

비트 폭에 따른 다양한 정수 곱셈 알고리즘의 자원 사용량 및 성능 분석

이찬별, 엄유진, 양희훈, 유호영*

충남대학교 전자공학과

e-mail : {cblee.cas, yjeom.cas, hhyang.cas}@gmail.com, hyyoo@cnu.ac.kr

Resource Utilization and Performance Analysis of Various Integer Multiplier Algorithms According to Bit Width

Chanbyeol Lee, Yujin Eom, Heehun Yang and Hoyoung Yoo*

Department of Electronics Engineering

Chungnam National University

Abstract

This paper presents an experimental analysis of the resource utilization and performance of various integer multiplication algorithms, including Schoolbook, Karatsuba, and 3-way Toom-Cook multipliers, for operand sizes ranging from 16 to 512 bits. Although algorithms with lower theoretical computational complexity are generally expected to achieve better performance, actual hardware implementations are affected by various physical factors, including limited hardware resources, routing overhead, and DSP mapping efficiency.

The experimental results show that the optimal multiplication algorithm varies depending on the operand bit width on FPGA platforms. Specifically, the Schoolbook multiplier shows high efficiency in small operands below 32 bits, the Karatsuba multiplier provides balanced performance and hardware efficiency in the intermediate range from 32 to 512 bits, and the Toom-Cook multiplier achieves better performance and ADP characteristics for 512 bits operations.

I. 서론

정수 곱셈 연산은 디지털 신호 처리, 공개키 암호 및 임베디드 시스템 등 다양한 분야에서 핵심적인 역할을 수행한다. 특히 최근에는 FPGA를 활용한 하드웨어 가속이 활발히 연구되면서, 제한된 하드웨어 자원 내에서 효율적인 곱셈 구조를 구현하기 위한 연구의 중요성이 증가하고 있다.

기존의 정수 곱셈 연산은 주로 단순한 Schoolbook 곱셈 기반으로 구현되어 왔다. 그러나 피연산자 크기가 늘어남에 따라 연산량 또한 증가하여 이를 효율적으로 계산하기 위해 Karatsuba, Toom-Cook 알고리즘과 같은 분할 정복(divide-and-conquer) 알고리즘이 등장하게 되었다. Karatsuba 알고리즘, Toom-Cook 알고리즘은 Schoolbook 방식에 비해 연산 복잡도가 낮아 성능이 높을 것으로 여겨지지만, 실제 하드웨어에서는 제한된 자원과 라우팅 오버헤드, DSP mapping 효율성 등의 요소가 존재하기 때문에 특정 알고리즘이 항상 적절한 것은 아니다. 이에 본 논문에서는 FPGA 플랫폼을 기반으로 다양한 비트 폭에 따른 곱셈 알고리즘을 구현하고, 자원 소모량과 연산 성능을 분석하여 피연산자의 비트 폭에 따른 최적의 곱셈 알고리즘 선택 기준을 제시하고자 한다.

II. 배경

2.1 Schoolbook Multiplier

Schoolbook 곱셈은 피연산자의 비트별로 곱셈을 수행하여 더하는 가장 기본적인 곱셈으로 수식 (1)로 표현된다.

$$\begin{aligned} A &= \sum_{i=0}^{n-1} a_i 2^i, & B &= \sum_{i=0}^{n-1} b_i 2^i \\ C &= A \times B = \sum_{i=0}^{n-1} a_i 2^i \times \sum_{j=0}^{n-1} b_j 2^j \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j 2^{i+j} \end{aligned} \quad (1)$$

여기에서 A 와 B 는 n 비트 정수이고, a_i 와 b_i 는 각각 A 와 B 의 i 번째 비트를 의미한다. A 와 B 의 곱인 C 는 최대 $2n$ 비트 정수로, 모든 a_i 와 b_i 를 교차하여 곱한 결과를 시프트하고 더함으로써 C 를 구할 수 있다. 이 방식에서는 2개의 n 비트 정수를 곱할 때, 모든 비트가 서로 한 번씩 곱해지므로 n^2 번의 1비트 곱셈이 수행되어 $\theta(n^2)$ 의 연산 복잡도를 갖는다.

2.2 Karatsuba Multiplier

Karatsuba 알고리즘은 대표적인 분할 정복(divide-and-conquer) 알고리즘으로 피연산자를 두 부분으로 쪼개어 연산을 재구성함으로써 곱셈 연산량을 줄일 수 있다. Karatsuba 알고리즘은 수식 (2)를 따른다.

$$\begin{aligned} A &= A_1 2^{n/2} + A_0, & B &= B_1 2^{n/2} + B_0 \\ C_0 &= A_0 B_0, & C_1 &= A_1 B_1, & C_2 &= (A_0 - A_1) \cdot (B_0 - B_1) \\ C &= C_1 2^n + (C_0 + C_1 - C_2) 2^{n/2} + C_0 \end{aligned} \quad (2)$$

여기에서 A_1, B_1 은 피연산자 A 와 B 의 상위 $n/2$ 비트, A_0, B_0 는 하위 $n/2$ 비트를 의미하며, C_0, C_1, C_2 는 A_1, A_0, B_1, B_0 로부터 정의되는 세 개의 중간 곱셈이다. 최종적으로 C 는 C_0, C_1, C_2 의 덧셈과 뺄셈, 시프트 연산으로 얻어진다. 즉, $n/2$ 비트 곱셈 3번으로 n 비트 곱셈 결과를 얻을 수 있어 이를 재귀적으로 수행하면 연산 복잡도가 $O(n^{1.585})$ 로 감소한다.

2.3 Toom-Cook Multiplier

Karatsuba 알고리즘의 일반화된 형태인 k -way Toom-Cook 알고리즘은 피연산자를 k 개의 부분으로 분할하고, 이를 $(k-1)$ 차 다항식으로 나타내어 다항식

평가(evaluation)와 보간(interpolation) 과정을 거쳐 곱셈을 수행한다. k 가 증가할수록 더 작은 크기의 부분 곱셈기로 재구성할 수 있어 점근적 연산량은 감소하지만, 평가와 보간 단계에서의 연산이 복잡해져 회로 오버헤드가 빠르게 증가한다. 이러한 상충 관계를 고려하여 본 논문에서는 가장 작은 k 값을 사용하는 3-way Toom-Cook 방식을 채택하였으며, 이는 수식 (3)~(5)의 연산 과정을 따른다.

$$\begin{aligned} A(x) &= A_2 x^2 + A_1 x + A_0 \\ B(x) &= B_2 x^2 + B_1 x + B_0 \\ C(x) &= C_4 x^4 + C_3 x^3 + C_2 x^2 + C_1 x + C_0 \end{aligned} \quad (3)$$

피연산자 A 와 B 를 이차 다항식의 형태로 표현한다. 여기서 A_2, A_1, A_0 와 B_2, B_1, B_0 는 피연산자 A 와 B 를 $n/3$ 비트 단위로 분할한 것이다. A 와 B 가 이차 다항식으로 표현되므로 두 수의 곱인 C 는 4차 다항식 $C(x)$ 로 표현할 수 있다. 이때 C 는 $n/3$ 에서 $C(x)$ 의 함숫값이다. 4차 다항식은 5개의 서로 다른 점으로 유일하게 결정되므로 5개의 평가점 $0, 1, -1, 2, \infty$ 에서 $C(x)$ 의 값을 계산하여 활용한다.

$$\begin{bmatrix} A(0) \\ A(1) \\ A(-1) \\ A(2) \\ A(\infty) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \\ A_2 \end{bmatrix}$$

$$\begin{bmatrix} B(0) \\ B(1) \\ B(-1) \\ B(2) \\ B(\infty) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix}$$

$$C(p) = A(p)B(p), p \in \{0, 1, -1, 2, \infty\} \quad (4)$$

마지막으로 보간 과정을 통해 $C(p)$ 로부터 $C(x)$ 의 계수 $C_0 \sim C_4$ 를 복원하고, 덧셈과 시프트 연산을 통해 C 를 재구성한다.

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} C(0) \\ C(1) \\ C(-1) \\ C(2) \\ C(\infty) \end{bmatrix}$$

$$C = \sum_{i=0}^{i=4} C_i 2^{i \times \lceil \frac{n}{3} \rceil} \quad (5)$$

즉, 3-way Toom-Cook 곱셈기에서는 n 비트 곱셈을 약 $n/3$ 비트의 곱셈 5번으로 바꾸어 이를 재귀적으로 수행하면 연산 복잡도가 $O(n^{1.465})$ 로 감소한다.

표 1. 곱셈기 구조별 FPGA 합성 결과

Bit width(n)	Algorithm	Freq.	LUT	FF	DSP	*Area	**ADP
16 bits	Schoolbook	723	0	0	1	5.0000	0.006915
	Karatsuba	400	307	213	0	51.6875	0.129218
	Toom-3	211	660	545	0	116.5625	0.552428
32 bits	Schoolbook	320	47	34	4	28.0000	0.087500
	Karatsuba	390	145	261	3	49.4375	0.126762
	Toom-3	166	743	764	6	170.6250	1.027861
64 bits	Schoolbook	204	146	102	16	104.6250	0.512867
	Karatsuba	314	431	728	12	159.3750	0.507563
	Toom-3	137	1883	1671	10	389.8125	2.845346
128 bits	Schoolbook	204	689	187	50	347.8125	1.704963
	Karatsuba	204	1135	1463	48	473.3125	2.320159
	Toom-3	99	4983	3141	37	1004.1875	10.143308
256 bits	Schoolbook	22	6721	323	225	1985.3125	90.241477
	Karatsuba	157	3778	3206	150	1422.6250	9.061305
	Toom-3	84	17381	6753	126	3224.6875	38.389136
512 bits	Schoolbook	9	25690	583	900	7747.6875	860.854166
	Karatsuba	9	10511	5478	675	5031.2500	559.027777
	Toom-3	21	48781	12986	501	9414.2500	448.297619

*Area: $\frac{1}{8} \times LUT + \frac{1}{16} \times FF + 5 \times DSP$, **ADP: Area/Freq.

III. 실험 결과

본 장에서는 Schoolbook, Karatsuba, Toom-Cook 곱셈기를 KCU-105 FPGA 보드에서 구현하여 하드웨어 자원 사용량과 최대 동작 주파수를 비교 분석한다. 표 1은 16, 32, 64, 128, 256, 512비트 곱셈기를 세 알고리즘에 따라 FPGA에서 합성한 결과를 동작 주파수와 하드웨어 자원(LUT, FF, DSP) 사용량, 하드웨어 면적, ADP로 나타낸 것이다. 합성 툴로는 AMD 사의 Vivado 2021.2를 사용하였다.

그림 1에 나타난 바와 같이 모든 곱셈기는 입력의 크기가 증가할수록 동작 주파수가 감소하며, 하드웨어 면적은 지수적으로 증가한다. 또한 대체적으로 Toom-Cook, Karatsuba, Schoolbook 방식 순서로 면적을 많이 차지한다. 세 곱셈기를 비교하였을 때, 16비트에서는 Schoolbook 곱셈기가 가장 높은 주파수로 동작하며, Karatsuba 곱셈기가 두 번째로 높은 주파수로 동작한다. 그러나 32비트 이상에서는 Karatsuba 곱셈기의 최대 동작 주파수가 Schoolbook 곱셈기보다 높게 나타난다. 또한 Toom-Cook 곱셈기는 복잡한 구조로 인해 작은 비트 폭에서 상대적으로 낮은 동작 주파수를 보이지만, 타 알고리즘에 비해 비트 폭 증가에 따른 동작 주파수 감소가 완만하여 연산 규모가 증가할수록 유리해지는 경향을 보인다. 이에 따라 256비트와 512비트 이후에서 각각 Schoolbook 곱셈기와 Karatsuba 곱셈기보다 높은 주파수로 동작한다.

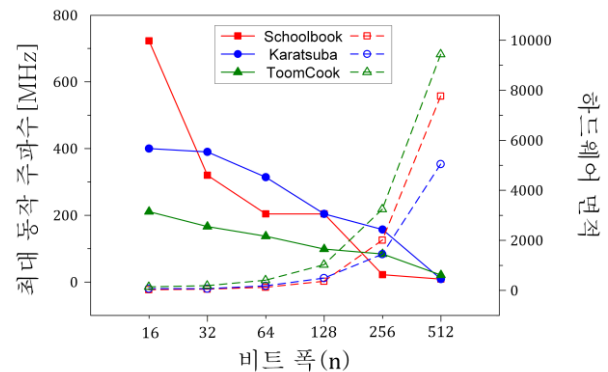


그림 1. 비트 폭에 따른 최대 동작 주파수, 하드웨어 면적

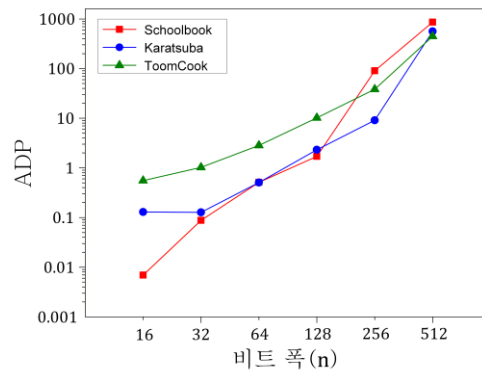


그림 2. 비트 폭에 따른 ADP

그림 2는 각 알고리즘의 비트 폭에 따른 Area-Delay Product(ADP)를 그래프로 나타낸 것이다. ADP는 하드웨어 자원 사용량과 최대 동작 주파수를 동시에 고려하기 위한 지표로 활용하였으며, 비트 폭이 증가할수록 ADP 역시 증가하는 경향을 보인다. 또한 Schoolbook 곱셈기는 128비트 이하의 환경에서는 ADP가 타 알고리즘에 비해 상대적으로 작아 면적 대비 우수한 동작 주파수를 보이고, Karatsuba 곱셈기는 64비트에서 512비트 이전까지 효율적인 ADP 지표를 나타낸다. 특히 256비트에서 Schoolbook 곱셈기보다 약 10배 낮은 ADP로 상당한 수준의 연산 복잡도 감소를 이루었다. Toom-Cook 곱셈기는 512비트에서 세 곱셈기 중 ADP가 가장 낮으며 비트 폭이 커질수록 상대적으로 효율적인 연산이 가능한 것으로 나타난다.

종합적으로 하드웨어 자원 사용량과 곱셈기의 동작 주파수를 고려하였을 때, 16비트에서는 Schoolbook 곱셈기가 자원 활용 및 최대 동작 주파수 측면에서 모두 효율적이며, 32비트부터 128비트까지의 비트 폭에서는 목표 동작 주파수와 하드웨어 효율성을 고려하여 각각 Karatsuba, Schoolbook 곱셈기를 선택하는 것이 합리적이다. 256비트 부근에서는 Karatsuba 곱셈기를 사용하는 것이 다른 알고리즘 대비 최대 동작 주파수가 높으면서도 하드웨어 자원을 효과적으로 활용할 수 있다. 마지막으로 512비트 연산에서는 Toom-Cook 곱셈기를 활용하여 큰 비트 폭에서의 곱셈 연산을 효율적으로 구현할 수 있다.

IV. 결론

본 논문에서는 Schoolbook, Karatsuba, Toom-Cook 곱셈 알고리즘을 KCU-105 FPGA 보드에서 구현하여, 다양한 비트 폭에서의 하드웨어 자원 사용량과 성능을 실험적으로 비교 분석하였다. 실험은 16비트부터 512비트까지의 피연산자 크기를 대상으로 수행되었으며 LUT, FF, DSP 사용량과 최대 동작 주파수, ADP(Area-Delay Product)를 기반으로 각 구조의 특성을 분석하였다.

실험 결과, 모든 알고리즘은 비트 폭이 증가함에 따라 최대 동작 주파수가 감소하고 면적은 늘어나는 경향을 보였다. 32비트 이하의 작은 비트 폭 영역에서는 구조가 단순한 Schoolbook 곱셈기가 최적의 자원 효율성을 나타내었으며, 32비트 이상 512비트 미만의 중간 비트 폭 영역에서는 Karatsuba 곱셈기가 동작 속도와 효율적인 자원 활용의 측면에서

균형적인 성능 지표를 보여주었다. 마지막으로 512비트 이상의 대규모 정수 연산 영역에서는 복잡한 구조적 오버헤드를 극복하고 Toom-Cook 구조가 상대적으로 높은 성능과 우수한 ADP 특성을 달성하였다.

본 연구를 통해 이론적 연산 복잡도가 낮은 알고리즘이 실제 FPGA 환경에서도 항상 가장 우수한 결과를 보장하는 것은 아님을 실험적으로 확인하였다. 하드웨어 가속기 구현 시에는 제한된 하드웨어 자원, 라우팅 오버헤드, DSP 매핑 효율성 등의 물리적 요소가 성능에 지대한 영향을 미치므로, 설계 목표 비트 폭에 부합하는 알고리즘 선택이 필수적이다. 본 연구의 정량적 분석 결과는 향후 다양한 비트 폭을 요구하는 FPGA 기반 다중 정밀도 곱셈기 설계 시, 자원 및 성능 제약 조건을 만족하는 효율적인 곱셈 알고리즘 선택 기준으로 활용될 것으로 기대된다.

Acknowledgments

This work was supported in part by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2026-RS-2024-00436406) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation); and in part by the IC Design Education Center (IDEC), Korea.

References

- [1] Y. Tu, S. Bai, J. Xiong, and J. Xie, "Novel Schoolbook-Originated Polynomial Multiplication Accelerators for NTRU-based PQC," in Proc. 5th NIST PQC Standardization Conference, 2024.
- [2] A. Karatsuba and Yu. Ofman, "Multiplication of Multidigit Numbers on Automata," Soviet Physics Doklady, vol. 7, pp. 595-596, 1963.
- [3] M. Das and B. Jajodia, "ATP-Optimized Implementation of Four Way Toom Cook Multiplications on FPGAs for Large Integer Arithmetic," Circuits, Systems, and Signal Processing, vol. 44, no. 5, pp. 3554-3569, 2025.
- [4] R. P. Brent and P. Zimmermann, Modern Computer Arithmetic, Cambridge University Press, Sec. 1.3.3, "Toom-Cook Multiplication."