Modulo-$(2^q - 3)$ Multiplication with Fully Modular Partial Product Generation and Reduction

Ghassem Jaberipur[1], Saeid Gorgin[1], Navid Ahamadian[2], Jeong-A Lee[1]

[1]Department of Computer Engineering, Chosun University, Gwangju, Republic of Korea

[2]Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran

30th IEEE International Symposium on Computer Arithmetic

ARITH 2023

Jeong-A Lee

Professor, Department of Computer Engineering, Chosun University.

Jeong-A Lee received a B.S. degree (Hons.) in computer engineering from Seoul National University, Seoul, South Korea, in 1982, an M.S. degree in computer science from Indiana University Bloomington, Bloomington, IN, USA, in 1985, and a Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 1990, From 1990 to 1995, she was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA. Since 1995, she has been with Chosun University, Gwangju, South Korea. From 2008 to 2009, she served as a Program Director of the ECE Division, at the National Research Foundation of Korea. Dr. Lee is a member of the National Academy of Engineering in South Korea. She has authored or co-authored more than 100 reviewed journal and conference papers. Her current interests include high-performance computer architectures, memory architecture, approximate computing, self-aware computing, and reliable computing.

Ghassem Jaberipur

Professor for the Brain Pool Program in Chosun University

Dr. Ghassem Jaberipur, Born in Tehran on the 26th of June 1952, is a graduate of UCLA, UW-Madison, and Sharif University of Technology (SUT). After 43 years of academic life based at Shahid Beheshti University (SBU), he retired on August 23 2022, as a Professor of the Computer Science and Engineering Department, Tehran, Iran. He is currently (as of September, 1st 2022) with the School of IT Convergence Engineering, Chosun University, Gwangju, South Korea, as a Professor for the Brain Pool Program. Besides SBU, he has taught for 4 decades in SUT and Tehran University. In 2016, he received the SUT semi-centennial medal as one of the 50 distinguished SUT graduates for his scientific achievements and services to Iranian society. Dr. Jaberipur's main research is in the field of Computer Arithmetic, for which he received a 2020 international Khwarizmi award.

Saeid Gorgin

Associate Professor of Computer Engineering at IROST & Brain Pool Program in Chosun University

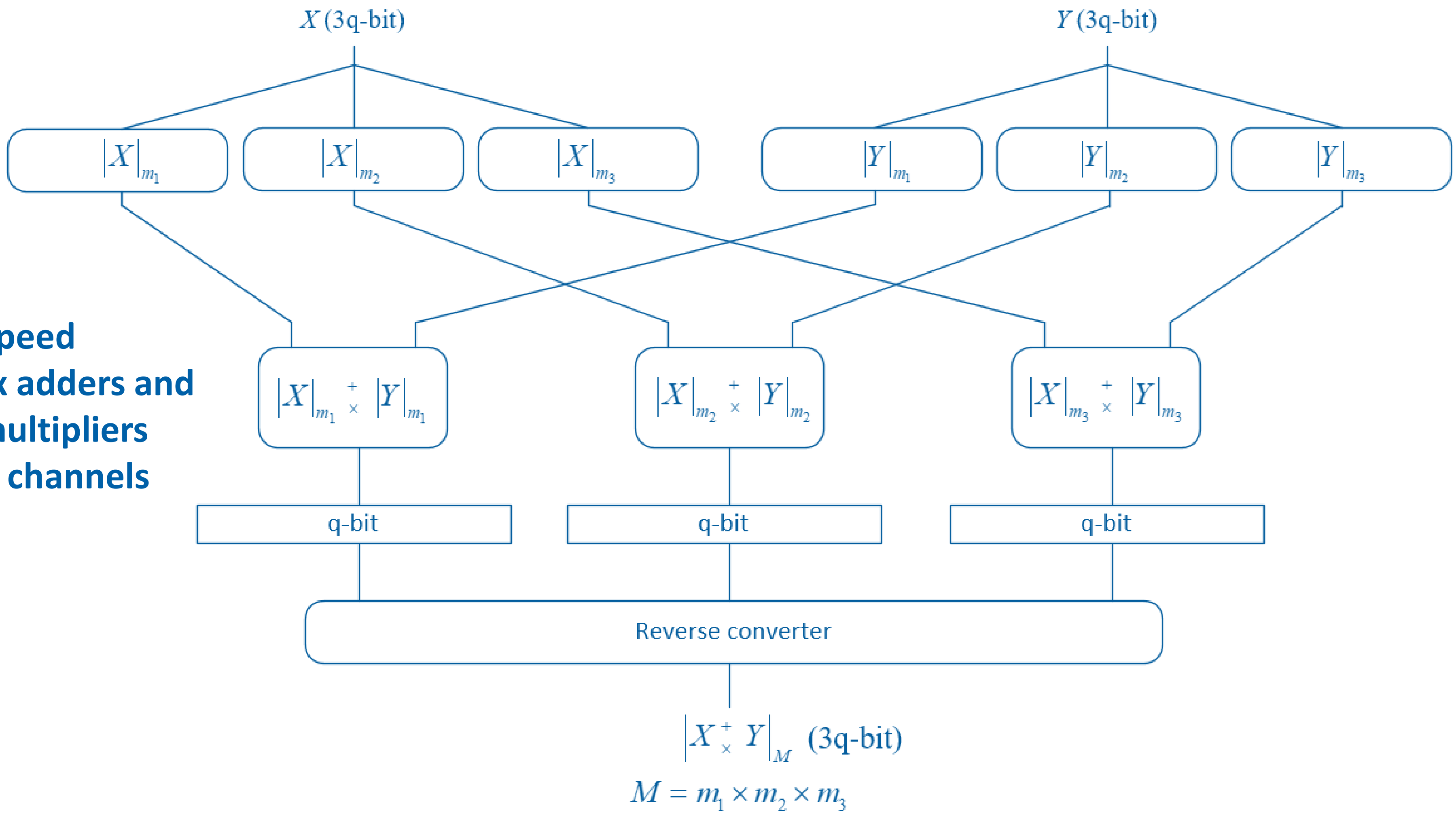
SAEID GORGIN received the B.S. degree in computer engineering from Azad University South Tehran Branch, Tehran, Iran, in 2001, the M.S. degree in computer engineering from Azad University Tehran Science and Research Branch, Tehran, in 2004, and the Ph.D. degree in computer system architecture from Shahid Beheshti University, Tehran, in 2010. He is currently an Associate Professor of computer engineering with the Department of Electrical Engineering and Information Technology, Iranian Research Organization for Science and Technology, Tehran. He is also a Visiting Scientist at the Computer Systems Laboratory, Department of Computer Engineering, Chosun University, South Korea. His current research interests include computing systems, Computer Arithmetic, Hardware Accelerators, Machine Learning, and FPGA.

Outline

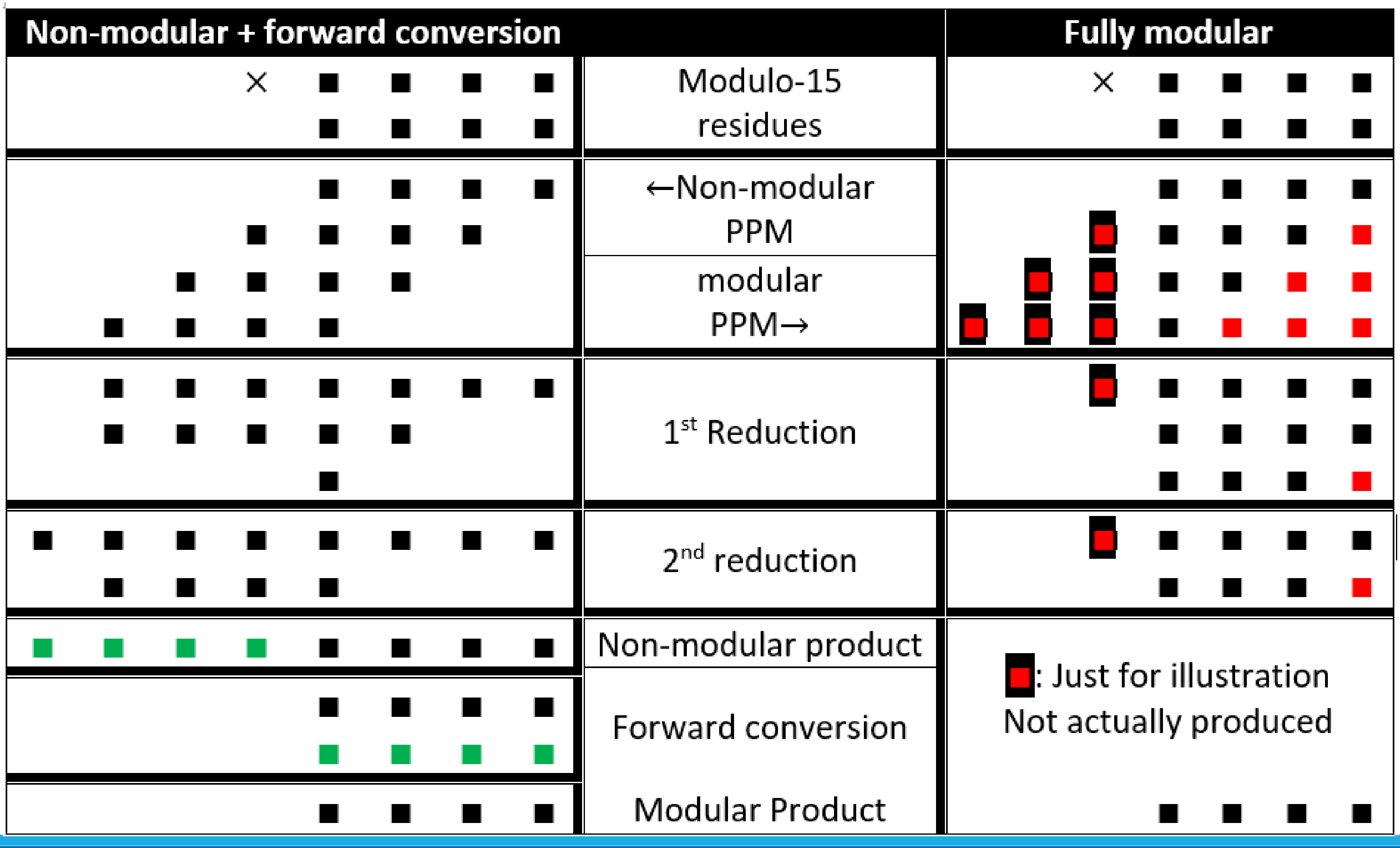
- **Limited DR of the most popular RNS moduli set $\tau = \{2^q - 1, 2^q, 2^q + 1\}$**
- **Balanced inter-moduli arithmetic speed**
- **The challenge of appropriate additional moduli for higher DR**
- **Existing τ-balanced parallel prefix modulo-$(2^q - 3)$ adders**
- **Modulo-$(2^q - 3)$ product via non-modular multiplication (2010 AND 2013)**
- **Via semi-modular multiplication (2018)**
- **The challenge of fully modular approach and the solution**
- **Results: Only 2 extra CSA levels for modulo-$(2^q - 3)$ vs. modulo-$(2^q - 1)$**
- **Results: Speedup and energy saving at the cost of more area and power consumption**

Most frequently used moduli forms: $\boldsymbol{m_1} = 2^q - 1, m_2 = 2^q, \boldsymbol{m_3} = 2^q + 1$

Balanced speed with parallel prefix adders and fully modular multipliers in the 3 residue channels



Modulo-$(2^q - 1)$ multiplication



Number of reduction levels $\mathcal{L}(q)$

Modulo-$(2^q - 1)$: $q \times q$ MPPM

$$2 \times \frac{3}{2} = 3; \left\lfloor 3 \times \frac{3}{2} \right\rfloor = 4; 4 \times \frac{3}{2} = 6; 6 \times \frac{3}{2} = 9$$

$$2\left(\frac{3}{2}\right)^{\mathcal{L}(q)} \approx q \Longrightarrow \mathcal{L}(q)\log\frac{3}{2} \approx \log\frac{q}{2} \Longrightarrow \mathcal{L}(q) \approx \left\lceil 1.7\log\frac{q}{2} \right\rceil$$

$$q = 4 \Longrightarrow \mathcal{L}(q) = \lceil 1.7 \rceil = 2(4 \rightarrow 3 \rightarrow 2)$$

$$q = 6 \Longrightarrow \mathcal{L}(q) = \lceil 1.7\log 3 \rceil = 3(6 \rightarrow 4 \ldots)$$

$$q = 9 \Longrightarrow \mathcal{L}(q) = \lceil 1.7\log 4.5 \rceil = 4(9 \rightarrow 6 \ldots)$$

Higher dynamic range without speed loss

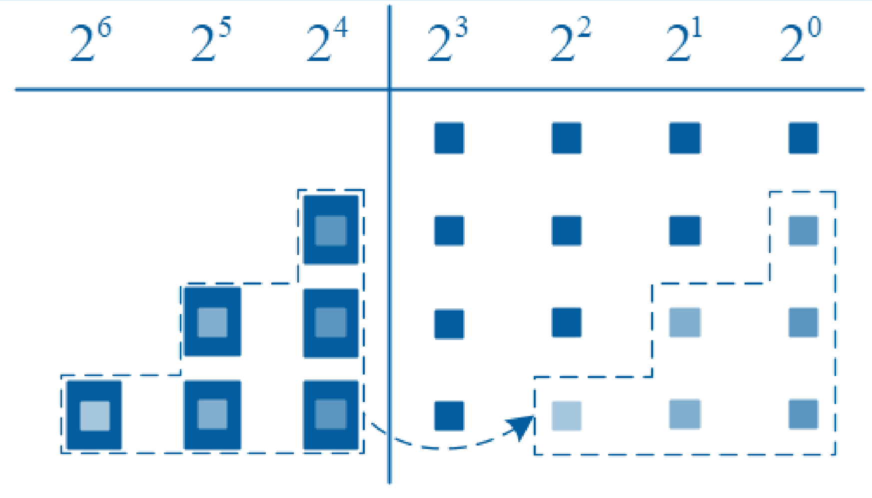
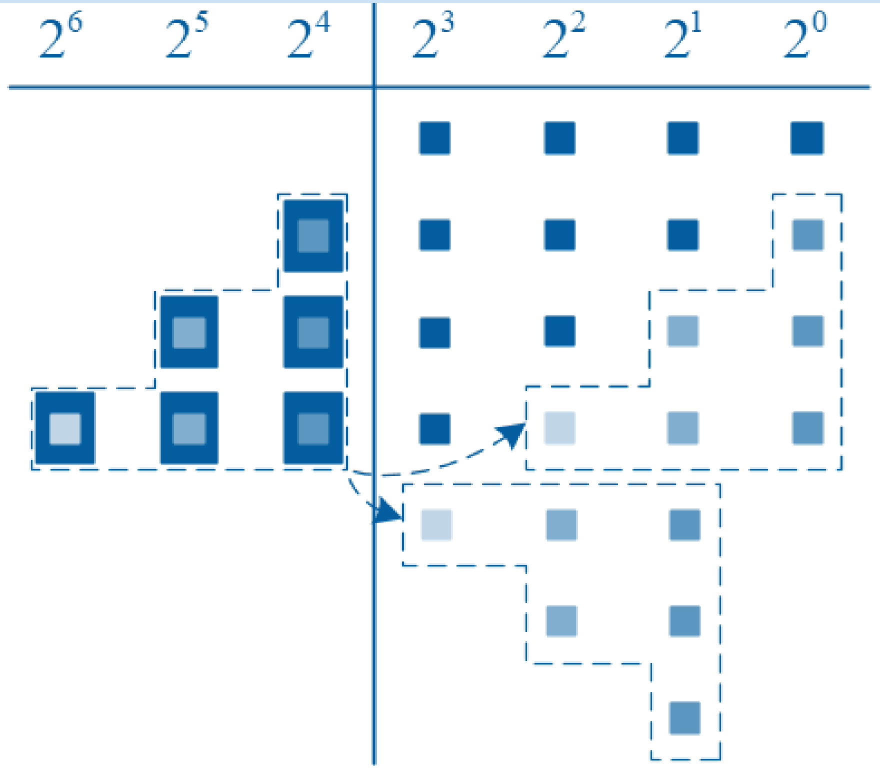
- $\tau = \{2^q - 1, 2^q, 2^q + 1\}$:
  - 2^{3q} bit DR
  - $+$ and $\times$: $O(log\ q)$ delay
- Increasing $\boldsymbol{q}$ for higher DR $\Longrightarrow$ Speed loss
- Higher DR with the same $\boldsymbol{q}$?
- Yes, via augmenting τ with $\{2^q \pm \delta, \text{for } \delta > 1\}$
- Challenge: τ-balanced $+$, $\times$ and $|\boldsymbol{X}|_{\boldsymbol{m}}$

The challenge of additional moduli of the form $(2^q \pm \delta)$

- **τ-balanced PPA for $\delta = 3$ exist**
- **Q1: Mod-$(2^q \pm 3)$ $\times$: As fast as Mod-$(2^q \pm 1)$ $\times$? NO!**
- **Q2: Complexity of $|X|_m$ for $m = 2^q \pm 3$?**
- **Do Q1 and Q2 share the same problem? Yes!**

Q1: 1st difference: Deeper MPPM ($2q-1$ vs. q)

$$q = 4$$

Modulo $2^q - 1$	**Modulo $2^q - 3$**
	

Q1: 2nd Difference: Deepening of Column 1 by two sources

$$|2^q c|_{2^q-3} = |(2^q - 3)c + 3c|_{2^q-3} = 3c = 2c + c$$

Column 1 receives carry bits from:

1) **Column $q-1$ via modular reduction**
2) **Column 0, via regular reduction:**

Q1: 3rd difference

1) **Non-modular product:** $P = A \times B = 2^q P_h + P_l$
2) **$2q$-bit P to residue conversion:**

$$|A \times B|_{2^q-3} = |2^q P_h + P_l|_{2^q-3} = |3P_h + P_l|_{2^q-3}$$

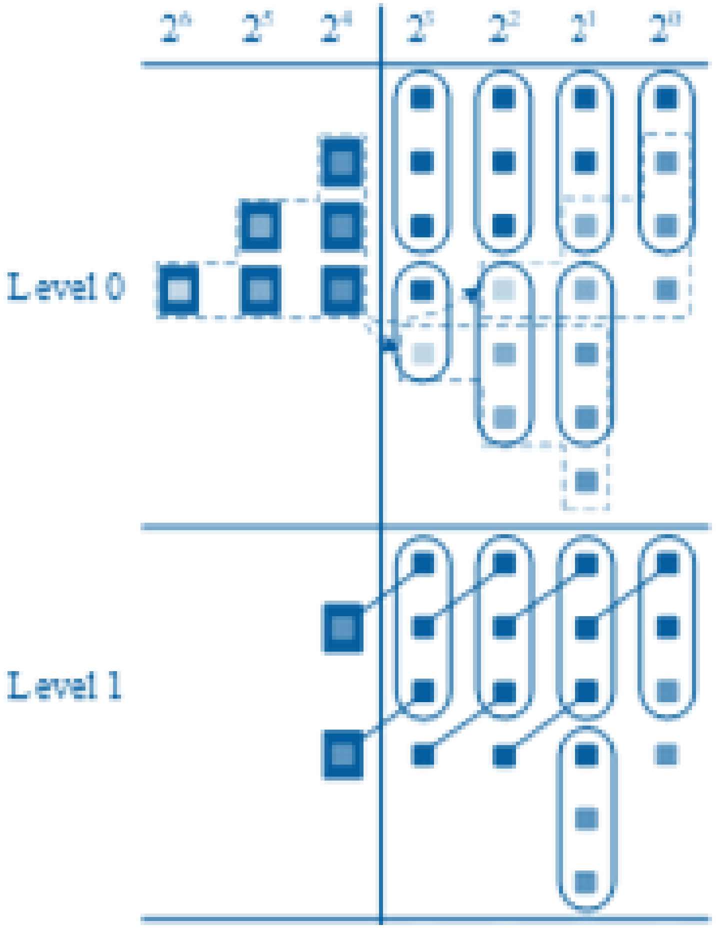
(for previous solutions of 2010 and 2013,

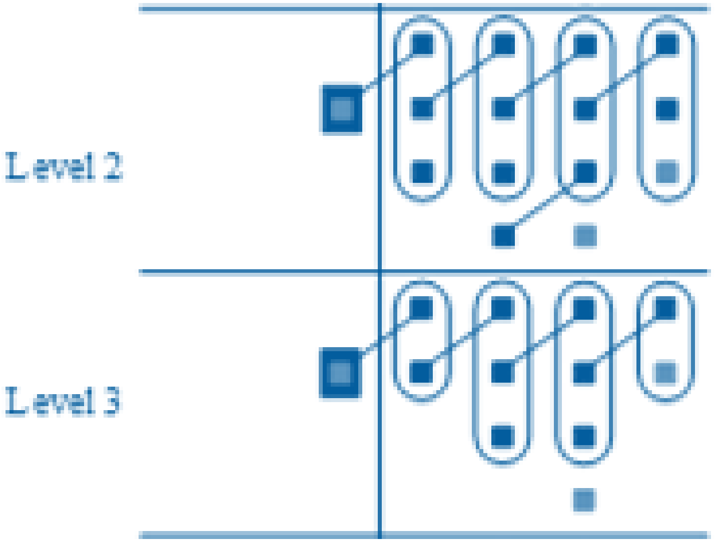
while the one of 2018 computes $\triangle = \left\lfloor \frac{P_l}{2^q} \right\rfloor$)

No reason giving for

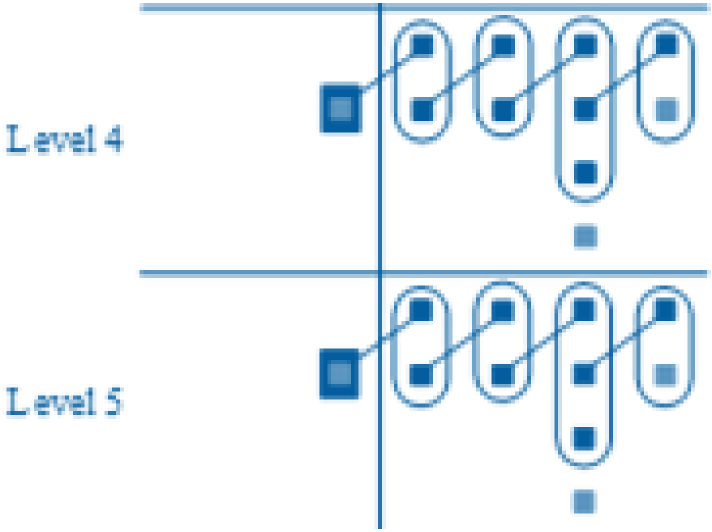
not using fully modular approach as in modulo $2^q - 1$?

Probable reason: Modulo-$(2^q - 3)$ Wallace PPR $\Longrightarrow$ Loop



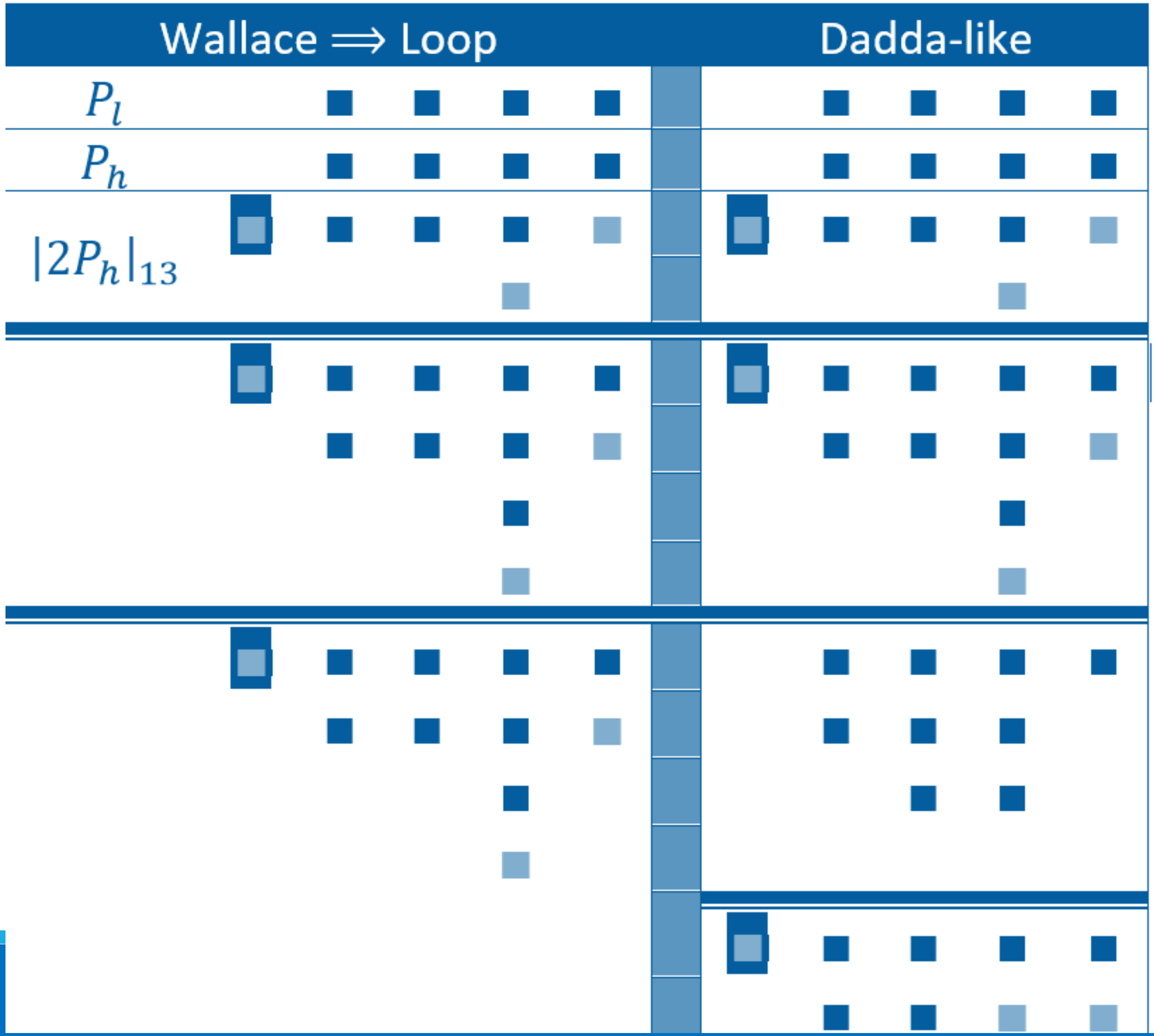


$$q = 4$$



Wallace-fail in residue generation $|3P_h + P_l|_{2^q-3}$

2010: Not addressed; 2013: Uses Dadda-like; 2018: Not applicable



Q2: Modulo-$(2^q - 3)$ residue generation

Example moduli set: $\{2^q, 2^q \pm 1, 2^q \pm 3\}$

$5q$-bit number X to $|X|_{2^q-3}$ residue:

$X = 2^{4q}X_4 + 2^{3q}X_3 + 2^{2q}X_2 + 2^qX_1 + X_0 \Longrightarrow$

$|X|_{2^q-3} = |81X_4 + 27X_3 + 9X_2 + 3X_1 + X_0|_{2^q-3}$

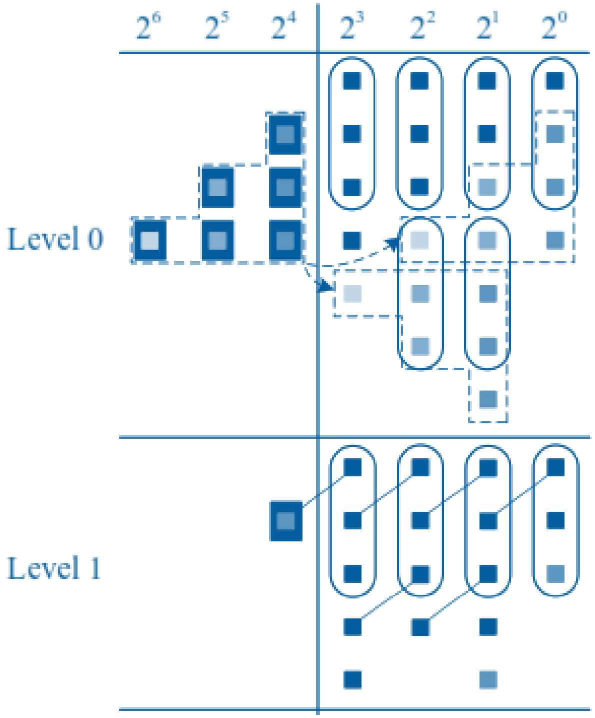
$= |(2^6 + 2^4 + 1)X_4 + (2^4 + 2^3 + 2 + 1)X_3 + (2^3 + 1)X_2 + (2 + 1)X_1 + X_0|_{2^q-3}$

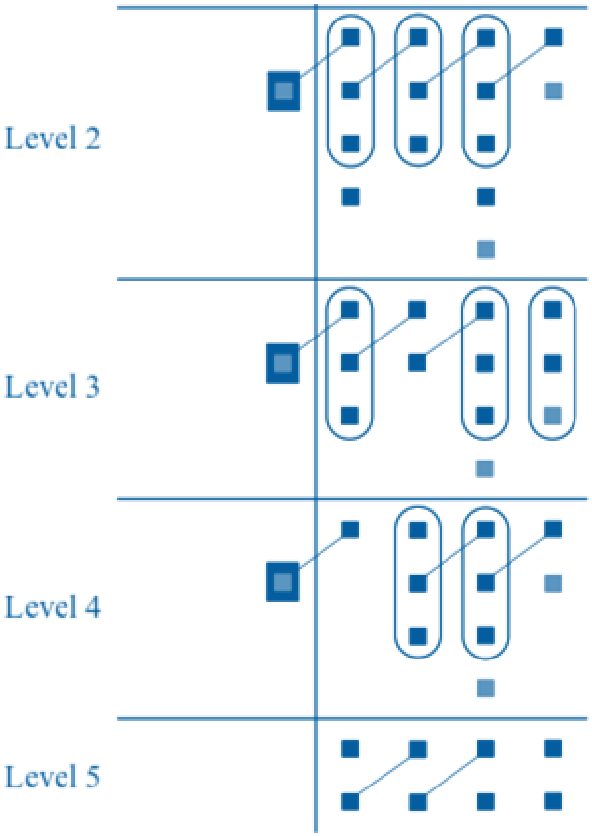
Modular multi-operand addition with

28 (=12+16) deep Column 1 $\Longrightarrow 2q - 1 = 28 \Longrightarrow$

As complex as PPR for modulo-$(2^{14} - 3)$ multiplication

Proposed design: Dadda-like reduction for $q = 4$





Proposed design: The general Algorithm

Do while there exists a column with a depth more than 2

a. Column #0: Apply $\left\lfloor \frac{d_0}{3} \right\rfloor$ FA reductions $\Longrightarrow d_0 = d_0 - 2\left\lfloor \frac{d_0}{3} \right\rfloor + \left\lfloor \frac{d_{q-1}}{3} \right\rfloor$;

b. Column #1: Apply $\left\lfloor \frac{d_1}{3} \right\rfloor$ FA reductions $\Longrightarrow d_1 = d_1 - 2\left\lfloor \frac{d_1}{3} \right\rfloor + \left\lfloor \frac{d_0}{3} \right\rfloor + \left\lfloor \frac{d_{q-1}}{3} \right\rfloor$;

c. Columns $2 \leq i \leq q - 1$:

For $i = 2$ to $q - 1$ do Apply $\left\lfloor \frac{d_i}{3} \right\rfloor$ FA reductions $\Longrightarrow d_i = d_i - 2\left\lfloor \frac{d_i}{3} \right\rfloor + \left\lfloor \frac{d_{i-1}}{3} \right\rfloor$;

End;

Number of reduction Levels $\mathcal{L}$

For Modulo $2^q - 1$:

$$\mathcal{L}(q) \approx \left\lceil 1.7 \log \frac{q}{2} \right\rceil; \qquad p = \lfloor \log q \rfloor \Longrightarrow q = 2^p \gamma (1 \le \gamma < 2) \Longrightarrow$$

$$\mathcal{L}(q) \approx \lceil 1.7(p - 1 + \log \gamma) \rceil \Longrightarrow \lceil 1.7(p-1) \rceil \le \mathcal{L}(q) \le \lceil 1.7p \rceil$$

$\mathcal{L}$ for Modulo $2^q - 3$ via the proposed algorithm $\approx \mathcal{L}(2q)$, since

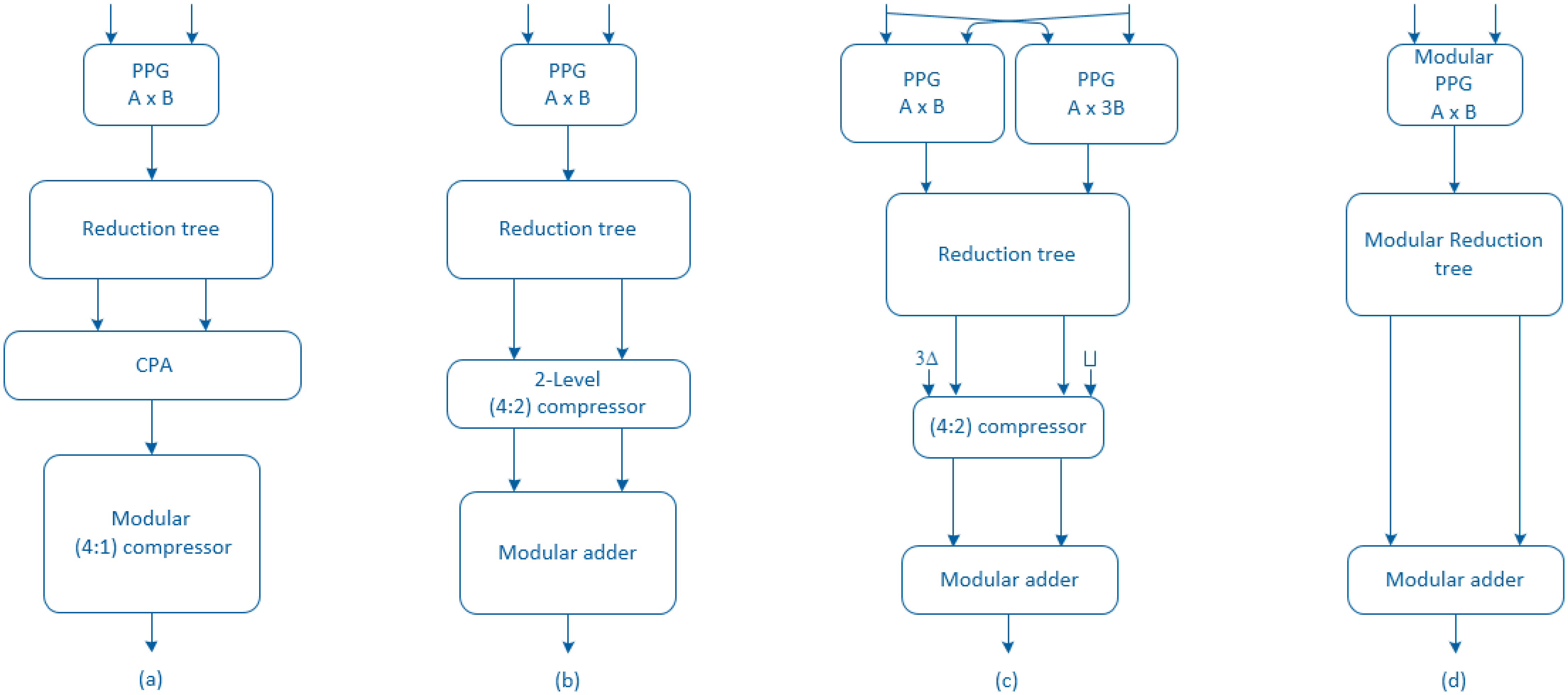
$d_1 = 2q - 1, d'_0 = 2q + 1$

$d_0 = q, d_{q-1} = q + 1, d'_0 = d_0 + d_{q-1} = q + q + 1$

Doubling q in $\lceil 1.7(p-1) \rceil \le \mathcal{L}(q) \le \lceil 1.7p \rceil$ extends the bounds by at most $\lceil 1.7 \rceil = 2$

$\Longrightarrow$ Only 2 extra levels

Schematic comparison of modulo-$(2^q - 3)$ multipliers



Modulo-13 example of Seidel's design

$A \times B = 2^q P_h + P_l, q = 4$						
			$\sqcup$	$= a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3$		
			a_3b_0	a_2b_0	a_1b_0	a_0b_0
		a_3b_1	a_2b_1	a_1b_1	a_0b_1	
	a_3b_2	a_2b_2	a_1b_2	a_0b_2		
a_3b_3	a_2b_3	a_1b_3	a_0b_3			
$P_h - \triangle$			$2^q \triangle + P_l, \triangle = \left\lfloor \frac{P_l}{2^q} \right\rfloor$			

The gray shaded parts are not implemented

$A \times 3B = A(2^4 b_3 + 2^3 B_3 + 2^2 B_2 + 2^1 B_1 + b_0) = 2^4 \times 3P_h + 3p_l$								
	128	64	32	16	8	4	2	1
				a_3B_1	a_3b_0	a_2b_0	a_1b_0	a_0b_0
$B_1 = b_1 + b_0$			a_3B_2	a_2B_2	a_2B_1	a_1B_1	a_0B_1	
$B_2 = b_2 + b_1$		a_3B_3	a_2B_3	a_1B_3	a_1B_2	a_0B_2		
$B_3 = b_3 + b_2$	a_3b_3	a_2b_3	a_1b_3	a_0b_3	a_0B_3			
	$3P_h - 3\triangle + \sqcup$							

Modulo-13 example of Seidel's design …

$\lvert A \times B \rvert_{2^q-3} = \lvert 3P_h + P_l \rvert_{2^q-3},\ q = 4$				
Actual column depth	5	6	7	8
	a_3b_0	a_2b_0	a_1b_0	a_0b_0
$B_1 = b_1 + b_0$	a_2b_1	a_1b_1	a_0b_1	a_3B_1
$B_2 = b_2 + b_1$	a_1b_2	a_0b_2	a_3B_2	a_2B_2
$B_3 = b_3 + b_2$	a_0b_3	a_3B_3	a_2B_3	a_1B_3
	a_3b_3	a_2b_3	a_1b_3	a_0b_3
				$3\triangle - \sqcup$
$\sqcup = a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3,\ \triangle = \left\lfloor \frac{P_l}{2^q} \right\rfloor$				

Modulo-13 example of Seidel's design …

$A \times B = 2^q P_h + P_l, q = 4$						
			$\sqcup$	$= a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3$		
			a_3b_0	a_2b_0	a_1b_0	a_0b_0
		a_3b_1	a_2b_1	a_1b_1	a_0b_1	
	a_3b_2	a_2b_2	a_1b_2	a_0b_2		
a_3b_3	a_2b_3	a_1b_3	a_0b_3			
$P_h - \triangle$			$2^q \triangle + P_l$, $\triangle = \left\lvert \frac{P_l}{2^q} \right\rvert$			

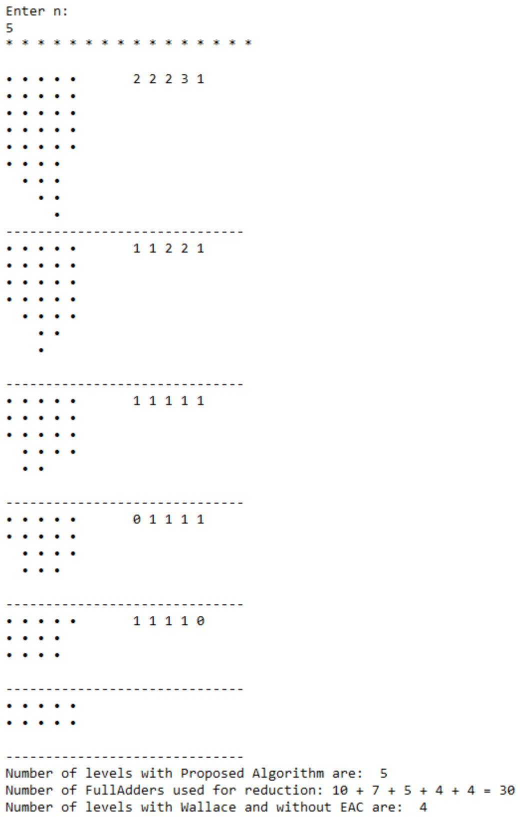
The gray shaded parts are not implemented

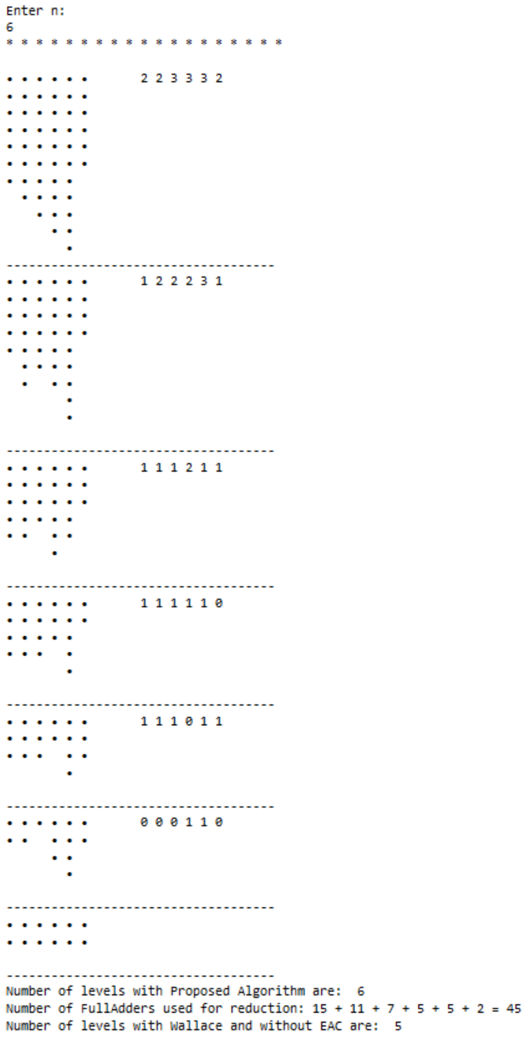
$A \times 3B = A(2^4 b_3 + 2^3 B_3 + 2^2 B_2 + 2^1 B_1 + b_0) = 2^4 \times 3P_h + 3p_l$								
	128	64	32	16	8	4	2	1
				a_3B_1	a_3b_0	a_2b_0	a_1b_0	a_0b_0
$B_1 = b_1 + b_0$			a_3B_2	a_2B_2	a_2B_1	a_1B_1	a_0B_1	
$B_2 = b_2 + b_1$		a_3B_3	a_2B_3	a_1B_3	a_1B_2	a_0B_2		
$B_3 = b_3 + b_2$	a_3b_3	a_2b_3	a_1b_3	a_0b_3	a_0B_3			
	$3P_h - 3\triangle + \sqcup$							

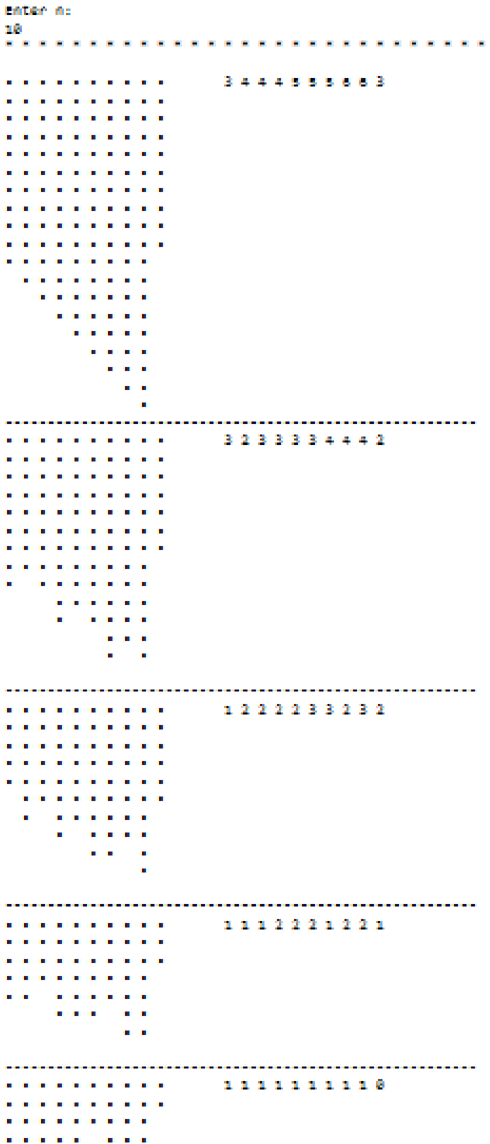
$\lvert A \times B \rvert_{2^q-3} = \lvert 3P_h + P_l \rvert_{2^q-3}, q = 4$				
Actual column depth	5	6	7	8
	a_3b_0	a_2b_0	a_1b_0	a_0b_0
$B_1 = b_1 + b_0$	a_2b_1	a_1b_1	a_0b_1	a_3B_1
$B_2 = b_2 + b_1$	a_1b_2	a_0b_2	a_3B_2	a_2B_2
$B_3 = b_3 + b_2$	a_0b_3	a_3B_3	a_2B_3	a_1B_3
	a_3b_3	a_2b_3	a_1b_3	a_0b_3
				$3\triangle - \sqcup$

Our in-house software

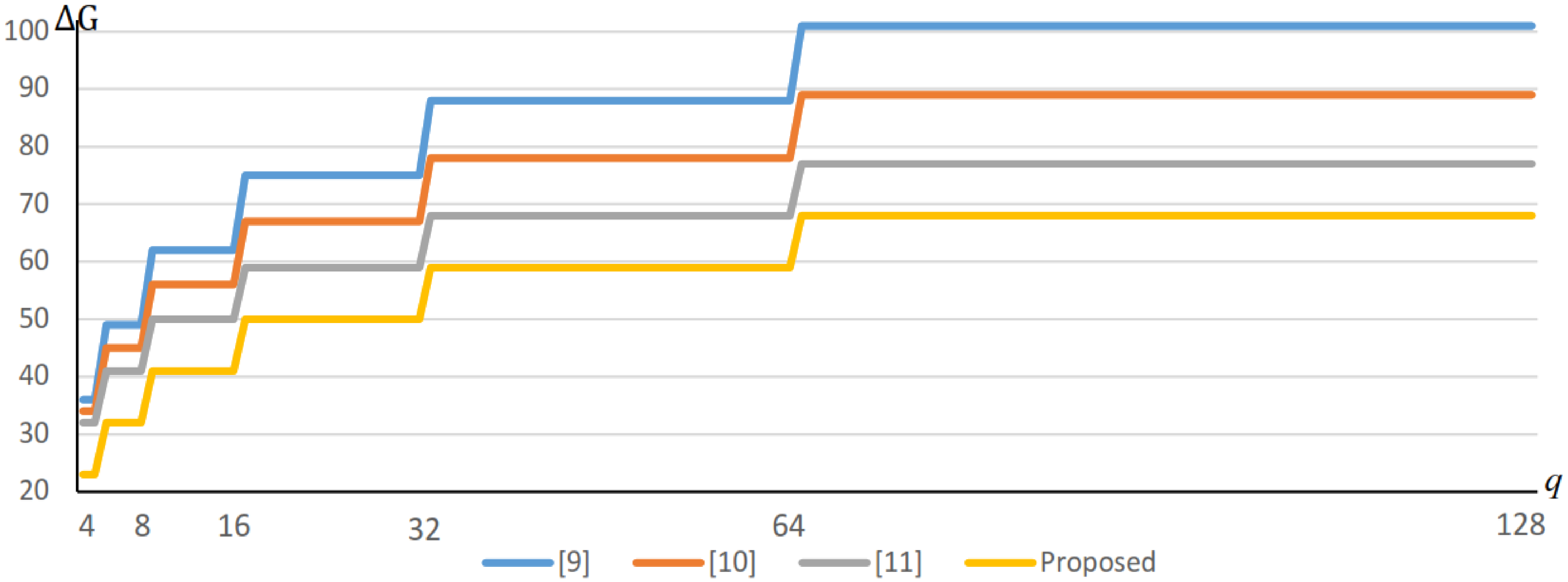
```python
import math
import array as arr

from termcolor import colored

dot = "\u2022"
blank = " "


def print_hardware(results):
    if len(results) > 0:
        delimiter = "--" * int(1.5 * len(results[0]))
        for i in range(0, len(results)):
            for hardware in results[i]:
                if hardware == blank:
                    hardware = " -"
                print(hardware, end=" ")
            print()
        print(delimiter)


def print_black(values, is_hardware_needed, results):
    row = len(values)
    col = len(values[0])
    numFA = 0
    delimiter = "--" * (int(2.5 * col) + 3)
    for i in range(0, row):
        is_blank = True
        for j in range(0, col):
            print(values[i][j], end=" ")
            if values[i][j] == dot:
                is_blank = False

        if is_hardware_needed:
            # print hardware needed for reduction
            print("     ", end=" ")
            if i == 0 and len(results)!=0:
                temp = [0]*len(results[0])
                for result in results:
                    for x in range(0, len(result)):
                        if result[x] != blank:
                            temp[x] += 1
                for value in temp:
                    print(value, end=" ")
                    numFA = numFA + value
        print()
        if is_blank:
            break

    print(delimiter)
    return numFA


def print_colored(values, colors):
    row = len(values)
    col = len(values[0])
```







Evaluations and Comparisons

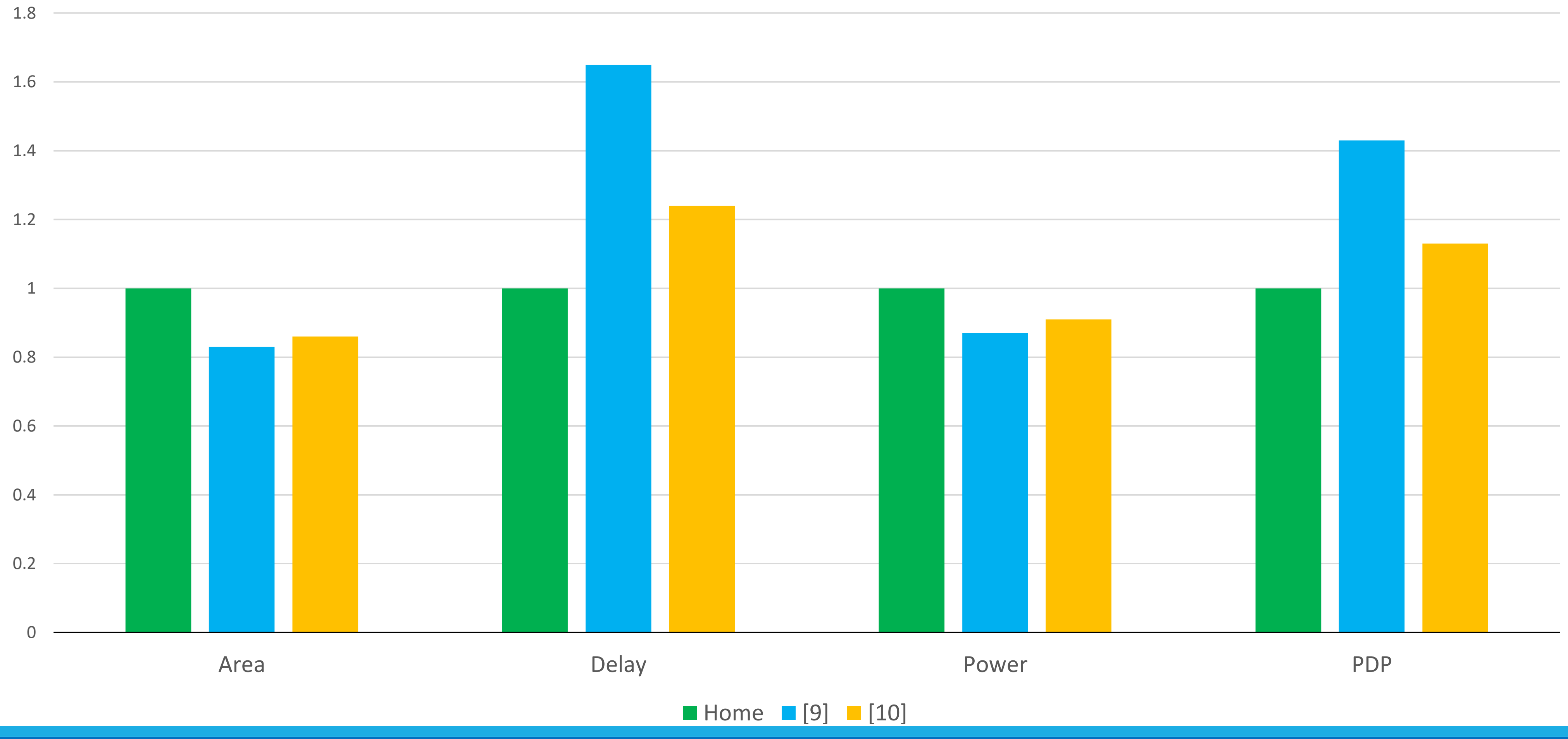


Evaluations and Comparisons …

	Area		Delay		Power		PDP	
	μm^2	Ratio	ns	Ratio	mW	Ratio	pj	Ratio
	$q = 4$							
Home	36314	**1**	4.34	**1**	0.73	**1**	3.19	**1**
[9]	37449	1.03	6.10	1.41	0.80	1.09	4.89	1.53
[10]	39349	1.08	6.41	1.48	0.86	1.17	5.52	1.73
	$q = 8$							
Home	158935	1	5.41	**1**	4.50	1	24.36	**1**
[9]	132620	**0.83**	8.92	1.65	3.90	**0.87**	34.83	1.43
[10]	137372	0.86	6.70	1.24	4.11	0.91	27.58	1.13
	$q = 16$							
Home	661472	1	6.32	**1**	22.80	1	144.11	**1**
[9]	530077	**0.80**	13.83	2.19	18.03	**0.79**	249.39	1.73
[10]	545047	0.82	8.40	1.33	18.88	0.83	158.60	1.10

Evaluations and Comparisons …

- $q = 8$



In short

Fully modular approach in the realization of modulo-($2^q - 3$) multiplier results in:

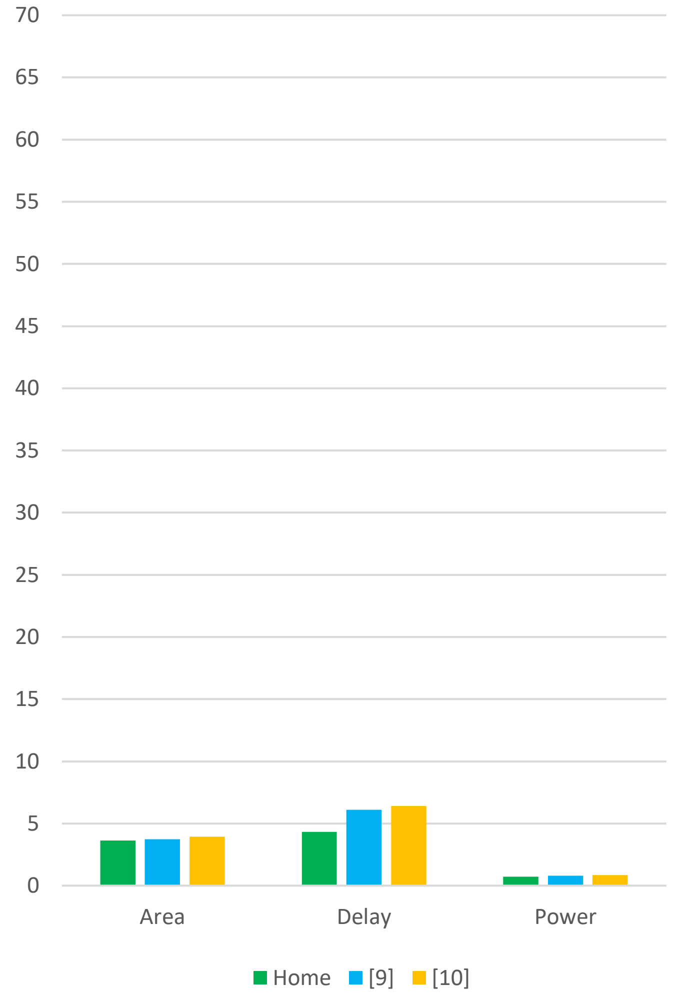
- ✓ **Less delay**
- ✓ **Less energy**
- ✓ **More speed-balance with companion moduli $2^q \pm 1$**

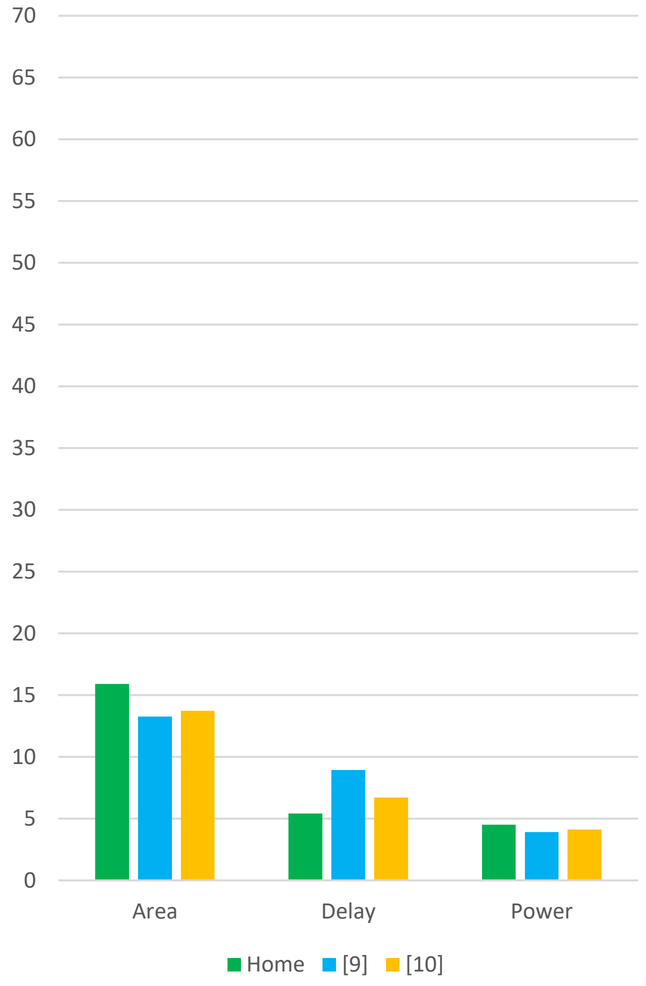
Ongoing and future relevant research:

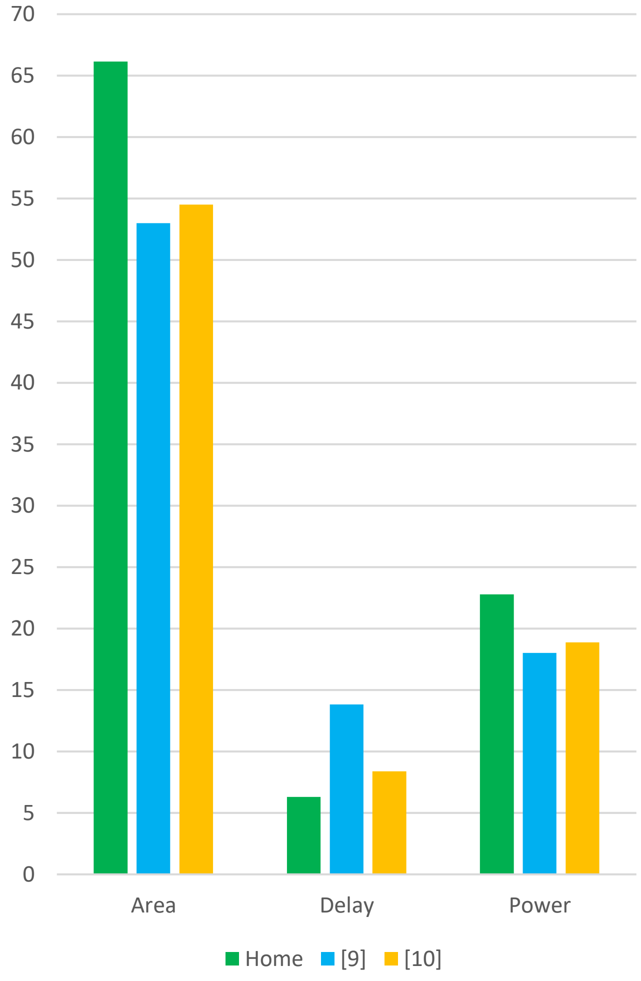
- ➢ **Fully modular modulo-($2^q + 3$) multiplier**
- ➢ **Study of fully modular approach for generic modulo-($2^q - \delta$) multiplier**

Greetings from Chosun University

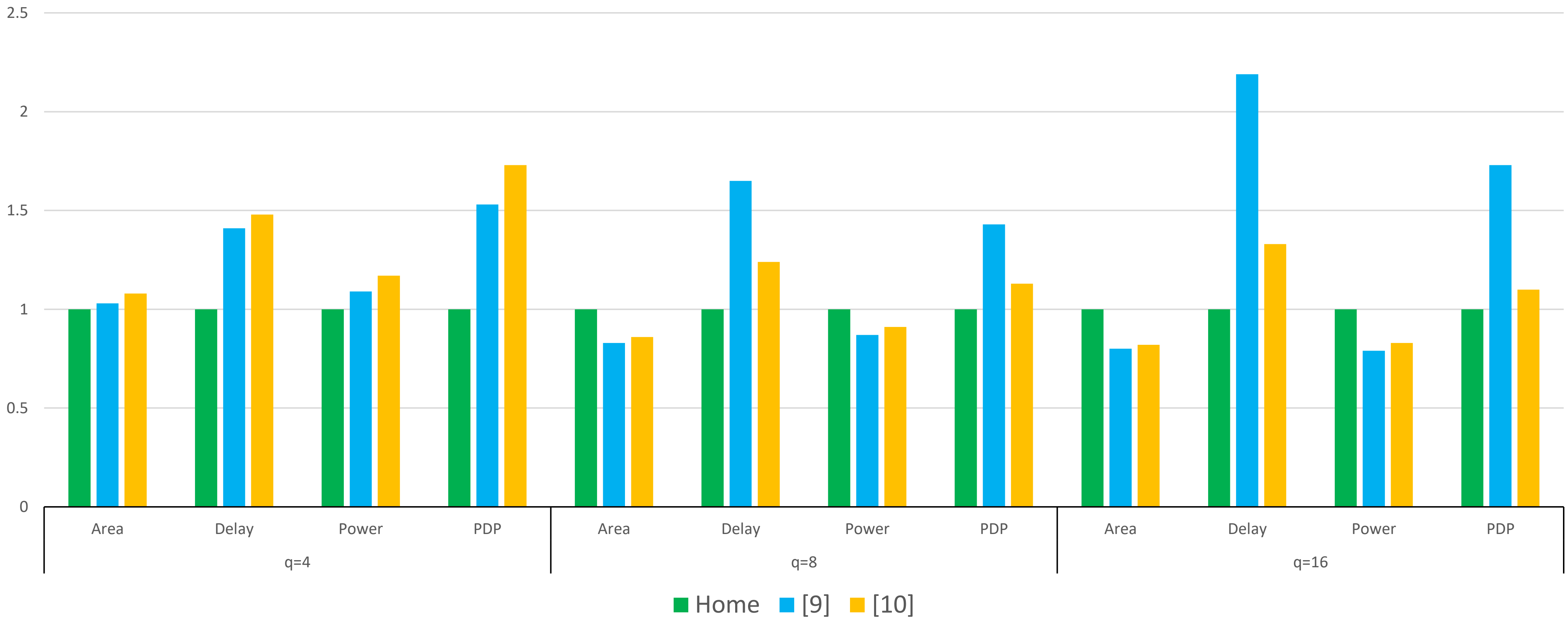
Evaluations and Comparisons



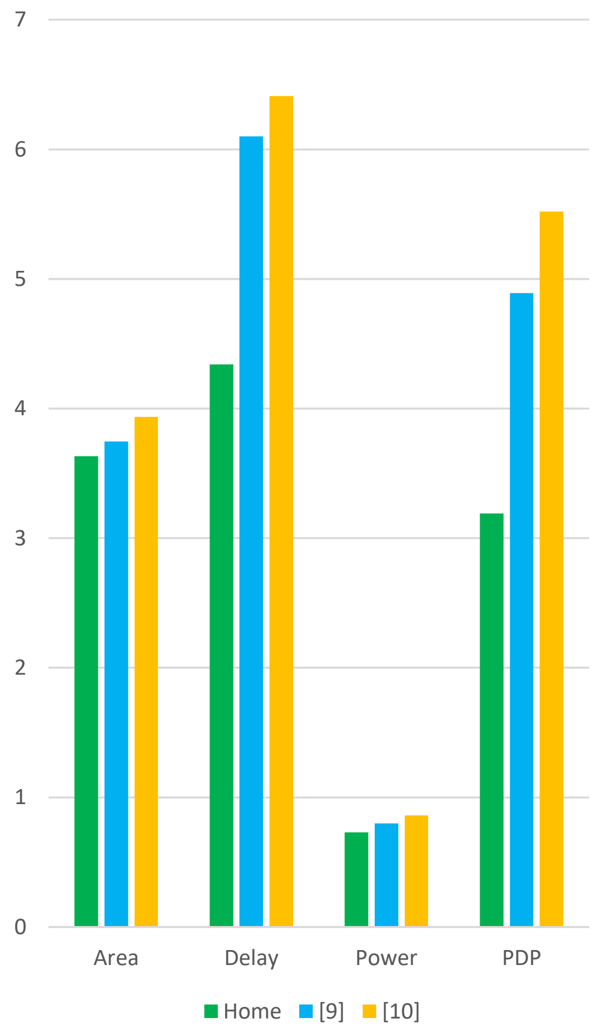


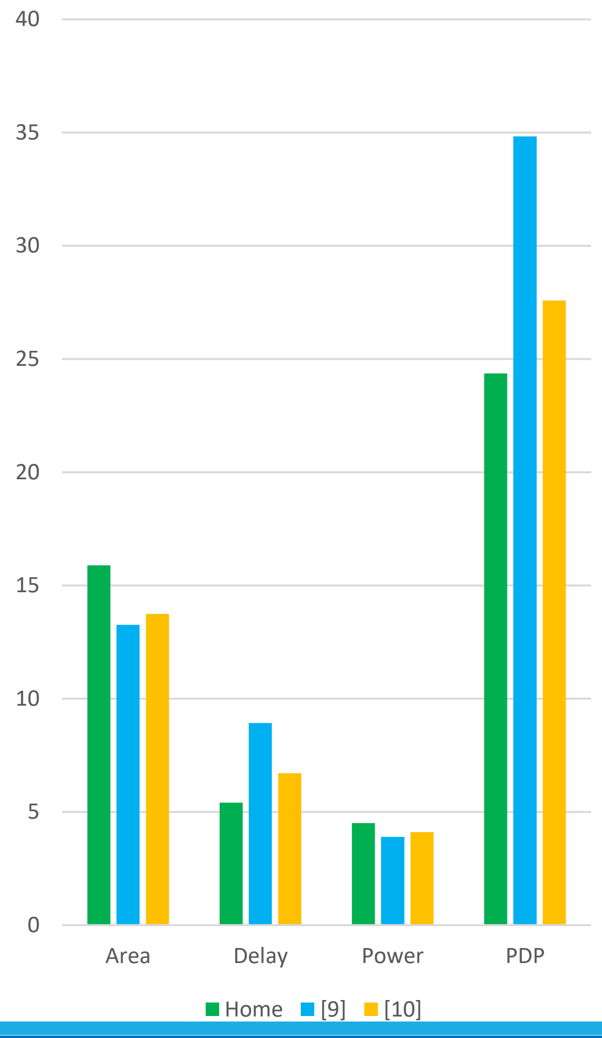


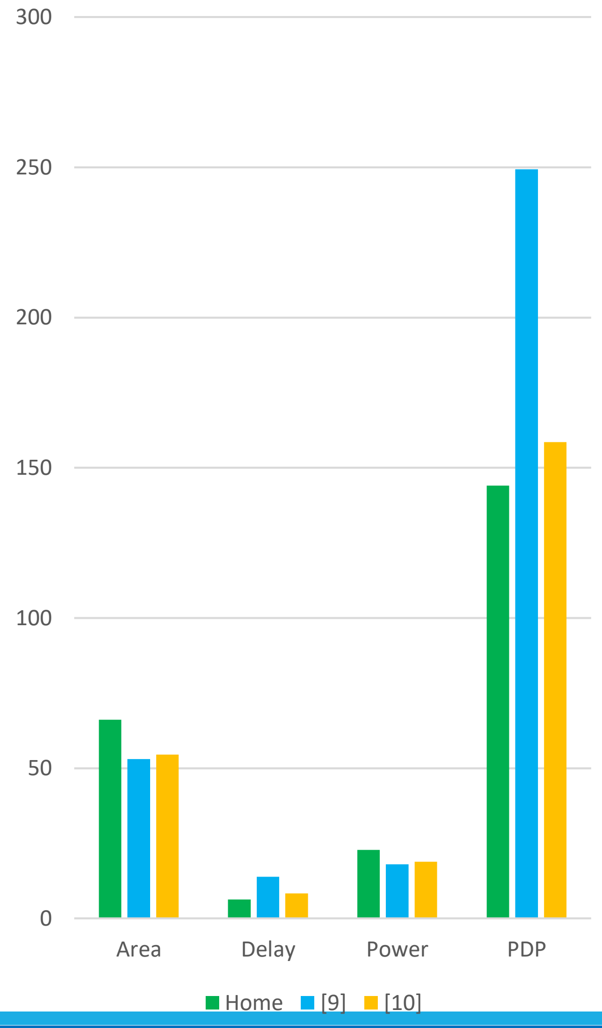
Evaluations and Comparisons



Evaluations and Comparisons







Evaluations and Comparisons