

Article

High-Performance SM2 Signature Hardware Architecture Based on Precomputation and Parallel Scheduling

Jie Huang ^{1,2,*}, Ming-Fu Zhong ¹  and Zuo-Nan Xiao ¹

¹ School of Cyber Science and Engineering Security, Southeast University, Nanjing 211189, China; 230219460@seu.edu.cn (M.-F.Z.); mfzh023@163.com (Z.-N.X.)

² Purple Mountain Laboratories, Nanjing 211111, China

* Correspondence: jhuang@seu.edu.cn

Abstract

In high-throughput, high-concurrency, low-latency cryptographic scenarios, the throughput of public-key cryptography is a decisive performance metric. As China's national elliptic curve cryptography standard, the SM2 signature algorithm has been widely adopted, yet scalar multiplication—the core primitive of SM2 signature—constitutes the dominant latency bottleneck. This paper proposes a high-throughput ASIC architecture that integrates precomputation with parallel task scheduling to accelerate SM2 signature generation. First, we design a Comb-algorithm-based precomputation hardware scheme for fixed-base scalar multiplication. The 256-bit scalar is partitioned into 32 segments, and 32 dedicated lookup tables are precomputed in on-chip SRAMs, which reduces fixed-base scalar multiplication to at most 31 elliptic curve point additions. Second, a six-arithmetic-unit parallel scheduling framework is developed for variable-base scalar multiplication. Equipped with two three-stage pipelined Montgomery multipliers and four modular adders, the design overlaps point addition and point doubling across pipeline stages to boost hardware resource utilization. Moreover, we build a 16-core parallel computing platform integrated with hardware task queues and DMA automatic scheduling, achieving efficient throughput scalability with the increase in core count. The proposed design has been taped out in the TSMC 28 nm CMOS process with completed physical design, including placement, clock tree synthesis, and routing. Post-layout simulation results, with full parasitic extraction (RC) and static timing analysis (STA), demonstrate that a single core achieves 89,593 signatures per second at a post-layout maximum frequency of 600 MHz. Its normalized area efficiency, measured as signatures per kilo gate equivalent (KGE), reaches 33.31 sig/s/KGE under the TSMC 28 nm process at 600 MHz. It should be noted that this metric is significantly influenced by the advanced process node and higher operating frequency; therefore, to enable a fairer assessment of intrinsic microarchitectural efficiency independent of process scaling, the frequency-normalized metric (sig/s/MHz/KGE) is adopted as the primary cross-design benchmark. Under this metric, our design achieves 55.5×10^{-3} sig/s/MHz/KGE, which is comparable to the 55.0×10^{-3} sig/s/MHz/KGE of the most area-efficient referenced design, with a marginal improvement of approximately 0.9%. The area efficiency comparison is presented only as a supplementary reference within a limited and clearly defined scope, acknowledging that the compared designs differ in functionality, process technology, and evaluation methodology. Furthermore, our design achieves the lowest Area–Time (AT) product of 30.03 KGE·ms among the compared works, indicating that our architectural innovation achieves a favorable AT trade-off for high-throughput applications rather than a fundamental shift in circuit efficiency. The 16-core parallel computing platform reaches an overall throughput of 1.03 million signatures per second. In addition, first-order arithmetic masking and key blinding are



Academic Editors: Paris Kitsos and John A. Kalomiros

Received: 1 July 2026

Revised: 10 August 2026

Accepted: 25 August 2026

Published: 27 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

integrated into the scalar-multiplication data path to resist first-order side-channel attacks, and simulation-based TVLA evaluation indicates its leakage suppression capability under simulated conditions.

Keywords: SM2 signature; scalar multiplication; side-channel resistance; multi-core parallel architecture; ASIC hardware acceleration

1. Introduction

1.1. Application Background and Practical Bottlenecks

In modern cryptographic application scenarios—such as national cryptographic infrastructures, cloud-based hardware security modules (HSMs), high-performance blockchain nodes, and financial transaction authentication systems—public-key cryptosystems are required to deliver extremely high throughput and low latency while supporting high concurrency. In some special application scenarios, public-key cryptosystems may be required to deliver extremely high throughput, massive concurrency, and strict low-latency constraints; however, single-core FPGA or ASIC public-key accelerators cannot satisfy such throughput demands. Meanwhile, traditional multi-core designs without dedicated hardware scheduling suffer from performance degradation caused by bus arbitration conflicts and uneven task allocation.

Since Miller [1] and Koblitz [2] independently proposed Elliptic Curve Cryptography (ECC) in 1986 and 1987, it has gradually superseded traditional RSA [3] due to its superior security-to-key-length ratio. As the national ECC standard of China, the SM2 algorithm [4] has been pervasively deployed in e-government, e-commerce, mobile payment, and Internet of Things (IoT) security. Nevertheless, elliptic curve scalar multiplication—the core operation underlying both ECC and SM2—remains the primary performance bottleneck in the entire signature generation process.

1.2. Related Work

Existing research on hardware acceleration of the SM2 signature algorithm can be systematically categorized into instruction-set coprocessors, pure hardware scalar-multiplication accelerators, GPU-based parallel implementations, and batch signature/verification optimizations. To clearly articulate the characteristics and limitations of each category, we conduct a targeted literature review and summarize their inherent drawbacks as follows:

Category 1: Instruction-set coprocessors. Wang et al. designed dedicated instruction hardware for SM2 on Xilinx ZYNQ-7 FPGA, consuming 7146 slices and requiring 2.25 ms for one scalar multiplication [5]. Satoh et al. optimized instruction decoding pipelines to reduce single-operation clock overhead; however, redundant fixed-base point operations in their design limit overall throughput [6]. Chen et al. proposed a high-performance elliptic curve cryptographic processor for general curves over GF(p) based on a systolic arithmetic unit, demonstrating efficient modular multiplication and point operation scheduling [7]. Although instruction-based designs simplify software–hardware interaction, nearly all implementations ignore the optimization potential of fixed-base scalar multiplication, lacking dedicated precomputation storage circuits.

Category 2: Pure hardware scalar-multiplication accelerators. Ding et al. implemented parallel SM2 on K-233 binary elliptic curves, achieving 35,880 signatures per second on Xilinx V6 FPGA; however, the binary curve cannot be directly applied to the standard prime-field SM2 curve specified by GM/T 32918 [8]. Chen et al. presented a dual-field elliptic

curve cryptographic processor featuring a radix-4 unified division unit, fabricated as an ASIC implementation [9]. While it supports both binary and prime fields, its area overhead from the unified division unit limits its applicability to resource-constrained SM2-only accelerators. Guneyasu et al. implemented ultra-high-performance ECC for NIST standard prime fields on commercial FPGAs [10]. This design achieves excellent performance for international ECC standards, but it is not compatible with the prime-field parameters of the domestic SM2 algorithm. Li et al. optimized the point multiplication logic of SM2 and implemented a secure and efficient point multiplication module for both single and batch tasks [11]. Kong completed a full FPGA implementation of a high-performance SM2 coprocessor with pure hardware logic [12]. The scheme optimizes the basic arithmetic units of scalar multiplication, but it does not adopt any precomputation strategy for frequent fixed-base operations. Xu et al. designed a high-speed and secure SM2 processor by using a modified Karatsuba multiplier [13]. This work optimizes the multiplication unit at the hardware level to raise the operating frequency, yet it still lacks dedicated precomputation circuits and overall parallel scheduling for scalar multiplication.

Category 3: GPU-based and batch optimization schemes. Yan et al. [14] and Huang et al. [15] adopted GPU-based computing to accelerate SM2 signatures. Yan et al. reached approximately 49,000 signatures per second, but only on non-standard elliptic curves [14]. GPUs are general-purpose computing devices with high power consumption and unstable real-time latency, which do not match the dedicated hardware requirements of cryptographic chips. Li et al. [16] optimized the SM2 algorithm with hybrid encryption technology. Ruan et al. [17] improved task-level efficiency via hybrid encryption and batch processing; however, these upper-layer optimizations do not reshape the underlying scalar-multiplication data path and are thus incomparable with hardware architecture innovations. We summarize the classification and comparison in Table 1.

Table 1. Classification and comparison of existing SM2 hardware acceleration approaches.

Reference	Representative Works	Key Techniques	Advantages	Limitations
Instruction-set Coprocessors	[5–7]	Custom instruction extensions, pipelined decoding	Simple SW-HW interaction, easy integration	Underutilized fixed-base precomputation, limited throughput
Pure Hardware Scalar Accelerators	[8–13]	Multi-channel parallelism, Karatsuba multiplier, pure-logic scheduling	High per-module performance, controllable resources	Lack of precomputation and global scheduling, poor area efficiency
GPU-based Implementations	[14,15]	CUDA/OpenCL threads, batch task mapping	High absolute throughput	Excessive power, unstable latency, non-dedicated architecture
Batch & Upper-layer Optimizations	[16,17]	Task-level batching, hybrid encryption scheduling	Flexible task management	Does not optimize the core scalar multiplier circuit

1.3. Motivation and Research Gaps

Among existing FPGA implementations, the scheme in [8] delivers the highest throughput, but it does not comply with the prime-field SM2 standard and consumes 768 DSP units. GPU-based designs in [14] achieve around 49,000 signatures per second only on non-standard elliptic curves, accompanied by high power consumption and unstable latency. Most existing ASIC implementations reported in the literature achieve a throughput below 2000 signatures per second. This limitation primarily stems from the lack of dedicated precomputation modules and fully serial operation scheduling for scalar multiplication.

In this paper, we select instruction-set coprocessors and pure hardware scalar multiplication accelerators as comparative references. GPU-based accelerators and batch signature optimization schemes are excluded from direct comparison. GPUs are general-purpose parallel computing devices originally developed for graphics and massively parallel workloads; their architectural hierarchy, optimization objectives, and design methodologies are fundamentally different from those of this work. This work focuses on microarchitectural innovation and parallel scheduling optimization of the ASIC-based scalar multiplication data path, which represents a different technological dimension that makes fair performance comparison inappropriate. Batch optimization only improves efficiency at the upper task-scheduling layer and cannot reshape the underlying scalar-multiplication data path, placing it at a different optimization level from our hardware-centric approach. Consequently, neither of these two categories is suitable for horizontal performance comparison in the context of hardware architecture evaluation. FPGA-based implementations are also excluded from direct comparison with our ASIC design. Although FPGA and ASIC are both hardware platforms, they differ fundamentally in design methodology, optimization objectives, and performance evaluation. FPGA designs typically prioritize flexibility, rapid prototyping, and reconfigurability, often operating at significantly lower frequencies and with different area–performance trade-offs compared to custom ASIC implementations. Moreover, the FPGA LUT and DSP-based arithmetic units introduce architectural overhead and routing constraints that do not directly translate to ASIC implementations. While FPGA results provide useful reference points for functional validation, direct quantitative comparisons—especially in terms of area efficiency and normalized throughput—are inherently unfair and potentially misleading due to the fundamentally different implementation paradigms. Therefore, we restrict our primary comparison to ASIC implementations that share similar design targets. However, it must be acknowledged that even among ASIC implementations, significant heterogeneity exists in terms of process technology spanning from 0.13 μm to 28 nm, functional scope covering full cryptographic processors and dedicated accelerators, security countermeasures, and evaluation methodologies including logic synthesis and post-layout simulation. These differences inherently limit the conclusiveness of any cross-design performance comparison. Consequently, all performance comparisons presented in this paper should be interpreted as reference points within a defined comparison scope, rather than as definitive demonstrations of architectural merit. We adopt the AT product as the holistic metric for cross-design evaluation, since this indicator inherently accounts for area–delay trade-offs across different technology nodes and architectural choices.

Based on the above analysis, three clear research gaps remain unaddressed by prior work:

First, few prior designs deploy a dedicated hardware precomputation architecture for the fixed-base point G of the prime-field SM2 standard. Most implementations treat scalar multiplication as a general-purpose operation and fail to exploit the fixed-base nature. Even the rare works that mention the Comb algorithm merely conduct functional simulations, rather than implementing a complete 32×256 -point lookup table with pipelined point-addition circuits on ASIC.

Second, existing designs fail to co-optimize variable-base multiplication together with the precomputation module. Although precomputation accelerates fixed-base operations, variable-base operations subsequently become the new performance bottleneck in the complete SM2 signature flow.

Third, current multi-core designs lack comprehensive hardware task queues and DMA-based automatic scheduling mechanisms. Most multi-core schemes simply replicate

computing cores without addressing bus contention and task distribution, resulting in simulated speedups far below ideal linear scaling.

1.4. Core Contributions of This Work

To tackle the above limitations, this paper proposes a high-performance SM2 signature hardware architecture that synergistically combines precomputation and parallel scheduling. This paper makes four core contributions as follows:

- (1) We propose a Comb-algorithm-based hardware precomputation scheme for fixed-base scalar multiplication. The 256-bit scalar is split into thirty-two 8-bit segments, and 32 lookup tables are deployed to convert fixed-base scalar multiplication into at most 31 point additions.
- (2) A unified parallel scheduling scheme with six arithmetic units for both fixed-base and variable-base scalar multiplication: Utilizing two three-stage pipelined multipliers and four adders, we overlap point-addition and point-doubling operations across different pipeline stages by sequentially scanning the binary bits of the scalar, achieving near-ideal hardware utilization. Although the SM2 signature generation primarily involves fixed-base scalar multiplication, the proposed scheduling framework is designed as a unified architecture that also supports variable-base operations required for SM2 signature verification and key exchange.
- (3) A 16-core parallel computing platform: Integrating task queues, DMA controllers, and multi-level caching, the platform supports up to 16 public-key coprocessor cores with hardware auto-scheduling, enabling efficient throughput scalability with increasing core count.
- (4) A high-performance ASIC implementation: Taped out in the TSMC 28 nm CMOS technology, a single module achieves 89,593 signatures per second at 600 MHz. Within the scope of the five compared ASIC designs and under the stated comparison assumptions, its normalized area efficiency reaches 33.31 sig/s/KGE, which is higher than the corresponding figures reported in the referenced designs. However, it is important to note that this advantage is largely attributed to the advanced 28 nm process technology employed in this work, whereas the referenced designs adopt older 0.13 μm , 90 nm, and 55 nm nodes. When frequency-normalized (sig/s/MHz/KGE), our design shows only a marginal 0.9% advantage over the best-performing referenced design [11], indicating that the primary contribution of this work lies in achieving competitive microarchitectural efficiency while leveraging advanced process technology for higher absolute throughput. When deployed on the proposed parallel computing platform, the overall throughput reaches 1.03 million signatures per second.

Implementation Status Statement: This design has completed RTL design, synthesis, placement-and-routing, and post-layout simulation with parasitic extraction in TSMC 28 nm CMOS, but has not yet undergone physical measurement after tape-out. All performance, throughput, and power data reported are derived from post-layout gate-level simulations and STA, representing pre-silicon performance estimations rather than actual silicon measurements. Final performance is subject to post-silicon validation.

2. Elliptic Curve Public Key Cryptography Algorithm

2.1. SM2 Algorithm

The SM2 algorithm is a standard published by the National Cryptography Administration of China as the GM/T 32918-2016 SM2 public key cryptography algorithm. It is recommended to use prime-field 256-bit elliptic curve parameters as follows: elliptic curve equation:

$$y^2 = x^3 + ax + b$$

where p is a 256-bit prime, a and b are curve parameters, and $G = (x_G, y_G)$ is the base point of order n . The SM2 parameters are specified as follows:

$$p = 0 \times \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFF}$$

$$a = 0 \times \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC}$$

$$b = 0 \times \text{28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93}$$

$$n = 0 \times \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123}$$

Assuming the message to be signed is M , user A has a distinguishable identifier IDA with a length of $entlen_A$ bits. $ENTLA$ is two bytes converted from the integer $entlen_A$, d_A is A's private key, and both the signer and verifier need to use cryptographic hash functions to obtain the hash value Z_A of user A. The coordinates x_G and y_G , the elliptic curve parameters a , b , and the public key coordinates x_A and y_A are converted into bit strings. The H_{256} hash algorithm used in the SM2 signature algorithm is the SM3 hash cipher algorithm [18].

2.2. Scalar Multiplication Algorithm

The performance of the elliptic curve encryption algorithm depends on the scalar multiplication operation in elliptic curve group operations, which can be divided into fixed-base scalar multiplication operations and non-fixed-base scalar multiplication operations. The fast algorithms for scalar multiplication mainly include binary algorithm, NAF algorithm, sliding window algorithm, and comb algorithm [19].

In SM2 signature computation, the Comb algorithm's precomputation strategy is adopted for fixed-base scalar multiplication [20]. The specific hardware design is as follows:

- (1) Split the 256-bit scalar k into 32 segments, denoted as $k_{255}k_{254} \cdots k_{248}k_{247}k_{246} \cdots k_{240} \cdots k_7k_6 \cdots k_0$, where each segment is 8 bits, denoted as $l_{31}l_{30} \cdots l_0$.
- (2) The scalar multiplication kG can be decomposed as follows:

$$kG = l_{31}G_{31} + l_{30}G_{30} + \cdots + l_0G_0$$

where $G_0 = G$, $G_1 = 2^8G_0$, $G_2 = 2^8G_1 \dots$, $G_{31} = 2^8G_{30}$.

- (3) For each 8-bit segment l_i , where i ranges from 0 to 31, compute all possible scalar multiplication values m_iG_i for $m_i \in [0, 255]$, and construct 32 precomputation tables D_i , each containing 256 points, denoted as $D_{31}, D_{30}, \dots, D_0$. $D_0 = m_0G_0$, $D_1 = m_1G_1 \dots$, $D_{31} = m_{31}G_{31}$. Each precomputation table stores the results of scalar multiplication in sequential order.
- (4) Input these 32 tables (each containing 256 points) into the precomputation storage SRAM sequentially. The scalar multiplication can be converted into a table lookup that only requires 31 point additions.
- (5) For scalar $k = l_{31}l_{30} \cdots l_0$, according to the value of l_0 , the hardware module performs a table lookup in Table D_0 to obtain point P_0 .
- (6) According to the value of l_1 , the hardware module performs a table lookup in Table D_1 to obtain point P_1 .
- (7) The hardware module performs the point addition of the two points P_0 and P_1 on the elliptic curve; $P = P_0 + P_1$.
- (8) According to the value of l_2 , the hardware module performs a table lookup in Table D_2 to obtain point P_2 .
- (9) The hardware module performs the point addition of the two points P and P_2 on the elliptic curve; $P = P + P_2$.
- (10) Following this process, the result of $kG = P + P_{31}$ can be obtained.

3. Hardware Design of Public Key Coprocessor

The algorithm module of the public key coprocessor mainly includes five sub-modules, namely the computing unit module, the multiplier module, the adder module, the storage module, and the control module. The overall hardware design framework for the scalar multiplication operation is shown in the following figure (Figure 1):

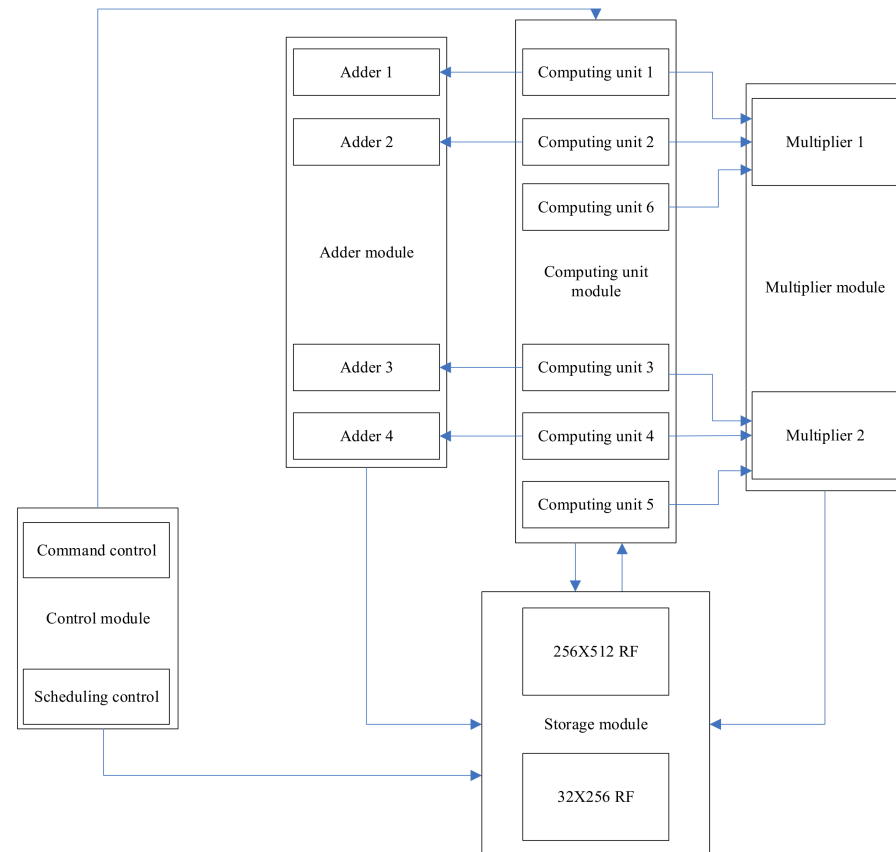


Figure 1. Overall hardware design framework for the scalar multiplication operation.

3.1. Hardware Design of Computing Unit Module

The scalar multiplication using precomputation is implemented in Jacobian coordinates using two 256-bit multipliers and four 256-bit adders. The multipliers adopt a three-stage pipeline structure, which means each multiplication requires three clock cycles. The modular multiplication operation adopts the Montgomery modular multiplication algorithm. The computing unit is implemented by calling a multiplier or adder. The input data of the computing unit comes from the storage unit, and the result is written back to the storage unit.

In the Jacobian projective coordinate system [4], the elliptic curve equation is transformed into $y^2 = x^3 + axz^4 + bz^6$, where $a, b \in F_p$, and $4a^3 + 27b^2 \neq 0 \pmod{p}$. For $z \neq 0$, the Jacobian coordinates (x, y, z) can be converted to affine coordinates (X, Y) via $X = \frac{x}{z^2}$ and $Y = \frac{y}{z^3}$, yielding the affine equation $Y^2 = X^3 + aX + b$.

The scheduling algorithm for scalar multiplication is as follows:

- (1) Input the elliptic curve parameters a and b , the scalar multiplication coefficient k , and a point $P (P_x, P_y, P_z)$ in Jacobian coordinates.
- (2) Expand the scalar k in binary as $k = (k_{255}k_{254} \cdots k_1k_0)_2$, where each bit is represented as k_i , and traverse from the most significant bit (MSB) to the least significant bit (LSB). When $k_i = 0$, only perform a point doubling operation on point P ; When $k_i = 1$, if it is

- the first bit 1, copy P to Q , then perform point doubling on P . If it is not the first bit 1, perform point doubling on P and point addition of P and Q .
- (3) Perform point addition and point doubling operations, using six computing units, where computing unit 1 and computing unit 2 are the computing units for doubling, computing units 3, 4, 5, and 6 are operational units for point addition. The multiplier resources are shared in groups: computing units 1, 2, and 6 share the first three-stage pipelined multiplier, whereas units 3, 4 and 5 share the second one. This resource-sharing pipeline schedule enables parallel computation on different multiplier stages and eliminates idle hardware cycles. Some existing designs choose to skip the first pipeline stage and only utilize the latter two stages to simplify control logic. Although this approach reduces control complexity, it sacrifices approximately one-third of the theoretical multiplication throughput. In contrast, our full-stage utilization strategy delivers higher computation throughput with no extra arithmetic units. The specific scheduling steps are detailed in Table 2. Table 2 details the parallel scheduling of six computing units across 11 steps for simultaneous point doubling and point addition. L_i denotes intermediate variables, where i ranges from 1 to 7. Seven variables L_1 to L_7 store intermediate values generated during point addition and point doubling operations. L_1 – L_6 are assigned values in Step 1, while computing unit 2 assigns a value to L_7 in Step 2. Iterative updates of L_1 to L_7 are then performed throughout the subsequent computation procedures. The pairs (X_2, Y_2, Z_2) and (X_3, Y_3, Z_3) are the resulting Jacobian coordinates for point doubling and addition, respectively. In addition, the term $a \cdot z^4$ is initialized as a constant multiplication of the curve parameter a with the fourth power of the z -coordinate. The initial value of $a \cdot z^4$ equals a since $P_z = 1$. Computing Unit 2 is adopted in Step 10 and Step 11 to iteratively update $a \cdot z^4$ for each subsequent point doubling operation. For the first point addition in the scalar multiplication sequence, point Q is initialized as a copy of point P when the first non-zero scalar bit is encountered. This initialization is performed by the control module before the scheduling sequence begins.

Table 2. Scheduling steps of computing units.

Step	Computing Unit 1	Computing Unit 2	Computing Unit 3	Computing Unit 4	Computing Unit 5	Computing Unit 6
1	$L_5 = P_x P_x$	$L_6 = P_y P_y$	$L_1 = P_z P_z$	$L_2 = Q_z Q_z$	$L_3 = P_z Q_y$	$L_4 = Q_z P_y$
2	$L_5 = L_5 + L_5$	$L_7 = L_5 + az^4$	$L_1 = Q_x L_1$	$L_2 = P_x L_2$	$L_3 = L_3 L_1$	$L_4 = L_4 L_2$
3	$L_6 = P_x L_6$	$X_2 = L_6 L_6$	$L_1 = L_1 - L_2$	$L_3 = L_3 - L_4$		
4	$L_5 = L_7 + L_5$	$L_6 = L_6 + L_6$	$X_3 = L_3 L_3$	$Z_3 = P_z L_1$	$Y_3 = L_1 L_1$	$L_4 = L_4 L_1$
5	$L_7 = L_5 L_5$	$Z_2 = P_y P_z$	$Z_3 = Z_3 Q_z$	$L_1 = L_1 Y_3$	$Y_3 = L_2 Y_3$	$L_4 = L_4 Y_3$
6	$L_6 = L_6 L_6$	$Y_2 = X_2 + X_2$	$X_3 = X_3 - L_1$	$L_2 = Y_3 + Y_3$		
7	$X_2 = L_6 + L_6$	$Y_2 = Y_2 + Y_2$	$X_3 = X_3 - L_2$			
8	$X_2 = L_7 - X_2$	$L_7 = Y_2 + Y_2$	$Y_3 = Y_3 - X_3$			
9	$L_6 = L_6 - X_2$	$Z_2 = Z_2 + Z_2$	$Y_3 = Y_3 L_3$			
10	$Y_2 = L_5 L_6$	$az^4 = L_7 az^4$	$Y_3 = Y_3 - L_4$			
11	$Y_2 = Y_2 - L_7$	$az^4 = az^4 + az^4$				

The six computing units time-share the two pipeline multipliers and four adders through interleaved scheduling. Specifically, for the two three-stage pipelined multipliers, the control logic schedules operations from different computing units to initiate in consecutive clock cycles. This time-multiplexing strategy fully utilizes all three pipeline stages,

allowing multiple multiplication operations to be processed concurrently without resource conflicts.

- (4) Output (X_2, Y_2, Z_2) as the result of the point doubling operation, stored in point P , and output (X_3, Y_3, Z_3) as the result of point addition, stored in point Q .
- (5) After traversing the scalar multiplication coefficient k , output point Q as the result of scalar multiplication.

Table 2 presents the high-level operation scheduling across six computing units, where each “Step” represents a logical scheduling stage rather than a single clock cycle. This computing unit module is implemented based on Jacobian coordinates and the Montgomery modular multiplication algorithm, and features a novel parallel scheduling scheme. It adopts a hybrid architecture consisting of six dedicated computing units, two three-stage pipeline multipliers and four adders, with partitioned scheduling for point doubling and point addition. The design fully overlaps different operations by utilizing all pipeline stages of multipliers, enabling parallel execution of point arithmetic and achieving near-optimal hardware utilization.

The scheduling logic is fully tailored for the complete SM2 scalar multiplication flow and supports both fixed-base and variable-base operations, which drastically reduces clock cycles and serves as the core foundation of high throughput for the whole coprocessor. Scalar multiplication dominates the overall latency of the SM2 signature. Traditional designs suffer from disordered unit allocation, low pipeline efficiency and long serial latency. Aiming at the operational characteristics of elliptic curve point arithmetic, we customize parallel scheduling, divide independent units and share pipeline resources. This design breaks the bottleneck of serial computation, greatly improves computational capability with limited hardware overhead, and realizes prominent throughput improvement.

The selection of two multipliers and four adders is grounded in both the choice of coordinate system and the intrinsic data-flow characteristics of the underlying formulae. We adopt Jacobian projective coordinates for scalar multiplication, as this representation eliminates the need for costly modular inversions within the iterative point operations, deferring the single inversion to the final affine conversion. Under this coordinate system, we performed a comprehensive data-flow analysis of the point doubling and point addition formulae, identifying all multiplication, squaring, and addition operations along with their interdependencies. This analysis reveals that the multiplication operations consistently form at most two independent streams that can proceed in parallel without data conflicts throughout the entire scheduling sequence, establishing a structural lower bound of two multipliers. Adding a third or fourth multiplier would not yield a proportional performance gain, as the dependency graph contains no additional independent multiplication stream to sustain an extra multiplier with useful work; any performance improvement would be marginal and quickly saturate, while the area and power overhead would increase substantially due to the large silicon footprint of 256-bit multipliers. For the addition operations, the data-flow analysis further reveals that certain scheduling steps contain up to four mutually independent additions that exhibit no data dependencies and can therefore be executed concurrently. Reducing the adder count to three would force one of these independent additions to be serialized into the next cycle, breaking the carefully aligned pipeline schedule and introducing idle bubbles that propagate through subsequent steps, ultimately increasing overall latency. Therefore, the configuration of two 256-bit multipliers and four 256-bit adders is not an empirical choice but a structural necessity.

3.2. Hardware Design of Storage Modules

According to the analysis of the comb algorithm steps for fixed-base, 256 points are precalculated before performing scalar multiplication. In this scheme, SRAM is used for storage.

Each individual SRAM features a 512-bit data width and a depth of 256 entries. To accelerate fixed-base scalar multiplication, we deploy thirty-two independent 256×512 SRAMs to store precomputed elliptic curve points. Different data are stored in each 256×512 SRAM. To store the intermediate variables and results of scalar multiplication operations in the computing units, four 16×256 SRAMs are adopted with uniform interfaces to facilitate hardware integration.

3.3. Hardware Design of Multiplier and Adder Modules

To significantly improve the performance of scalar multiplication, two 256-bit multipliers and four 256-bit adders are used, with the multipliers adopting a three-stage pipeline structure. Each stage of the four adders and two multipliers can perform different operations, and the two multipliers can be divided into a total of six pipeline stages for use. The point doubling occupies two pipeline stages, and the point addition occupies the remaining four pipeline stages. When traversing scalar multiplication coefficients, a sequential traversal from MSB to LSB is used to enable parallel computation of multiplication and addition operations, thereby maximizing the utilization of hardware resources and improving computational performance. The multiplier produces a pipelined result. In the first clock cycle, the operand of computing unit 1 enters the first stage, in the second clock cycle, the operand of computing unit 1 enters the second stage, the operand of computing unit 2 enters the first stage, and so on.

3.4. Hardware Design of Control Module

The control module mainly includes scheduling algorithm control and instruction control units. The scheduling algorithm mainly includes the operation scheduling control of 2 multipliers and 4 adders, as well as the control of data update and storage. Instruction control mainly includes the control of various instructions of this coprocessor, such as some basic large number operation instructions. The hardware implementation of this scheme includes large number operation instructions such as modular addition, modular subtraction, Montgomery modular multiplication, modular multiplication, modular exponentiation, point addition, point doubling, and scalar multiplication.

3.5. Key Masking and Blinding Design for Side-Channel Resistance

3.5.1. Security Design

This design adopts first-order arithmetic masking and key blinding, targeting the first-order Differential Power Analysis (DPA), which is the most common side-channel attack (SCA) for commercial cryptographic chips. The threat model assumes that adversaries can collect single-cycle power traces and analyze the correlation between power consumption and secret data. Chosen-message scenarios fall within the scope of our DPA threat model, since adversaries are capable of collecting power traces using arbitrary self-selected messages. The proposed countermeasures are designed specifically to resist first-order DPA and first-order Electromagnetic Analysis (EMA), as the same intermediate-value randomization applies to EM emanations. Higher-order DPA/EMA, template attacks, and differential fault analysis are outside the intended protection scope of this work. To resist side-channel attacks and eliminate the correlation between private keys and power consumption, this design integrates key masking and key blinding hardware modules for SM2 scalar multiplication and signature operations.

To eliminate the correlation between secret keys and power consumption in security-oriented hardware designs, key masking is applied to the private key [21]. Let d_A denote the private key. We transform the original private key into a masked version $d_{A'}$ using the formula: $d_{A'} = d_A + k_1n$, where k_1 is a 256-bit random number and n represents the order of the base point G on the elliptic curve. The masked key $d_{A'}$ is adopted for elliptic curve scalar multiplication operations. The introduced random number effectively breaks the correlation between the original private key d_A and the circuit power consumption.

This masking scheme does not affect the correctness of the final calculation results, due to the mathematical property $nG = O$ (where O is the point at infinity on the elliptic curve). Taking the scalar multiplication operation $d_{A'}G$ as an example:

$$\begin{aligned} d_{A'}G &= (d_A + k_1n)G \\ &= d_AG + k_1nG \\ &= d_AG + k_1(nG) \\ &= d_AG + k_1O \\ &= d_AG \end{aligned}$$

In the SM2 signature, the signature S relies on the private key d_A and thus needs key blinding [22]. The original formula is $S = [(1 + d_A)^{-1}(k - rd_A)] \bmod n = [(1 + d_A)^{-1}(k + r) - r] \bmod n$. After algebraic rearrangement, the blinded formula is derived as

$$\begin{aligned} S &= \{[m(1 + d_A)]^{-1}(k + r)m - r\} \bmod n \\ &= \{[m^{-1}(1 + d_A)^{-1}](k + r)m - r\} \bmod n \\ &= [(1 + d_A)^{-1}(k + r)m^{-1}m - r] \bmod n \\ &= [(1 + d_A)^{-1}(k + r) - r] \bmod n \\ &= [(1 + d_A)^{-1}(k - rd_A)] \bmod n \end{aligned}$$

where k and m are both 256-bit random numbers. Therefore, another 256-bit random number m is introduced in the transformed signature formula. The two random numbers fully blind the operations of d_A and eliminate key-related power leakage.

3.5.2. Security Analysis

We implemented a simulation-based first-order DPA attack targeting the intermediate value after key loading. For the unprotected design, the attack recovered the full 256-bit private key with only 1000 simulated power traces. For the masked design proposed, no key bit could be recovered even after 500,000 simulated power traces, and the correlation coefficients exhibited a Gaussian distribution centered at zero, with no significant peaks, indicating the absence of key-dependent leakage and confirming the effectiveness of the proposed countermeasures.

To preliminarily evaluate the side-channel security performance of the proposed SM2 hardware accelerator via simulation, Test Vector Leakage Assessment (TVLA) [23] is performed on the RTL design integrated with key masking and blinding design protection schemes. In this experiment, 500,000 simulated power traces are generated based on the Hamming-weight power consumption model following the conventional fixed-versus-random non-specific TVLA scheme, with each test group containing 50,000 traces. The corresponding t-statistic heatmap characterizes the spatial leakage distribution throughout the full computation latency, covering all clock cycles and sampling points within each cycle. The maximum absolute t-statistic value obtained in the test is 3.1, which is lower than the widely recognized side-channel leakage detection threshold of 4.5 [24]. The simulated TVLA results, based on the Hamming-weight power model, indicate that the adopted key

masking and blinding design strategy has the potential to suppress first-order side-channel information leakage under idealized conditions. However, it must be emphasized that this simulation model does not capture physical phenomena such as glitches, coupling effects, or non-ideal noise present in actual silicon. Therefore, while the algorithm demonstrates promising resistance in simulation, these results cannot be extrapolated to certify practical suitability for commercial cryptographic chips or guarantee security under physical operating conditions. Physical TVLA measurement on the fabricated chip is indispensable for drawing definitive conclusions on side-channel resistance and is planned as the next step after chip return. None of the sampling points exceed the threshold, which demonstrates that the design achieves preliminary performance in the simulation-oriented TVLA test.

All side-channel analysis results in this paper are obtained from simulated power traces built upon the Hamming-weight power model. It is necessary to clarify that the TVLA results above only reflect the first-order leakage resistance under the ideal simulated leakage model and cannot be regarded as formal security certification. Actual TVLA physical measurement on tape-out silicon chips is indispensable for drawing a definitive conclusion on anti-side-channel performance, which is planned as future work.

This module adopts a dual protection scheme combining first-order arithmetic masking and signature key blinding. Based on the mathematical property of elliptic curve order, the masked private key will not affect the correctness of results. Two independent 256-bit random numbers are introduced to reconstruct the signature formula, which thoroughly eliminates the correlation between private keys and power consumption. The on-chip TRNG generates fresh random numbers in real time to resist repeated random number attacks. This scheme features low hardware overhead and good compatibility, and can be integrated into the original data path to effectively defend against first-order DPA and other common side-channel attacks. SM2 hardware for commercial secure chips faces severe side-channel threats, where adversaries can recover private keys by analyzing power traces. Unprotected circuits have critical key leakage risks. The proposed masking and blinding scheme hides key characteristics at the hardware layer without changing algorithm functions or degrading performance.

3.5.3. Simulation Methodology for TVLA Evaluation

The TVLA evaluation reported in this paper was performed on simulated power traces rather than physical silicon measurements. The methodology is as follows:

Simulation Setup: The power traces are generated using a cycle-accurate gate-level simulation with a Hamming-weight power model. The power consumption at each clock cycle is computed as the Hamming-weight of all switching signals, multiplied by a unit capacitance factor. The simulation captures all internal registers and combinational logic transitions.

Noise Configuration: Additive white Gaussian noise (AWGN) with standard deviation $\sigma = 0.15$ is added to each sampling point to model measurement noise and process variations.

Test Configuration: Each TVLA test group consists of 50,000 traces, with a total of 500,000 traces across all groups. The fixed-versus-random non-specific methodology is adopted: the fixed group uses a constant message input, while the random group uses uniformly random messages. All traces are aligned prior to t-statistic computation.

Scope Limitation Statement: The TVLA evaluation methodology described above is inherently limited by its reliance on a simplified Hamming-weight power model and the absence of physical measurement noise, electromagnetic coupling, and environmental variations. The fixed-versus-random non-specific TVLA test with a t-statistic threshold of 4.5 is a standard detection methodology; however, passing this threshold in simulation does

not guarantee that the design will exhibit identical leakage characteristics when fabricated and measured under real operating conditions. The maximum absolute t-statistic of 3.1 reported herein should therefore be understood as an upper-bound estimate of first-order leakage under idealized conditions, and the actual t-statistic measured on physical silicon may differ significantly. We explicitly refrain from claiming that these simulation results demonstrate practical suitability for security-critical commercial applications.

Limitations and Future Physical Testing: We acknowledge that the Hamming-weight model is a simplification that does not capture glitches, propagation delays, or coupling effects present in physical silicon. Therefore, these simulation results are presented as preliminary validation of the mathematical soundness of the masking scheme, not as formal security certification. Physical TVLA measurement on the fabricated chip is planned as the next step after chip return, with the following timeline: (1) chip packaging and test board preparation; (2) power measurement setup with near-field probes; and (3) TVLA statistical analysis with 500,000 physical traces.

4. Hardware Design of Parallel Computing Platform for Public-Key Algorithm

The parallel computing platform supports a public key coprocessor module that includes modular and scalar operations for large numbers, and integrates a hardware true random number generator. The public key algorithm parallel computing platform exposes two buses to the external system: one is the slave bus, which is used by the processor to configure the parallel computing platform and read its status; the other is a master bus used by the parallel computing platform to retrieve data from the system memory.

The parallel computing platform mainly consists of the following functional units: DMA control unit, task queue control unit, command execution unit, and cryptographic hardware accelerator. The cryptographic hardware accelerator includes public key cryptographic engines, and reconfigurable symmetric and hash algorithms engines. The structure of the parallel computing platform is shown in the following figure (Figure 2).

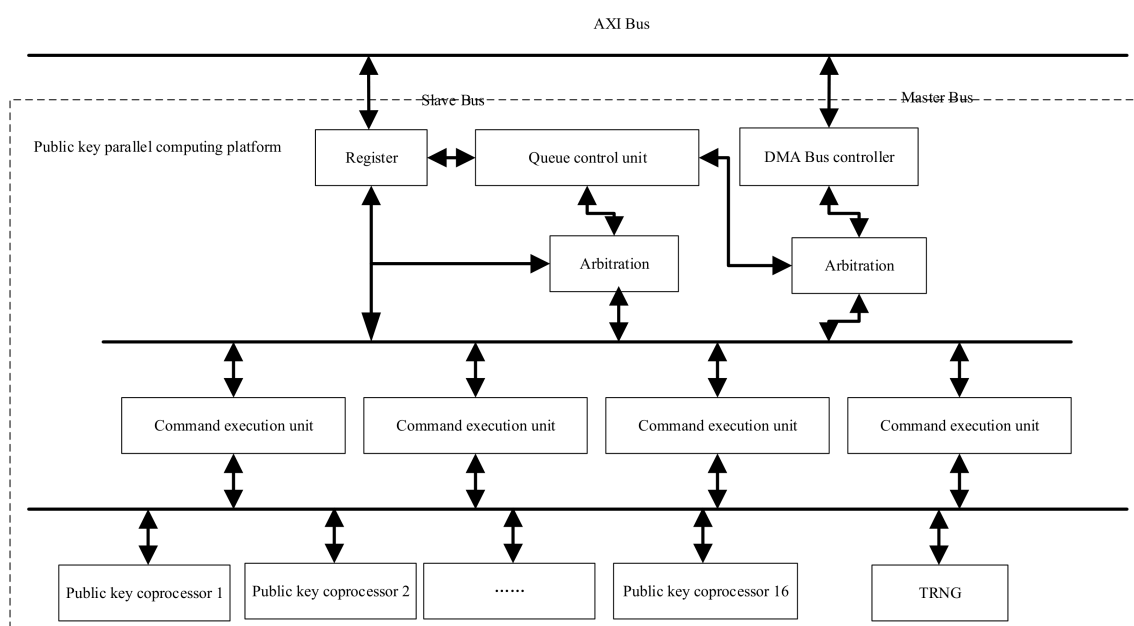


Figure 2. Structural diagram of parallel computing platform.

The public key parallel computing platform module adopts a parallel algorithm module design, multi-level caching, queue management, channel and resource isolation, and

hardware automatic scheduling design technology to achieve high-performance cryptographic algorithm coprocessors and cryptographic algorithm cross-parallel operation functions.

The parallel computing platform module supports up to 16 independent modules for parallel scheduling. The task queue is configured through the register configuration interface. The task queue retrieves tasks from devices on the bus through the DMA controller and issues them to the command execution unit. The order of tasks in each queue is sent to the command execution unit. The command execution unit schedules the algorithm based on the commands in the executed tasks, calls the corresponding algorithm unit for computation, and accesses the source and destination data from the devices on the bus through the DMA controller.

In this design scheme, public key parallel computing platform modules can be used simultaneously, with 16 public key coprocessor hardware modules used in each computing platform. This public-key parallel computing platform integrates a DMA controller, a task queue, a command execution unit and reconfigurable cryptographic engines, and supports parallel scheduling of up to 16 coprocessor cores. Equipped with multi-level caching, channel isolation, resource isolation and hardware auto-scheduling, it resolves bus contention and unbalanced task distribution in multi-core scenarios. The dual-bus architecture supports host configuration and high-speed data access. A single coprocessor cannot cope with massive concurrent signature requests in finance, blockchain and government service scenarios. Traditional multi-core designs simply duplicate computing cores without optimized task and bus management, leading to resource conflict and performance degradation. This dedicated parallel computing platform realizes automatic task distribution and resource isolation, fully exploiting multi-core computing power. It adapts to high-concurrency application scenarios and guarantees system stability and scalability.

5. Analysis of Hardware Design Results

The RTL design was synthesized, placed, and routed using the TSMC 28 nm HPC+ process technology. The design has successfully passed the entire physical implementation flow, including floor planning, power grid synthesis, clock tree synthesis (CTS), routing, and sign-off STA with Standard Parasitic Exchange Format (SPEF)-based parasitic extraction. All performance, power, and area results reported in this section are derived from post-layout gate-level simulations with RC parasitics, representing the pre-silicon estimated performance.

5.1. Functional Verification

We performed an RTL functional simulation of the SM2 Verilog module and established a UVM-based automated test platform for test case execution, waveform analysis, and result comparison. Additionally, the post-layout netlist was back-annotated with SDF (Standard Delay Format) for gate-level functional simulation to ensure timing correctness. FPGA-based emulation was used during the RTL development phase for pre-tapeout validation. The verification covers official SM2 Known Answer Test (KAT) vectors, all-zero/all-one boundary cases and one billion random test cases. The code statement coverage and branch coverage are both above 99%.

All functional tests follow the national commercial cryptographic standards GM/T 0003-2020 [4], GM/T 0009-2012 [25] and GM/T 0039-2024 [26]. The module's parameters, workflows, data formats and exception handling comply with standard requirements, and all test outputs match the standard references, validating the correctness of the implementation.

Post-Silicon Validation Plan: Upon chip return, our post-silicon validation plan includes: (a) functional testing using a custom PCB with high-speed FPGA interface and automated test pattern generation; (b) performance measurement using an on-chip frequency counter to determine maximum stable operating frequency; (c) power measurement using on-die thermal sensors and external current monitoring; and (d) side-channel evaluation using the methodology described in Section 3.5.3.

5.2. Performance Analysis

5.2.1. Single-Module Performance Analysis

The design is implemented in Verilog HDL for RTL (Register Transfer Level) implementation of the overall scheme.

In the SM2 cryptographic algorithm, core computational operations consist of fixed-base scalar multiplication and modular inversion. Based on the proposed hardware architecture, the latency of each arithmetic primitive is detailed below. For elliptic curve point addition, the latency ranges from 98 to 108 cycles: the operation completes in 98 cycles when all modular addition results are less than the modulus P ; when the result is greater than or equal to P , an extra subtraction is performed, increasing the latency to 108 cycles. For the 31 point additions involved in scalar multiplication, the total cumulative latency is 3328 cycles. This total includes not only the raw point-addition operations but also the following overheads: (1) SRAM lookup latency for retrieving precomputed points from the 32 tables; (2) pipeline fill and drain cycles between consecutive point additions; and (3) control logic cycles for table-index decoding and operand routing.

The modular inversion operation requires 3242 clock cycles, accompanied by additional overhead for control logic. The system adopts a 128-bit bus width. The input data containing the private key and message totals 512 bits, requiring 4 clock cycles for data loading. The output signature is also 512 bits, taking another 4 clock cycles for data transmission. In summary, the full execution of one complete SM2 signature generation occupies approximately a total of 6697 clock cycles on average. All operation latency values are summarized in Table 3.

Table 3. Operation latency statistics.

Operation Category	Specific Operation	Specification	Clock Cycles
ECC Point Operation	Single Point Addition (Minimum)	Modular addition result $< p$	98
ECC Point Operation	Single Point Addition (Maximum)	Modular addition result $\geq p$ (with extra subtraction)	108
ECC Point Operation	Total of 31 Point Additions	Cumulative latency for 31 rounds of point addition	3328
Core Operation	Modular Inversion	Full modular inversion computation	3242
Data I/O Interface	Input Data Loading	128-bit bus, 512-bit input (private key + message)	4
Data I/O Interface	Output Signature Transmission	128-bit bus, 512-bit signature output	4
Auxiliary Logic	Additional Control Operations	Scheduling and peripheral control logic.	119
Overall System	Full SM2 Signature Generation	End-to-end total latency	6697

It is worth noting that all latency figures reported in Table 3 already include the full overhead of the key masking and blinding modules described in Section 3.5. Compared with the unprotected design, the integration of security countermeasures introduces 2

additional Montgomery modular multiplications and 2 modular additions, all of which have been incorporated into the 119 clock cycles allocated for auxiliary control logic in Table 3. However, these operations are not executed on the critical path; they are scheduled during the idle slots of the scalar multiplication pipeline when the arithmetic units would otherwise be stalled waiting for memory accesses or operand availability. Consequently, these additional operations have been fully absorbed within the 119 clock cycles allocated for auxiliary control logic and scheduling overhead in Table 3. The end-to-end signature latency remains unchanged compared to the unprotected baseline, and all security-related operations are already included in the reported cycle count. Random number generation is performed by the on-chip true random number generator during computation idle slots, and its generation process is fully parallelized with the scalar multiplication pipeline, thus incurring no impact on the end-to-end signature throughput. Therefore, the performance evaluation results presented above reflect the simulated system performance with security countermeasures fully integrated. In summary, the introduction of the security modules incurs negligible overhead in both latency and chip area, achieving both high performance and side-channel attack resistance simultaneously.

Note that the latency breakdown in Table 3 focuses on the public-key cryptographic operations within the SM2 signature generation. The hash computation $e = H_v(M)$ (Step 2 of Algorithm 1), where $M = Z_a || M$, is performed by the on-chip SM3 hash engine. For a typical message length of 256–512 bits, the SM3 hash computation requires approximately 100–200 clock cycles. However, since the hash engine operates in parallel with the scalar multiplication pipeline and message preprocessing that computes Z_a can be executed offline or during system initialization, hash latency does not add to the critical path of signature generation for preprocessed messages.

Algorithm 1 SM2 signature algorithm.

Input: elliptic curve parameter set $(p, E(Fp), G, n)$, private key d_A , message M , signature identifier Z_A

Output: Signature (r, s)

- 1: $Z_A = H_{256}(ENTLA || IDA || a || b || x_G || y_G || x_A || y_A)$. Then set $M' = Z_A || M$.
 - 2: Calculate $e = H_v(M')$, convert the data type of e to an integer, where $H_v()$ represents a cryptographic hash function with a message digest length of v bits, which is the SM3 hash algorithm.
 - 3: Generate a 256-bit random number $k \in [1, n - 1]$.
 - 4: Calculate the elliptic curve point $(x_1, y_1) = kG$ and convert x_1 to an integer.
 - 5: Calculate $r = (e + x_1) \bmod n$, if $r = 0$ or $r + k = n$, return to step 3.
 - 6: Calculate $s = ((1 + d_A)^{-1}(k - rd_A)) \bmod n$, if $s = 0$, return to step 3.
 - 7: Convert r, s into a byte string, and obtain the signature of message M as (r, s) .
 - 8: Return (r, s) .
-

Under the TSMC 28 nm process, the operating frequency of this hardware coprocessor module is 600 MHz. Therefore, the single module achieves 89,593 signatures per second.

As shown in Table 4, our design scheme achieves the highest performance of the SM2 signature algorithm, reaching 89,593 signatures per second per module. This is mainly due to the precomputation in the fixed-base scalar multiplication operation of the SM2 signature algorithm, as well as the scheduling optimization of the non-fixed-base scalar multiplication operation, which greatly improves the overall performance of the SM2 signature algorithm.

Within the scope of the compared designs and under the stated assumptions, our single-module design achieves the highest absolute throughput among the five compared ASIC implementations. Its normalized area efficiency reaches 33.31 sig/s/KGE, which is higher than the corresponding figures in the referenced designs. This advantage is primarily attributable to two factors: (a) the Comb-algorithm precomputation significantly reduces the cycle count for fixed-base scalar multiplication; and (b) the 28 nm process enables a 600 MHz operating frequency, substantially higher than the frequencies of the referenced designs in older process nodes. When frequency-normalized (sig/s/MHz/KGE), our design shows only a marginal 0.9% advantage over Ref. [11], indicating that the microarchitectural innovations of this work achieve competitive normalized efficiency. This observation suggests that the primary contribution of this work lies in achieving a favorable area–time trade-off while leveraging advanced process technology for higher absolute throughput. The AT product, which inherently accounts for the area–delay cost of delivering one complete signature operation, serves as the most meaningful cross-design comparison metric given the heterogeneity of the referenced implementations.

Table 4. Performance comparison of the proposed and referenced ASIC designs.

	Curve	Platform	Freq. (MHz)	Original Resources	Cycles	Performance (sig/s)	Equivalent Gate Count	Norm. Area Efficiency (sig/s/KGE)	Norm. Performance (sig/s/MHz/KGE) ($\times 10^{-3}$)
Ref. [6]	$GF(p)$ 256-bit	0.13 μ m	137.7	120 K	340 K	405	120 K	3.38	24.5
Ref. [7]	$GF(p)$ 256-bit	0.13 μ m	556	122 K	562 K	990	122 K	8.11	14.6
Ref. [9]	$GF(p)$ 256-bit	90 nm	250	122 K	193 K	1298	122 K	10.64	42.6
Ref. [11]	$GF(p)$ 256-bit	55 nm	143	29 K	640 K	228	29 K	7.86	55.0
Our work	$GF(p)$ 256-bit	28 nm	600	2.69 M	6697	89,593	2690 K	33.31	55.5

We report absolute area efficiency (sig/s/KGE) as a supplementary metric for completeness, as it reflects the throughput per unit of silicon area at the native 600 MHz operating frequency. However, given the significant process heterogeneity across the compared designs, we caution against overinterpreting this metric as a measure of architectural merit. The principal architectural benchmarks adopted in this work are the frequency-normalized efficiency (sig/s/MHz/KGE) and the AT product, as these metrics inherently account for process scaling and area–delay trade-offs across different technology nodes. Our frequency-normalized efficiency is comparable to that of Ref. [11], with a marginal difference of approximately 0.9%. This confirms that the proposed microarchitecture achieves competitive normalized efficiency, and that the higher absolute throughput is primarily enabled by the advanced 28 nm process.

5.2.2. Performance Comparison with Prior ASIC Designs

Comparison Scope and Limitations. Before presenting the quantitative comparison, we emphasize that the referenced designs exhibit substantial heterogeneity that limits the conclusiveness of any direct performance comparison. Table 5 summarizes the key functional and methodological differences among the compared designs. Specifically, (a) the process technology varies from 0.13 μ m to 28 nm, with corresponding differences in device performance and density; (b) the functional scope ranges from full cryptographic processors [6,7,9] to dedicated point-multiplication macros [11], affecting area allocation and optimization priorities; (c) the evaluation methodologies differ between logic synthesis and post-layout simulation with parasitic extraction; and (d) side-channel countermeasures

are present only in Ref. [11] and our work, while others are unprotected. These differences are inherent to the state of the art and cannot be eliminated through normalization. Therefore, the following comparison should be interpreted as a reference under the stated assumptions, not as definitive proof of architectural merit. We adopt the area–time (AT) product as the holistic comparison metric because it inherently accounts for area–delay trade-offs across different technology nodes.

Table 5. Assessment of functionality and data completeness.

Design	Platform	Functional Scope	Chip Positioning	Side-Channel Protection	Memory/DMA
Ref. [6]	0.13 μm	Dual-field	Full crypto processor	None	Dual banks; DMA unspecified
Ref. [7]	0.13 μm	GF(p)-only	Systolic-array dedicated accelerator	None	No independent memory
Ref. [9]	90 nm	Dual-field	Dual-field coprocessor	None	No independent DMA/memory
Ref. [11]	55 nm	SM2-only + SCA	Point-multiplication macro only	Masking + randomization	No DMA/memory specs
Our work	28 nm	SM2-only + SCA	Standalone SM2 accelerator	Masking + blinding	DMA + 512 KB SRAM

To ensure a fair and transparent comparison, we categorize existing ECC hardware implementations according to their supported finite fields, algorithmic scope, memory architecture, control-logic complexity, and bus and DMA infrastructure, as well as integrated side-channel attack countermeasure modules. The component-level accounting covers arithmetic core, memory, control logic, bus/interface, SCA countermeasures, and auxiliary logic, to the extent that such data are available from the original publications. Missing items are denoted ‘N/A.’

Prior to the performance comparison, Table 5 systematically assesses the functionality and data completeness of the four referenced designs [6,7,9,11] against our work. As shown, the referenced designs differ substantially in process technology, function scope, chip positioning, side-channel protection, and memory/DMA configurations. This heterogeneity renders direct comparisons of total gate counts or absolute cycle counts methodologically problematic. Therefore, we adopt the Area–Time product ($AT = \text{Area} \times \text{Time}$) and frequency-normalized metrics as the primary comparison scales in the following analysis, as these metrics inherently account for architectural heterogeneity by incorporating area, latency, and frequency into a unified evaluation framework.

Table 6 presents a component-level area breakdown to the extent that data are available from the original publications, covering the arithmetic core, memory, control logic, bus interface, SCA countermeasures, and total area, alongside aligned performance metrics including latency, AT product, area efficiency, and frequency-normalized efficiency. Consequently, the comparisons presented in this section are explicitly confined to a preliminary, scope-limited assessment rather than a definitive ranking. The AT product is reported as a supplementary holistic metric; however, we emphasize that AT alone cannot compensate for differences in functionality, security features, or evaluation methodology.

Table 6. Component-level area breakdown.

Design	Eval.	Core (Gates)	Mem (Gates)	Ctrl (Gates)	Bus (Gates)	SCA (Gates)	Total (Gates)	Time (ms)	AT (KGE·ms)
Ref. [6]	Syn.	106.7 K	11.4 K	N/A	N/A	0	120 K	2.47	296.3
Ref. [7]	Syn.	N/A	N/A	N/A	N/A	0	122 K	1.01	123.2
Ref. [9]	Syn.	N/A	N/A	N/A	N/A	0	122 K	0.77	94.0
Ref. [11]	Syn.	N/A	N/A	N/A	N/A	N/A	29 K	4.386	127.2
Our work	P-layout	1675 K	920 K	77 K	16 K	2 K	2690 K	0.011	30.03

Note: “Syn.” = Logic synthesis; “P-layout” = post-layout simulation. This table is for illustrating available data; due to missing information, it is not used to construct a rigorous quantitative comparison. Entries marked “N/A” indicate that the original literature did not disclose the corresponding component data; these missing items preclude a rigorous component-level normalization, and readers are cautioned against drawing strong conclusions from incomplete data.

Primary Benchmark: Frequency-Normalized Area Efficiency. To mitigate the dominant impact of process-driven frequency scaling, we adopt the frequency-normalized metric (sig/s/MHz/KGE) as the primary benchmark for intrinsic microarchitectural comparison. As shown in Table 4, our design achieves 55.5×10^{-3} sig/s/MHz/KGE, while Ref. [11] achieves 55.0×10^{-3} sig/s/MHz/KGE. The marginal difference of approximately 0.9% indicates that, after removing the frequency advantage conferred by the 28 nm process, the proposed scheduling and precomputation architecture delivers microarchitectural efficiency comparable to the most area-optimized prior design [11]. This near-identical normalized efficiency suggests that the additional architectural complexity of this work does not introduce significant inefficiency.

Secondary Metric: Absolute Area Efficiency (sig/s/KGE). Our design achieves 33.31 sig/s/KGE at its native 600 MHz operating point. This figure is higher than those reported for Refs. [6,7,9,11]; however, this advantage is overwhelmingly attributable to the higher operating frequency enabled by the TSMC 28 nm process rather than to intrinsic architectural merit. The reference designs operate at significantly lower frequencies on older process nodes, where lower transistor switching speeds inherently constrain throughput. Therefore, the absolute area efficiency is presented only as a supplementary datapoint under the explicit condition that the compared designs target different process generations.

Area–Time Product Analysis: Our design achieves an AT value of 30.03 KGE·ms, which is lower than the AT values of the other four referenced designs under the same evaluation framework. Compared with Ref. [9], our AT corresponds to approximately 31.9% of that benchmark, reflecting a 68.1% reduction in area–time product. Compared with Ref. [11], our AT amounts to roughly 23.6%. This advantage stems from the trade-off between hardware overhead and computing latency: our design incurs approximately 92 times the area of Ref. [11] but delivers a latency roughly 1/393 of that design. While this trade-off yields a favorable AT within the defined comparison framework, we caution that AT is sensitive to the inclusion or exclusion of memory, I/O, and security components, and the referenced designs report these components with varying levels of completeness.

Synthesis of Metrics: Taken together, the three metrics support a conditional and scope-limited conclusion:

First, in terms of frequency-normalized efficiency (sig/s/MHz/KGE), our design exhibits comparable performance to the best prior design [11], suggesting that the proposed architecture does not degrade intrinsic microarchitectural efficiency.

Second, in terms of absolute area efficiency (sig/s/KGE), our design achieves higher values, primarily due to the advanced 28 nm process and 600 MHz frequency, not solely due to architectural innovation.

Third, in terms of AT value (KGE·ms), our design yields the lowest numerical value among the five compared schemes, reflecting a favorable area–delay trade-off within the bounds of the available data.

Therefore, considering the methodological constraints outlined in Section 1.3 and Table 5, the core contribution of this work lies in an optimized hardware architecture that achieves a desirable area–time trade-off and maintains competitive normalized microarchitectural efficiency when benchmarked against selected prior implementations under explicitly acknowledged limitations.

5.2.3. Performance Loss Analysis of Multi-Core Scaling

When deployed on the proposed 16-core parallel computing platform, the overall throughput reaches 1.03 million signatures per second. To quantitatively evaluate the multi-core parallel efficiency of the proposed platform, we compare the ideal linear scaling throughput against the simulated throughput. Based on the single-core throughput of 89,593 signatures per second and the 16-core platform throughput of 1.03 million signatures per second, the ideal linear scaling throughput for 16 cores would be 1,433,488 signatures per second, which is calculated by multiplying the single-core throughput by the core count. The actual platform achieves only 1.03 million signatures per second, resulting in a performance loss of approximately 28.1 percent. This corresponds to a parallel efficiency of 71.9 percent and a simulated speedup of 11.5 times over the single-core performance.

This performance loss primarily originates from five sources within the parallel computing platform architecture described in Section 4. To quantitatively evaluate multi-core parallel efficiency, we extracted the performance loss components using cycle-accurate RTL simulation logs captured during 1000 consecutive signature generation tasks across all 16 cores. We instrumented the RTL code with performance counters recording: (a) bus wait cycles for each DMA request; (b) queue empty/idle cycles for each core; (c) DMA setup latency; (d) cache miss counts and associated stall cycles; (e) isolation protection cycles. The primary source is shared master bus arbitration contention, which accounts for an estimated 14% of the total loss. When all 16 cores simultaneously initiate DMA accesses to the system memory via the shared master bus, the bus can serve only one request at a time, forcing other requesting cores to wait for bus arbitration and data transfer completion. The second source is task queue scheduling and dispatch overhead, contributing approximately 5% of the loss. This overhead encompasses task enqueue and dequeue operations, command parsing and decoding by the command execution unit, and load imbalance across cores caused by uneven task completion times, all of which introduce additional clock cycles before computation begins. The third source is DMA configuration and handshake latency, accounting for roughly 3% of the loss. Before each task execution, the DMA controller requires register configuration for source addresses, destination addresses, and transfer lengths; moreover, the request-response handshake protocol between the DMA and system memory introduces extra latency. In short-task scenarios where each core processes a relatively small number of operations per task, this setup overhead is amplified as a percentage of total execution time. The fourth source is multi-level cache misses and memory access latency, responsible for about 4% of the loss. When a cache miss occurs in one or multiple cores, the system must access off-chip system memory, which incurs significantly higher access latency and may saturate the memory bandwidth when multiple misses happen concurrently. The fifth source is resource isolation and protection mechanisms, contributing approximately 2% of the loss. To prevent data interference between different tasks and mitigate side-channel leakage risks, the hardware inserts isolation protection cycles and may flush pipelines or cache states during channel switching, introducing idle clock cycles that reduce peak throughput. The simulated results are summarized in Table 7.

Table 7. Performance loss breakdown for 16-core scaling.

Loss Source	Simulated Avg. Cycles per Task	Percentage of Non-Computation Cycles	Contribution to Overall Throughput Loss
Bus Arbitration	1314	50.2%	14.1%
Task Queue Scheduling	466	17.8%	5.0%
DMA Setup/Handshake	279	10.7%	3.0%
Cache Misses	373	14.2%	4.0%
Resource Isolation	186	7.1%	2.0%
Total Loss	2618	100%	28.1%

The ‘Contribution to Overall Throughput Loss’ column indicates each source’s share of the 28.1% total throughput degradation observed in the 16-core system. We validated the accuracy of these estimates by comparing the summed cycles of all individual loss sources, which amounted to 2618 cycles, with the total non-computation cycles directly counted from the trace logs at 2616 cycles; this comparison yields a discrepancy of merely 0.08%. In summary, the five sources collectively cover the observed 28.1% performance loss, demonstrating that the platform sustains necessary hardware management overheads while still achieving an 11.5 times speedup. The parallel efficiency of 71.9% is considered acceptable for a 16-core shared-bus cryptographic accelerator, particularly when compared to traditional multi-core designs without dedicated scheduling, which suffer from severe performance degradation caused by bus arbitration conflicts and uneven task distribution as discussed in Sections 1.1 and 1.3.

5.2.4. Design Space Exploration Analysis of Segment Count

To determine the optimal number of segments, we conducted a quantitative trade-off analysis evaluating configurations from 4 to 64 segments. Let S represent the number of segments. We quantitatively analyze the trade-off between storage overhead and scalar multiplication latency under different S values, with all comparison results listed in Table 8.

Table 8. Segment count trade-off analysis.

Segments (S)	Bits/Seg	Point Additions	LUT Entries	SRAM Area	Feasibility (On-Chip)	Area Efficiency (sig/s/KGE)
4	64	3	264	4.72×1021 Bytes	Infeasible (Astronomical)	-
8	32	7	232	2.20×1012 Bytes	Infeasible (Off-chip only)	-
16	16	15	65,536	64 MB	Impractical (Too large for on-chip SRAM)	-
32	8	31	256	512 KB	Feasible & Optimal	33.3
64	4	63	16	64 KB	Feasible	31.4

Based on the physical capacity analysis above, $S = 32$ achieves the best trade-off between on-chip SRAM area overhead and scalar multiplication latency. The advantages are as follows:

First, $S = 4$ and $S = 8$ entail prohibitively large storage requirements, which are completely infeasible for on-chip SRAM in any modern CMOS process. $S = 16$ requires 64 MB of on-chip SRAM. Although $S = 16$ is theoretically feasible, its required large SRAM capacity would substantially increase die area and power consumption, making the chip too

costly for commercial deployment. Consequently, among physically realizable candidates, only two viable configurations remain: $S = 32$ with 512 KB of on-chip SRAM and $S = 64$ with 64 KB of on-chip SRAM.

Second, a quantitative comparison between these two candidates reveals that $S = 32$ is clearly superior. Compared to $S = 32$, $S = 64$ reduces SRAM area by 448 KB (saving approximately 0.8 M gates), but it doubles the number of point additions from 31 to 63. Based on Table 3, each point addition averages 107 cycles, adding $32 \times 107 = 3424$ extra cycles. Consequently, the total signature latency increases from 6697 to 10,121 cycles, reducing throughput from 89,593 sig/s to 59,300 sig/s. The area for $S = 64$ is estimated at 1.89 M gates, yielding a normalized area efficiency of 31.4 sig/s/KGE, compared to 33.3 sig/s/KGE for $S = 32$. Thus, $S = 32$ outperforms $S = 64$ in area efficiency while delivering substantially higher absolute throughput.

Third, compared to 64 SRAM instances, 32 instances significantly ease clock tree synthesis (CTS), placement, and routing. Fewer memory macros reduce signal integrity issues, insertion delays, and the area overhead associated with additional buffering.

5.3. Area and Power Analysis

This scheme is implemented using the hardware description language Verilog. To significantly improve the performance of the fixed-base scalar multiplication operation in the elliptic curve signature algorithm, thirty-two 256×512 SRAMs are used for storing precomputed points. In the entire public key algorithm coprocessor module, four 16×256 SRAMs are also used. The public-key coprocessor core has a total equivalent gate count of 2.69 M, including the 512 KB precomputation point lookup SRAM array. Under normal temperature conditions, a single SM2 algorithm module consumes 0.25 W of power when operating at a frequency of 600 MHz. The security masking and blinding modules are embedded in the main data path with minimal additional combinational logic and only a small number of registers for random-number buffering; no additional SRAM macros are introduced. Only a minimal amount of combinational logic is added, and the resulting area overhead is entirely negligible relative to the total core area of 2.69 M gates. Therefore, the security countermeasures effectively enhance side-channel attack resistance with virtually no chip area penalty, achieving a favorable balance between area efficiency and security.

The physical core area in the TSMC 28 nm HPC+ process is 4.0 mm^2 . The standard cell utilization rate is approximately 60% after placement and routing, with the remaining area allocated for routing and power distribution. The relatively conservative utilization was intentionally chosen to improve power-delivery and IR-drop integrity, as the crypto core contains two large-number multipliers that generate significant instantaneous current spikes. Spreading these multipliers with greater physical distance helps reduce local current density and improves overall power delivery network robustness. The total die area, including the I/O pad ring and seal ring, is approximately 5.2 mm^2 .

The power consumption of 0.25 W at 600 MHz is obtained from post-layout simulation using Synopsys PrimeTime PX with the following conditions: TSMC 28 nm HPC+ process, typical corner (TT), supply voltage of 0.9 V, and junction temperature of 25°C .

To achieve a 600 MHz timing closure, our design addresses two key timing bottlenecks introduced by complex scheduling logic, 32 SRAM banks, and large multipliers, namely SRAM CK-to-Q read delay and long multiplication path latency. We resolve these issues via targeted optimizations: splitting large SRAM banks to reduce bitline capacitance and read latency, adopting Low Threshold Voltage (LVT) memory instances to mitigate SRAM timing delay, deploying optimized DesignWare multipliers and inserting pipeline registers to shorten combinational multiplication paths, and tuning clock tree synthesis

to minimize skew between scheduling logic and SRAM banks for reliable setup and hold timing margins.

6. Discussion

6.1. Simulation Results and Performance-Security Analysis

The simulation results demonstrate that the proposed architecture achieves higher throughput and area efficiency than the selected prior-art ASIC designs within the scope of this comparison. As shown in Table 4, under the native 600 MHz operating frequency, our single-module design achieves an absolute area efficiency of 33.31 sig/s/KGE, which is higher than the corresponding figures reported in the referenced designs. However, as discussed in Section 5.2.2, this advantage is primarily attributable to the advanced 28 nm process and higher operating frequency; the frequency-normalized efficiency is comparable to that of Ref. [11], with only a marginal difference of approximately 0.9%. Therefore, the absolute area efficiency improvement should be understood as a process-driven benefit rather than a fundamental microarchitectural breakthrough. This improvement can be attributed to two primary factors:

First, the hardware precomputation scheme based on the Comb algorithm transforms fixed-base scalar multiplication into 31 point additions using precomputed SRAM lookup tables, since the first point is loaded directly from the lookup table. This architecture eliminates massive iterative point doublings adopted by conventional binary scalar multiplication algorithms, cutting the total cycle overhead of fixed-base scalar multiplication from hundreds of thousands to only 3328 cycles, consistent with the latency data listed in Table 3. Second, the novel six-unit parallel scheduling architecture maximizes hardware utilization. By strategically assigning two three-stage pipelined multipliers and four adders to concurrently execute point addition and point doubling operations, the design effectively eliminates pipeline stalls. The sequential traversal method ensures that all pipeline stages are continuously fed with operands, breaking the serial computation bottleneck inherent in conventional designs.

Furthermore, the integration of a 16-core parallel computing platform with DMA-based auto-scheduling and task queues resolves the typical bus contention issues in multi-core systems. The significant throughput, scaling up to 16 cores, demonstrates the effectiveness of the hardware task management, suggesting its potential suitability for high-concurrency applications such as blockchain nodes and financial gateways.

From a security perspective, the implemented first-order arithmetic masking and key blinding schemes are intended to decouple the private key from power consumption traces. Simulation-based DPA attack testing provides preliminary evidence of the proposed masking and blinding schemes' effectiveness within the constraints of the simulated leakage model. However, these results are strictly preliminary. Simulation TVLA results only reflect first-order leakage suppression capability under an idealized leakage model and cannot be regarded as formal security certification, nor do they establish practical resistance under physical operating conditions. Physical TVLA measurement on the fabricated chip, including near-field EM probes and real power-trace acquisition under varying temperature and voltage conditions, is indispensable for drawing definitive conclusions on side-channel resistance. Until such physical measurements are completed, no claim is made regarding the design's adequacy for commercial security-chip deployment.

6.2. Portability to Other Prime-Field Curves

The proposed architecture is directly portable to other prime-field curves of the same bit length, including NIST P-256 and SECP256k1. The required modifications are minimal: (a) updating the curve parameters (p , a , b , n , G) in the configuration registers, and

(b) regenerating the precomputation SRAM tables for the new base point G . No changes to the arithmetic data path, scheduling logic, or control unit are required, as these are parameterized by the bit width and not by the specific curve parameters.

For curves with different bit lengths, the architecture would require more significant modifications: (a) scaling the bit width of all multipliers, adders, and registers; (b) adjusting the segment count to maintain the same segment size or re-optimizing it based on the new bit length; and (c) modifying the SRAM depth and data width accordingly.

The Comb-algorithm-based precomputation scheme remains effective for all curves with a fixed-base point. For curves where the base point changes frequently, precomputation would need to be regenerated on demand, with regeneration latency scaling linearly with the number of segments.

6.3. Quantum Threat and Crypto-Agility

The proposed architecture is optimized for elliptic curve cryptography, which is known to be vulnerable to Shor's algorithm on a sufficiently large fault-tolerant quantum computer. While no such quantum computer currently exists, the cryptographic community is actively standardizing post-quantum cryptographic (PQC) algorithms in response to this long-term threat.

The parallel computing platform infrastructure—including the DMA controller, hardware task queues, multi-level caching, and 16-core scheduling framework—is hardware-agnostic and can support PQC algorithm cores with appropriate modifications to the cryptographic engine module. Specifically: (a) the DMA and task queue infrastructure can handle any compute-intensive workload, including lattice-based or code-based PQC; (b) the arithmetic units would need to be replaced with PQC-specific data paths; (c) the 512 KB precomputation SRAM could be repurposed for PQC-specific tables or reallocated as general-purpose memory.

It must be emphasized that such PQC support has not been implemented in the current design. No PQC algorithms have been implemented in RTL, synthesized, simulated, or performance-evaluated in this work. For practical deployment, we recommend a hybrid signature approach that combines SM2 with a post-quantum signature algorithm during the transition period. PQC extension and hybrid signature performance evaluation constitute explicit future work. Subsequent research will design and implement PQC-specific accelerator cores while retaining the parallel computing platform infrastructure proposed herein, with actual RTL implementation and synthesis simulation to verify throughput and area efficiency.

6.4. Manufacturing Cost and Target Applications

Taking TSMC 28 nm mature-process wafer pricing into account, the die area of this design includes the I/O pad ring. The per-die cost remains low under large-volume manufacturing, and the total unit cost rises only slightly after adding packaging and testing. This architecture addresses two distinct market segments. The first covers high-end server hardware security modules, cloud security gateways and financial infrastructure equipment, where the ultra-high signature processing throughput of the 16-core design offsets the hardware cost incurred by the larger die area. The second targets high-performance edge servers equipped with 2 to 4 cores, with well-controlled overall area and power consumption metrics. For cost-sensitive IoT edge terminals constrained by gate count budgets, a scaled-down single-core version can be developed by simplifying on-chip SRAM configurations to shrink the chip footprint, accompanied by a corresponding loss in processing performance.

7. Conclusions

This paper proposes a high-throughput SM2 signature ASIC architecture integrating precomputation and multi-core parallel scheduling. It must be clearly stated that all performance data are derived from post-layout gate-level simulations and STA with RC parasitic extraction in TSMC 28 nm, representing pre-silicon performance estimations. Physical chip measurements await tape-out return and will be essential for final performance validation. Similarly, all side-channel security claims are based on simulation-based TVLA under a Hamming-weight power model and remain preliminary until physical measurements on fabricated silicon are completed.

First, a Comb-algorithm on-chip SRAM precomputation scheme is designed for fixed-base scalar multiplication, which reduces massive iterative point doublings to a limited number of point additions and greatly cuts scalar multiplication latency.

Second, a unified six-arithmetic-unit collaborative parallel scheduling framework is presented for both fixed-base and variable-base scalar multiplication, ensuring architectural flexibility for SM2 signature generation, verification, and other protocols.

Third, a DMA-driven 16-core parallel computing platform with hardware task queues is constructed, supporting up to 16 coprocessor cores and realizing significant throughput scaling under high-concurrency service loads.

Fourth, within the scope of the five compared ASIC designs and under the stated comparison assumptions, post-layout simulation results indicate that a single core delivers a throughput of 89,593 signatures per second at 600 MHz, with a normalized area efficiency of 33.31 sig/s/KGE. This figure is higher than the corresponding values reported in the referenced designs. However, it must be emphasized that the bulk of this advantage stems from the frequency scaling enabled by the advanced 28 nm process, whereas the referenced designs adopt older 0.13 μm , 90 nm, and 55 nm nodes. When frequency-normalized (sig/s/MHz/KGE), our design shows only a marginal 0.9% advantage over Ref. [11], indicating that the microarchitectural innovations achieve competitive normalized efficiencies. The area–time product of our design is the lowest among the five compared designs, reflecting a favorable area–delay trade-off. These comparisons should be interpreted as reference points within a defined scope, not as definitive demonstrations of architectural merit, given the inherent heterogeneity of the referenced implementations in terms of process technology, functional scope, and evaluation methodology. When evaluating the area–time product metric, our single-core design attains an area–time product of 30.03 kilo gate milliseconds, the minimum value across all five benchmarked architectures, which translates to a nearly 68.1% reduction in area–time product compared to Ref. [9]. This outcome demonstrates a well-balanced trade-off between hardware area overhead and computation latency. Moreover, the 16-core parallel computing platform implemented in this work can sustain an overall throughput of 1.03 million signatures per second.

It must be reiterated that all side-channel security claims in this paper are based exclusively on simulation results using an idealized Hamming-weight power model. These results validate the mathematical construction of the masking and blinding countermeasures but do not constitute formal security proof or practical certification for physical operating conditions. Comprehensive TVLA physical measurement on the fabricated chip is planned as mandatory future work to validate the preliminary simulation findings.

Although the proposed design achieves high signature throughput within the scope of the compared designs, the thirty-two 256×512 precomputation SRAMs introduce considerable on-chip storage area overhead, which restricts its deployment on ultra-lightweight IoT edge terminals. In addition, the embedded first-order masking scheme is designed to defend against first-order side-channel attacks; however, its practical effectiveness against higher-order attacks and under physical measurement conditions remains to be verified.

Future work will focus on compressing precomputed elliptic point storage to reduce area cost, and exploring low-overhead second-order masking architectures to further strengthen side-channel security robustness, subject to validation through physical chip testing.

Author Contributions: Conceptualization, J.H. and M.-F.Z.; Methodology, M.-F.Z.; Software, M.-F.Z.; Validation, M.-F.Z. and Z.-N.X.; Formal analysis, M.-F.Z. and Z.-N.X.; Investigation, M.-F.Z. and Z.-N.X.; Resources, M.-F.Z.; Data curation, M.-F.Z.; Writing—original draft, M.-F.Z.; Writing—review & editing, J.H., M.-F.Z. and Z.-N.X.; Supervision, J.H.; Project administration, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Miller, V.S. Use of Elliptic Curve in Cryptography. In *Advances in Cryptology—CRYPTO 1985*; Springer: Heidelberg, Germany, 1986; pp. 417–426. [\[CrossRef\]](#) [\[PubMed\]](#)
2. Koblitz, N. Elliptic Curve Cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [\[CrossRef\]](#)
3. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [\[CrossRef\]](#)
4. GM/T 0003-2020; State Cryptography Administration of China. SM2 Elliptic Curve Public Key Cryptography Algorithm. China Standards Press: Beijing, China, 2020.
5. Wang, T.F.; Zhang, H.F.; Xu, S. Design and Implementation of SM2 Special Instruction Coprocessor. *Comput. Eng. Appl.* **2022**, *58*, 102–109. [\[CrossRef\]](#)
6. Satoh, A.; Takano, K. A Scalable Dual-Field Elliptic Curve Cryptographic Processor. *IEEE Trans. Comput.* **2003**, *52*, 449–460. [\[CrossRef\]](#)
7. Chen, G.; Bai, G.; Chen, H. A High-Performance Elliptic Curve Cryptographic Processor for General Curves over GF(p) Based on a Systolic Arithmetic Unit. *IEEE Trans. Circuits Syst. II Express Briefs* **2007**, *54*, 412–416. [\[CrossRef\]](#)
8. Ding, F. Hardware Design of Digital Signature System Based on SM2. Master's Thesis, Southeast University, Nanjing, China, 2015. [\[CrossRef\]](#)
9. Chen, Y.L.; Lee, J.W.; Liu, P.C.; Chang, H.C.; Lee, C.Y. A Dual-Field Elliptic Curve Cryptographic Processor with a Radix-4 Unified Division Unit. In *Proceedings of the 2011 IEEE International Symposium on Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011*; IEEE: Piscataway, NJ, USA, 2011; pp. 713–716. [\[CrossRef\]](#)
10. Guneyusu, T.; Paar, C. Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In *Cryptographic Hardware and Embedded Systems—CHES 2008*; LNCS 5154; Springer: Berlin, Germany, 2008; pp. 62–78. [\[CrossRef\]](#)
11. Li, M.; Li, N.; Liu, H.; Cheng, S.; Hu, X.; Li, J. Implementation of a Safe and Efficient Point Multiplication for SM2 Algorithm. In *Proceedings of the 6th IEEE Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 16–18 June 2023*; IEEE: Piscataway, NJ, USA, 2023; Volume 6, pp. 137–141. [\[CrossRef\]](#)
12. Kong, T. FPGA Implementation of a High-Performance SM2 Coprocessor. In *Proceedings of the 3rd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), Chongqing, China, 19–21 May 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 174–179. [\[CrossRef\]](#)
13. Xu, X.; Song, M.; Zeng, X. High-Speed and Secure ECC Processor for Chinese SM2 Using Modified Karatsuba Multiplier Based on FPGA. *IEICE Trans. Commun.* **2025**, *108*, 1030–1037. [\[CrossRef\]](#)
14. Yan, M. Rapid Implementation and Optimization of GPU-Based SM2 Digital Signature and Verification Algorithm. Master's Thesis, Shanghai Jiao Tong University, Shanghai, China, 2020. Available online: <https://www.chndoi.org/Resolution/Handler?doi=10.27307/d.cnki.gsجتu.2020.002136> (accessed on 6 August 2025).
15. Huang, Y.K. High Throughput Optimization and Security Implementation of State Secrets SM2 Algorithm. Master's Thesis, Xidian University, Xi'an, China, 2023. Available online: <https://www.chndoi.org/Resolution/Handler?doi=10.27389/d.cnki.gxadu.2023.003084> (accessed on 24 August 2026).
16. Li, J.L.; Mo, Y.N.; Su, T.; Chen, D.H. Hardware Design of High-Speed Hybrid Encryption System Based on State Secrets Algorithms SM2, SM3, SM4. *Comput. Appl. Res.* **2022**, *39*, 2818–2825+2831. Available online: <https://www.chndoi.org/Resolution/Handler?doi=10.19734/j.issn.1001-3695.2022.02.0072> (accessed on 11 November 2024).

17. Ruan, O.; Chen, J.C.; Mao, H. An Efficient Batch Verification Algorithm for SM2 Digital Signatures. *Comput. Eng. Sci.* **2021**, *43*, 1236–1242. [[CrossRef](#)]
18. GM/T 0004-2012; State Cryptography Administration of China. SM3 Cryptographic Hash Algorithm. China Standards Press: Beijing, China, 2012.
19. Hankerson, D.; Menezes, A.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer: New York, NY, USA, 2004; pp. 93–113. [[CrossRef](#)]
20. Lim, C.H.; Lee, P.J. More flexible exponentiation with precomputation. In *Proceedings of the 14th Annual International Cryptology Conference (CRYPTO '94)*, Santa Barbara, CA, USA, 21–25 August 1994; Springer: Berlin, Germany, 1994; pp. 95–107. [[CrossRef](#)] [[PubMed](#)]
21. Coron, J.S. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Cryptographic Hardware and Embedded Systems—CHES 1999*; LNCS 1717; Springer: Berlin, Germany, 1999; pp. 292–302. [[CrossRef](#)] [[PubMed](#)]
22. Zhou, J.; Yuan, Y.; Qin, F. Research on Side-Channel Attack Mitigation Technology for PCI-E Cryptographic Card Supporting SM2. *Commun. Technol.* **2023**, *56*, 521–529. [[CrossRef](#)]
23. Goodwill, G.; Jun, B.; Jaffe, J.; Rohatgi, P. A Testing Methodology for Side-Channel Resistance Validation. In *Proceedings of the NIST Non-Invasive Attack Testing Workshop*; Gaithersburg, MD, USA, NIST: Gaithersburg, MD, USA, 2011; pp. 115–136. Available online: https://csrc.nist.gov/CSRC/media/Events/Non-Invasive-Attack-Testing-Workshop/documents/08_Goodwill.pdf (accessed on 11 February 2025).
24. Schneider, T.; Moradi, A. Leakage Assessment Methodology: A Clear Roadmap for Side-Channel Evaluations. In *Cryptographic Hardware and Embedded Systems—CHES 2016*; LNCS 9813; Springer: Berlin, Germany, 2015; pp. 495–513. [[CrossRef](#)]
25. GM/T 0009-2012; State Cryptography Administration of China. SM2 Cryptography Algorithm Usage Specification. China Standards Press: Beijing, China, 2012.
26. GM/T 0039-2024; State Cryptography Administration of China. Security Test Requirements for Cryptographic Modules. China Standards Press: Beijing, China, 2024.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.