ISSN: 2366-1313

AN EFFICIENT AND HIGH-SPEED OVERLAP FREE KARATSUBA BASED FINITE FIELD MULTIPLIER FOR FPGA IMPLEMENTATION

[1] GATLA SANDEEP, [2] Mr. B. SHIVA KUMAR
[1] M Tech Student, Dept. Of ECE, Vaagdevi Engineering College Bolikunta, Warangal
[2] Assistant Professor, Dept. Of ECE, Vaagdevi Engineering College Bolikunta, Warangal

Abstract: *There is no kind of electronic communication that does not today include some kind of cryptography technology. Elliptic curve cryptography (ECC), a branch of public-key cryptography, is now the most used technique for using cryptographic protocols. In ECC systems, the operation that requires the greatest space and time is polynomial multiplication. In order to maximize the utilization of field-programmable gate arrays (FPGAs), this research introduces novel hardware architecture for ECC finite-field multipliers. In order to determine the performance criterion, the suggested hardware was implemented on many FPGA devices with different operand sizes. When compared to state-of-the-art works, the proposed method shows design efficiency with a reduced combinational delay and area-delay product.*

Implementation of field-programmable gate arrays (FPGAs), binary polynomial multiplier, finite-field multiplier, Galois field, hardware cryptography, and overlap-free Karatsuba are all terminology that is connected to this issue.

I. INTRODUCTION

Both public-key and symmetric-key encryption techniques are widely used [7, 6]. Thanks to public-key cryptography, all participants in a communication may exchange keys securely without revealing any private information. There is no other method to do this except using digital signatures and the correct key configuration for encrypted communications. Daffier-Hellman, elliptic curve cryptography (ECC), and RSA are only a few of the cryptosystems that rely on public keys [8, 9]. The ECC algorithm is often regarded as the most advanced public-key cryptosystem because of its tiny key size, which enhances security and makes implementation easier [12], [13].When it comes to energy-efficient electrical equipment, hardware cryptography is chosen over software cryptography due to its speed and lower cost. Some examples of cryptographic methods used in these contexts are field-programmable gate arrays (FPGAs), very large scale integration (VLSI), and ECC.

ISSN: 2366-1313

An essential part of the ECC hardware implementation is the finite-field multiplier, which determines the points of the elliptic curve. Throughput and system area are governed by the size and latency of the multiplier. This prompted the development and intensive study of cryptosystem finite-field multipliers [17]-[26].

The Karatsuba algorithm (KA) is a famous way of multiplying numbers [27]. By substituting addition operations for multipliers, this strategy seeks to reduce the number of multipliers.

Unfortunately, due to algorithms' high temporal complexity and KA's iterative nature, processing time increases and efficiency decreases. As a result, while deciding between the KA and CA algorithms, latency and area are also considered. The hardware implementation's constraints dictate whether the Karatsuba approach can be fine-tuned to increase speed or decrease area. Maximizing multipliers' performance is possible with the use of hardware implementation approaches such as pipelines. Titles of pages 29–32.

To lessen the combinational latency of the KA, academics have suggested many methods, one of which is the overlap-free Karatsuba algorithm (OKA) [33]-[35]. The objective of this approach is to remove an XOR gate from the critical path of KA. With this enhancement, the possible latency is much lower than with the original KA technique. However, because to it's no iterative nature, CA achieves better latency performance than OKA.

In this paper, we provide a new hardware approach to efficient multiplication that maintains its performance even while avoiding very large size constraints. The method produces an output comparable to the OKA by using a base unit developed at a lower level in a similar fashion to the earlier technique. The predicted method of multiplication was determined to be

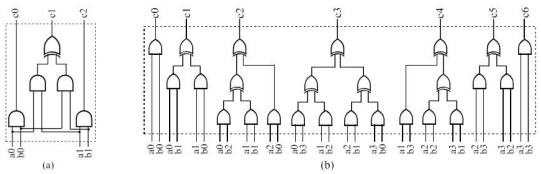

Fig. 1. DFG for hardware implementation of (a) 2- and (b) 4-bit conventional binary polynomial multiplier.

Near the Karatsuba utilizing resources at the same pace as the CA.

Three algorithms were chosen for FPGA implementation: OKA, KA, and the classic method of multiplication. At the end of this round of the research, we get an overview, in terms of area and latency, of the FPGA implementations for various operand sizes. This proves that the CA is the fastest of the three algorithms. However, for operands of a bigger size, more lookup tables are required. In contrast to previous algorithms, the ones that were proposed demonstrated much

ISSN: 2366-1313

higher speed while using significantly less space. The significance of this study is best shown by two key points. We started by comparing the results to theoretical analysis and other methods, and then we evaluated FPGA implementations of overlap-free Karatsuba binary polynomial multiplications for various operand sizes. We also proposed an overlap-free lookup table-based method to obtain a quick and efficient polynomial multiplier.

II BACK GROUND

Polynomial multiplication and modular reduction are two common operations in GF (2n) that have a major impact on the efficiency and cost of the system [30].

Space consumption and total multiplication delay are theoretically utilized to calculate the multiplier's efficiency when utilizing ideal two-input AND XOR gates [30], [33], [36]. For example, the issue of limited gate fan-out in these devices is often ignored in this research, which means that the hardware limits are not considered. We can simply get the multiplier's space and delay by taking the ideal system configuration and adding the usual gate delays and area needs in a linear fashion. These considerations may be irrelevant in theoretical studies, but they become crucial in real-world applications, such as when buffers are necessary due to hardware limitations [37]. This article presents results that have been derived from theoretical and practical sources.

The A. Gold Standard

A brief summary of the CA is given here for binary polynomial multiplication. We continue on to n-bit multipliers after we lay the groundwork with 2- and 4-bit multipliers. In GF (2n), consider two polynomials of degree one, A(x) = a1x + a0 and B(x) = b1x + b0. We do 1-bit addition and multiplication using logical XORs and ANDs, respectively, since we are in GF (2n). For the first-order and 4-bit multiplier examples, respectively, the data flow graphs (DFGs) are shown in images 1(a) and (b).

A simple 2-bit multiplier might be implemented using only one XOR and four AND gates. These are the values that make up a conventional n-bit multiplier:

$$CA_{XOR}(n) = (n-1)^2 \quad (1)$$

$$CA_{AND}(n) = (n)^2 \quad (2)$$

Where (CA_{AND}) and (CA_{AND}) are the total number of AND XOR gates, respectively. Assuming ideal hardware condition and signal strength (no buffers required), the delay of the CA multiplier for the given example in Fig. 1(a) is

$$T_{CA}(2) = T_x + T_a \quad (3)$$

ISSN: 2366-1313

Where it is assumed that T_x and T_a are the delay of an XOR

And an AND gate, respectively.

Generally, the highest delay of CA for multiplying two *n*- bit polynomials happens at term *(n 1)* of output and is equal to

$$T_{CA}(n) = T_a + \log_2(n)\, T_x. \quad (4)$$

Next, the original KA will be briefly reviewed.

A. Karatsuba Algorithm

Since the conventional multiplication method is not the mostefficient method, other methods, such as KA [27] and its variations, have been developed. We will briefly review the original KA in the following.

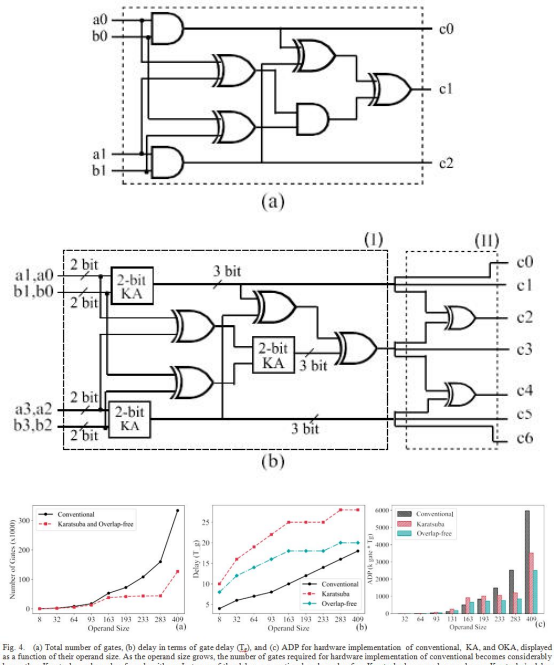

Fig. 4. (a) Total number of gates, (b) delay in terms of gate delay (T_x), and (c) ADP for hardware implementation of conventional, KA, and OKA, displayed as a function of their operand size. As the operand size grows, the number of gates required for hardware implementation of conventional becomes considerably larger than Karatsuba and overlap-free algorithms. In terms of the delay, conventional and overlap-free Karatsuba become closer, whereas Karatsuba's delay rises faster than these two algorithms. Overlap-free has the minimum ADP for operand sizes larger than 163.

III PROPOSED MULTIPLICATION STRATEGY

Here we provide a novel and efficient finite-field multiplier implementation. The suggested implementation approach is derived on research on the theoretical bounds of area and delay for conventional, Karatsuba, and overlap-free systems. A finite-field multiplier of different size is generated using an observed trend as a template. Additionally, two implementation techniques, theoretical gate-based analysis and FPGA, are assessed for their hardware resource needs and combinational latency.

The hardware implementation of the binary polynomial multiplication algorithms is shown in Fig. 4(a) for various operand sizes. Taking into account tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. On the other hand, compared to Karatsuba and overlap-free, the number of gates needed to achieve CA increases dramatically as the operand size increases. To illustrate, compared to the Karatsuba or overlap-free, the CA needs over 163% more gates for an operand size of 409 bits.

The overall combination delay, expressed as gate delays, for all three techniques is shown in Figure 4(b). We started with the

ISSN: 2366-1313

premise that Tx Ta Tg, the delays of the AND and XOR gates, are equivalent. While the CA has the smallest delay in this figure, the delays of conventional and overlap-free Karatsuba converge to almost the same value as the operand size expands. Contrarily, the latency for Krartsuba method increases more rapidly than that of the other two algorithms. The durations of recursive multipliers with 16, 19, and 233 bits are same because

A new and efficient implementation of the finite-field multiplier is given here. Findings from studies of conventional, Karatsuba, and overlap-free systems' theoretical area and delay limitations inform the proposed implementation strategy. Using a trend as a template, a finite-field multiplier of varying sizes is produced. There is also an evaluation of the hardware resource requirements and combinational delay of two implementation strategies, namely FPGA and theoretical gate-based analysis.

Figure 4(a) displays the hardware implementation of the algorithms for binary polynomial multiplication for different sizes of operands. The graphic illustrates that fewer gates are required to implement CAs in comparison to the KA, even when considering the small operand sizes. The number of gates required to accomplish CA, in contrast to Karatsuba and overlap-free, grows substantially with increasing operand size. The CA requires more gates—more than 163% more—to handle an operand size of 409 bits, in comparison to the Karatsuba or overlap-free.

Figure 4(b) shows the total combination delay, here shown as gate delays, for all three methods. The assumption that the AND and XOR delays, Tx Ta Tg, are equal was our starting point. Conventional and overlap-free Karatsuba delays approach one another as the operand size increases, with the CA having the least delay in this figure. Interestingly, compared to the other two algorithms, the latency for the Krartsuba approach grows at a faster rate. There is no difference in the runtimes of recursive multipliers with 16, 19, and 233 bits due to

The number of levels is same for each of them. To further understand the problem, see Figure 5, which depicts the construction of these multipliers.

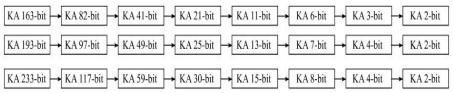

Fig. 5. Recursive Karatsuba polynomial multiplication for operand sizes of 163, 193, and 233 bits. Theoretical VLSI delay for implementation of these algorithms would be the same since they have an equal number of multiplier stages. The same structure was also used to implement an overlap-free algorithm.

ISSN: 2366-1313

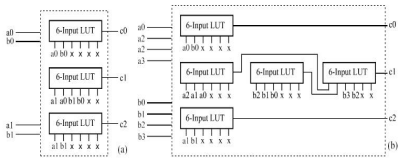

Fig. 6. DFG for FPGA implementation of (a) 2- and (b) 4-bit binary polynomial multipliers using six-input LUTs. Regardless of the different DFGs, FPGA implementation of all methods for 2- and 8-bit multipliers is the same. It is worth noting that this is a simple demonstration and actual architecture and routing is more complex.

Here we provide a novel and efficient finite-field multiplier implementation. The suggested implementation approach is derived on research on the theoretical bounds of area and delay for conventional, Karatsuba, and overlap-free systems. A finite-field multiplier of different size is generated using an observed trend as a template. Additionally, two implementation techniques, theoretical gate-based analysis and FPGA, are assessed for their hardware resource needs and combinational latency.

The hardware implementation of the binary polynomial multiplication algorithms is shown in Fig. 4(a) for various operand sizes. Taking into account tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. On the other hand, compared to Karatsuba and overlap-free, the number of gates needed to achieve CA increases dramatically as the operand size increases. To illustrate, compared to the Karatsuba or overlap-free, the CA needs over 163% more gates for an operand size of 409 bits.

The overall combination delay, expressed as gate delays, for all three techniques is shown in Figure 4(b). We started with the premise that Tx Ta Tg, the delays of the AND XOR gates, are equivalent. While the CA has the smallest delay in this figure, the delays of conventional and overlap-free Karatsuba converge to almost the same value as the operand size expands. Contrarily, the latency for Krartsuba method increases more rapidly than that of the other two algorithms. The durations of recursive multipliers with 16, 19, and 233 bits are same because

A new and efficient implementation of the finite-field multiplier is given here. Findings from studies of conventional, Karatsuba, and overlap-free systems' theoretical area and delay limitations inform the proposed implementation strategy. Using a trend as a template, a finite-field multiplier of varying sizes is produced. There is also an evaluation of the hardware resource requirements and combinational delay of two implementation strategies, namely FPGA and theoretical gate-based analysis.

Figure 4(a) displays the hardware implementation of the algorithms for binary polynomial multiplication for different sizes of operands. The graphic illustrates that fewer gates are required to

ISSN: 2366-1313

implement CAs in comparison to the KA, even when considering the small operand sizes. The number of gates required to accomplish CA, in contrast to Karatsuba and overlap-free, grows substantially with increasing operand size. The CA requires more gates—more than 163% more—to handle an operand size of 409 bits, in comparison to the Karatsuba or overlap-free.

Figure 4(b) shows the total combination delay, here shown as gate delays, for all three methods. The assumption that the AND and XOR delays, Tx Ta Tg, are equal was our starting point. Conventional and overlap-free Karatsuba delays approach one another as the operand size increases, with the CA having the least delay in this figure. Interestingly, compared to the other two algorithms, the latency for the Krartsuba approach grows at a faster rate. There is no difference in the runtimes of recursive multipliers with 16, 19, and 233 bits due to

An updated and more effective version of the finite-field multiplier is shown here. The suggested approach to implementation is based on research into the theoretical area and delay constraints of conventional, Karatsuba, and overlap-free systems. One may generate finite-field multipliers of different sizes by using a trend as a model. Two implementation options, theoretical gate-based analysis and field-programmable gate array (FPGA), are also compared in terms of hardware resource needs and combinational latency.

The methods for binary polynomial multiplication are shown in Figure 4(a) for various operand sizes. Even with the tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. As the operand size increases, the number of gates needed to achieve CA increases significantly, in contrast to Karatsuba and overlap-free. When compared to the Karatsuba or overlap-free, the CA needs more gates—over 163% more—to accommodate an operand size of 409 bits.

Figure 4(b) displays the three approaches' total combination delays, which are shown as gate delays. We began with the premise that the AND XOR delays, denoted as Tx Ta Tg, are equal. In this image, the conventional and overlap-free Karatsuba delays are becoming closer to one other as the operand size grows, but the CA has the shortest delay. Latency increases more rapidly for the Krartsuba method than for the other two methods, which is an interesting finding. For recursive multipliers, the runtimes with 16, 19, and 233 bits are same because

ISSN: 2366-1313

In contrast to linear growth, the number of steps in recursive multipliers like KA and OKA grows logarithmically with the size of the operand. This is an intriguing property. The first four steps in the 233-bit example, for example, multiply up to fifteen-bit multipliers repeatedly. But you'll have to add four more steps if you want to multiply by fifteen bits. Keep in mind that the total of the delays of these multipliers is directly proportional to the number of steps they have.

Figure 4(c) compares and contrasts the outcomes of computing the area-delay product (ADP) for each method. The traditional technique often yielded the greatest ADP, whereas the overlap-free approach yielded the lowest.

Since our attention was directed on FPGAs and not DGBAs, we examined the on-FPGA space and time analysis of these methods and verified their findings by putting them into practice. Lookup tables (LUTs) are used to implement most functions in FPGAs instead of combinational gates. In essence, LUTs are universal gates due to the fact that they may substitute for any function. Implementing these structures on the FPGA with the use of LUTs might lead to more accurate estimations for complexity and delay analysis.

Two FPGA-based binary polynomial multipliers with six input LUTs are shown in Figure 6. The image shows that the LUT-based implementations are the same, even when the number of gates and DFGs vary. As the number of LUTs grows and the operands become larger, the performance gap between these algorithms' LUT implementations becomes more noticeable. This means that it is not enough to simply apply the theoretical estimates used to measure the algorithms' efficiency to their FPGA implementation.

It is possible to predict the time required to construct combinational circuits with the use of LUTs and FPGAs. However, a more realistic strategy would have been to implement each of those algorithms for varying operand sizes using FPGA. Figure 7(a) and (b) show the results of the Vivado synthesizer tool reporting the number of LUTs and combinational delay for the conventional, KA, and OKA implementations on an Artix-7 XC7A200TTFV1156-1 FPGA.

In terms of footprint, almost all algorithms use the same number of LUTs when operand sizes are minimal. The area difference, however, grows nonlinearly as operand sizes increase. As an example, consider a 283-bit traditional multiplier; FPGAs are used 69% more often

ISSN: 2366-1313

compared to KAs. Compared to the Karatsuba approach, 409 bit requires almost twice as many LUTs. It is anticipated that the disparity would widen with increasing operand sizes.

When compared to Karatsuba, the overlap-free technique uses a somewhat larger number of LUTs. A 409-bit overlap-free approach requires more LUTs than the Karatsuba method, which is more than 2.7% more.

The average combinational latency achieved by the CA is more than 44% lower than that of Karatsuba. Taking into account the absence of overlap reduces this figure by around 36%. For small operand sizes, the overlap-free and Karatsuba delays are strongly related. For example, the overlap-free approach is almost 14% quicker for a 409-bit multiplier; nevertheless, the performance disparity becomes much more noticeable with bigger operands.

With a few tweaks for better readability, Figure 7 displays the outcomes of using FPGAs, which are comparable to the theoretical values given in Figure 4. Both the Karatsuba and overlap-free methods utilize fewer resources than the traditional approach while being almost indistinguishable from it. When comparing theoretical and hardware implementations, the trend for space complexity is rather comparable.

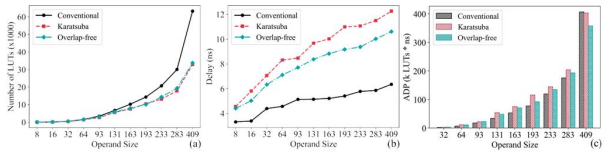

Fig. 7. (a) Number of LUTs, (b) combinational delay, and (c) ADP for FPGA implementation of bit-parallel binary polynomial multipliers using different algorithms. CA has the maximum LUTs and minimum delay. Overlap-free's resource utilization is close to Karatsuba, while it has a relatively higher speed. ADP for conventional method is smaller than other algorithms for all operand sizes except for 409 operand size where overlap-free the minimum ADP has.

TABLE I

NUMBER OF LUTS AND COMBINATIONAL DELAY FOR FPGA IMPLEMENTATION OF THE PROPOSED BIT-PARALLEL OVERLAP-FREE KARATSUBA-BASED MULTIPLICATION STRATEGY FOR DIFFERENT TRANSITION LEVELS. MINIMUM OF ADP FOR EACH OPERAND SIZE IS HIGHLIGHTED WITH A BOLD FONT

n	Level 1			Level 2			Level 3			Level 4		
	LUT	Delay	ADP	LUT	Delay	ADP	LUT	Delay	ADP	LUT	Delay	ADP
163	9307	5.786	53850	7996	6.460	51654	7102	6.832	**48520**	6634	7.565	50186
193	11786	6.811	80887	11640	6.683	77790	9683	6.887	**66686**	8843	7.780	68798
233	16814	6.841	115024	16797	7.475	125557	12349	6.930	**85578**	12439	6.930	86202
283	24251	6.846	166022	21596	7.155	154519	18746	7.662	**143631**	16475	8.720	143662
409	49211	6.856	337390	40199	8.085	325008	32880	7.889	**259390**	32361	9.170	296750
537	84199	7.524	633513	66957	8.098	542217	59863	8.559	51367	54177	8.906	**482500**
617	110119	7.528	828975	87007	8.106	705278	77623	8.556	664142	69967	9.405	**658039**

Here we provide a novel and efficient finite-field multiplier implementation. The suggested implementation approach is derived on research on the theoretical bounds of area and delay for conventional, Karatsuba, and overlap-free systems. A finite-field multiplier of different size is generated using an observed trend as a template. Additionally, two implementation techniques, theoretical gate-based analysis and FPGA, are assessed for their hardware resource needs and combinational latency.

The hardware implementation of the binary polynomial multiplication algorithms is shown in Fig. 4(a) for various operand sizes. Taking into account tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. On the other hand,

ISSN: 2366-1313

compared to Karatsuba and overlap-free, the number of gates needed to achieve CA increases dramatically as the operand size increases. To illustrate, compared to the Karatsuba or overlap-free, the CA needs over 163% more gates for an operand size of 409 bits.

The overall combination delay, expressed as gate delays, for all three techniques is shown in Figure 4(b). We started with the premise that Tx Ta Tg, the delays of the AND XOR gates, are equivalent. While the CA has the smallest delay in this figure, the delays of conventional and overlap-free Karatsuba converge to almost the same value as the operand size expands. Contrarily, the latency for Krartsuba method increases more rapidly than that of the other two algorithms. The durations of recursive multipliers with 16, 19, and 233 bits are same because

The traditional technique provides the quickest results for theoretical gate-based analysis and FPGA implementation in terms of latency, followed by the overlap-free and Karatsuba methods. Furthermore, the overlap-free latency is near the Karatsuba on FPGA, although it is closer to the CA in theoretical calculations.

On top of that, Figure 7(c) shows the ADP graphs for all three techniques. According to the findings, when the operand size is less than 409 bits, the ADP for the traditional approach is less compared to the other ways. In terms of numerical efficiency, a 283-bit multiplier implemented using the traditional way outperforms Karatsuba by 14% and the overlap-free method by 9%. For a multiplier of 93 bits, these are the numbers 64% and 66% in that order.

For lower operand sizes, the trend suggests that the CA is the most efficient approach. Also, keep in mind that overlap-free outperforms the KA when the operand size is greater than 93 bits. When dealing with bigger operand sizes, the overlap-free technique is likely to continue to be the most efficient. A hybrid technique might be proposed to achieve finite-field multiplication, as the efficiency is still leaning toward the overlap-free method for large operand sizes.

Figure 8 shows the DFG for the OBS technique, which is a suggested overlap-free multiplication algorithm. Based on the overlap-free, the maximum level is reached. At the first level, however, the traditional method is employed.

ISSN: 2366-1313

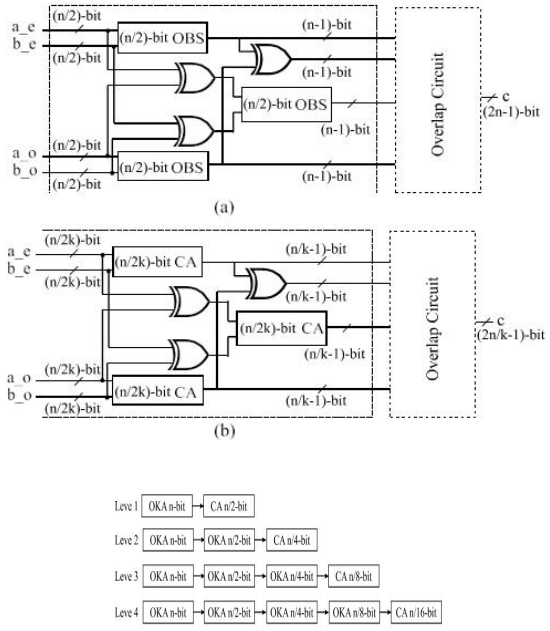

Fig. 9. Structure of the proposed multiplier where overlap-free transits to conventional at different levels. Choosing the transition level is a tradeoff between area and delay.

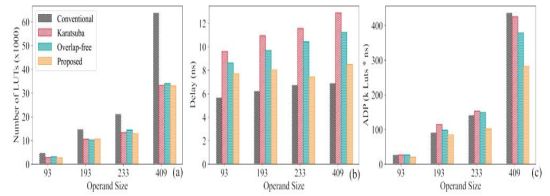

Fig. 10. (a) Number of LUTs and (b) combinational delay for FPGA implementation of bit-parallel finite-field multiplier using different algorithms. The number of LUTs for FPGA implementation of the proposed OBS is lower than conventional and is close to Karatsuba and overlap-free. However, in terms of delay, it is nearly as fast as CA.

TABLE II

AREA–DELAY IMPROVEMENT PERCENTAGE OF THE PROPOSED OVERLAP-FREE-BASED MULTIPLIER COMPARING TO OTHER ALGORITHMS

n	Conventional	Karatsuba	Overlap-free Karatsuba
93	23.8	23.7	24.0
193	5.2	25.0	12.2
233	32.8	38.5	36.5
409	36.1	34.5	26.4

TABLE III

SPEED CHANGE PERCENTAGE OF THE PROPOSED ALGORITHM WITH REGARD TO OTHER ALGORITHMS

n	Conventional	Karatsuba	Overlap-free Karatsuba
93	-36.1	19.8	10.8
193	-29.1	26.8	17.2
233	-10.4	35.8	28.9
409	-23.6	34.2	24.3

Implementation of the multipliers. As an example, a 283-bit proposed multiplier with a CA at level 4 utilizes 32% less LUTs compared to that of the CA at level 1.

On the other hand, delay for level 4 is 27% higher than that for level 1. Therefore, choosing the transition level is tradeoffs between area and delay. In this work, the level that resulted in the minimum value of ADP (highlighted with a bold font in Table I) was selected to construct the proposed multipliers for each operand size. The most optimum transition level may varywith the operand size as in this table for a 537-bit multiplier level 4 has the minimum ADP, while for other sizes, transitionto CAs at level 4 was the most efficient.

IV RESULTS AND DISCUSSION

Here we provide a novel and efficient finite-field multiplier implementation. The suggested implementation approach is derived on research on the theoretical bounds of area and delay for conventional, Karatsuba, and overlap-free systems. A finite-field multiplier of different size is generated using an observed trend as a template. Additionally, two implementation techniques, theoretical gate-based analysis and FPGA, are assessed for their hardware resource needs and combinational latency.

The hardware implementation of the binary polynomial multiplication algorithms is shown in Fig. 4(a) for

ISSN: 2366-1313

various operand sizes. Taking into account tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. On the other hand, compared to Karatsuba and overlap-free, the number of gates needed to achieve CA increases dramatically as the operand size increases. To illustrate, compared to the Karatsuba or overlap-free, the CA needs over 163% more gates for an operand size of 409 bits.

The overall combination delay, expressed as gate delays, for all three techniques is shown in Figure 4(b). We started with the premise that Tx Ta Tg, the delays of the AND XOR gates, are equivalent. While the CA has the smallest delay in this figure, the delays of conventional and overlap-free Karatsuba converge to almost the same value as the operand size expands. Contrarily, the latency for Krartsuba method increases more rapidly than that of the other two algorithms. The durations of recursive multipliers with 16, 19, and 233 bits are same because

A new and efficient implementation of the finite-field multiplier is given here. Findings from studies of conventional, Karatsuba, and overlap-free systems' theoretical area and delay limitations inform the proposed implementation strategy. Using a trend as a template, a finite-field multiplier of varying sizes is produced. There is also an evaluation of the hardware resource requirements and combinational delay of two implementation strategies, namely FPGA and theoretical gate-based analysis.

Figure 4(a) displays the hardware implementation of the algorithms for binary polynomial multiplication for different sizes of operands. The graphic illustrates that fewer gates are required to implement CAs in comparison to the KA, even when considering the small operand sizes. The number of gates required to accomplish CA, in contrast to Karatsuba and overlap-free, grows substantially with increasing operand size. The CA requires more gates—more than 163% more—to handle an operand size of 409 bits, in comparison to the Karatsuba or overlap-free.

Figure 4(b) shows the total combination delay, here shown as gate delays, for all three methods. The assumption that the AND XOR delays, Tx Ta Tg, are equal was our starting point. Conventional and overlap-free Karatsuba delays approach one another as the operand size increases, with the CA having the least delay in this figure. Interestingly, compared to the other two algorithms, the latency for the Krartsuba approach grows at a faster rate. Recursive multipliers with 16, 19, and 233 bits all have the same durations since this

ISSN: 2366-1313

section covers the results of employing the proposed approach and comparing them to other relevant research in this field.

A hardware multiplier that utilizes approaches that avoid overlap

Theoretical limits for area, latency, and ADP were often computed to ascertain an algorithm's performance. It becomes evident upon closer study that this may not be true when applied to an FPGA. Since the consumption of FPGA hardware is reliant on LUTs, such theoretical analysis needs to be revised. The proposed method is based on LUT implementation, the core component of the FPGA; therefore the estimates are also more accurate in terms of actual performance and cost.

Binary polynomial multiplication followed by a modular reduction is a common way to build a finite-field multiplier. You can see the relative performance and amount of resources used by several methods in Figure 10(a) and (b). An FPGA (Artix-7 XC7A200TTFV1156-2) was used to implement these algorithms that were developed using irreducible trinomials. There is a wide variety of multipliers, from 93 to 409 bits.

The proposed approach is much closer to KAs, utilizes a fraction of the resources, and is almost as fast as the standard procedure (Fig. 10). Nevertheless, the following delves into the effectiveness, the principal advantage of the proposed approach.

Figure 10(c) shows that compared to other methods, the ADP of the proposed method are the lowest. As compared to the alternative approaches, Tables II and III summarize the ADP and speed improvements. As a whole, the proposed method achieves 25% better ADP performance than the conventional algorithm, 31% better than the Karatsuba method, and 25% better than the overlap-free algorithm.

TABLE IV

NUMBER OF LUTS AND DELAY FOR FPGA IMPLEMENTATION OF GF(2^{233}) FINITE-FIELD MULTIPLIERS ON DIFFERENT DEVICES USING VARIOUS SYNTHESIS TOOLS

Device Synthesis Tool	Virtex-5 XC6SLX75 XILINX ISE		Zynq XC7Z030 XILINX ISE		Spartan-7 XC7S100 XILINX Vivado		Cyclone-IV EP4CGX150DF Intel Quartus Prime	
Method	LUTs	Delay	LUTs	Delay	LUTs	Delay	LUTs	Delay
Proposed	14275	9.144	14275	4.955	12720	4.929	18901	7.633
Conventional	22821	6.319	22821	4.168	20920	4.146	37142	7.191
Karatsuba	17613	15.342	17644	7.756	13315	6.868	19814	10.792
Overlap-free Karatsuba	21376	14.116	21359	7.345	14255	6.340	20830	9.783

TABLE V

COMPARING FPGA RESOURCE UTILIZATION AND DELAY OF THE PROPOSED MULTIPLIER WITH THOSE OF RELEVANT WORKS

Reference	Algorithm	Slices	LUTs	Delay (ns)	ADP	n	Device & Tool	I/O Buffer	Reduction
Samanta et al. [38]	Modified Karatsuba	36	62	13.95	1367	8	Spartan-3E XC3S100E	Yes	No
This work	OBS	24	46	11.01	770		Xilinx ISE		
Zhou et al. [39]	Modified Karatsuba	3320	11404	7.68	113080	233	Virtex-5 XC5VLX50	No	Yes
This work	OBS	3857	11476	6.61	101351		Xilinx ISE		
J. Imaña [40]	Imaña's method	2354	5501	20.56	161498	113	ARTIX-7 XC7A200T	No	Yes
This work	OBS	1084	3792	10.52	51295		Xilinx Vivado		
Xie et al. [35]	Digit-serial Karatsuba	NA	1420	49.5	70290	233	Stratix II EP2S180F	No	Yes
This work	OBS	1872	17738	9.39	166559		Quartus Prime		
Arish et al. [41]	Karatsuba-Urdhva	972	1018	13.00	25870	24	Virtex-4	Yes	No
This work	OBS	184	360	12.51	6805		Xilinx ISE		
Rashidi et al. [42]	Pipelined Bit-parallel	21195	36812	7.19	3702090	233	Virtex-4 XC4VLX200	No	Yes
This work	OBS	1147	19804	8.29	173683		Xilinx ISE		

Here we provide a novel and efficient finite-field multiplier implementation. The suggested implementation approach is derived on research on the theoretical bounds of area and delay for conventional, Karatsuba, and overlap-free systems. A finite-field multiplier of different size is generated using an observed trend as a template. Additionally, two implementation techniques, theoretical gate-based analysis and FPGA, are

ISSN: 2366-1313

assessed for their hardware resource needs and combinational latency.

The hardware implementation of the binary polynomial multiplication algorithms is shown in Fig. 4(a) for various operand sizes. Taking into account tiny operand sizes, the figure shows that fewer gates are needed to implement CAs compared to the KA. On the other hand, compared to Karatsuba and overlap-free, the number of gates needed to achieve CA increases dramatically as the operand size increases. To illustrate, compared to the Karatsuba or overlap-free, the CA needs over 163% more gates for an operand size of 409 bits.

V CONCLUSION

A new finite-field multiplier is suggested in this paper. We compared the suggested method's performance metrics with those of other algorithms after implementing it on FPGA for varying operand sizes. On average, the suggested strategy outperformed Karatsuba and the OKA by 30% and 20%, respectively, according to the results of the implementation. Quicker than Karatsuba, 4% smaller than overlap-free Karatsuba, and 43% smaller than the CA, all while using 1% less land. The design outperforms traditional, Karatsuba, and OKA by 25%, 30%, and 25%, respectively, according to ADP comparisons. By comparing with state-of-the-art works, it was found that the design is more efficient, with greater speed and lower ADP.

REFERENCES

[1] R. Abu-Salma, M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith, "Obstacles to the adoption of secure communication tools," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 137–153.

[2] B. Vembu, A. Navale, and S. Sadhasivan, "Creating secure communication channels between processing elements," U.S. Patent 9 589 159, Mar. 7, 2017.

[3] J. Yoo and J. H. Yi, "Code-based authentication scheme for light-weight integrity checking of smart vehicles," *IEEE Access*, vol. 6, pp. 46731–46741, 2018.

[4] K. Shahbazi and S. B. Ko, "Area-efficient nano-AES implementation for Internet-of-Things devices," *IEEE Trans. Very Large Scale Integer. (VLSI) Syst.*, vol. 29, no. 1, pp. 136–146, Jan. 2021.

[5] P. Aparna and P. V. V. Kishore, "Biometric-based efficient medical image watermarking in E-healthcare application," *IET Image Process.*, vol. 13,no. 3, pp. 421–428, Feb. 2019.

ISSN: 2366-1313

[6] S. Raza, L. Seitz, D. Sitenkov, and G. Selander, "S3K: Scalable security with symmetric keys—DTLS key establishment for the Internet of Things," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1270–1280, Jul. 2016.

[7] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public-key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[8] A. Faz-Hernandez, F. Rodriguez-Henriquez, E. Ochoa-Jimenez, and J. Lopez, "A faster software implementation of the super singular isogenies Daffier-Hellman key exchange protocol," *IEEE Trans. Compute.*, vol. 67, no. 11, pp. 1622–1636, Nov. 2018.

[9] X. Zhou and X. Tang, "Research and implementation of RSA algorithm for encryption and decryption," in *Proc. 6th Int. Forum Strategic Technol.*, vol. 2, Aug. 2011, pp. 1118–1121.

[10] F.-Y. Rao, "On the security of a variant of ElGamal encryption scheme," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 4, pp. 725–728, Jul. 2019.

[11] Z. U. A. Khan and M. Benaissa, "High-speed and low-latency ECC processor implementation over GF (2^m) on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 165–176, Jan. 2017.

[12] F. Mallouli, A. Hellal, N. S. Saeed, and F. A. Alzahrani, "A survey on cryptography: Comparative study between RSA vs ECC algorithms, and RSA vs El-Gamal algorithms," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (Edge Com)*, Jun. 2019, pp. 173–176.