Area-Time Efficient Hardware Architecture for Signature Based on Ed448

Mojtaba Bisheh-Niasar, Reza Azarderakhsh, and Mehran Mozaffari Kermani, *Senior Member, IEEE*

***Abstract*—In this brief, we proposed a highly-optimized FPGA-based implementation of the Ed448 digital signature algorithm. Despite significant progress in elliptic curve cryptography (ECC) implementations, Ed448 hardware architecture, to the best of our knowledge, has not been investigated in the literature. In this work, we demonstrate a high throughput while maintaining low resource architecture for Ed448 by employing a new combined algorithm for refined Karatsuba-based multiplier with precise scheduling. Furthermore, a compact distributed memory unit is developed to increase speed while keeping the area low. Our variable-base-point Ed448 architecture performs 327 signatures and 189 verifications per second at a notably higher security level of 224 bits, using not more than 6,617 Slices and 16 DSPs on a Xilinx Artix-7 FPGA. We also proposed possible countermeasures and extensions to Ed448 to counter the physical attacks.**

***Index Terms*—Elliptic curve cryptography, EdDSA, Ed448, FPGA, side-channel.**

I. Introduction

EDWARDS-CURVES Digital Signature Algorithm (EdDSA) is a secure digital signature algorithm (DSA) supporting unified addition formulas without exception. EdDSA was first proposed by Bernstein *et al.* [1], including two efficient Ed25519 and Ed448 schemes offering 127 and 224-bit security, respectively. EdDSA satisfies high-performance and constant-time requirements for implementing applications at a higher security level as a part of the TLS and OpenSSH protocols. Moreover, Safe-Curve policies are considered in designing EdDSA to address existing weaknesses in many ECDSA curves [2]. Since hybrid cryptosystems are crucial to transition to post-quantum cryptography (PQC), implementing a highly optimized EdDSA scheme is an active research area. Nevertheless, Ed448 has not got sufficient study, especially in the field of hardware implementation.

Most existing implementations of DSA bring into focus on at most 128-bit security level curves, such as NIST P256 and Ed25519. However, [3] proposed a scalable architecture in prime fields over NIST recommended field sizes up to 521-bit, employing a hardware/software co-design approach. Additionally, a multi-core FPGA scheme was presented in [4] with an arbitrary prime modulus up to 528-bit. It was applied over different curves, including NIST curves, Brainpool P512 r1, and SEC P256 k1. However, leakage exploitation using horizontal attacks was not studied in these works.

Recently, few works implement Curve448 scalar multiplication on FPGA. The authors in [5] proposed a high-performance architecture employing schoolbook multiplication. It has been extended in [6] by adding some countermeasures to protect against side-channel analysis (SCA) attacks. The work of [7] proposed a compact and fast point multiplication targeting area-constrained and time-constrained applications. Furthermore, [8] presented a LUT-based architecture employing the most significant digit multiplier. However, FPGA-based implementations of Curve448 cannot be directly compared to ours because Ed448 requires more computation units, e.g., hash core and modulo L reduction, and it takes more clock cycles. In particular, some dedicated optimizations cannot be applied to the Ed448 scheme since the architecture is reused to perform non-modular multiplication and modulo L reduction.

The work of [9] presented an implementation of AVR and MSP embedded processors for Ed448. It employed the subtractive Karatsuba approach to implementing three-level Karatsuba multiplication and the two-level Karatsuba squaring on low-end Internet-of-Thing devices.

This brief proposes, to the best of our knowledge, the first hardware implementation of the Ed448, while previous works implemented ECDSA. The EdDSA provides a high-secure scheme employing a hash function to generate a random number in a secretly deterministic way which copes with the well-known pitfall in implementing DSA and ECDSA. We propose a new approach for implementing variable-base-point Ed448 architecture employing refined Karatsuba multiplication combined with computation of the restricted-X coordinates of a point over its isogenous map to reduce the computation complexity. Our proposed pure hardware implementation presents a novel design of the Ed448 algorithm, with higher efficiency, enhanced maximum frequency, and lower area costs, which is an essential aspect of the embedded system's architecture. The previous works [3], [4] implemented a significant number of DSP and BRAM cores; this could seriously reduce the efficiency. Our work utilizes only a few DSPs, employing pipeline architecture with comparable cycle counts, advancing the hardware implementation efficiency.

The compact structure of the proposed design over other competitive designs can be used in embedded systems and Internet-of-Things (IoT) devices where energy consumption is critical as a hardware accelerator to offload computations from the microcontroller units (MCU). In particular, our objective is to create a system-on-chip (SoC) crypo-accelerator with an

Manuscript received February 22, 2021; revised March 12, 2021; accepted March 19, 2021. Date of publication March 23, 2021; date of current version July 30, 2021. This work was supported in part by NSF under Grant 1801341, and in part by NIST under Grant 60NANB16D246. This brief was recommended by Associate Editor C. W. Sham. *(Corresponding author: Mojtaba Bisheh-Niasar.)*

Mojtaba Bisheh-Niasar and Reza Azarderakhsh are with CEECS Department, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: mbishehniasa2019@fau.edu; razarderakhsh@fau.edu).

Mehran Mozaffari Kermani is with the CSE Department, University of South Florida, Tampa, FL 33620 USA (e-mail: address: mehran2@usf.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3068136.

Digital Object Identifier 10.1109/TCSII.2021.3068136

MCU such as ARM processors that achieve high area-time efficiency, rather than creating a very low area or ultra-high-performance implementations at the high cost of the other. Our implementation can also be integrated as an off-chip solution; however, other criteria, such as performance, are often as important or more important than efficiency in the external crypto-chip design, which is beyond of this work. Not only does our architecture inherently provide protection against timing and SPA attacks, but also advanced security mechanisms to avoid DPA attacks are included, which is missing in the literature. We implement the proposed architecture in a Xilinx Artix-7 FPGA and provide performance evaluations.

The rest of this brief is organized as follows: Section II reviews the preliminaries. Our proposed architecture is presented in Section III. Section IV presents the results and comparison with previous work. Finally, we conclude this brief in Section V.

II. Preliminaries

Twisted Edwards curve E is defined by $E : ax^2 + y^2 = 1 + dx^2y^2$ where $\mathcal{P} = (x, y) \in \mathbb{F}_p \times \mathbb{F}_p$. Twisted Edwards curve offers fast execution of high-security elliptic curve cryptography employing a unified addition formula between $\mathcal{P}_3 = (x_3, y_3) = (x_1, y_1) + (x_2, y_2)$ such that:

$$(x_3, y_3) = \left(\frac{x_1 \times y_2 + x_2 \times y_1}{1 + d \times x_1 \times x_2 \times y_1 \times y_2}, \frac{y_1 \times y_2 - a \times x_1 \times x_2}{1 - d \times x_1 \times x_2 \times y_1 \times y_2}\right) \quad (1)$$

EdDSA provides a high-performance digital signature algorithm, which is more resistant to side-channel attacks [10]. The detailed schedule of the EdDSA algorithm can be described into three major functions as follows:

- *KeyGen*(s): This algorithm takes a security parameter s as input and outputs a secret (signing) key sk and a public (verification) key pk with associated message space $\mathcal{M}$.
- *Sign*(sk, m): This algorithm takes a secret key sk and a message $m \in \mathcal{M}$ as input and outputs a signature (R, S).
- *Verify*(pk, m, R, S): This algorithm takes a public key pk, a message $m \in \mathcal{M}$, and a signature (R, S) as input and outputs an acceptance validation $b \in \{0, 1\}$.

For correctness, we require that for all $s \in \mathbb{F}_p$, for all $(sk, pk) \leftarrow keyGen(s)$, and for all $m \in \mathcal{M}$ it holds that:

$$Verify(pk, m, sign(sk, m)) = 1 \quad (2)$$

Ed448 is an equivalent to a Montgomery curve and an untwisted Edwards curve called Ed448-Goldilocks introduced by Hamburg [11]. The centerpiece of Ed448 is elliptic curve point multiplication (ECPM) $\mathcal{Q} = [k]\mathcal{P}$ including k times consecutive point addition over prime field $p = 2^{448} - 2^{224} - 1$. Ed448 group operation can be performed efficiently using homogeneous coordinates introduced in [12], [13].

For group arithmetic based on Ed448, the computation can be performed on projective coordinates [10]. A mapping between affine coordinates (x, y) and projective coordinates (X, Y, Z) for a point $\mathcal{P}$ is defined by $x = X/Z$ and $y = Y/Z$. Let $\mathcal{P}_1 = (X_1, Y_1, Z_1)$ and $\mathcal{P}_2 = (X_2, Y_2, Z_2)$, $\mathcal{P}_3 = \mathcal{P}_1 + \mathcal{P}_2$ can be computed using the following formula:

$$\begin{aligned} &A = Z_1 \cdot Z_2,\ B = A^2,\ C = X_1 \cdot X_2,\ D = Y_1 \cdot Y_2,\\ &E = d \cdot C \cdot D,\ F = B - E,\ G = B + E,\\ &X_3 = A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D),\\ &Y_3 = A \cdot G \cdot (D - C),\ Z_3 = F \cdot G \end{aligned} \quad (3)$$

Algorithm 1 Proposed Point Multiplication Over Ed448

Input: k, $\mathcal{P}_{Ed\_Base} = (x_{Ed\_Base}, y_{Ed\_Base})$
Require: $\mathcal{Q}_{Ed} = k \cdot \mathcal{P}_{Ed\_Base}$
Initial step: convert the base point from Edwards curve to its isogenous Montgomery curve: $\mathcal{P}_{Mont\_Base} = isogenous\_map(\mathcal{P}_{Ed\_Base})$, $k_{Mont} = k \gg 2$, $\mathcal{P}_0 = 0$, $\mathcal{P}_1 = \mathcal{P}_{Mont\_Base}$

```
1: for i from 445 downto 0 do //Montgomery Ladder
2:     if $k_{Mont_i} = 0$ then
3:         $\mathcal{P}_1 = \mathcal{P}_0 + \mathcal{P}_1$
4:         $\mathcal{P}_0 = 2 \times \mathcal{P}_0$
5:     else
6:         $\mathcal{P}_0 = \mathcal{P}_0 + \mathcal{P}_1$
7:         $\mathcal{P}_1 = 2 \times \mathcal{P}_1$
8:     end if
9: end for
10: $Y_{Mont} = y\_recovery(\mathcal{P}_0 = (X_{Mont}, Z_{Mont}))$
11: $(X_{Ed}, Y_{Ed}, Z_{Ed}) = dual\_isogenous\_map(X_{Mont}, Y_{Mont}, Z_{Mont})$
12: $(x_{Ed}, y_{Ed}) = (X_{Ed}/Z_{Ed}, Y_{Ed}/Z_{Ed})$
13: return $\mathcal{Q}_{Ed} = (x_{Ed}, y_{Ed})$
```

In [12], the authors introduced an efficient unified point addition and a dedicated point doubling formula at the cost of $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{k} + 7\mathbf{A}$ and $3\mathbf{M} + 4\mathbf{S} + 5\mathbf{A}$, where $\mathbf{M}$, $\mathbf{S}$, $\mathbf{k}$, and $\mathbf{A}$ are a multiplication, a squaring, a multiplication by a constant, and an addition cost, respectively.

A. Side-Channel Protection

Since ECDSA is vulnerable against SCA shown in [14], implementing the SCA-protected EdDSA algorithm gained more attention. Implementing a resistant scheme for Ed448 signatures can be achieved easier due to the unified addition formula. On the other hand, although ECDSA can be broken in case of reusing nonce in only a few signatures, Ed448 does not employ a unique random number for each signature.

A constant-time and secret-independent computation should be performed to protect architecture against timing and simple power analysis (SPA) attacks. Furthermore, point randomization can be applied to the projective coordinate representation of the base point to prevent differential power analysis (DPA) attacks [15]. The authors in [6] introduced point re-randomization applied to the Montgomery ladder to enhance architecture against horizontal attacks.

III. Target Architectures for Ed448

A. Proposed ECPM Algorithm

To reduce the computation complexity, we propose performing ECPM over Montgomery curve instead of the Edwards curve. Algorithm 1 describes our proposed Ed448 point multiplication, including four major steps:

- *Step 1:* The base point should be mapped from Edwards domain to Montgomery domain such that:

$$x_{Mont} = y_{Ed}^2 / x_{Ed}^2 \quad (4)$$
$$y_{Mont} = y_{Ed} \cdot (2 - x_{Ed}^2 - y_{Ed}^2) / x_{Ed}^3 \quad (5)$$

However, as the base point is constant, we assume the Montgomery base point is available without any cost.

TABLE I
IMPLEMENTATION RESULTS ON ARTIX-7 FPGA

Logic Utilization	Design Utilization Summary		
	Used	Available	Utilization
Look-Up tables	25,095	63,400	39.58%
Flip-flops	12,544	126,800	9.89%
Slices	6,617	15,850	41.75%
DSP	16	240	6.67%
Block-RAMs	0	135	0.0%
Logic Clock Frequency	123 MHz		
Total clock cycle	Keygen	Signing	Verifying
	376,095	376,120	650,123
Total Time [ms]	3.06	3.06	5.29

- *Step 2:* The efficient Montgomery ladder in projective coordinates should be performed to achieve the Montgomery domain result.
- *Step 3:* Since computation is implemented using restricted-X coordinate on the Montgomery curve, we have to recover the Y-coordinate result proposed by [16].
- *Step 4:* A map from the Montgomery domain is implemented to achieve a result in Edwards domain such that:

$$x_{Ed} = \frac{4 \cdot (x_{Mont}^2 - 1) \cdot y_{Mont}}{(x_{Mont}^2 - 1)^2 + 4 \cdot y_{Mont}^2} \tag{6}$$

$$y_{Ed} = \frac{x_{Mont} \cdot ((x_{Mont}^2 - 1)^2 - 4 \cdot y_{Mont}^2)}{2 \cdot (x_{Mont}^2 + 1) \cdot y_{Mont}^2 - x_{Mont} \cdot (x_{Mont}^2 - 1)^2} \tag{7}$$

Since the dual isogenous map has a degree of 4, the Montgomery ladder in step 2 should be performed for two fewer iterations.

B. Proposed Hardware Architecture

Although Ed448 is implemented over an extended field size of 448-bit long, it provides impressive flexibility to design an efficient architecture for different platforms. Moreover, its special prime with golden ratio 2^{224} makes it suitable in many security applications employing fast Karatsuba multiplication.

The proposed architecture consists of three stages: (i) the top stage includes FSM, controller, and ROM, (ii) the lower stage includes the field arithmetic logic unit, and (iii) the middle stage includes hash function, reduction handlers, memory unit, and secret key buffer. Redundant representation is employed in the proposed Ed448 architecture. Hence, we decompose an integer into four chunks in radix $2^{448/4} = 2^{112}$. Therefore, the data path is considered 128-bit to allow several operations before causing an overflow.

1) Modular Multiplication: Suppose A (and B) is decompused such that $A = A_1 \times 2^{224} + A_0$, $A_i = a_{2i+1} \times 2^{112} + a_{2i}$, and $a_i = a_{i1} \times 2^{64} + a_{i0}$. Employing Karatsuba multiplication for the top level results in:

$$C = A \cdot B = (A_{10} \cdot B_{10} - A_0 B_0) 2^{224} + (A_1 B_1 + A_0 B_0) \tag{8}$$

where $A_{10} = (A_1 + A_0)$ and $B_{10} = (B_1 + B_0)$.

The authors in [17] introduced the refined Karatsuba identity formula to decrease the number of required addition. Applying refined Karatsuba identity in middle-level can decompose A_0B_0, A_1B_1, and $A_{10}B_{10}$ to reduce a 225×225-bit multiplication to three 114×114-bit multiplications. For example, A_0B_0 is decomposed such that:

$$A_0 B_0 = (1 - 2^{112}) \cdot (a_0 b_0 - 2^{112} a_1 b_1) + 2^{112}(a_{10} b_{10}) \tag{9}$$

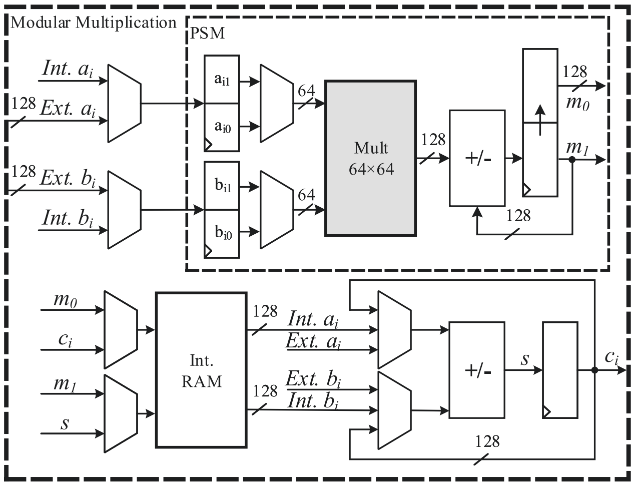

Fig. 1. Modular multiplier in the proposed efficient Ed448 scheme. $Ext.a_i$ and $Ext.b_i$ are read from, and c_i are stored to the main memory unit. The Int. RAM module is a dedicated RAM only for multiplier.

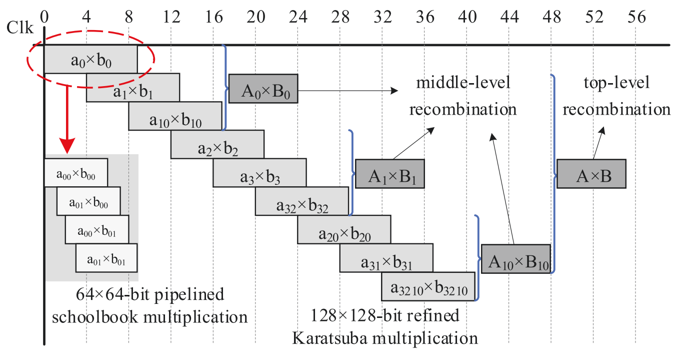

Fig. 2. Timing diagram of the proposed scheduling scheme for $A \times B$, where A (and B) is decomposed such that $A = A_1 \times 2^{224} + A_0$, $A_i = a_{2i+1} \times 2^{112} + a_{2i}$, and $a_i = a_{i1} \times 2^{64} + a_{i0}$.

We implement a pipelined schoolbook multiplier (PSM) illustrated in Fig. 1. To control the execution sequence efficiently, we design a specific controller and an internal memory for multiplier using a dedicated ROM and RAM. In our proposed architecture, the partial multiplications are computed with PSM by selecting the operands from the input registers with two multiplexers. Results of the partial multiplications are accumulated into a 256-bit register. Triggering the enable command starts the multiplier to read from the external memory and write the intermediate data to its internal memory. Eventually, applying the last reduction stage prepares the product to store in the external memory.

We also design a precise scheduling to increase efficiency presented in Fig. 2. Since PSM requires 4 cycles to read the input registers, the next decomposed part in middle-level is started each 4-cycle. The i^{th} middle-level recombination can be performed after $i \times 3 \times 4 + 5$ clock cycles. After performing a_0b_0, a_1b_1, and $a_{10}b_{10}$, only 5 additions are needed in middle-level recombination to compute A_0B_0 which are pipelined with A_1B_1 computing. For example, in order to compute $a_0b_0 - 2^{112}a_1b_1$, only the second digit of a_0b_0 is subtracted by the first digit of a_1b_1. The second digit of a_1b_1 should be subtracted from 0, which can be neglected in this step. Therefore, at the end of this step, the result is represented in three digits. Then, we shift $a_0b_0 - 2^{112}a_1b_1$ and subtract from itself to compute $(1 - 2^{112})(a_0b_0 - 2^{112}a_1b_1)$. Thus, two subtractions are executed which results in four digits. To compute $(1 - 2^{112})(a_0b_0 - 2^{112}a_1b_1) + 2^{112}(a_{10}b_{10})$, two additions are performed between the second and third digits of the previous result and the first and second digits of $a_{10}b_{10}$. Eventually, four digits are achieved to represent A_0B_0 with performing only five additions. The same procedure is considered to produce A_1B_1 and $A_{10}B_{10}$.

We also employ the interleaved reduction technique into the top-level recombination. In this approach, the $2^{224} \times (A_{10}B_{10} - A_0B_0)$ is reduced before adding with $(A_0B_0 + A_1B_1)$. In the top level, the two most significant digits of A_0B_0 cancel themselves considering the reduction algorithm.

2) Inversion: FLT-based inversion is used in the last stage of ECPM to convert back to affine from projective coordinates.

3) Hash Unit: Generating a random number in Ed448 is considered by employing a highly secure hash function rather than entrusting it to an implementer, such as in ECDSA. Although SHAKE256 recommended by [10] is particularly defined not to be a hash function, it can be employed considering (i) a fixed-output size of 114-byte long, and (ii) its sufficient 256-bit security level against collisions and preimages. In our design, SHAKE256, based on Keccak$[r = 1088, c = 512]$, takes an arbitrary input in 1088-bit chunks and provides 912-bit output. The SHA-3 was designed to be fast in hardware. Hence, its latency is significantly smaller than the ECPM.

4) Mod L Reduction and Non-Modular Multiplication: A 912-bit scalar taken from SHAKE256 should be reduced by modulus 446-bit value of L. We propose a constant-time execution to implement this reduction. Let x be $x = x_1 2^{456} + x_0$, where x_1 and x_0 have 456-bit in four 114-bit chunks. The group order can be presented by $L = 2^{446} - l_0$, where l_0 has 224-bit. Hence, the first round of reduction is performed as follows:

$$x \bmod L \equiv x_1 2^{10} \times (L + l_0) + x_0 \equiv x_1 \cdot l_0 2^{10} + x_0 \quad (10)$$

The product of 456×224-bit non-modular multiplication is shifted by 10 bits and added with x_0. The output is a 691-bit long value shown by x'. For the next round, let $x' = x'_1 2^{446} + x'_0$, hence, x'_1 and x'_0 have 245 and 446-bit, respectively. The second and third round of the reduction can be performed to achieve a 446-bit long result as follows:

$$x' \bmod L \equiv x'_1 2^{446} + x'_0 \equiv x'_1 \times (L + l_0) + x'_0 \equiv x'_1 \cdot l_0 + x'_0 \quad (11)$$

To implement non-modular multiplication, we reuse our modular multiplication architecture while the modulo p is excluded. This technique keeps the utilized area low.

5) Double-Point Multiplication: Verifying algorithm requires two ECPMs, while double-point multiplication can be used to reduce two scalar multiplications to only one and results in improving efficiency significantly. We employ a modified version of Strauss' trick presented in [21]. Since both scalars in the verifying operation are not secret, SCA countermeasures are not required for the verifying.

6) SCA Countermeasures: As we implement constant-time and secret-independent ECPM algorithms, e.g., Montgomery ladder and FLT, our design is resistant against timing and SPA attacks. To enhance our architecture against DPA attacks, we randomize the base point in projective coordinates as follows:

$$\mathcal{P}^{randomized}_{Mont\_Base} = (\lambda \cdot X_{Mont\_Base}, \lambda \cdot Y_{Mont\_Base}, \lambda) \quad (12)$$

This technique provides basic protection against information leakage through a DPA attack. Designing a random number generator is not in the scope of this study, so we assume $\mathcal{P}^{randomized}_{Mont\_Base}$ are provided externally to be used in our variable-base-point architecture. Point re-randomization against horizontal attacks requires two additional multiplications in each step of the Montgomery ladder, which results in extending algorithm latency.

IV. Experimental Results and Comparison

We implement the proposed architecture on a Xilinx Artix-7 (XC7A100TFGG484-3) using Xilinx Vivado 2018.2. The implementation results for the protected scheme are presented in Table I. Our design occupied 25k LUTs, 12.5k FFs, and 16 DSPs, while the required memory is implemented using the distributed memory. The maximum frequency is 123 MHz; hence, our efficient Ed448 scheme can generate 327 keys per second. Moreover, it signs and verifies a 1088-bit message in 3.06 ms (327 OP/s) and 5.29 ms (189 OP/s), respectively.

With a variety of performance optimizations in hardware implementations, the number of total clock cycles is reduced notably compared with AVR implementation as well as MSP results. Hence, our design achieves almost 1,000 times speedup compared to [9].

This brief proposes the first FPGA-based EdDSA architecture over Ed448. Moreover, complete signature and verification implementations with certificate handling are scarce. Hence, a direct comparison of the area utilization and performance is difficult since the implementations target different schemes and security levels, and they use different platforms and technologies. Nevertheless, we compare our design with previous DSA hardware designs with security levels 81-bit to 260-bit. The comparison results are reported in Table II, while the area-time product is used to evaluate efficiency. As far as different designs have different security levels, we extrapolate their performance, assuming time complexity is growing cubic in the field size [5]. To have a fair comparison, we evaluate the performance of the proposed design on the state-of-the-art targeted platforms, which changes performance by a factor of 1.34×, 0.79×, and 0.62× on Kintex-7, Virtex-5, and Virtex-6 compared to Artix-7.

Compared with [3], our design reduces the required resources and total time using refined Karatsuba multiplication in a pipelined architecture. Thus, we improve efficiency by 87% and 92% compared with [3] at security level 192 and 260-bit, respectively. Furthermore, compared with [4], which improves performance through homogeneous Co-Z coordinate representation, our proposed architecture achieves faster speed due to computing over restricted-X coordinates of a point on the Montgomery curve. Our architecture achieves 74% and 53% improvement compared to [4] at security level 192 and 260-bit, respectively. As one can see, implementing a highly-protected scheme reduces 15% efficiency due to adding two additional modular multiplications at the end of each Montgomery ladder iteration. The proposed design requires 21.4% less energy for generating a signature considering the cubic time complexity compared to the best previous work.

The work of [6] implemented a Curve448 ECPM based on [5] results. While a direct comparison is not possible, this architecture can operate at 2.8× higher frequency since Curve448 requires smaller circuit compared to Ed448.

TABLE II
COMPARISON OF DIFFERENT DESIGNS FOR DIGITAL SIGNATURE ALGORITHM

Work	Curve	Device	Freq	Area				Keygen		Signing		Verifying		Power/Energy	$A \times T$[b]	Normalized
			[MHz]	Slices	DSP	BRAM	Est. Area[a]	[kCCs]	[ms]	[kCCs]	[ms]	[kCCs]	[ms]	[mW]/[mJ]		$A \times T$[c]
[9][p]	*Ed448*	AVR	32	-	-	-	-	103,229	3,225.9	-	-	-	-	-	-	-
[9][p]	*Ed448*	MSP	25	-	-	-	-	73,478	2,939.1	-	-	-	-	-	-	-
[18][u]	*NIST B-163*	Virtex-5	149	20,628	-	-	20,628	-	-	-	1,901	-	1,943	228/433.4	39,213.8	829,331.6 (65)
[19][u]	*NIST B-163*	Virtex-5	107	18,504	-	-	18,504	-	-	84	0.78	83	0.72	105.7/0.08	14.4	304.5 (65)
[3][u]	*NIST P-384*	Virtex-6	83	16,747	196	NA	36,347	-	-	284	3.42	301	3.62	-	124.3	197.4 (40)
[4][p]	*NIST P-384*	Kintex-7	32	8,759[†]	0	60	8,759	234	6.94	-	-	-	-	-	60.8	96.5
[3][u]	*NIST P-521*	Virtex-6	71	23,633	280	NA	51,633	-	-	651	9.17	855	12.05	-	473.5	302.8 (40)
[4][p]	*Brainpool P512 r1*	Kintex-7	32	8,759[†]	0	60	8,759	311	9.62[‡]	-	-	-	-	-	84.3	53.9
This work[p]	*Ed448*	Artix-7	123	6,617	16	0	8,217	376	3.06	376	3.06	650	5.29	436/1.33	25.1	25.1
This work[hp]	*Ed448*	Artix-7	123	6,713	16	0	8,313	439	3.57	438	3.56	650	5.29	441/1.57	29.6	29.6

[u]Unprotected. [p]Protected by SPA countermeasures. [hp]Highly protected by DPA countermeasures
[†]Slices are estimated as KLUT/4.
[‡]These values are only for the core operations scalar multiplication, without all pre- and post-processing and hashing.
[a]To estimate the area equivalent, we converted 1 Xilinx DSP48 to 100 Slices following earlier work [20].
[b]Area-time product ($A \times T$) is the product of time (keygen or signing) and estimated resources, which has a unit of Slice·s.
[c]Normalized area-time product in 224-bit security level is the quotient of a design area-time product in the security level of l-bit by dividing $l^3/224^3$.

However, this design needs almost 25% more cycles and heavily relies on FPGA primitives occupying 14 BRAMs and 2× more DSPs compared to ours. The work of [7] utilized 9 BRAMs, while needs 37% more cycles. However, the three functions, i.e., keygen, signing, and verifying, are all realized on a single hardware component in our design.

V. CONCLUSION

In this brief, we proposed a method for FPGA-based implementation of the digital signature algorithm on Ed448 using optimized refined Karatsuba multiplication over its isogenous map and high-throughput pipelined architecture. Implementation results show that our architecture can significantly improve the efficiency in terms of area-time product, which made it practical for resource constraint applications in higher-level security requirements.

The trick of working on the isogenous twisted Edwards curve introduced in [22] by Hamburg can reduce the cost of point addition. A combination of this trick with a window-based ladder can be employed to design a high-performance Ed448 scheme. We keep this as future work.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their comments. Also, they thank Mike Hamburg for his constructive comments.

REFERENCES

[1] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, “High-speed high-security signatures,” in *Proc. 13th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Nara, Japan, 2011, pp. 124–142.
[2] D. J. Bernstein and T. Lange. (May 2011). *Security Dangers of the NIST Curves*. [Online]. Available: https://www.hyperelliptic.org/tanja/vortraege/20130531.pdf
[3] B. Panjwani, “Scalable and parameterized hardware implementation of elliptic curve digital signature algorithm over prime fields,” in *Proc. Int. Conf. Adv. Comput. Commun. Informat. (ICACCI)*, Udupi, India, 2017, pp. 211–218.
[4] B.-Y. Peng, Y.-C. Hsu, Y.-J. Chen, D.-C. Chueh, C.-M. Cheng, and B.-Y. Yang, “Multi-core FPGA implementation of ECC with homogeneous Co-Z coordinate representation,” in *Proc. 15th Int. Conf. Cryptol. Netw. Security (CANS)*, Milan, Italy, 2016, pp. 637–647.
[5] P. Sasdrich and T. Güneysu, “Cryptography for next generation TLS: Implementing the RFC 7748 elliptic Curve448 cryptosystem in hardware,” in *Proc. 54th Annu. Design Autom. Conf. (DAC)*, Austin, TX, USA, 2017, pp. 1–6.
[6] P. Sasdrich and T. Güneysu, “Exploring RFC 7748 for hardware implementation: Curve25519 and Curve448 with side-channel protection,” *J. Hardw. Syst. Security*, vol. 2, no. 4, pp. 297–313, 2018.
[7] M. Bisheh Niasar, R. Azarderakhsh, and M. Mozaffari Kermani, “Efficient hardware implementations for elliptic curve cryptography over Curve448,” in *Proc. 21st Int. Conf. Cryptol. India (Indocrypt)*, Dec. 2020, pp. 228–247.
[8] Y. A. Shah, K. Javeed, M. I. Shehzad, and S. Azmat, “LUT-based high-speed point multiplier for Goldilocks-Curve448,” *IET Comput. Digit. Techn.*, vol. 14, no. 4, pp. 149–157, Jul. 2020.
[9] H. Seo, “Compact implementations of curve Ed448 on low-end IoT platforms,” *ETRI J.*, vol. 41, no. 6, pp. 863–872, 2019.
[10] S. Josefsson and I. Liusvaara, “Edwards-curve digital signature algorithm (EdDSA),” IETF, RFC 8032, 2017.
[11] M. Hamburg, “Ed448-goldilocks, a new elliptic curve,” IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2015/625, 2015.
[12] D. J. Bernstein and T. Lange, “Faster addition and doubling on elliptic curves,” IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2007/286, 2007.
[13] H. Hisil, K. K.-H. Wong, G. Carter, and E. Dawson, “Twisted edwards curves revisited,” IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2008/522, 2008.
[14] K. Ryan, “Return of the hidden number problem. A widespread and novel key extraction attack on ECDSA and DSA,” *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 1, pp. 146–168, 2019.
[15] J.-S. Coron, “Resistance against differential power analysis for elliptic curve cryptosystems,” in *Cryptographic Hardware and Embedded Systems (CHES)*, Ç. K. Koç and C. Paar, Eds. Heidelberg, Germany: Springer, 1999, pp. 292–302.
[16] K. Okeya and K. Sakurai, “Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve,” in *Proc. 3rd Int Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Paris, France, May 2001, pp. 126–141.
[17] D. J. Bernstein, “Batch binary Edwards,” in *Proc. 29th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 2009, pp. 317–336.
[18] E. Wajih, B. Noura, M. Mohsen, and T. Rached, “Low power elliptic curve digital signature design for constrained devices,” *Int. J. Security*, vol. 6, no. 2, pp. 1–14, 2012.
[19] A. Sghaier, M. Zeghid, C. Massoud, and M. Mahchout, “Design and implementation of low area/power elliptic curve digital signature hardware core,” *Electronics*, vol. 6, no. 2, p. 46, 2017.
[20] M. Bisheh Niasar, R. El Khatib, R. Azarderakhsh, and M. Mozaffari-Kermani, “Fast, small, and area-time efficient architectures for key-exchange on Curve25519,” in *Proc. 27th IEEE Symp. Comput. Arithmetic (ARITH)*, Portland, OR, USA, Jun. 2020, pp. 72–79.
[21] P. Schwabe. (Sep. 2013). *Scalar-Multiplication Algorithms*. [Online]. Available: https://cryptojedi.org/peter/data/eccss-20130911b.pdf
[22] M. Hamburg, “Twisting Edwards curves with isogenies,” IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2014/027, 2014.