and Systems I: Regular Papers **68**(3), 1146–1159 (2021). https://doi.org/10.1109/TCSI.2020.3048395

[21] A. Ghosh, J. Mera, A. Karmakar, D. Das, S. Ghosh, I. Verbauwhede, S. Sen. A $334\mu w$ $0.158mm^2$ saber learning with rounding based post-quantum crypto accelerator (2022). Preprint at https://arxiv.org/pdf/2201.07375.pdf

[22] Y. Zhu, W. Zhu, M. Zhu, C. Li, C. Deng, C. Chen, S. Yin, S. Yin, S. Wei, L. Liu. A 28nm 48kops 3.4µj/op agile crypto-processor for post-quantum cryptography on multi-mathematical problems (2022). IEEE International Solid State Circuits Conference (ISSCC), San Francisco, CA, USA, p. 514–516, February 20-26, 2022.

[23] M. Imran, F. Almeida, J. Raik, A. Basso, S.S. Roy, S. Pagliarini. Design space exploration of saber in 65nm asic (2021). Paper presented at the proceedings of the 5th workshop on attacks and solutions in hardware security, virtual event, Republic of Korea, p. 85–90, November 19, 2021.

[24] M. Imran, S. Pagliarini. saber-chip (last accessed on March 21, 2022). Available at https://github.com/Centre-for-Hardware-Security/saber-chip

[25] A. Basso, J.M.B. Mera, J.P. D'Anvers, A. Karmakar, S.S. Roy, M.V. Beirendonck, F. Vercauteren. Saber: Mod-lwr based kem (round 3 submission) (last accessed on March 23, 2022). Available at https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf

[26] A. Basso, S.S. Roy. Optimized polynomial multiplier architectures for post-quantum kem saber (2021). Paper presented at the 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, p. 1285–1290, December 5–9 2021.

[27] NIST. Sha-3 standard: Permutation-based hash and extendable-output functions. FIPS PUB 202 (last accessed on March 9, 2022). Available at https://doi.org/10.6028/NIST.FIPS.202

[28] STM32. Nucleo-64 development board with stm32f446re mcu (last accessed on February 19, 2022). Available at https://www.st.com/en/evaluation-tools/nucleo-f446re.html

[29] M. Imran, Z.U. Abideen, S. Pagliarini. An open-source library of large integer polynomial multipliers (2021). Paper presented at the proceedings of the 24th international symposium on design and diagnostics of electronic circuits systems (DDECS), Vienna, Austria, p. 145–150, April 7-9 2021.