IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 2, pp. 403–425. DOI:10.46586/tches.v2024.i2.403-425

Efficient ASIC Architecture for Low Latency Classic McEliece Decoding

Daniel Fallnich[1,2], Christian Lanius[1], Shutao Zhang[1] and Tobias Gemmeke[1]

[1] RWTH Aachen University, Aachen, Germany
{lanius,zhang,gemmeke}@ids.rwth-aachen.de
[2] now with IBM, Böblingen, Germany
fallnich@ibm.com

Abstract.
Post-quantum cryptography addresses the increasing threat that quantum computing poses to modern communication systems. Among the available "quantum-resistant" systems, the Classic McEliece key encapsulation mechanism (KEM) is positioned as a conservative choice with strong security guarantees. Building upon the code-based Niederreiter cryptosystem, this KEM enables high performance encapsulation and decapsulation and is thus ideally suited for applications such as the acceleration of server workloads. However, until now, no ASIC architecture is available for low latency computation of Classic McEliece operations. Therefore, the present work targets the design, implementation and optimization of a tailored ASIC architecture for low latency Classic McEliece decoding. An efficient ASIC design is proposed, which was implemented and manufactured in a 22 nm FDSOI CMOS technology node. We also introduce a novel inversionless architecture for the computation of error-locator polynomials as well as a systolic array for combined syndrome computation and polynomial evaluation. With these approaches, the associated optimized architecture improves the latency of computing error-locator polynomials by 47% and the overall decoding latency by 27% compared to a state-of-the-art reference, while requiring only 25% of the area.

Keywords: Application-Specific Architecture · Post-Quantum Cryptography · Classic McEliece · Niederreiter Cryptosystem · Hardware Implementation

1 Introduction

Advances in quantum computing are raising concerns that large-scale quantum computers could threaten the confidentiality of modern communications systems, realized by cryptographic algorithms. In order to maintain secure communications, post-quantum cryptography (PQC) aims to defend against attacks from quantum computers by the introduction of so-called quantum-resistant cryptosystems. To assess the suitability of these cryptosystems with respect to diverse applications, the National Institute of Standards and Technology (NIST) is currently evaluating post-quantum key encapsulation mechanisms (KEM) and digital signature algorithms, with the goal to standardize at least one system from each category. For key encapsulation, NIST announced a KEM candidate to be standardized, but continues the KEM evaluation process in a fourth round [AAC+22]. One of the fourth round candidates is a scheme called *Classic McEliece*, which is based on the Niederreiter cryptosystem. Apart from Niederreiter decryption, the Classic McEliece KEM decapsultation comprises a hashing operation (using SHAKE256) and a plaintext confirmation routine. Since these operations are either well studied or straightforward to implement, in the following we focus on the core operations of Classic

Received: 2023-10-15 Accepted: 2023-12-15 Published: 2024-03-12

McEliece decoding. The Classic McEliece KEM, whose associated characteristics allow for high-speed operations, represents a conservative choice among quantum-resistant systems. Confidence in its security follows from an extensive history of cryptanalysis.

Despite its conservative and well-researched security guarantees, the Niederreiter cryptosystem, on which Classic McEliece is based, never experienced wide-spread adoption, due to relatively large key sizes. Nevertheless, the accomplishment of high-speed operations as well as strong security levels suggest the suitability of this cryptosystem for applications in data centers and other application fields, where security and performance are critical. These fields are expected to rank among the early adopters of post-quantum cryptography, where hardware-accelerated high-speed operations are desirable. However, up to now, no ASIC architecture has been proposed for the Classic McEliece KEM and its underlying Niederreiter cryptosystem. Therefore, the present work targets the design and implementation of an ASIC architecture for the code-based Niederreiter cryptosystem, suitable to accelerate Classic McEliece decapsulation. The specifics of an ASIC implementation are thereby taken into account, especially for the selection of algorithms and approaches which are suited to facilitate an efficient implementation. We also show that these approaches and the respective efficient design points differ between ASIC and FPGA implementations. Aligned with the application scenarios described above, the focus of this work lies on facilitating a low latency decoding operation of the Classic McEliece KEM with high area efficiency.

Contributions. In this paper, we present the first optimized and highly area-efficient ASIC implementation of the Classic McEliece decoding operation. This ASIC implementation was taped-out in a 22 nm FDSOI CMOS node. The contributions furthermore comprise a novel constant-time architecture for computing error-locator polynomials based on the inversionless Berlekamp-Massey algorithm, which allows for a significant latency reduction compared to prior approaches. Additionally, we propose a hardware architecture relying on a specialized systolic array, which combines syndrome computation and polynomial evaluation into a single area-efficient module. Lastly, we demonstrate the achievable performance, area and power characteristics of our proposed decoding architecture using simulation results as well as measurements of the manufactured chip.

The remainder of this paper is structured as follows: Section 2 gives a brief background of code-based cryptography as well as binary Goppa codes and their associated decoding procedure. Section 3 provides an overview of previous hardware implementation approaches of code-based cryptosystems, while the proposed ASIC architecture is detailed in Section 4. Implementation aspects of this architecture and the test chip are described in Section 5. Section 6 discusses results of the proposed architecture as well as the manufactured decoding test chip and gives a comparison to previous approaches. Finally, Section 7 summarizes the findings and results.

2 Code-Based Cryptography

The use of error-correcting codes in the design of cryptosystems was already proposed in 1978 by Robert McEliece [McE78]. The code-based Classic McEliece KEM builds upon the *Niederreiter* cryptosystem, which is a "dual" variant of the McEliece cryptosystem [ABC+20]. However, the aforementioned KEM bears the name of the original proposal by Robert McEliece, which used binary Goppa codes and remains unbroken, apart from parameter modifications. Niederreiter's variant of this system allows for an increase in performance, when considering key encapsulation, due to smaller ciphertext and key sizes [WSN17]. However, the original publication also proposed the use of Reed-Solomon codes, which led to successful attacks of this system [SS92]. Therefore, this work considers code-based cryptography using binary Goppa codes.

2.1 Binary Goppa Codes

Binary Goppa codes are a class of linear error-correcting codes, which, due to their structure, exhibit certain characteristics, that are advantageous for applications in code-based cryptography. As a sub-class of Goppa codes, binary Goppa Codes operate in $GF(2^m)$ and possess a minimum distance of $d_{\min} \geq 2t+1$, where t is the number of correctable errors [Ber73].

A binary Goppa code is defined by a monic *generator polynomial* $g(x)$ and a *support vector* of field elements α, described by

$$g(x) = x^t + \sum_{i=0}^{t-1} g_i x^i, \quad g_i \in GF(2^m) \tag{1}$$

and

$$\alpha = (\alpha_0, ..., \alpha_{n-1}), \quad \alpha_i \in GF(2^m), \tag{2}$$

where n is the code length [HP03]. When the generator polynomial $g(x)$ is an irreducible polynomial, the resulting code is called an irreducible Goppa code and in the following this property is assumed for all discussed Goppa codes.

2.2 Decoding Binary Goppa Codes

In order to decode binary Goppa codes and recover a transmitted codeword $c \in GF(2^n)$ from an erroneous codeword $\tilde{c} = c + e$ with error vector e, a decoding procedure is applied, which comprises three major steps: *Syndrome computation*, *solving the key equation* and *determining the roots of the error locator polynomial* [McE02].

The first step in the decoding process is the computation of the syndrome polynomial S. This syndrome is obtained from a received word $\tilde{c}$ as the product $S = H \times \tilde{c}$, where the $t \times n$ matrix H is called a parity check matrix, with $H_{ij} = \alpha_{j-1}^{i-1} / g(\alpha_{j-1})$ [LC87].

Subsequently, the computed syndrome is utilized in order to construct an *error locator polynomial* λ, whose roots correspond to the error positions. The process of computing the aforementioned error locator polynomial is also referred to as solving the key equation [Ber15] given by $S(x)\lambda(x) \equiv \omega(x) \bmod g(x)$, where the error evaluator polynomial ω can be omitted in the present case of binary codes [Hey13]. Since the error locations are represented by the roots of the error locator polynomial, these locations can subsequently be determined by evaluating the error locator polynomial λ for all support elements α_i, where the indices of support elements that are roots of λ indicate an erroneous position. By obtaining the error vector from the roots of λ, the initial codeword can be reconstructed as $c = \tilde{c} - e$, which equals $c = \tilde{c} + e$ in the binary case.

Various algorithms are available for solving the key equation of binary Goppa codes. Since Goppa codes are a sub-class of alternant codes, algorithms designed for alternant codes can be utilized in the decoding process of Goppa codes [MBR15]. However, when applying algorithms, which were not specifically designed for Goppa codes, only $t/2$ errors can be corrected directly, while the Classic McEliece KEM requires a correction capability of t errors. This limitation is overcome by computing a *double-sized syndrome* $S^{(2)} = H^{(2)} \times (S|0)$ [HG13], with the double-sized parity check matrix

$$H^{(2)} = \begin{bmatrix} \frac{1}{g^2(\alpha_0)} & \frac{1}{g^2(\alpha_1)} & \cdots & \frac{1}{g^2(\alpha_{n-1})} \\ \frac{\alpha_0}{g^2(\alpha_0)} & \frac{\alpha_1}{g^2(\alpha_1)} & \cdots & \frac{\alpha_{n-1}}{g^2(\alpha_{n-1})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha_0^{2t-1}}{g^2(\alpha_0)} & \frac{\alpha_1^{2t-1}}{g^2(\alpha_1)} & \cdots & \frac{\alpha_{n-1}^{2t-1}}{g^2(\alpha_{n-1})} \end{bmatrix} . \tag{3}$$

By using $S^{(2)}$ as an input to a general algorithm for solving the key equation instead of S, up to t errors are correctable. Even though this approach allows for the selection of a

Algorithm 1 Inversionless Berlekamp-Massey algorithm

1: **function** INVERSIONLESS BERLEKAMP-MASSEY(S)
2: $\lambda(x) \leftarrow 1$, $b(x) \leftarrow 1$, $l \leftarrow 0$, $\gamma \leftarrow 1, \delta \leftarrow 0$
3: **for** k **from** 0 **to** $2t-1$ **do**
4: $d \leftarrow \delta$, $\delta \leftarrow \sum_{i=0}^{t} \lambda_i \cdot S_{k-i}$
5: $\lambda(x) \leftarrow \gamma \cdot \lambda(x) - \delta \cdot b(x) \cdot x$
6: **if** $\delta = 0$ or $l < 0$ **then**
7: $b(x) \leftarrow x \cdot b(x)$, $l \leftarrow l+1$, $\gamma \leftarrow \gamma$
8: **else**
9: $b(x) \leftarrow \lambda(x)$, $l \leftarrow -1-l$, $\gamma \leftarrow \delta$
10: **end if**
11: **end for**
12: **return** $\lambda(x)$
13: **end function**

suitable algorithm from a broad spectrum, this work focuses on an inversionless variant of the *Berlekamp-Massey algorithm* that allows for an efficient constant-time hardware implementation.

We also investigated the use of *Patterson's algorithm* for solving the key equation, i.e. for the computation of error-locator polynomials. While this specialized algorithm allows for a speedup of the decoding operation, some steps of Patterson's algorithm (e.g. the extended Euclidean algorithm) are not inherently constant-time operations. Without further modifications, the use of Patterson's algorithm renders a decoding design susceptible to timing side-channel attacks and thus undermines its security. Since even a Patterson-based design without constant-time modifications proved to be less area-efficient than the Berlekamp-Massey-based design described in the following, this approach was not pursued further.

2.2.1 Inversionless Berlekamp-Massey Algorithm

A commonly employed algorithm for the construction of error-locator polynomials for binary Goppa codes is the *Berlekamp-Massey algorithm* (BM). Even though this algorithm relies on mostly simple field operations, it also requires field inversions in an iterative loop. In order to facilitate an efficient hardware implementation without repeated inversions, a variant of the BM algorithm is used for the proposed ASIC architecture, called the *inversionless Berlekamp-Massey algorithm* (iBM) [Bur71].

The pseudocode of the iBM algorithm is given in Algorithm 1. It can be seen, that during the coefficient update step (line 5 of Algorithm 1), the coefficients of the error-locator polynomial are multiplied by a scalar field element γ, which causes the inversion of the discrepancy in the subsequent steps to vanish. Due to this scalar multiplication of the coefficients of $\lambda(x)$, the resulting error-locator polynomial is a scalar multiple of the polynomial determined by the original BM algorithm [SS01]. Since only the roots of $\lambda(x)$, which remain unaffected by scalar multiplication, are of interest for decoding Goppa codes, this property of the iBM algorithm does not impact the final solution.

2.3 Classic McEliece KEM

In 1986, Niederreiter proposed an asymmetric code-based cryptosystem [Nie86], which is considered a variant of the McEliece cryptosystem [BLP08]. Instead of encoding a plaintext message in a codeword, Niederreiter's approach employs the error vector e as the plaintext and consequentially the syndrome S as the ciphertext. The resulting cryptosystem exhibits a smaller ciphertext than McEliece's system and requires no CCA2-conversion [HG13],

Algorithm 2 Simplified operations of the Classic McEliece KEM [ABC+20].

1: System parameters m, n, t
2: **function** KEYGEN (SYSTEMATIC FORM)(m, n, t)
3: Generate an uniform random 256 bit seed Δ
4: Generate a pseudo-random bitstring $E = G(\Delta)$ with PRNG G
5: Define s as the first n bits and Δ' as the last 256 bits of E
6: Select a random permutation of 2^m field elements $\alpha = \alpha_0, ..., \alpha_{2^m-1}$ using E
7: Select a random irreducible polynomial $g(x)$ of degree t using E
8: Determine the associated parity check matrix H using $g(x)$ and $\alpha_0, ..., \alpha_{n-1}$.
9: If any of the above three steps fail, set $\Delta \leftarrow \Delta'$ and goto Line 4.
10: Find the systematic form of H using Gaussian elimination as $H = [I_{mt}|T]$.
11: **return** the public key T and the private key $(\Delta, g(x), \alpha, s)$.
12: **end function**
13: **function** ENCAP(T)
14: Generate a random vector $e \in GF(2^n)$ with Hamming weight t.
15: Compute $C_0 = \text{ENCODE}(e, T)$.
16: Compute $C_1 = \text{H}(2, e)$ with hash function H and set $C = (C_0, C_1)$.
17: Compute the session key $K = \text{H}(1, e, C)$.
18: **return** the ciphertext C and the session key K.
19: **end function**
20: **function** ENCODE(e, T)
21: $C_0 \leftarrow [I_{mt}|T] \times e$
22: **return** the partial ciphertext C_0.
23: **end function**
24: **function** DECAP$((\Delta, g(x), \alpha, s), C)$
25: Split the ciphertext C into C_0, C_1 and set $b \leftarrow 1$.
26: Compute $e = \text{DECODE}(C_0, (g(x), \alpha))$.
27: If $\mathbf{wt}(e) \neq t$ or $C_0 \neq [I_{mt}|T] \times e$, set $e \leftarrow s$ and $b \leftarrow 0$.
28: Compute $C_1' = \text{H}(2, e)$ with hash function H.
29: If $C_1' \neq C_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
30: Compute $K = \text{H}(b, e, C)$.
31: **return** the session key K.
32: **end function**
33: **function** DECODE$(S, (g(x), \alpha))$
34: Determine the double-sized parity check matrix $H^{(2)}$ according to Equation 3.
35: Determine the double-sized syndrome $S^{(2)} \leftarrow H^{(2)} \times (S|0)$
36: Compute the error-locator polynomial $\lambda(x)$ using $S^{(2)}$.
37: Retrieve the roots of $\lambda(x)$ as e, i.e. $e_i = 1 \iff \lambda(\alpha_i) = 0$.
38: **return** the error vector e.
39: **end function**

thus this system is favorable for key exchange applications. Nevertheless, the Niederreiter cryptosystem is equivalent to the McEliece cryptosystem in terms of security [LDW94].

Classic McEliece is a key encapsulation mechanism based upon the Niederreiter cryptosystem. A simplified pseudocode of its operations for the case of a systematic parity-check matrix is shown in Algorithm 2, while we refer to [ABC+20] for a detailed description.

Table 1: Parameter sets and key sizes for the Classic McEliece KEM [ABC+20].

Parameter set[a]	Parameters			Public key size [B]	Private key size [B]	Ciphertext size [B]
	n	m	t			
mceliece348864(f)	3488	12	64	261120	6492	96
mceliece460896(f)	4608	13	96	524160	13608	156
mceliece6688128(f)	6688	13	128	1044992	13932	208
mceliece6960119(f)	6960	13	119	1047319	13948	194
mceliece8192128(f)	8192	13	128	1357824	14120	208

[a] Parameter sets with suffix "f" use a parity check matrix of semi-systematic form.

Key generation of the Classic McEliece KEM allows (among others) for the selection of system parameters m (field size), n (code size) and t (maximum error number). With these parameters and a random seed Δ, a random permutation $\alpha = (\alpha_0, ..., \alpha_{2^m-1})$, with $\alpha_i \in GF(2^m)$ of 2^m distinct field elements is selected, which is called the *support vector* [WSN18]. By storing a permutation implicitly in the support vector, the use of a permutation matrix $\bar{P}$, as it is employed in the McEliece cryptosystem, can be avoided [HG13]. Thereafter, a random irreducible *generator polynomial* $g(x)$ of degree t is chosen. The support vector and generator polynomial subsequently allow for the computation of the $t \times n$ parity check matrix H. This parity check matrix is then transformed into its systematic form $H = [I_{mt}|T]$, which reduces the size of the public key to $mt \times (n - mt)$ [WSN18]. Afterwards, the public key is given by the non-systematic part of H, i.e. T, while the private key comprises the generator polynomial $g(x)$ as well as the support vector α and the random bit-strings Δ and s.

Encapsulation in the Classic McEliece KEM requires the generation of a plaintext message represented by an error vector e of Hamming weight t. The subsequent ENCODE subroutine is equivalent to Niederreiter encryption, corresponding to syndrome computation of binary Goppa codes, given by the product of the plaintext e and the parity check matrix $[I_{mt}|T]$, yielding a partial ciphertext, i.e. the syndrome $C_0 = [I_{mt}|T] \times e$. From the error vector e and the partial ciphertext C_0 both the session key K and the ciphertext C are derived by using a hash function H.

Decapsulation of Classic McEliece ciphertexts is constructed from plaintext checks, the hash function H, which is instantiated as SHAKE256, as well as a DECODE subroutine, which corresponds to decryption of Niederreiter ciphertexts and retrieves the error vector e from the ciphertext. Since plaintext checks are straightforward to implement and hardware implementations of SHAKE256 are already well-studied, we focus our work on the acceleration of the *decoding* core operation. Assuming the application of a general algorithm for computing error-locator polynomials, the first step in this decoding operation is the computation of the *double-sized syndrome* as the product $S^{(2)} = H^{(2)} \times (S|0)$, where $H^{(2)}$ denotes the $2t \times n$ *double-sized parity check matrix* given by Equation 3 and $(S|0)$ denotes the syndrome, right-padded with zeros to n bit. Afterwards, an error-locator polynomial $\lambda(x)$ of degree t is constructed from $S^{(2)}$. By evaluating the error-locator polynomial for each element α_i of the secret support α, its roots can be found, where indices of support elements that are roots of $\lambda(x)$ correspond to indices of bits in the error-vector e for which $e_i = 1$.

2.3.1 Parameter Selection

The system parameters n, m and t of the Classic McEliece KEM allow for a tradeoff between security level and performance. The parameter sets given in the Classic McEliece NIST

Table 2: Overview of code-based PQC hardware implementations.

Design[a]		Device / Technology	Security Level [Bit]	Key Eq. Algorithm[b]	f_{clk} [GHz]	Slices / Area[c]	Latency[d] [µs]		
							Key.	Enc.	Dec.
[SWM+10]	(M)	Xilinx LX110T	103	Patterson	0.163	14537	9000	500	1290
[HG13]	(N)	Xilinx LX240	80	Sugiyama	0.220	2474	-	0.91	49.72
[GV14]	(M)	Xilinx XC6VHX255T	128	Patterson	0.254	5357	-	4.74	920
[MBR15][e]	(M)	Xilinx 3AN-1400	80 — 256	Arguello	0.123	2108	-	-	601
[HDYC18]	(N)	Xilinx XC6VLX240T	76.5	Patterson	0.250	4252	-	1.41	798.57
[WSN18]	(N)	Altera 5SGXEA7N	266	BM	0.248	121806	3896.52	21.83	68.77
[CCKA21]	(M)	Xilinx XC7K70T	266	Patterson	0.050	n.d.	-	n.d.	n.d.
[CCD+22][f]	(N)	Xilinx XCZU49DR	266	-	0.155	55489	17200	-	-
		Xilinx XC7A200T		BM	0.147	30786	-	398	1230
[QSTW23]	(N)	Altera 5SGXEA7N	266	ePiBM	0.340	14913	-	-	49.5
This work	(N)	GF 22 nm FDSOI	266	iBM	2	0.075 mm^2	-	-	6.64

[a] (M) = McEliece, (N) = Niederreiter

[b] Algorithm for solving the key equation

[c] Results are given for the total slices/area of an implementation.

[d] Key. = key generation, Enc. = encryption, Dec. = decryption

[e] Results are listed for a 128 bit parameter set.

[f] Results are listed for complete KeyGen., Encap. and Decap. operations.

submission are listed in Table 1. Due to its high-speed operations and confidence in its security guarantees, the Classic McEliece KEM is inherently well suited for applications in critical environments, such as data centers. The proposed architecture targets this scenario with its high-speed and low-area objective. Therefore, an architecture supporting long-term security for critical data is appropriate. Due to this reason, a parameter set resulting in a security level of 266 bit was selected, with the associated parameters $n = 6960$, $m = 13$ and $t = 119$. This parameter set still provides a 128 bit "quantum-resistant" security level when considering attacks using quantum computers executing Grover's algorithm [WSN18] and follows the recommendations for PQC given in [ABB+15]. While in the following we limit out discussion to the aforementioned parameter set, the described algorithms and approaches are transferable to other parameter sets as well.

3 Previous Work

While only very few hardware implementations for the Classic McEliece KEM exist, several FPGA implementations were proposed for the associated code-based cryptosystems. Due to its history and associated position as a reliable conservative choice, the McEliece cryptosystem has received significantly more attention than the variant proposed by Niederreiter. Nevertheless, several implementations of the Niederreiter cryptosystem do exist, which are listed in Table 2, in addition to McEliece implementations as well as the iBM-based test chip proposed below for comparison. "Low-reiter", for instance, is a Niederreiter software implementation, which targets 8-bit AVR microcontrollers and provides a security level of 80 bit, while utilizing Patterson's algorithm for the computation of error-locator polynomials [Hey10]. Considering FPGA implementations, architectures for Niederreiter encryption and decryption with 80 bit security were proposed in [HG12] and [HG13]. This led to two designs employing Patterson's algorithm and the BM algorithm, respectively, although the results are not directly transferable[1] to Niederreiter implementations conforming to the Classic McEliece KEM submission, which was proposed later. In 2018, Hu et al. presented an ASIP design implemented on an FPGA, which supports Niederreiter encryption as well as decryption. This architecture is furthermore capable of generating

[1] This is due to the fact, that the implementation in [HG13] assumes the double-sized parity check matrix $H^{(2)}$ as a part of the private key.

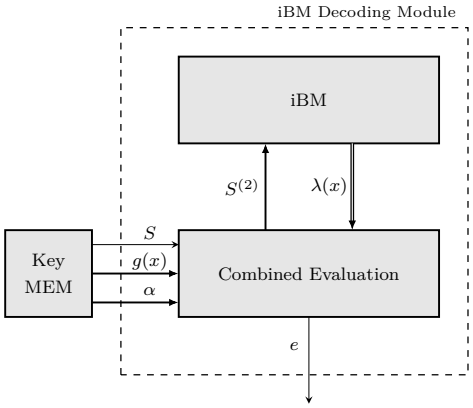

Figure 1: Overview of the proposed iBM-based Classic McEliece decoding architecture.

signatures using the Niederreiter cryptosystem [HDYC18]. Furthermore, an FPGA implementation of the complete Niederreiter system providing long-term security with a security level of 266 bit is given in [WSN17] and [WSN18]. The aforementioned implementation is scalable with respect to different parameter sets and employs the BM algorithm for constant-time decryption. Building upon this implementation, Chen et al. introduced the first complete standard-compliant FPGA implementation for the Classic McEliece KEM in [CCD+22]. This constant-time design supports the different Classic McEliece parameter sets and proposes several optimizations, especially for key generation. Lastly, an FPGA implementation using a BM variant and an optimized additive FFT polynomial evaluation scheme is given in [QSTW23]. To the best of our knowledge, no ASIC architecture apart from a HLS-based comparison of PQC KEM algorithms [BSNK19] exists for the Classic McEliece KEM and its underlying Niederreiter cryptosystem.

4 Architecture Design

An overview of our proposed application-specific hardware architecture is shown in Figure 1. For this decoding architecture, ciphertext and private keys are assumed to be located in an external key memory, which facilitates a fair comparison of architectures without the influence of a constant large area contribution of the key memory.

The iBM-based decoding module comprises two sub-modules: A *combined evaluation module* for computation of double-sized syndromes and polynomial evaluation as well as an *iBM module* for computation of the error-locator polynomial. A decoding operation is executed by first computing the double-sized syndrome using the combined evaluation module. This double-sized syndrome is then employed by the iBM module to construct an error-locator polynomial $\lambda(x)$ from the syndrome. This error-locator polynomial is fed back into the combined evaluation module, which evaluates the polynomial at all points corresponding to support vector elements α_i and returns the plaintext represented by an error-vector e, thus completing the decoding operation.

4.1 Finite Field Arithmetic

Since arithmetic modules for finite field arithmetic represent the fundamental components of the aforementioned modules of the Classic McEliece decoding architecture, they should be carefully designed, such that a low latency architecture with reasonable area efficiency is facilitated. In the following, irreducible polynomials required for operations in $GF(2^m)$ are assumed to match the polynomial given in the Classic McEliece KEM proposal

[ABC+20]. For all arithmetic modules a standard basis representation, i.e. a representation as coefficients of a polynomial, is assumed, thus allowing for fast multiplier implementations [DInS09]. Efficient design points for these arithmetic modules will be detailed in the following.

4.1.1 Operations in $GF(2^m)$

Addition in a finite field $GF(2^m)$ with elements represented as polynomials equals the addition of polynomials. $GF(2^m)$ addition is straightforward and performed by bit-wise XOR of field elements, because polynomial addition is achieved by addition of coefficients, which in $GF(2)$ is equivalent to the logical XOR operation.

Multiplication in a finite field can be implemented by using a multitude of approaches. For fast multiplication algorithms, such as Montgomery or Karatsuba-Ofman multiplication, it is assumed that these algorithms do not allow for efficient implementations for the choice of $m = 13$, which is congruent with the findings of Wang et al. [WSN17]. Hence, Mastrovito multiplication is employed in the proposed decoding architecture, as an approach featuring low latency multiplication with moderate area footprint. Low latency operations are achieved by combining the partial product computation with the reduction steps [Mas89]. A finite field multiplier can thereby be designed as a combinatorial function with low latency. Although optimizations for Mastrovito's approach exist (see e.g. [PDCS07a] or [PDCS07b]), improvements for the present case of an irreducible pentanomial are marginal compared to the additional design effort, hence the original approach by Mastrovito is used here.

Squaring in $GF(2^m)$ can be implemented using a field multiplier. However, squaring using multipliers is relatively costly in terms of area footprint. Exploiting the observation that in binary fields, squaring can be expressed as

$$c = a^2 \bmod f(x) \equiv a_{m-1}x^{2(m-1)} + a_{m-2}x^{2(m-2)} + ... + a_1x^2 + a_0 \bmod f(x)\,, \qquad (4)$$

less complex implementations are possible [DInS09]. By applying a reduction to the aforementioned squared polynomial, squaring is therefore implemented by a combinatorial low latency approach similar to the Mastrovito multiplication.

Inversion is an expensive operation in a finite field $GF(2^m)$. Available inversion approaches, such as application of the extended Euclidean algorithm, exponentiation or table lookup, differ in the attainable latencies and area costs. Since for iBM-based decoding, inversion operations are only performed outside of iterative loops, these operations are mainly required to match the throughput of the polynomial evaluation module described below (one field element per cycle), to avoid introducing bottlenecks. Therefore, field inversion was implemented using a small array of multipliers and squaring modules, in order to balance area footprint and decoding latency. By using Fermat's little theorem, inversion can thereby be evaluated by computing a^{2^m-2} in a square-and-multiply scheme [DPBM00]. As the squared inverses of field elements are required for the computation of a double-sized syndrome, these squared inverses can be directly obtained by computing the power $a^{2^m-3} \equiv a^{-2} \bmod f(x)$ instead of $a^{2^m-2} \equiv a^{-1} \bmod f(x)$.

4.2 Error-Locator Polynomial Computation

Computing error-locator polynomials by utilizing the inversionless Berlekamp-Massey algorithm allows for a constant time implementation while employing simple finite field operations. The iBM module presented below aims to reduce the latency of error-locator polynomial construction compared to previous implementations while at the same time maintaining a balanced design point with high area efficiency. Latency reduction without

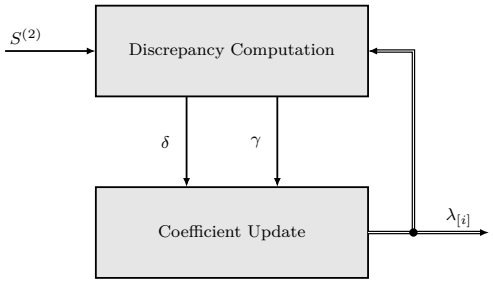

Figure 2: Block diagram of the pipelined iBM module, operating on blocks $\lambda_{[i]}$ of the error-locator polynomial.

adverse effects on the area footprint is achieved by various novel approaches[2], which are described below.

The architecture of the proposed iBM module (shown in Figure 2) comprises two sub-modules, associated with the primary steps of the iBM algorithm, *discrepancy computation* and *coefficient update*. In order to avoid a large and thus inefficient design, fully parallelized operation on all $t+1 = 120$ coefficients of the error-locator polynomial should be avoided. Therefore, the iBM module operates on subsets of 20 coefficients in parallel. This block-wise computation allows for the introduction of two primary measures for latency reduction: *pipelined operation* and *coefficient update bypass*.

Pipelined operation thereby implies that as soon as the first block of coefficients is updated during an iteration, this block is fed into the discrepancy computation module, in order to start discrepancy computation of the next iteration. With this scheme the cycles for a single iBM operation are reduced from $2(t+1)/20+1 = 13$ to $(t+1)/20+2 = 8$ with a constant number of 2 cycles for the final accumulation step and the update of the first coefficient block. Thus, the total latency for the computation of an error-locator polynomial is reduced by approximately 38%.

It was shown that for the iBM algorithm the upper $t-k$ coefficients of the error-locator polynomial are 0 in iteration k [SS01]. Therefore, considering architectures with block-wise operations on these coefficients, it is possible to bypass updates of coefficient blocks that only contain zero coefficients. This measure reduces the total amount of cycles for our pipelined iBM module from $2t \cdot (t+1/20+2) = 1904$ to $t \cdot (t+1/20+2) + \sum_{i=1}^{(t+1)/20} 20 \cdot (i+2) = 1612$, corresponding to a relative latency reduction of approximately 15%.

4.2.1 Discrepancy Computation

Discrepancy computation constitutes the first step in an iBM iteration. As described before, the discrepancy $\delta^{(k+1)}$ in an iteration k is computed as

$$\delta^{(k+1)} = \sum_{i=0}^{t} \lambda_i^{(k)} \cdot S_{k-i}^{(2)} \,. \tag{5}$$

Due to the relative shift of syndrome against error-locator coefficients in each iteration, implementation of this structure can be performed using a shift register. The proposed discrepancy computation module features a shift register that shifts only once per iteration and selects blocks of coefficients via multiplexers, in order to reduce switching activity. This approach, which is shown in Figure 3, furthermore facilitates the coefficient update bypass described before.

[2] After tape-out of the proposed decoding ASIC, a similar approach to the iBM implementation described below was published in [QSTW23]. However, at equivalent folding levels, our proposed design exhibits a 15% lower cycle count, due to the implemented coefficient update bypass optimization.

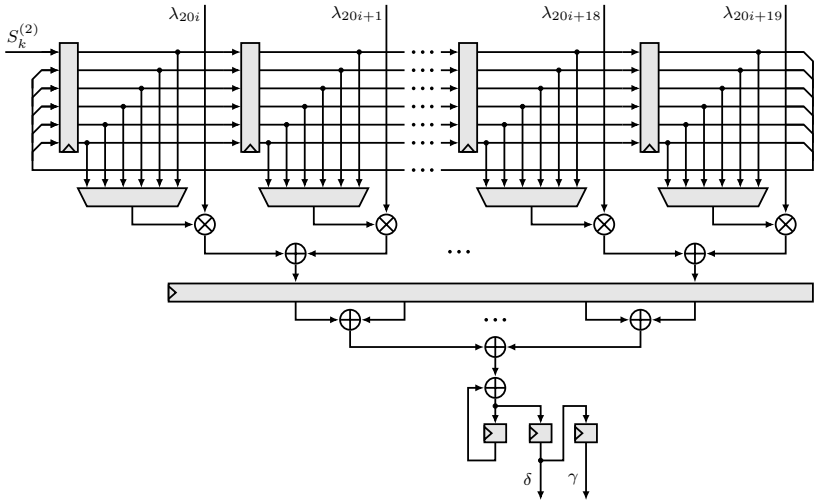

Figure 3: Block diagram of the iBM discrepancy computation module.

It can be observed in Figure 3 that syndrome-error-locator products are pairwise added followed by a register stage. The remaining additions after the register stage are realized as an adder tree with an additional accumulator register. This split, involving an intermediate register stage, ensures a short critical path of the whole module.

4.2.2 Coefficient Update

Following the discrepancy computation, coefficients of the error-locator polynomial are updated in the coefficient update module according to

$$\lambda(x)^{(k+1)} = \gamma^{(k)} \cdot \lambda(x)^{(k)} - \delta^{(k+1)} \cdot b(x)^{(k)} \cdot x\,. \tag{6}$$

Since the formulation of the coefficient update procedure exhibits a regular structure, this procedure is ideally suited to be implemented using a systolic array. Therefore, the coefficient update module relies on such a systolic array, which is depicted in Figure 4. In the proposed systolic array, blocks of error-locator polynomial coefficients remain stationary in an associated processing element (PE), while the auxiliary coefficients b_i are shifted between the PEs, which corresponds to the multiplication by x. Furthermore, the currently updated subset of coefficients is also stored in dedicated registers, which makes these coefficients accessible for the discrepancy computation module. The use of multiplexed registers in this coefficient update module allows for the coefficient update bypass.

4.3 Polynomial Evaluation

Polynomial evaluation in finite fields is an essential operation for the decoding of Classic McEliece ciphertexts, as it is necessary for computation of the double-sized syndrome as well as root searching of the error-locator polynomial. Assuming the availability of a low latency module for error-locator polynomial computation, polynomial evaluation might account for the majority of the resulting decoding latency and area footprint of a Classic McEliece decoding architecture, e.g. as seen in [WSN18]. Therefore, careful design of a polynomial evaluation architecture is mandatory for efficient implementation of the whole decoding process.

Even though sophisticated polynomial evaluation approaches, e.g. FFT-based schemes, are available, the large number of intermediate results that have to be stored when

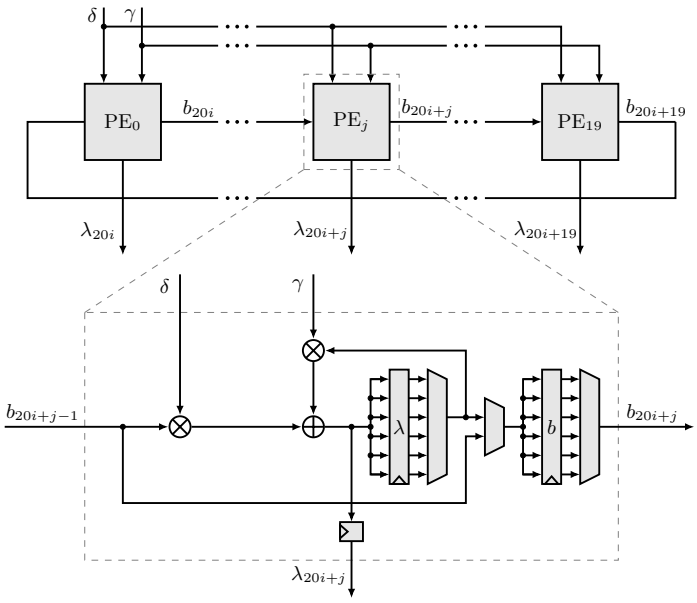

Figure 4: Block diagram of the iBM coefficient update module's systolic array (top) with detail view of a processing element (PE) at the bottom.

using these schemes results in large memories, thus negatively impacting area efficiency. Furthermore, evaluating a polynomial in a specific order results in a high memory read-out latency, when memory access schemes supplying one field element per cycle are considered. Therefore, Horner's method is employed in this work for polynomial evaluation. This approach, which will be described in the following, brings the additional advantage that the resulting architecture can be reused for double-sized syndrome computation.

4.3.1 Horner's Method

Horner's rule allows for the computation of the polynomial point $g(\alpha_i)$ as

$$g(\alpha_i) = (((g_t\alpha_i + g_{t-1})\,\alpha_i + g_{t-2})\,...)\,\alpha_i + g_0\,, \tag{7}$$

which eliminates exponentiations and thus allows for the evaluation of a polynomial using solely finite field multiplication and addition.

The proposed polynomial evaluation module considers *coefficient stationary* processing. Hereby $t+1$ coefficients are stored in t PEs and field elements are fed into a systolic array, while partial sums are transferred between PEs of such an array. Even though a polynomial of degree t possesses $t+1$ coefficients, the coefficient of x^t can be treated as the first partial sum and thus only t PEs are required. A systolic array was derived from this approach, which features reduced fanout and memory bandwidth requirements.

The aforementioned systolic array for polynomial evaluation is suited to directly evaluate the error-locator polynomial. For iBM-based decoding, however, in addition to polynomial evaluation, computation of a double-sized syndrome is necessary, which is described in the following.

4.3.2 Double-Sized Syndrome

The double-sized syndrome, which is required for correcting t errors when utilizing the Berlekamp-Massey algorithm for constructing error-locator polynomials, can be computed as the product of the zero-padded syndrome S and a double-sized parity check matrix $H^{(2)}$

as $H^{(2)} \times (S|0)$. With the definition of each element of the double-sized parity check matrix as $H^{(2)}_{i,j} = \alpha_j^i / g^2(\alpha_j)$, with $i \in [0, 2t-1]$ and $j \in [0, n-1]$, several observations facilitate optimizations of the double-sized syndrome computation. First, the computation of the double-sized parity check matrix $H^{(2)}$ can be merged with the following vector-matrix multiplication by multiplying each value of $g^2(\alpha_j)$ by the corresponding syndrome bit before constructing the double-sized parity check matrix and accumulating its columns [WSN18]. Since a field element thereby gets multiplied by a single syndrome bit, this operation is easily realized by the AND operation of each element bit and the syndrome bit. Furthermore, due to zero-padding of the syndrome polynomial, the last columns of $H^{(2)}$ have no influence on the final vector-matrix product and can thus be omitted from computation entirely, where only the first $mt = 1547$ columns have to be computed. Lastly, it can be observed that each row of the double-sized parity check matrix can be obtained from the previous row by element-wise multiplication of a row by the truncated support vector $(\alpha_0, ..., \alpha_{mt-1})$. This allows for the iterative computation of the double-sized parity check matrix by a single multiplication per matrix element.

Using above observations, a systolic array can be designed for double-sized syndrome computation. The double-sized syndrome systolic array operates in a *partial sum stationary* scheme, where generator polynomial values and support vector elements are transferred between PEs. Instead of focusing on a single row or column of the double-sized parity check matrix, the array computes entries of multiple rows and columns in parallel, i.e. the entries $H^{(2)}_{i,0}, H^{(2)}_{i-1,1}, H^{(2)}_{i-2,2}, ..., H^{(2)}_{i-mt,mt}$ are calculated in parallel in iteration i of this scheme. This approach exhibits the advantage of reduced storage and memory bandwidth requirements, with additional improvements for fanout of support vector elements. In the present case, an array consisting of t PEs computes t coefficients of the double-sized syndrome concurrently.

4.3.3 Combined Evaluation Module

Even though the aforementioned optimizations for polynomial evaluation and double-sized syndrome systolic arrays enable a significant circuit size reduction compared to unoptimized designs, these arrays would still occupy a majority of the area of an associated Classic McEliece decoding architecture. Hence, it is desirable to further decrease the area footprint of the modules for these operations. When analyzing the systolic arrays for polynomial evaluation and double-sized syndrome computation, it becomes apparent that these arrays resemble each other. Instead of instantiating two distinct arrays for evaluation and syndrome computation, it is therefore advantageous to employ a single combined systolic array, which executes both operations. This *combined evaluation* module[3] will be described in the following in greater detail.

The systolic array of the combined evaluation module employed for iBM-based decoding is illustrated in Figure 5. Apart from the combined systolic array, this module comprises $m = 13$ parallel AND gates for multiplying generator polynomial values by syndrome bits and a constant delay FIFO used to buffer generator polynomial values as well as values of the double-sized parity check matrix. Furthermore, a shift register for parallel-to-serial and serial-to-parallel conversion, a comparator for determining roots of the error-locator polynomial, and a squared inversion module are used in the combined evaluation module.

Double-sized syndrome computation is initiated on the combined evaluation module by sequentially loading coefficients of the generator polynomial $g(x)$ into the shift-register. Subsequently, these coefficients are loaded into the combined systolic array and support vector elements are read from memory to obtain generator polynomial values $g(\alpha_i)$. The

[3]Note, that the combined double-sized syndrome computation and polynomial evaluation approach was developed independently from another implementation, which also combines these two operations [MBR15]. However, the implementation of Massolino et al. employs a different dataflow and does not consider interleaving of syndrome and error-locator polynomial computations.

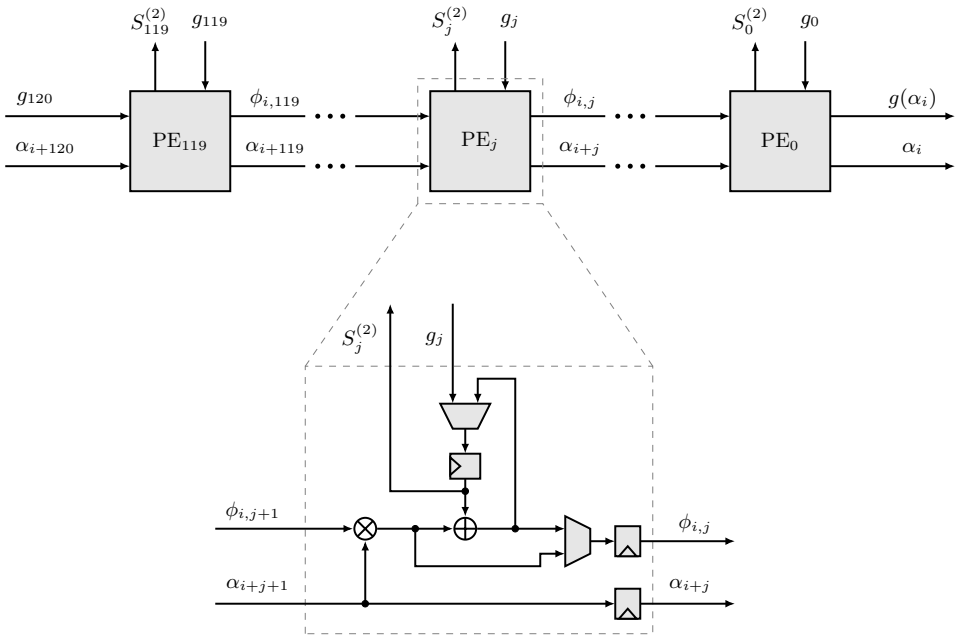

Figure 5: Block diagram of the combined systolic array (top) with detail view of the jth PE (bottom).

squared inverses of these values are then stored in the constant delay FIFO. Afterwards, these polynomial values are multiplied by the corresponding syndrome bits and are fed back into the systolic array[4], in order to compute the first half of a double-sized syndrome, which is loaded into the shift register, hence allowing to immediately resume double-sized syndrome computation. The product of support element powers and the squared inverses of generator polynomial values that exit the systolic array in this first iteration are required to resume computation of the double-sized syndrome and are thus stored in the constant delay FIFO. After completing the computation of the first half of the double-sized syndrome, the respective products and support vector elements are read from memory and from the FIFO to obtain the second half of the double-sized syndrome. Using this sequence, the double-sized syndrome computation is completed within

$$N_{\mathrm{cyc},t,S^{(2)}} = t + m - 1 + mt + 1 + t + mt + 1 + t + mt = 5012 \tag{8}$$

cycles, where t or $t + m - 1$ cycles are required to fill the array, mt cycles are required to iterate over support vector elements and single cycles are required for multiplying the generator values by syndrome bits and to store the first half of the double-sized syndrome.

For root search of the error-locator polynomial the sequence of operations is considerably simpler: After block-wise loading of error-locator coefficients into the combined systolic array (not shown in Figure 5), the error-locator polynomial is evaluated for all support vector elements and the downstream comparator module converts polynomial values to bits of the error vector, where a root of this polynomial corresponds to a 1 in the error vector.

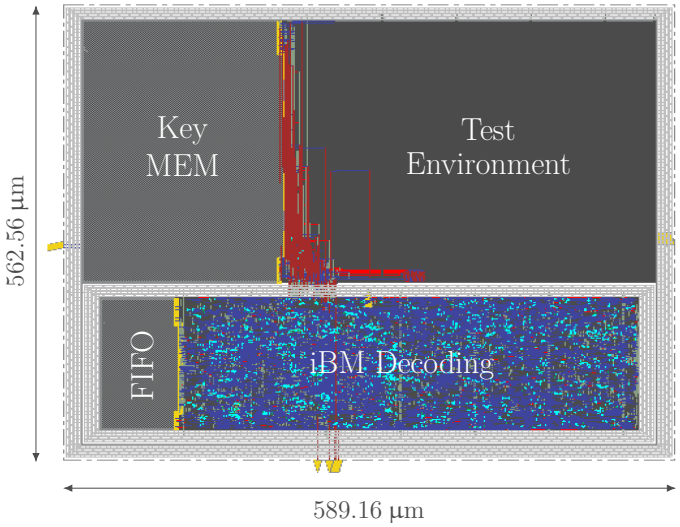

Figure 6: Layout of the iBM-based test chip with the test environment above the iBM decoding module containing the combined evaluation module's FIFO.

5 Implementation

The modules described in Section 4 were implemented in Verilog and SystemVerilog. An ASIC design, corresponding to the iBM-based decoding module, was developed using a digital design flow relying on Cadence Genus and Innovus software systems for the 22 nm FDSOI CMOS technology node from GlobalFoundries (GF). This design was integrated into a test chip, which combines the iBM decoding module with additional on-chip test modules, in order to facilitate straightforward evaluation. Apart from on-chip clock generation logic, these test modules also comprise a CRC unit used to compute and compare CRC values of a decoded plaintext to an expected CRC value as well as an UART interface for control and memory access. The layout of the test chip is shown in Figure 6, with the decoding module at the bottom and the key memory as well as the test modules at the top. The test chip was taped out and manufactured by GF in the aforementioned 22 nm FDSOI node.

The combined evaluation module's FIFO was implemented using a memory macro from the GF 22 nm FDSOI memory portfolio. While the decoding ASIC architecture operates at a clock frequency of 1 GHz, synthesis results suggest (see Figure 7) that higher clock frequencies are achievable, since the clock frequency is limited by the delay of memory macros and not by the decoding logic itself. Therefore, the test chip design adopts multicycle paths across the key memory and evaluation module's FIFO, which allows for clock frequencies of up to 2 GHz. In this scheme, SRAM macros are instantiated with a doubled read and write width, in order to read or write two field elements in two cycles (corresponding to a single 1 GHz clock cycle). From these double width elements, high and low elements are selected individually in consecutive cycles. Due to the linear memory access patterns found in the Classic McEliece decoding operation, this scheme thus effectively mimics the behaviour of a memory macro with read and write port widths of a single field element operating at 2 GHz.

For the sake of accelerating the design process, we implemented the described ASIC in a non-parametrized fashion and only directly support the parameter set described in Sub-

[4]Note, that the first PE of the combined systolic array is slightly modified, since it is possible to omit the multiplication by a support element for the first row of $H^{(2)}$.

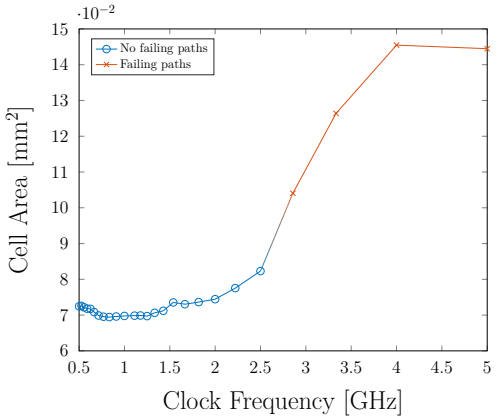

Figure 7: Clock frequency dependent cell area of the iBM decoding module with differentiation of design points with and without failing paths after synthesis.

subsection 2.3.1. However, the employed optimization techniques are readily transferable to other parameter sets as well. Due to its systolic design, the combined evaluation module can simply be adapted to other parameter sets by instantiating a different number of PEs corresponding to the system parameter t and adapting the depth of shift registers and the instantiated FIFO SRAM. The iBM module can be adapted to support different parameter sets by adjusting the number of iBM iterations as well as the number of PEs, shift register stages and PE registers according to a different number of coefficients processed in parallel. Additionally, adjustments to control and dataflow logic would be necessary, in order to ensure the correct interaction of all modules.

6 Evaluation

In order to evaluate the designed Classic McEliece decoding ASIC architecture and to assess its suitability for efficient low latency decoding, this architecture should be compared to previous state-of-the-art approaches. However, no previous ASIC design is available for comparison. The FPGA design introduced by Wang et al. is the only available open-source Classic McEliece decoding architecture that supports the parameter set selected in our research[5]. The authors of that architecture state that the majority of their FPGA design can be re-used for the development of an ASIC design [WSN17]. Therefore, such an ASIC design was created from the given Verilog source using the given speed-optimized design point for comparison purposes, where only the block RAM modules, instantiated for polynomial evaluation, were exchanged with appropriate memory macros. The resulting ASIC design will subsequently be referred to as the *baseline design*.

6.1 Error-Locator Polynomial Computation

Results for the proposed iBM module as well as the BM module from the baseline design are shown in Table 3. When juxtaposing the iBM module and the baseline Berlekamp-Massey module, it becomes apparent that the proposed iBM design requires significantly fewer adders and multipliers. Additionally, due to the utilization of the inversionless Berlekamp-Massey variant, the iBM module does not rely on a fast field inversion module and

[5]The authors of [CCD+22] also provide an open-source FPGA implementation for the selected parameter set. However, since this design adopts the decoding architecture from [WSN17], our comparison can be applied to this implementation as well.

allows for a substantial reduction of the number of registers, compared to the Berlekamp-Massey implementation of the baseline design. Nevertheless, the proposed iBM design still allows for a speedup of approximately 1.91, which can be explained by the systolic array approach of the iBM module that specifically considers operations on coefficient blocks and introduces various optimizations, such as the pipelined discrepancy and coefficient update computations. In contrast, the baseline Berlekamp-Massey implementation takes operations on coefficient blocks into account only by reducing the number of parallel multipliers, without further optimizations.

Table 3: Arithmetic module and cycle counts for different error-locator polynomial computation designs.

Design	Adders	Multipliers	Inversions	Registers	Cycles
iBM	40	60	0	5109	1619
Baseline	240	80	1	13079	3095

6.2 Error-Locator Polynomial Evaluation

The proposed iBM-based ASIC architecture employs a systolic array derived from Horner's method for polynomial evaluation, while the baseline design uses an additive FFT approach, which allows for a reduction of the multiplicative complexity [GM10]. With the latency of Horner's method for polynomial evaluation depending on the number of evaluated elements the scenario of error-locator polynomial evaluation for $n = 6960$ elements is assumed here for comparison.

It can be seen in Table 4 that the use of the FFT in the baseline designs proves to be very effective in reducing the multiplicative complexity. As a result, not only the number of required multipliers is drastically reduced, compared to the proposed evaluation module, but also the number of required cycles for polynomial evaluation. These reductions, however, constitute a trade-off, associated with more than tripling the number of required registers, compared to the proposed systolic array. Additionally, strongly parallelized evaluation of multiple coefficients requires wide high-bandwidth memories, to store a large number of polynomial values computed in parallel, which further negatively impact the area footprint. The introduced systolic array using Horner's method, on the other hand, allows for local data transfer, hence no additional large memories are needed.

In addition to the area implications described above, the utilization of an additive FFT approach in the baseline design entails another operation: Because polynomial values of field elements are evaluated in a specific order, these values need to be accessed from memory according to the permutation of the support vector. Thus, a subsequent memory read-out operation of polynomial roots is required for the baseline design, which ultimately eliminates the latency advantage of the additive FFT, as shown in Table 4. Obtaining the roots of the polynomial without a sequential read-out of the FFT memory would therefore require some changes to the design approach in [WSN17, WSN18] and some algorithmic optimizations.

Table 4: Module and cycle counts for error-locator polynomial root search approaches.

Design	Adders	Multipliers	Registers	Memory Size	Cycles[a]	
					Eval.	Mem.
Horner	119	119	6240	-	7086	-
FFT	128	40	20069	$2 \cdot (768 \times 70)$	1082	6972

[a] Eval. = evaluate error-locator polynomial, Mem. = memory read-out

6.3 Double-Sized Syndrome Computation

Double-sized syndrome computation in the iBM-based decoding architecture is executed on the combined evaluation module, while in the baseline design this operation is performed by the FFT polynomial evaluation module and a dedicated double-sized syndrome module. For the baseline design this double-sized syndrome module comprises 40 multipliers and a fast field inversion module. The proposed combined evaluation module, on the other hand, computes squared inverses using a square-and-multiply approach.

Results of both approaches are shown in Table 5. Syndrome computation in the baseline design requires fewer multipliers than in the proposed iBM architecture, resulting from the use of FFT-based polynomial evaluation, which leads to an increased area requirement from the associated memory macros. While the polynomial evaluation portion exhibits a longer latency in the proposed iBM architecture, the actual syndrome construction is faster than in the baseline design, due to the use of the combined evaluation and syndrome computation approach using a higher number of field multipliers.

Table 5: Module and cycle counts for double-sized syndrome computation approaches.

Design	Adders	Multipliers / Squarers	Inversions	Registers	Memory Size	Cycles[a]	
						Eval.	Synd.
iBM	119	130 / 12	0	6552	1568×13	1800	3338
Baseline	170	80 / 1	1	25757	$2 \cdot (768 \times 70)$	1082	4658

[a] Eval. = evaluate generator polynomial, Synd. = compute double-sized syndrome

6.4 Design Space Comparison

In addition to the assessment of decoding architectures on a module-wise basis, as described above, these architectures should also be evaluated in their entirety, in order to consider dataflow dependencies and interactions across module boundaries. Therefore, the proposed ASIC design is compared to the baseline design with respect to the key performance indicators decoding latency, area footprint and power dissipation, which allow to determine different points in the design space. The designs are compared at a clock frequency of $f_{\text{clk}} = 1$ GHz.

Results of the aforementioned designs after Place-and-Route are summarized in Table 6, where additional area efficiency figures are given as the reciprocal of the product of decoding latency and area requirements. Table 6 also includes results of the 2 GHz iBM test chip design, as an example of a decoding design operating at a higher clock frequency. The difference in cycle counts between the proposed 1 GHz and 2 GHz decoding architectures results from synchronization cycles, required by the introduced multicycle memory access scheme of the 2 GHz design. The aforementioned results prove the effectiveness of the introduced optimization measures for the design of an efficient Classic McEliece decoding architecture.

In terms of latency, the proposed iBM-based decoding architecture achieves the lowest latency of the compared designs, which follows from the fast iBM module in conjunction with optimized polynomial evaluation and double-sized syndrome computation. While the baseline design features the fastest polynomial evaluation approach, the longer latency of remaining decoding steps in this architecture outweigh this advantage. Therefore, it follows that the total decoding latency for the baseline design is approximately 28% higher than the latency of the proposed iBM-based decoding architecture.

The differences of the compared approaches also manifests in the attainable area of the associated implementations. By combining multiple decoding steps into the same module and balancing the number of parallel units, the iBM-based architecture achieves a

Table 6: Summary of simulation results of the compared ASIC designs after Place-and-Route[a].

Design		f_{clk} [GHz]	Latency [Cycles]	Latency [µs]	Area [mm^2]	Power [mW]	Energy/Op.[b] [nJ]	AT-Efficiency [(µs · mm^2)$^{-1}$]
Test Chip	Total	2	13271	6.64	0.1402	116.85	775.4	-
	Dec.[c]				0.0750	107.91	716.0	2.0080
iBM		1	13185	13.19	0.0744	56.80	748.9	1.0190
Baseline		1	16889	16.89	0.2955	248.11	4190.0	0.2004

[a] All designs were implemented using the GF 22nm FDSOI CMOS node.

[b] Energy per decoding operation.

[c] Results of the decoding module without key memory and test environment.

very low area requirement, which positively impacts the area efficiency. Even though the memory-intensive FFT polynomial evaluation approach of the baseline design might prove advantageous for FPGA implementations[6], in the derived ASIC architecture, this approach leads to a considerable larger area footprint compared to the proposed design, with a 297% area increase relative to the iBM decoding design. The impact of large memory macros in the baseline design furthermore becomes apparent for power dissipation figures, were the iBM-based architecture achieves approximately four times lower power dissipation compared to the baseline design.

It should also be mentioned, that the iBM decoding architecture as well as the baseline design only employ constant-time operations and are thus not vulnerable to timing side-channel attacks.

6.5 Measurement Results

The manufactured test chip was functionally verified and measured by utilizing the on-chip test environment. In order to verify functional correctness and to determine the maximum attainable clock frequency, test vectors were loaded alongside an expected CRC value into the key memory. By setting the frequency of the internal clock generator and observing the comparison of the CRC value of the computed plaintext with the expected CRC value, the configuration corresponding to the maximum frequency was found. Subsequently, this maximum frequency was determined by measuring a divided clock on an output pin. Figure 8a shows the maximum clock frequency that led to a correct decoding result. While the test chip was able to compute a decoding operation with the target clock frequency of 2 GHz, a higher supply voltage of at least 1.15 V (compared to the design point of 0.8 V) was required to achieve a maximum clock frequency of 2.06 GHz. We identified the cause of this mismatch as significant IR drop at the start of a decoding operation, resulting from switching from an external clock while writing the key memory to an internally generated high-speed clock for the actual decoding operation. This IR drop as well as the high-speed single-rail memory macro used for the combined evaluation modules FIFO are likely the reason for the voltage scaling behaviour shown in Figure 8a.

Power measurement of the test chip was executed slightly differently than functional verification. Instead of running a single decoding operation, the test chip was configured to continuously decode the same ciphertext stored in the key memory, in order to eliminate influences of clock switching and to allow measurements over a longer period than the relatively short decoding latency. Leakage, dynamic energy per cycle as well as total

[6] We estimate that our proposed design would also allow for an advantageous decoding latency when implemented on an FPGA, although at the cost of slightly increased logic utilization, compared to the baseline design. Since our design was optimized for an ASIC implementation, we focused on reduced SRAM utilization, while on an FPGA SRAM blocks are readily available.

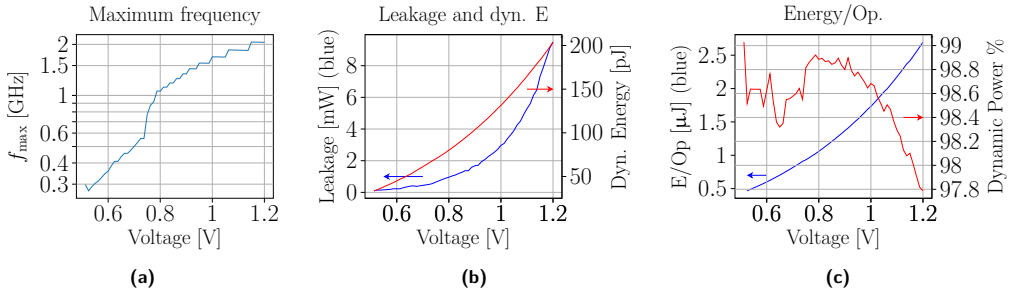

Figure 8: Measurement results of the iBM test chip: (a) Maximum clock frequency with verified correct decoding result. (b) Leakage power (blue) and dynamic energy (red) over V_{DD}. (c) Energy per decoding operation (blue) and percentage of dynamic energy (red) over V_{DD}.

energy per decoding operation and the ratio of dynamic to total energy were determined by regression and are shown in Figure 8b and Figure 8c, respectively. Due to the aforementioned IR drop at the beginning of a decoding operation, operating the decoding module at a clock frequency of $f_{\mathrm{clk}} = 2.06$ GHz with a supply voltage of $V_{\mathrm{DD}} = 1.15$ V results in a power consumption of 366 mW, of which 8.7 mW are attributed to leakage. The initial design point of $V_{\mathrm{DD}} = 0.8$ V allows for a maximum clock frequency of $f_{\mathrm{max}} = 1.06$ GHz. With these parameters, a total power consumption of 83.9 mW (including 1.3 mW leakage power) was measured for the decoding module. Even though a decoding operation exhibits an increased power dissipation compared to simulation results, the achievable area and energy efficiencies are still significantly better compared to the described state-of-the-art baseline implementation, due to a reduced utilization of SRAM macros in the proposed polynomial evaluation architecture.

7 Conclusion

The presented work aims to facilitate low latency decoding for the Classic McEliece KEM with high area efficiency. This objective is achieved by the design, implementation as well as the optimization of an ASIC architecture for Classic McEliece decoding, which targets the GF 22 nm FDSOI CMOS technology node. Furthermore, optimizations considering the memory bottleneck allow to place-and-route the proposed decoding architecture at 2 GHz. An associated decoding ASIC was manufactured and verified to achieve the high AT-efficiency suggested by simulation results.

The presented decoding ASIC architecture enables an unprecedented decoding latency, area footprint and power dissipation. Compared to previous solutions, the improved performance in this work is achieved due to the proposed novel dataflow optimizations, especially for inversionless computation of error-locator polynomials, which allows for a 1.91x speedup in terms of cycle count compared to previous state-of-the-art approaches. At the same time, the occupied area is reduced to approximately 25% of the area of previous approaches. Due to the introduced optimization techniques, the aforementioned design exhibits an area efficiency that is significantly higher than the efficiency of prior approaches. By selecting a large parameter set, the implemented design was shown to support decoding of long-term secure Classic McEliece ciphertexts. With the constant-time operations of the proposed decoding design, this architecture is hardened against timing side-channel attacks. Hence, it can be concluded that the proposed design is ideally suited for applications in high security and high performance environments.

References

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process, 2022.

[ABB+15] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Johannes Buchman, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsin, Tanja Lange, Mohamed S. E. Mohamed, Christian Rechberger, Peter Schwab, Nicolas Sendrier, Frederik Vercauteren, and Bo-Yin Yang. Initial recommendations of long-term secure post-quantum systems, 2015.

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece: conservative code-based cryptography. In *NIST Post-Quantum Cryptography Standardization Round 3 Submission*, 2020.

[Ber73] E. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, 1973.

[Ber15] Elwyn R. Berlekamp. *Algebraic Coding Theory - Revised Edition.* World Scientific Publishing Co., Inc., 2015.

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and Defending the McEliece Cryptosystem. In *Post-Quantum Cryptography*, pages 31–46, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[BSNK19] Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri. NIST Post-Quantum Cryptography- A Hardware Evaluation Study. Cryptology ePrint Archive, Report 2019/047, 2019. https://ia.cr/2019/047.

[Bur71] H. Burton. Inversionless decoding of binary BCH codes. *IEEE Transactions on Information Theory*, 17(4):464–466, 1971.

[CCD+22] Po-Jen Chen, Tung Chou, Sanjay Deshpande, Norman Lahr, Ruben Niederhagen, Jakub Szefer, and Wen Wang. Complete and Improved FPGA Implementation of Classic McEliece. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(3):71–113, Jun. 2022.

[CCKA21] Alvaro Cintas Canto, Mehran Mozaffari Kermani, and Reza Azarderakhsh. Reliable Architectures for Composite-Field-Oriented Constructions of McEliece Post-Quantum Cryptography on FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(5):999–1003, 2021.

[DInS09] Jean-Pierre Deschamps, Jose Luis Imaña, and Gustavo D. Sutter. *Hardware Implementation of Finite-Field Arithmetic.* McGraw-Hill, 2009.

[DPBM00] A.V. Dinh, R.J. Palmer, R.J. Bolton, and R. Mason. A low latency architecture for computing multiplicative inverses and divisions in $GF(2^m)$. In *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492)*, volume 1, pages 43–47 vol.1, 2000.

[GM10] Shuhong Gao and Todd Mateer. Additive Fast Fourier Transforms Over Finite Fields. *IEEE Transactions on Information Theory*, 56(12):6265–6272, 2010.

[GV14] Santosh Ghosh and Ingrid Verbauwhede. BLAKE-512-based 128-bit CCA2 secure timing attack resistant McEliece cryptoprocessor. *IEEE Transactions on Computers*, 63:1–1, 05 2014.

[HDYC18] Jingwei Hu, Wangchen Dai, Liu Yao, and Ray C.C Cheung. An application specific instruction set processor (ASIP) for the Niederreiter cryptosystem. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–6, 2018.

[Hey10] Stefan Heyse. Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers. In *Post-Quantum Cryptography*, pages 165–181, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Hey13] Stefan Heyse. *Post Quantum Cryptography: Implementing Alternative Public Key Schemes On Embedded Devices - Preparing for the Rise of Quantum Computers*. PhD thesis, Ruhr-University Bochum, 2013.

[HG12] Stefan Heyse and Tim Güneysu. Towards One Cycle per Bit Asymmetric Encryption: Code-Based Cryptography on Reconfigurable Hardware. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 340–355, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[HG13] Stefan Heyse and Tim Güneysu. Code-based cryptography on reconfigurable hardware: tweaking Niederreiter encryption for performance. *Journal of Cryptographic Engineering*, 3(1):29–43, 2013.

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.

[LC87] Mansour Loeloeian and Jean Conan. A Transform Approach to Goppa Codes. *IEEE Trans. Inf. Theor.*, 33(1):105–115, January 1987.

[LDW94] Yuan Xing Li, R.H. Deng, and Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, 1994.

[Mas89] Edoardo D. Mastrovito. VLSI designs for multiplication over finite fields $GF(2^m)$. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[MBR15] Pedro Maat C. Massolino, Paulo S. L. M. Barreto, and Wilson V. Ruggiero. Optimized and Scalable Co-Processor for McEliece with Binary Goppa Codes. *ACM Trans. Embed. Comput. Syst.*, 14(3), 2015.

[McE78] R. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, 1978.

[McE02] R. McEliece. *The Theory of Information and Coding*. Cambridge University Press, 2002.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. In *Problems of Control and Information Theory*, 1986.

[PDCS07a] Nicola Petra, Davide De Caro, and Antonio G.M. Strollo. A Novel Architecture for Galois Fields $GF(2^m)$ Multipliers Based on Mastrovito Scheme. *IEEE Transactions on Computers*, 56(11):1470–1483, 2007.

[PDCS07b] Nicola Petra, Davide De Caro, and Antonio G.M. Strollo. High Speed Galois Fields $GF(2^m)$ Multipliers. In *2007 18th European Conference on Circuit Theory and Design*, pages 468–471, 2007.

[QSTW23] Xinyuan Qiao, Suwen Song, Jing Tian, and Zhongfeng Wang. Efficient Decryption Architecture for Classic McEliece. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pages 1–7, 2023.

[SS92] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[SS01] D.V. Sarwate and N.R. Shanbhag. High-speed architectures for Reed-Solomon decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(5):641–655, 2001.

[SWM+10] Abdulhadi Shoufan, Thorsten Wink, H. Gregor Molter, Sorin A. Huss, and Eike Kohnert. A Novel Cryptoprocessor Architecture for the McEliece Public-Key Cryptosystem. *IEEE Transactions on Computers*, 59(11):1533–1546, 2010.

[WSN17] Wen Wang, Jakub Szefer, and Ruben Niederhagen. FPGA-based Key Generator for the Niederreiter Cryptosystem Using Binary Goppa Codes. In *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 253–274. Springer International Publishing, 2017.

[WSN18] Wen Wang, Jakub Szefer, and Ruben Niederhagen. FPGA-Based Niederreiter Cryptosystem Using Binary Goppa Codes. In *Post-Quantum Cryptography*, pages 77–98. Springer International Publishing, 2018.