Volume 9, Issue 5, May – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY261

FPGA Based Accelerator for Implementation of Large Integer Polynomials

Dr. J. Kamala
Department of ECE
Anna University Chennai, India

M. V. Tejendra Prasad
Department of ECE Anna
University Chennai, India

Abstract:- A 13-bit multiplier is implemented on the Artix-7 100T FPGA using a divide-and-conquer algorithm. The design is coded in SystemVerilog, leveraging its powerful features for hardware description and synthesis. The divide-and-conquer approach breaks down the multiplication task into smaller sub- tasks, enhancing efficiency and reducing complexity. The FPGA's high-performance capabilities, particularly on the Artix-7 100T board, make it well-suited for accelerating the computations involved. Additionally, Area Delay Product (ADP) tools are employed to evaluate the algorithm's efficiency. This project aims to showcase the synergy between algorithmic design, hardware implementation, and FPGA capabilities, emphasizing the versa- tility of the Artix-7 100T FPGA in handling complex arithmetic operations.

Keywords:- *Multiplier, Verilog, FPGA, Areadelayprod-UCT(ADP).*

I. INTRODUCTION

In the realm of cryptography, the choice of a 13-bit multiplier holds particular significance, serving as a foundational operation within cryptographic algorithms and protocols. This specific multiplier implies that the operands involved in the multiplication process are each 13 bits in length, representing a careful consideration of bit size in cryptographic computations.[6]

Multiplication, as a core mathematical operation, plays a crucial role in various cryptographic processes such as key generation, encryption, and signature schemes. The decision to employ a 13-bit multiplier is strategic, striking a delicate balance between computational complexity and resource efficiency. This balance is pivotal in ensuring that cryptographic algorithms perform optimally without sacrificing computational speed or requiring excessive resources.[6]

The application of a 13-bit multiplier is notably prevalent in cryptographic systems that involve modular arithmetic and finite field operations. These operations are fundamental to many cryptographic algorithms, including those based on elliptic curve cryptography, where finite field multiplication is a key component. The 13-bit size is carefully chosen to align with the specific requirements of these cryptographic schemes, contributing to their security and efficiency.[7]

The use of a 13-bit multiplier in cryptography reflects a thoughtful consideration of the intricate interplay between security needs and practical implementation constraints. Whether in hardware or software, the choice of bit size in multiplication is a critical decision, and the 13-bit multiplier exemplifies a meticulous approach to achieving a harmonious blend of computational effectiveness and resource optimization within the cryptographic domain.[7]

II. RELATED WORKS

The systolic implementation of the Karatsuba algorithm (KA)-based digit-serial multiplier over GF(2m) on FPGA platforms. The primary focus is on addressing the high register-complexity issues associated with existing designs. The proposed approach introduces a novel KA-based algorithm that significantly reduces computational complexity. Furthermore, efficient register minimization techniques, including redundant register removal, two-stage pipelining,and register sharing, are proposed to mitigate the register complexity of the suggested structure. The study also employs an FPGA-specific digit-parallel implementation strategy to optimize area, time, and power complexities. Comparativeresults with existing designs, particularly using NIST- recommended polynomials, highlight substantial reductionsin area-delay product (ADP) and power-delay product (PDP). The proposed multiplier demonstrates superior performanceon FPGA platforms and holds promise for applications in resource-constrained platforms such as wearable devices and deeply embedded systems[1]

A lightweight, FPGA-based hardware implementation for polynomial multiplication, addressing a key bottleneck in the NTRU public-key cryptographic scheme. Focused on IoT applications, the proposed constant-time implementation with optimized hardware consumption utilizes a single-step multiplexer-based iterative architecture. By eliminating the need for a modular arithmetic unit and replacing it with an accumulator, substantial device resource savings are achieved. Experimental results on an FPGA demonstrate an impressive 2.86× reduction in area and a 1.23× increase in throughput compared to state-of-the-art strategies. The proposed architecture, resilient against timing attacks, proves well-suited for IoT applications, showcasing constant execution time and significant area reduction, outperforming existing works in terms of area utilization. Future work aims to extend this approach to complete NTRU encryption and decryption module[2]

The vulnerability of the baseline schoolbook polynomial multiplication architecture in lattice-based cryptography to power side-channel leakage. The authors conduct power analysis on the FPGA implementation, identifying substantial power leakage in the R-LWE crypto-system. The study emphasizes the need for countermeasure strategies due to the susceptibility of naive implementations to side-channel attacks.Future work aims to explore SCA-countermeasure strategies and assess their effectiveness, considering power and timing analyses, along with area-delay product (ADP) evaluations.[3]

This paper focuses on enhancing the vulnerability-prone schoolbook polynomial multiplication architecture in lattice-based cryptography by addressing power side-channel leakage. Through power analysis on FPGA, significantpower leakage is identified in the R-LWE crypto-system, emphasizing the need for countermeasure strategies. Future work aims to explore SCA-countermeasure effectiveness, incorporating power and timing analyses, and evaluating area-delay product (ADP). In a related context, the widelyused Schoolbook algorithm in mainstream post-quantum cryptography is optimized by leveraging Toeplitz matrix features. Implemented with the Saber architecture on FPGA, the proposed multiplier exhibits a notable 3.33× higher throughput and 1.58× higher throughput-per-slice compared to state-of-the-art implementations. These resultsunderscore the advantages of the proposed multiplier for high-performance post-quantum cryptography hardware implementations.[4]

KaratSaber introduces an optimized Karatsuba polynomial multiplier for Saber, achieving a 7.47× speed improvement over SPMA Saber and 2.11× higher throughput than LWRPro,with a 46.04 percent efficiency gain. This FPGA-based architecture sets new speed records for Saber polynomial multiplication. Future work includes extending KaratSaber for other Lattice-Based Cryptography schemes and optimizing energy consumption for IoT on ASIC platforms. The proposed improvements position KaratSaber as a notable advancement in FPGA-based polynomial multiplication cores for Saber, a NIST PQC Round 3 KEM scheme.[5]

III. METHODS

A. Divide and Conquer Algorithm

The Divide and Conquer algorithm for integer multiplicationis a transformative approach that optimizes the computation of large integer products. This method strategically divides the numbers into smaller segments, significantly reducing the number of necessary multiplications. Throughout this project, we explore the fundamental principles, advantages, and implementation considerations of this Divide and Conquer algorithm. Its efficiency in managing extensive integer multiplication tasks positions it as a key strategy in algorithmic design, offering notable improvements over traditional multiplication methods. [1]

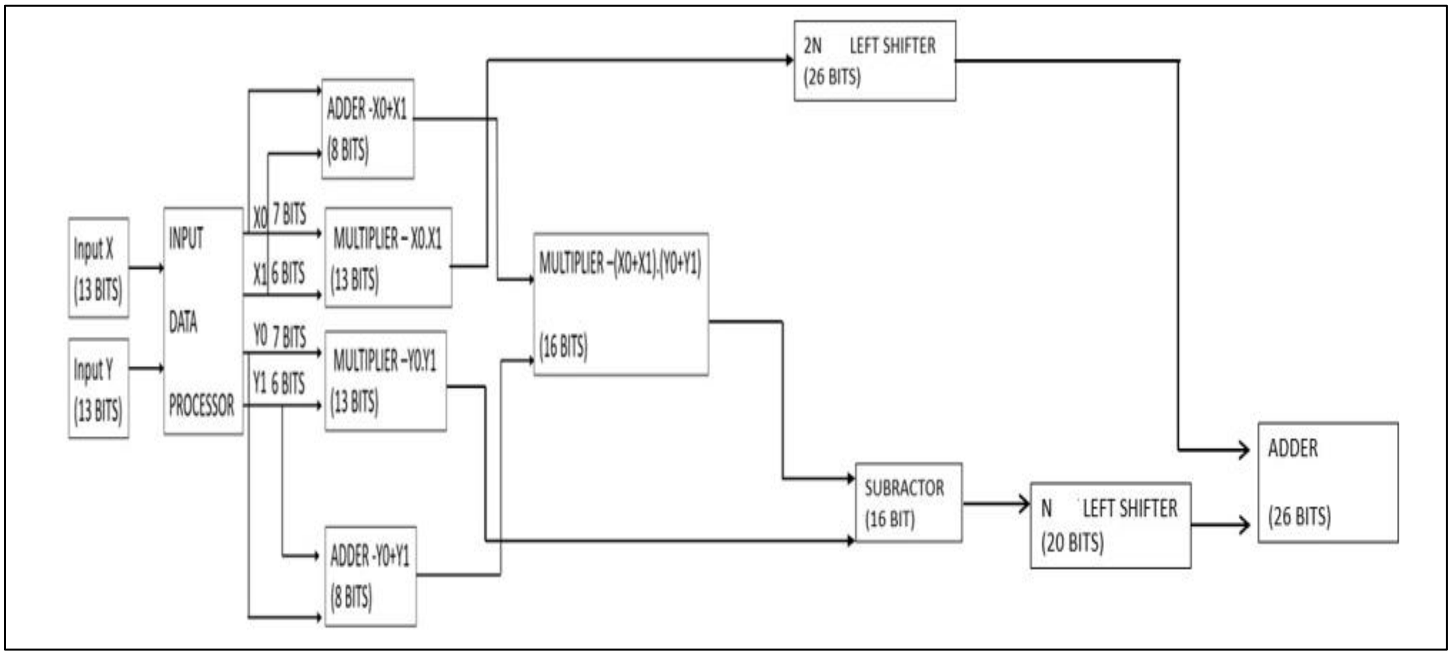


Fig 1: Divide and Conquer Algorithm Flow

The divide and conquer algorithm is a fast multiplication algorithm that divides numbers into smaller chunks and recursively multiplies them to achieve a more efficient multiplication process. In the context of 13-bit multiplication using the Karatsuba algorithm, each 13-bit number is divided into two 6.5-bit parts. The algorithm then performs three recursive multiplications instead of four traditional ones. This reduction in subproblems improves the overall computational efficiency. The process involves calculating three partial products, combining them with appropriate shifts, and performing additions to obtain the final product. Despite its simplicity, the Karatsuba algorithm showcases the power of divide-and-conquer techniques, optimizing multiplication for relatively small bit lengths like 13 bits. [1]

B. School Book Multiplication

Multiplying 13-bit numbers involves multiplying two binarynumbers, each consisting of 13 bits (binary digits). In binary multiplication, the process is similar to decimal multiplication,but it only involves the digits 0 and 1. The multiplication is carried out bit by bit, starting from the rightmost bit (the least significant bit) and progressing towards the left.// For each bit position, the product is calculated by multiplyingthe corresponding bits of the two numbers and consideringany carry from the previous step. The partial products arethen added together to obtain the final result. The result may exceed 13 bits, so overflow bits need to be managed accordingly. multiplying 13-bit numbers involves a systematic process of multiplying individual bits, managing carries, and summing up partial products to arrive at the correct binary representation of the product of the two original numbers. This process ensures an accurate result within the constraints of the 13-bit representation.

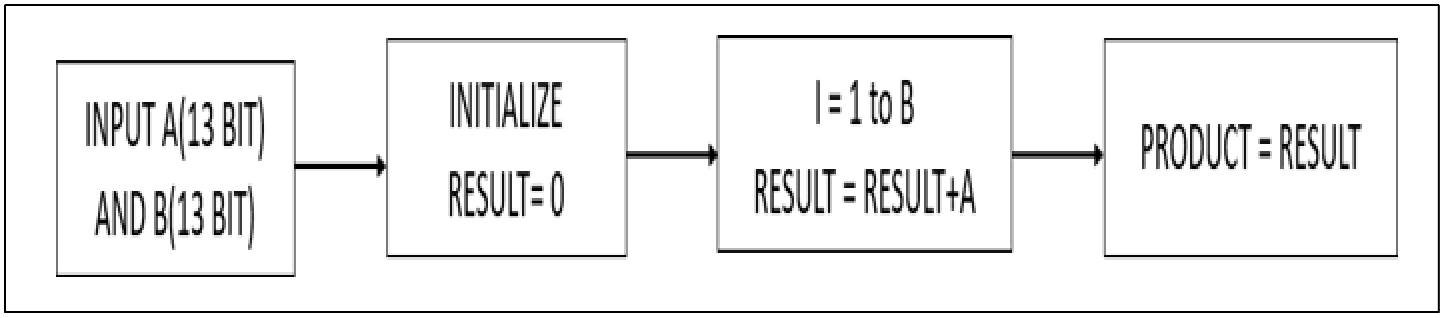


Fig 2: School Book Algorithm Flow

IV. RESULT

A. Output

The algorithm successfully produces for multiplying num-bers from 0 to 8191.

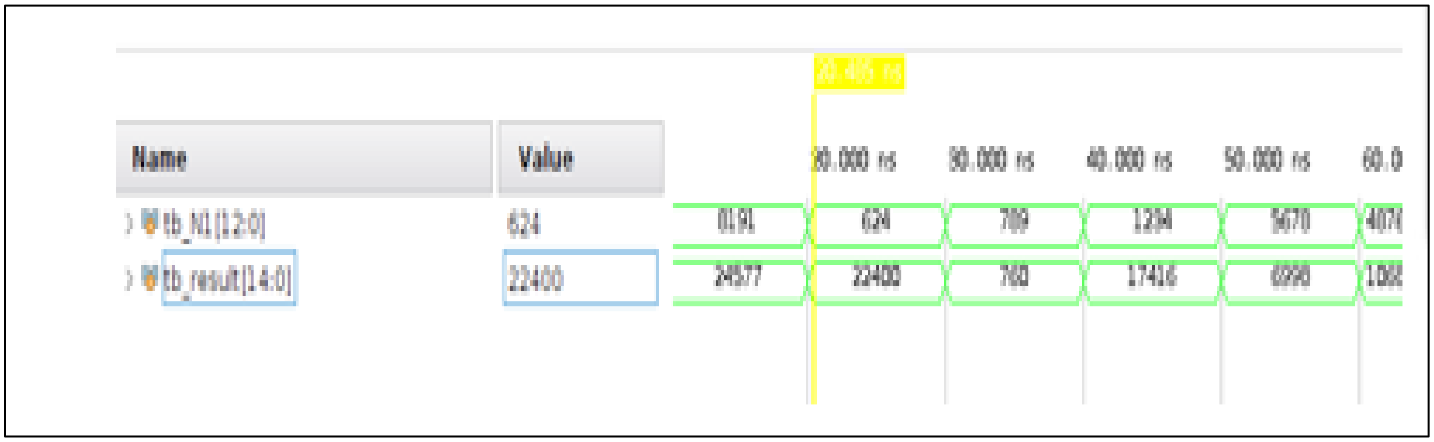
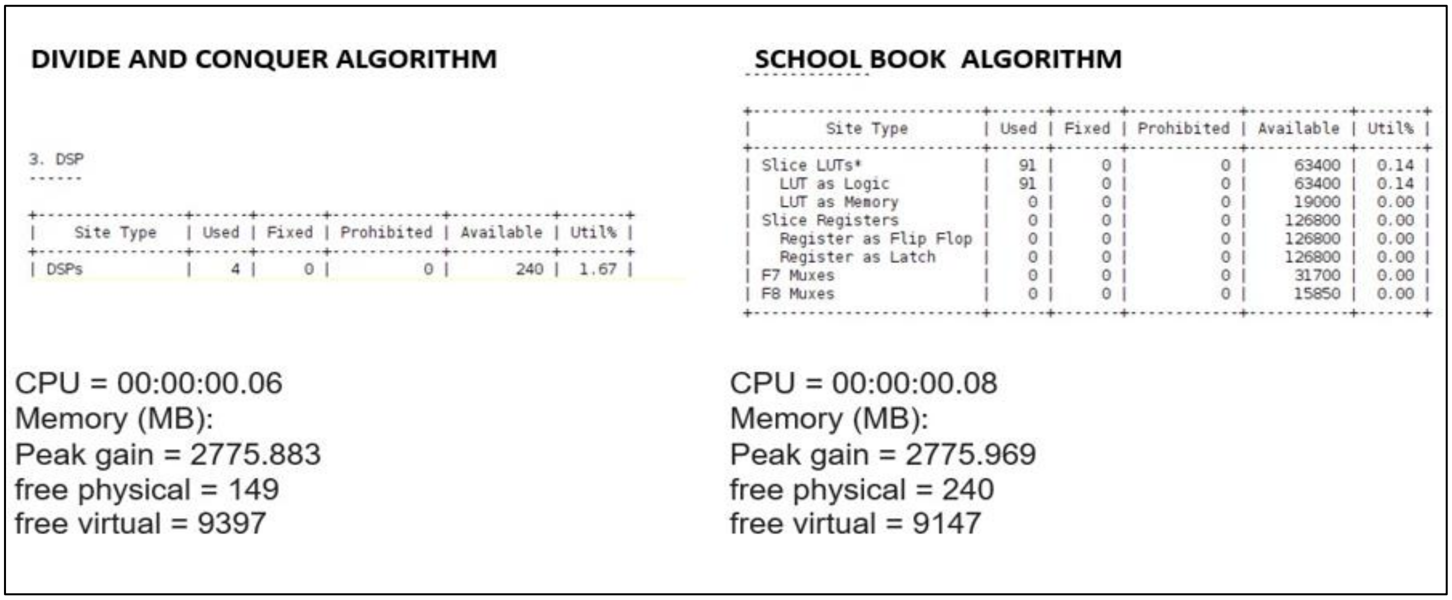


Fig 3: Simulation Output

B. Synthesis

Divide and conquer algorithm uses only 4 DSP's whereas the School book polynomial algorithm consumes 91 LUT's.



DIVIDE AND CONQUER ALGORITHM

```
3. DSP
------

+-----------+------+-------+------------+-----------+-------+
|  Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs       |    4 |     0 |          0 |       240 |  1.67 |
```

CPU = 00:00:00.06
Memory (MB):
Peak gain = 2775.883
free physical = 149
free virtual = 9397

SCHOOL BOOK ALGORITHM

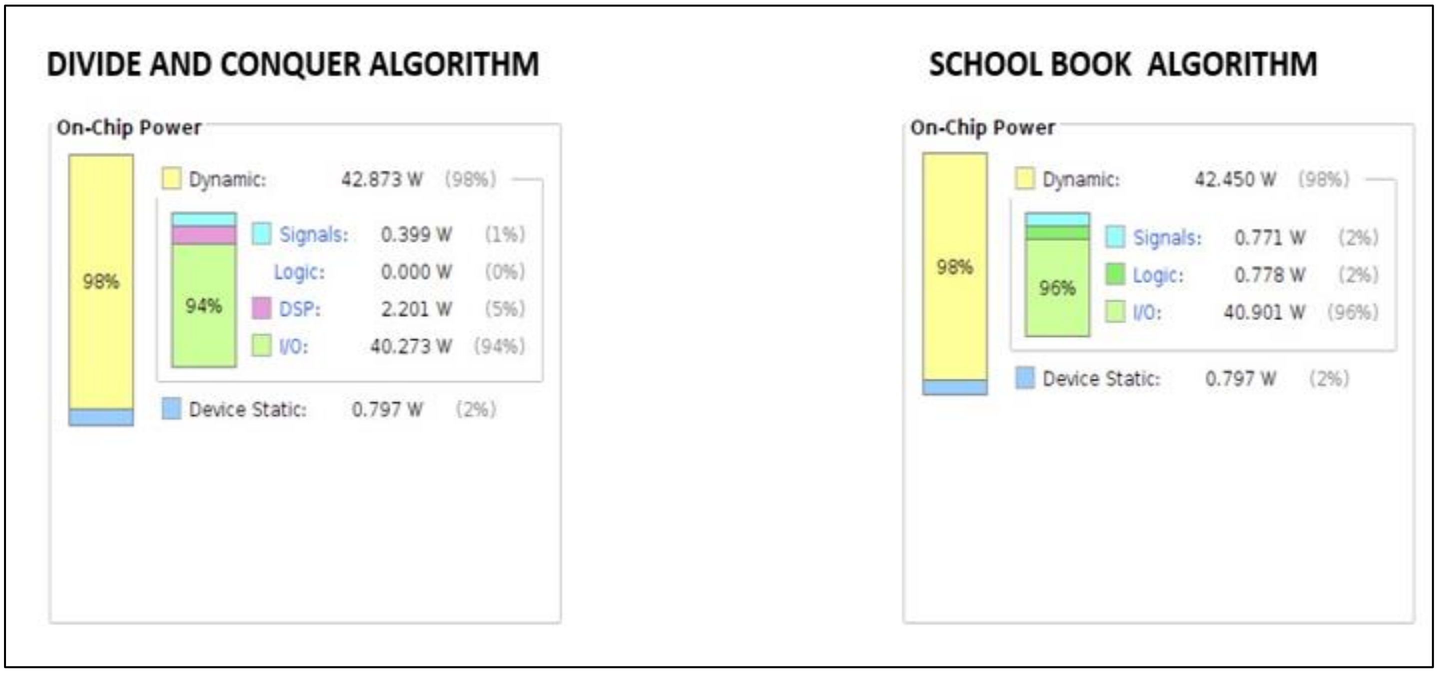
```
+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |   91 |     0 |          0 |     63400 |  0.14 |
|   LUT as Logic          |   91 |     0 |          0 |     63400 |  0.14 |
|   LUT as Memory         |    0 |     0 |          0 |     19000 |  0.00 |
| Slice Registers         |    0 |     0 |          0 |    126800 |  0.00 |
|   Register as Flip Flop |    0 |     0 |          0 |    126800 |  0.00 |
|   Register as Latch     |    0 |     0 |          0 |    126800 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     31700 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     15850 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

CPU = 00:00:00.08
Memory (MB):
Peak gain = 2775.969
free physical = 240
free virtual = 9147

Fig 4: Synthesis Output

C. Power and Delay

Power analysis in electronics assesses the energy consump- tion of a circuit, crucial for optimizing energy efficiency. It involves evaluating dynamic power (related to switching activities) and static power (leakage). Delay analysis measuresthe time it takes for signals to propagate through a circuit, influencing the overall performance. Balancing power and delay is essential in designing efficient and high-performance electronic systems.



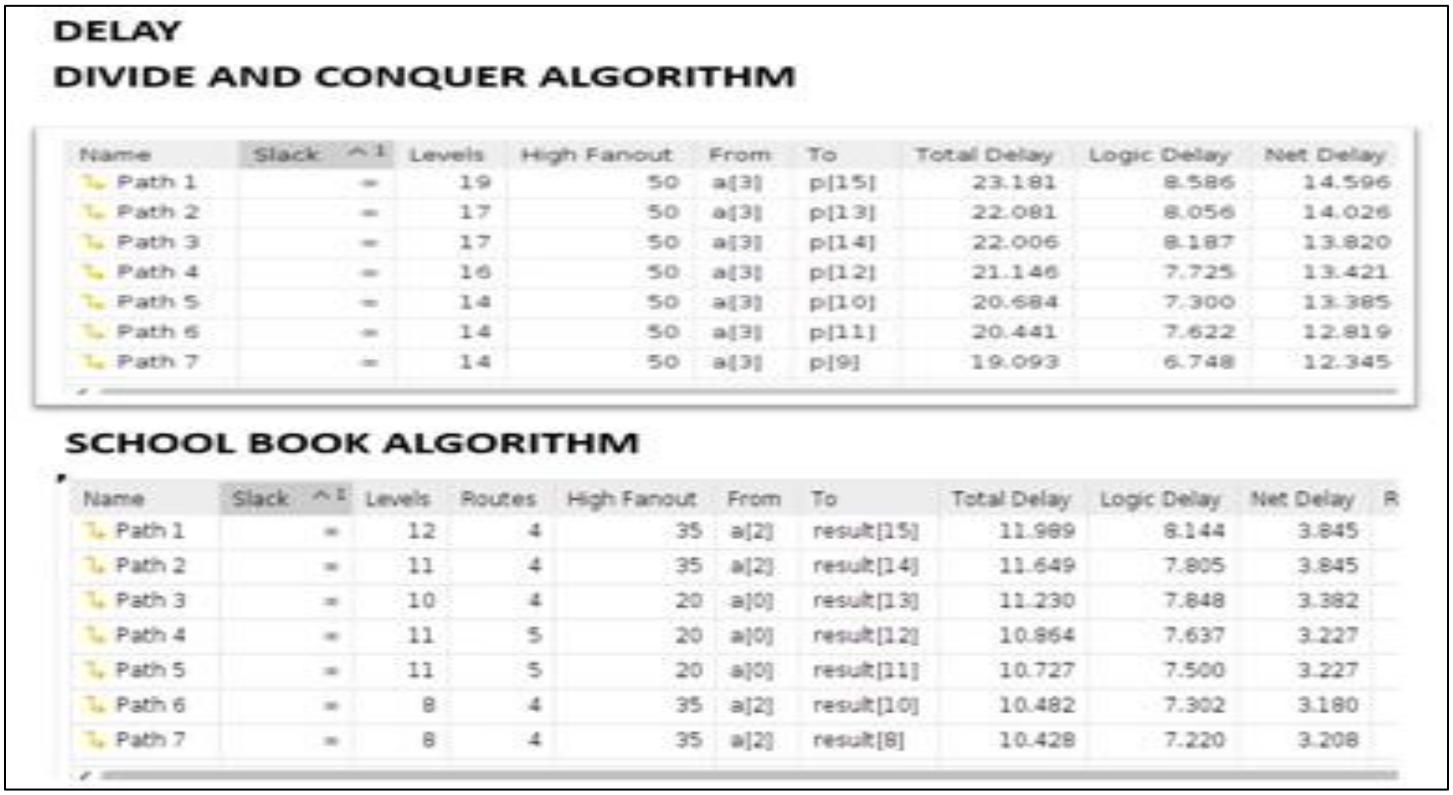
DIVIDE AND CONQUER ALGORITHM

```
On-Chip Power
Dynamic:         42.873 W  (98%)
  Signals:        0.399 W   (1%)
  Logic:          0.000 W   (0%)
  DSP:            2.201 W   (5%)
  I/O:           40.273 W  (94%)
Device Static:    0.797 W   (2%)
```

SCHOOL BOOK ALGORITHM

```
On-Chip Power
Dynamic:         42.450 W  (98%)
  Signals:        0.771 W   (2%)
  Logic:          0.778 W   (2%)
  I/O:           40.901 W  (96%)
Device Static:    0.797 W   (2%)
```

Fig 5: Power Analysis



DELAY

DIVIDE AND CONQUER ALGORITHM

Name	Slack ^ 1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	19	50	a[3]	p[15]	23.181	8.586	14.596
Path 2	∞	17	50	a[3]	p[13]	22.081	8.056	14.026
Path 3	∞	17	50	a[3]	p[14]	22.006	8.187	13.820
Path 4	∞	16	50	a[3]	p[12]	21.146	7.725	13.421
Path 5	∞	14	50	a[3]	p[10]	20.684	7.300	13.385
Path 6	∞	14	50	a[3]	p[11]	20.441	7.622	12.819
Path 7	∞	14	50	a[3]	p[9]	19.093	6.748	12.345

SCHOOL BOOK ALGORITHM

Name	Slack ^ 1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	R
Path 1	∞	12	4	35	a[2]	result[15]	11.989	8.144	3.845	
Path 2	∞	11	4	35	a[2]	result[14]	11.649	7.805	3.845	
Path 3	∞	10	4	20	a[0]	result[13]	11.230	7.848	3.382	
Path 4	∞	11	5	20	a[0]	result[12]	10.864	7.637	3.227	
Path 5	∞	11	5	20	a[0]	result[11]	10.727	7.500	3.227	
Path 6	∞	8	4	35	a[2]	result[10]	10.482	7.302	3.180	
Path 7	∞	8	4	35	a[2]	result[8]	10.428	7.220	3.208	

Fig 6: Timing Analysis

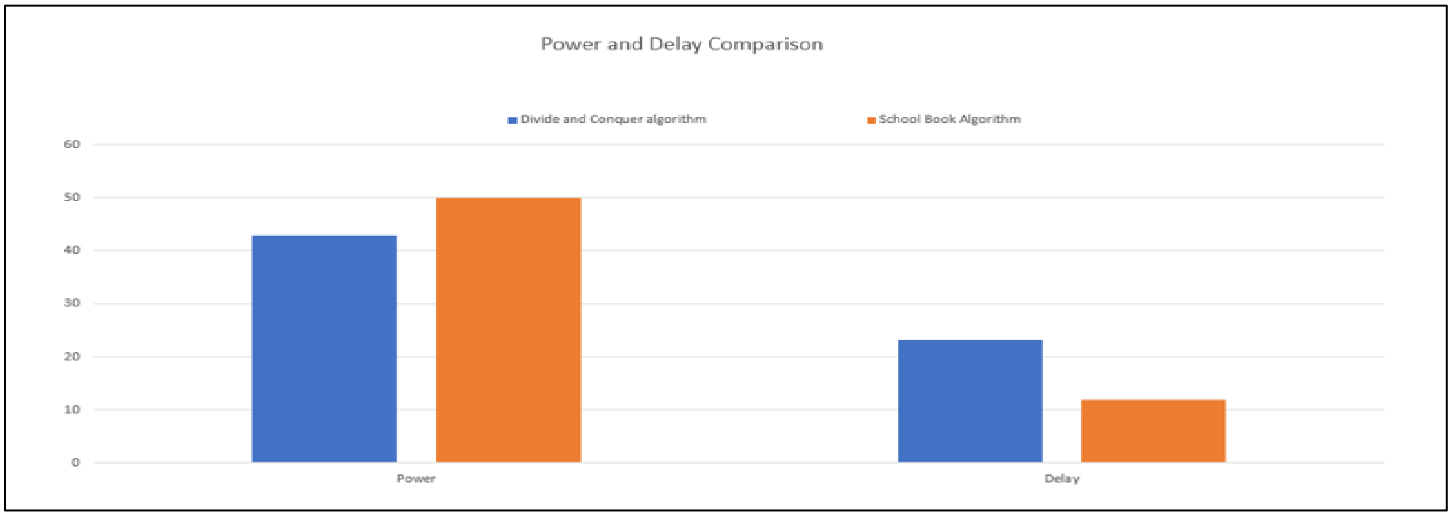


Fig 7: Comparison of Power and Delay

D. Hardware Implementation

Loading code onto an FPGA using Xilinx Vivado involves several steps. Firstly, the hardware description, typically written in a hardware description language (HDL) like Verilog or VHDL, is synthesized to generate a netlist. This netlist is then implemented, mapping the design onto the specific FPGA's architecture. After successful implementation,a bitstream file is generated, representing the configuration data for the FPGA.

In Vivado, the bitstream file is loaded onto the FPGA using programming tools such as Xilinx's Hardware Manager. This process may involve configuring the FPGA through JTAGor other programming interfaces. Once loaded, the FPGA effectively becomes a customized hardware circuit, executing the logic described in the HDL code. This approach allows for flexible and reconfigurable hardware designs in various applications such as digital signal processing, communication, and embedded systems.

E. Area Delay Product

The Area-Delay Product (ADP) is a metric used in digital circuit design to evaluate the trade-off between the physical area occupied by a circuit and its propagation delay. It is calculated by multiplying the total logic area of a circuit (in square units) by its corresponding delay (in time units). A lower ADP signifies a more efficient design as it implies a balance between circuit size and performance speed. Designersoften aim to minimize the ADP to optimize both area and delay, achieving a desirable compromise in integrated circuit performance.

- **INPUT :**
- DECIMAL - 456
- BINARY – 000111001000 [SWITCHES]
- **OUTPUT :**
- DECIMAL-284544
- BINARY - 1000101011110000000
- RESULT PRINTED – 010101111000000 [LED]

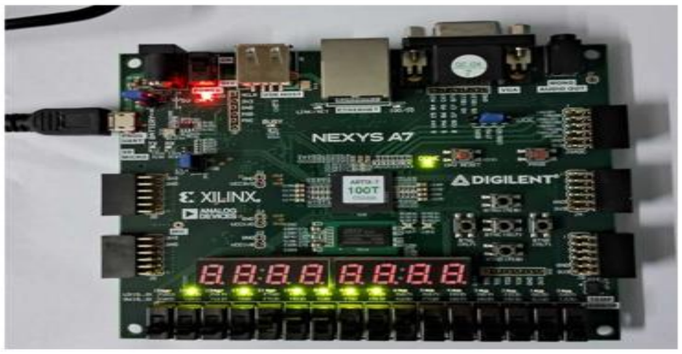


Fig 8: Hardware Implementation on Artix-7 100T

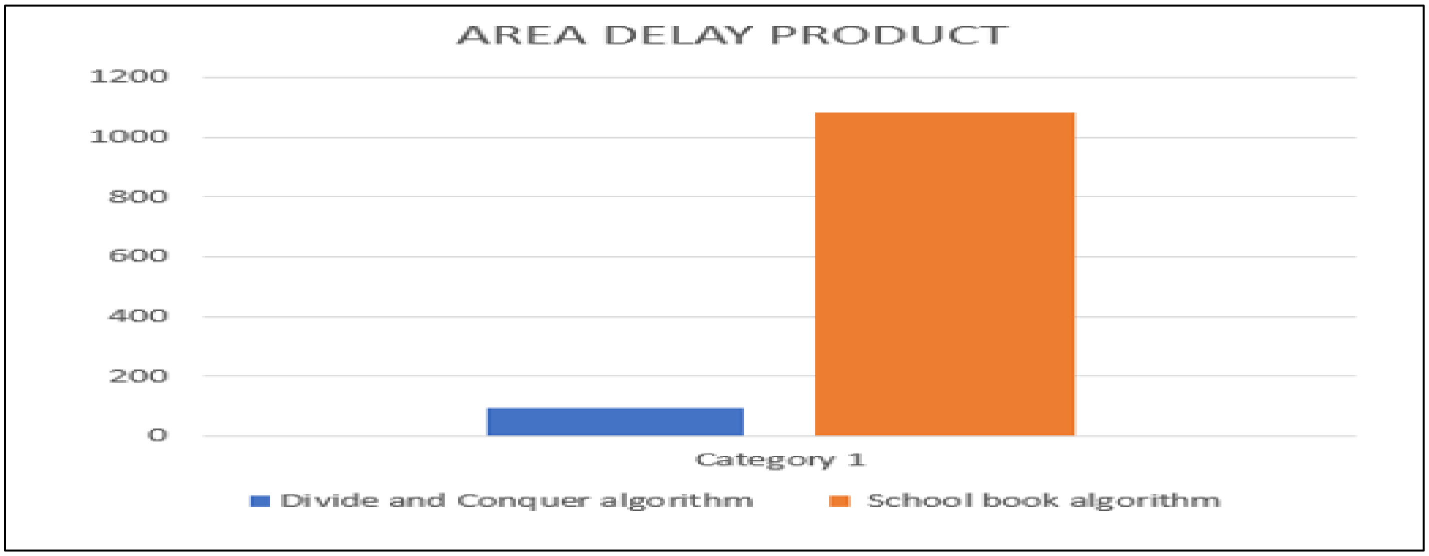


Fig 9: Area Delay Product Comparison

V. CONCLUSION

The comparison of the divide and conquer algorithm withan ADP of 92.724 um**2ps and the schoolbook algorithm with an ADP of 1084.54 um**2ps on the FPGA Artix 7 100T indicates a significant performance advantage for the divide and conquer algorithm. The lower ADP value of 92.724 um**2ps suggests that the divide and conquer algorithm is more power-efficient compared to the schoolbook algorithm, which has a higher ADP of 1084.54 um**2ps.

This performance difference can be attributed to the inherent nature of the divide and conquer algorithm, which optimally utilizes the FPGA's resources and architecture to achieve better power efficiency. The FPGA Artix 7 100T appears to be well-suited for the implementation of the divide and conquer algorithm, contributing to its superior performance in terms of ADP.

In practical terms, a lower ADP value is desirable as it indicates that the algorithm consumes less dynamic powerper unit area, making it more energy-efficient. Therefore, based on the provided ADP values, it can be concluded that the divide and conquer algorithm is a better choice for implementation on the FPGA Artix 7 100T in comparison to the schoolbook algorithm.

REFERENCES

[1]. J. Xie, P. K. Meher, M. Sun, Y. Li, B. Zeng, and Z.-H. Mao, ”Efficient FPGA Implementation of Low-Complexity Systolic Karatsuba Multi- plier Over GF(2m) Based on NIST Polynomials,” in IEEE Transactions on Circuits and Systems–I: Regular Papers, vol. 64, no. 7, pp. 1815, July 2017. DOI: 10.1109/TCSI.2017.2667164

[2]. S. Khan, W.-K. Lee, A. Khalid, A. Majeed, and S. O. Hwang, ’Area- Optimized Constant-Time Hardware Implementation for Polynomial Multiplication,’ in IEEE Embedded Systems Letters, vol. 15, no. 1, pp. 5, March 2023. DOI: 10.1109/LES.2023

[3]. Lu, Y. Cui, A. Khalid, C. Gu, C. Wang, and W. Liu, ”A Novel Combined Correlation Power Analysis (CPA) Attack on Schoolbook Polynomial Multiplication in Lattice-based Cryptosystems,” in 2022 IEEE 35th International System-on-Chip Conference (SOCC), Nanjing, China, 2022, pp. 1-6, DOI: 10.1109/SOCC56010.2022.9908076.

[4]. Y. Cui, Y. Zhang, Z. Ni, S. Yu, C. Wang, and W. Liu, ”High-Throughput Polynomial Multiplier for Accelerating Saber on FPGA,” in IEEE Transactions on Circuits and Systems—II: Express Briefs, vol. 70, no. 9, September 2023.

[5]. Z.-Y. Wong, D. C.-K. Wong, W.-K. Lee, K.-M. Mok, W.-S. Yap, and A. Khalid, ”KaratSaber: New Speed Records for Saber Polynomial Multiplication Using Efficient Karatsuba FPGA Architecture,” in IEEE Transactions on Computers, vol. 72, no. 7,July 2023

[6]. P. He, Y. Tu, Ç. K. Koç, and J. Xie, ”Hardware-Implemented Lightweight Accelerator for Large Integer Polynomial Multiplication,” in IEEE Computer Architecture Letters, vol. 22, no. 1, pp. 1-1, January-June 2023.

[7]. Y. Cui et al., ”High-Throughput Polynomial Multiplier for Accelerating Saber on FPGA,” in IEEE Transactions on Circuits and Systems—II: Express Briefs, vol. 70, no. 9, pp. 1465-1469, September 2023.

[8]. Zoni, A. Galimberti, and W. Fornaciari, ”Flexible and Scalable FPGA-Oriented Design of Multipliers for Large Binary Polynomials,” in IEEE Access, DOI: 10.1109/ACCESS.2022.3084732.