Efficient FPGA Implementation of Compressor Trees Based on Generalized Parallel Counter Chains

Mugi Noda
Graduate School of Science and Technology
Kwansei Gakuin University
Sanda, Hyogo, Japan

Nagisa Ishiura
School of Engineering
Kwansei Gakuin University
Sanda, Hyogo, Japan

***Abstract*—This paper proposes a circuit design method based on chained connection of generalized parallel counters (GPCs) as an efficient FPGA implementation technique for multi-input adders, which are key components in multipliers, multiply-accumulate units, and neural networks. Conventional methods construct carry-save adders using GPCs, which are extended full adders, but the most bit-efficient GPC, the GPC(6,0,7;5), has not been effectively utilized. The proposed technique reduces circuit area by fully leveraging this GPC through chained GPC connections. Experimental implementations on a Xilinx Artix-7 FPGA demonstrated an average area reduction of 13.06% for 8- to 32-bit multipliers and 11.08% for multi-input adders compared to the best-known methods, while maintaining a critical path delay comparable to conventional designs.**

***Index Terms*—multi-input adders, chained connection of generalized parallel counters, FPGA, compressor trees**

I. Introduction

Multi-input addition, which computes the sum of multiple binary numbers, serves as a core component of various arithmetic circuits such as multipliers and multiply-accumulate units. In recent years, its importance has also been increasing in the context of hardware acceleration for neural networks.

Classical implementation methods for multi-input adders involves constructing a tree of carry-save adders using 3-input, 2-output full adders as the fundamental building blocks [1], [2]. However, in implementations using LUT-based FPGAs, the basic building blocks of carry-save adders are often extended to 6-input, 3-output full adders [3], or *generalized parallel counters (GPCs)*, which allow not only weight-1 but also power-of-two weighted inputs [4]–[7]. Various GPC designs on Xilinx 7-series FPGAs [8] have been proposed [9]–[11], which make full use of built-in carry logic as well as LUTs.

A carry-save adder tree composed of GPCs is referred to as a *compressor tree*. Since compressor trees have a far more complex structure than trees full adders, various heuristic algorithms and formulations as integer linear programming have been proposed to minimize the level and size of the circuits [3], [5]–[7], [9], [10].

In compressor trees, GPCs reduce the number of bits; thus, using GPCs with high bit-reduction efficiency helps minimize overall circuit size. Among the GPCs implementable within a single slice, the most efficient is the GPC (6,0,7;5), which compresses 13 input bits into 5. However, this GPC fits within a single slice only when its least significant input is driven by the carry output of another GPC; otherwise, it requires two slices.

Kanai et al. [12] focused on this issue and proposed a method for constructing multi-input adders by chaining the carry logic of the GPC (6,0,7;5) to build a 6-2 adder that reduces six binary numbers to two, and connecting these adders in a tree structure. This method yields multi-input adders with smaller area and delay than those generated by the optimal compressor tree construction techniques proposed in [7], [9]. However, this approach is only applicable to simple input patterns, such as summing multiple binary numbers.

This paper proposes a new method that generalizes the concept of chaining the GPC (6,0,7;5), as introduced in [12], to enable the construction of optimal circuits for arbitrary input patterns in compressor trees. The proposed approach allows chaining of various GPCs and determines the optimal configuration using integer linear programming, as in conventional compressor tree designs.

In experiments constructing compressor trees for multiplier and multi-input adder circuits using the proposed method, we achieved circuits with an average area approximately 10% smaller than those produced by the optimal compressor tree construction methods presented in [7], [9], while maintaining a comparable critical path delay in all cases.

II. Multi-Input Adders Based on Generalized Parallel Counters

A. Generalized Parallel Counters and Compressor Trees

The input/output specification of a GPC (generalized parallel counter) is represented as $(p_{k-1}, p_{k-2}, \ldots, p_0; q)$. This GPC takes p_i input bits with weights of 2^i and outputs their weighted sum using q bits. For example, a GPC (1,3,2,5;5) has 1, 3, 2, and 5 input bits with weights of 8, 4, 2, and 1, respectively, and outputs their weighted sum using 5 bits.

To date, efficient designs of GPCs using a single slice have been proposed for Xilinx 7 series FPGAs, where a slice (or a logic block) is composed of four LUTs and carry logic as illustrated in Fig. 1. Eight fundamental types of GPCs that can be implemented in a single slice are known, and approximately 70 variations shown in TABLE I, obtained by omitting or merging certain inputs and outputs, are used as basic building blocks for multi-input adders.

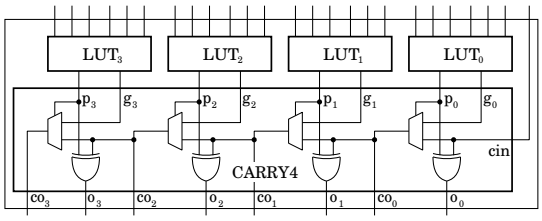


Fig. 1: Model of FPGA slice (logic block)

TABLE I: Single-slice GPCs on Xilinx 7 series [5], [9]–[11]

(3;2)	(7;3)	(1,5;3)	(2,3;3)	(3,1;3)
(4,4;4)	(6,3;4)	(7,1;4)	(1,1,7;4)	(1,2,6;4)
(1,3,5;4)	(1,4,3;4)	(1,5,1;4)	(2,0,7;4)	(2,1,5;4)
(2,2,3;4)	(2,3,1;4)	(3,0,3;4)	(3,1,1;4)	(4,2,5;5)
(4,3,3;5)	(4,4,1;5)	(6,0,6;5)	(6,0,7;5)	(6,1,5;5)
(6,2,3;5)	(6,3,1;5)	(7,0,3;5)	(7,1,1;5)	(1,1,6,3;5)
(1,1,7,1;5)	(1,2,4,4;5)	(1,2,5,3;5)	(1,2,6,1;5)	(1,3,1,6;5)
(1,3,2,5;5)	(1,3,3,4;5)	(1,3,4,3;5)	(1,3,5,1;5)	(1,4,0,6;5)
(1,4,0,7;5)	(1,4,1,5;5)	(1,4,2,3;5)	(1,4,3,1;5)	(1,5,0,3;5)
(1,5,1,1;5)	(2,0,4,4;5)	(2,0,6,3;5)	(2,0,7,1;5)	(2,1,1,6;5)
(2,1,1,7;5)	(2,1,2,6;5)	(2,1,3,5;5)	(2,1,4,3;5)	(2,1,5,1;5)
(2,2,0,6;5)	(2,2,0,7;5)	(2,2,1,5;5)	(2,2,2,3;5)	(2,2,3,1;5)
(2,3,0,3;5)	(2,3,1,1;5)	(3,0,0,6;5)	(3,0,0,7;5)	(3,0,1,5;5)
(3,0,2,3;5)	(3,0,3,1;5)	(3,1,0,3;5)	(3,1,1,1;5)	

A carry-save adder tree composed of GPCs is referred to as a *compressor tree*. Fig. 2 shows an example of a multi-input adder that sums n binary numbers, each with m bits. Each dot represents a single bit. A stage—defined as a set of GPCs separated by dashed lines—receives bits from the previous stage and outputs fewer bits to the next. The total number of bits decreases at each stage. Once the inputs are reduced to two values by the compressor tree, a row adder with a carry chain is typically used to compute the final sum [7], [9], [10].

The critical path delay of the circuit (indicated by the orange arrow) is on the order of $O(m + \log n)$—with $O(\log n)$ from the compressor tree and $O(m + \log_2 n)$ from the row adder. Although using a carry-lookahead adder for the row addition can reduce the delay to $O(\log m + \log n)$, employing the built-in carry chain within a slice is generally faster for m values up to around 256.

B. Optimization of Compressor Tree Structure

To construct compressor trees with fewer stages and smaller circuit sizes, heuristic algorithms and formulations as integer linear programming (ILP) problems have been proposed so far [7], [9], [10].

In [7], [10], for a fixed number of stages S, the construction of a GPC-based compressor tree with minimal circuit size is formulated as an optimization problem. By incrementally solving this problem while increasing S ($S = 0, 1, 2, \cdots$), a compressor tree structure with both the minimal number of stages and minimal circuit size can be obtained.

The formulation of this method is shown in TABLE II. The objective function to be minimized is the total cost of the GPCs used. Constraint C1 ensures that, in the first stage ($s = 0$) of the compressor tree, the number of bits in each column c is equal to the number of input bits I_c in that column. Constraint

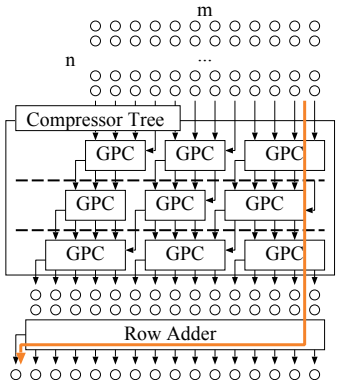


Fig. 2: Multi-input adder based on compressor tree

TABLE II: Formulation of the conventional method [7]

$k_{s,c,e} \in \mathbb{N}_0$	Number of GPCs e used at stage s, column c
$N_{s,c} \in \mathbb{N}_0$	Number of bits at stage s, column c
$S \in \mathbb{N}_0$	Number of stages in the compressor tree
$C \in \mathbb{N}_0$	Number of columns in the compressor tree
E	Set of available GPCs
$c_e \in \mathbb{R}$	Cost of GPC e
$M_{e,c} \in \mathbb{N}_0$	Number of input bits to GPC e at column c
$q_e \in \mathbb{N}_0$	Output bit width of GPC e
$K_{e,c} \in \mathbb{N}_0$	Number of output bits from GPC e at column c
$I_c \in \mathbb{N}_0$	Number of input bits at column c
$L \in \mathbb{N}_0$	Max number of output rows of the adder

	minimize $\sum_{s=0}^{S-1} \sum_{c=0}^{C-1} \sum_{e \in E} k_{s,c,e} \cdot c_e$
C1	$N_{0,c} = I_c$ for $\forall c \in \{0, 1, \ldots, C-1\}$
C2	$N_{S-1,c} \leq L$ for $\forall c \in \{0, 1, \ldots, C-1\}$
C3	$\sum_{e \in E} \sum_{c'=0}^{q_e - 1} k_{s,c-c',e} \cdot M_{e,c'} \geq N_{s,c}$ for $\forall s \in \{0, 1, \ldots, S-1\}$, $\forall c \in \{0, 1, \ldots, C-1\}$
C4	$\sum_{e \in E} \sum_{c'=0}^{q_e - 1} k_{s,c-c',e} \cdot K_{e,c'} = N_{s+1,c}$ for $\forall s \in \{0, 1, \ldots, S-2\}$, $\forall c \in \{0, 1, \ldots, C-1\}$

C2 guarantees that the number of bits in each column c in the final stage ($s = S - 1$) does not exceed the maximum number of output rows L. Constraint C3 requires that all bits in each stage be input to GPCs and propagated to the next stage. Constraint C4 states that the number of output bits from the GPCs in each stage must equal the number of bits in the subsequent stage, ensuring that all output bits from the GPCs are transferred to the next stage.

C. Exploiting GPCs with Large Compression Ratios

In compressor trees, GPCs serve to reduce the number of bits. For example, a full adder—represented as GPC (3;2)—takes three inputs and produces two outputs, effectively reducing one bit. This corresponds to a compression ratio of 2/3. Using GPCs with higher compression ratios reduces the total number of GPCs required and, consequently, the overall circuit size.

Among the GPCs implementable in a single slice (see TABLE I), the 13-input, 5-output GPC (6,0,7;5) has the highest compression ratio, followed by the 12-input, 5-output GPC. Although the (6,0,7;5) GPC is desirable due to its efficiency, it fits in a single slice only when its cin port connects to the co3 port of an adjacent slice; otherwise, it requires two slices.

Kanai et al. [12] proposed a method to build multi-input adders with smaller circuit size than the optimal solution in [7], by connecting GPC (6,0,7;5) units via carry chains. By chaining these GPCs, they construct a 6-2 adder, which sums six binary numbers into two, and use it to build an adder tree.

Since a 6-2 adder tree is simpler than a GPC tree, it can be constructed without solving complex optimization problems. However, optimal circuits can be obtained only for simple input patterns such as multi-operand addition, and it is not applicable to partial product summation in multiplication.

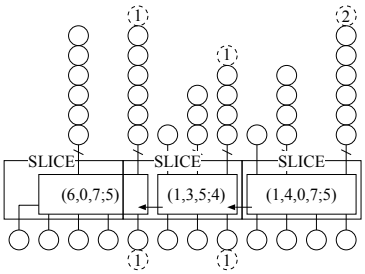
III. Multi-Input Adder Based on GPC Chaining

A. GPC Chaining

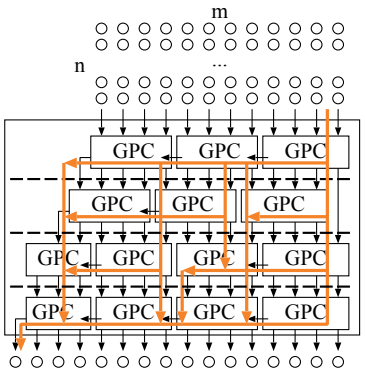
In this paper, we generalize the concept of the 6-2 adder and apply a chained connection of GPCs to a compressor tree. This enables efficient use of the GPC (6,0,7;5), reducing circuit size. It also allows optimal circuit design for general input patterns, including those in multipliers.

In our method, a chain of GPCs connected via carry chains, as shown in Fig. 3 (a), is referred to as a *GPC chain*. Fig. 3 (b) illustrates a multi-input adder using the GPC chains. Each stage of the conventional compressor tree is connected via carry chains. These chains do not need to span from the least to the most significant bits and may be segmented. While conventional methods use a row adder to sum the two outputs compressed by the tree, as in Fig. 2, our approach treats the row adder as part of the GPC chain, allowing the entire circuit, including the row adder, to be optimized as a whole.

The critical path of the adder is shown by the orange arrows in Fig. 3 (b). Signals may propagate within each stage of the GPC chain, but all paths from the top-right to the bottom-left have equal length. Thus, the delay remains $O(m + \log n)$, the same as in conventional compressor trees.



(a) GPC chaining



(b) Structure of multi-input adder

Fig. 3: Multi-input adder based on GPC Chaining

TABLE III: Formulation for GPC Chaining

$w_{s,c} \in \mathbb{N}_0$	Number of input bits directly propagated to the next stage at stage s, column c
$t_{s,c} \in \mathbb{N}_0$	Number of chain connections from below to the GPC at stage s, column c

C3′	$w_{s,c} - t_{s,c} + \sum_{e \in E} \sum_{c'=0}^{q_e - 1} k_{s,c-c',e} \cdot M_{e,c'} \geq N_{s,c}$ for $\forall s \in \{0, 1, \ldots, S-1\}$, $\forall c \in \{0, 1, \ldots, C-1\}$
C4′	$w_{s,c} - t_{s,c} + \sum_{e \in E} \sum_{c'=0}^{q_e - 1} k_{s,c-c',e} \cdot K_{e,c'} = N_{s+1,c}$ for $\forall s \in \{0, 1, \ldots, S-2\}$, $\forall c \in \{0, 1, \ldots, C-1\}$
C5	$\sum_{e \in E} k_{s,c,e} \geq t_{s,c}$ for $\forall s \in \{0, 1, \ldots, S-1\}$, $\forall c \in \{0, 1, \ldots, C-1\}$
C6	$\sum_{e \in E} k_{s,c-q_e,e} \geq t_{s,c}$ for $\forall s \in \{0, 1, \ldots, S-1\}$, $\forall c \in \{0, 1, \ldots, C-1\}$

B. Circuit Optimization via Integer Linear Programming

As in previous methods [7], [10], our approach also uses integer linear programming to determine the optimal compressor tree structure. It builds on the existing formulation, with additions and modifications to support chained connections.

(1) Formulation for GPC chaining

When two GPCs are connected in a chain, the highest bit of the lower GPC and the lowest bit of the upper GPC (indicated as 1 in Fig. 3 (a)) are removed. The corresponding formulation is shown in TABLE III.

The variable $w_{s,c}$ represents the number of bits directly propagated to the next stage. In previous methods, this was modeled using GPC (1;1), but it cannot form chains, so our approach introduces a dedicated variable instead. Let $t_{s,c}$ denote the number of GPCs connected at stage s and column c; the number of bits removed from the inputs and outputs due to chaining is equal to $t_{s,c}$.

Constraints C3 and C4 are replaced with C3′ and C4′, and new constraints C5 and C6 are introduced. C3′ and C4′ are extensions of the original C3 and C4, incorporating the number of directly propagated bits $w_{s,c}$ and the number of bits removed by chaining $t_{s,c}$. Constraints C5 and C6 limit the number of chained GPCs $t_{s,c}$ at stage s and column c. Since chaining is only possible when GPCs exist both above and below, the number of chains must not exceed the number of GPCs in the upper stage (C5) or the lower stage (C6).

TABLE IV: Formulation for GPC with 7 inputs at LSB

$r_{s,c} \in \mathbb{N}_0$	Number of bits reduced from the GPC at stage s, column c
E_7	Set of GPCs not implementable in a single slice

C3″	$w_{s,c} - r_{s,c} + \sum_{e \in E} \sum_{c'=0}^{q_e-1} k_{s,c-c',e} \cdot M_{e,c'} \geq N_{s,c}$ for $\forall s \in \{0,1,\ldots,S-1\}, \forall c \in \{0,1,\ldots,C-1\}$
C7	$t_{s,c} \leq r_{s,c}$ for $\forall s \in \{0,1,\ldots,S-1\}, \forall c \in \{0,1,\ldots,C-1\}$
C8	$\sum_{e \in E_7} k_{s,c,e} \leq r_{s,c}$ for $\forall s \in \{0,1,\ldots,S-1\}, \forall c \in \{0,1,\ldots,C-1\}$

(2) Formulation for a GPC with 7 inputs at LSB

In the formulation so far, a GPC chain with a 7-input GPC at the least significant position consumes two slices. To avoid this, when a 7-input GPC is placed at the bottom of a GPC chain, it is treated as having 6 inputs by reducing one input. The corresponding formulation is shown in TABLE IV.

In our method, both GPC chaining and the adjustment for 7-input GPCs reduce the number of input bits. We define the total number of reduced bits as $r_{s,c}$. Constraint C3″ is a modified version of C3′, replacing $t_{s,c}$ with $r_{s,c}$. Constraints C7 and C8 define the behavior of $r_{s,c}$.

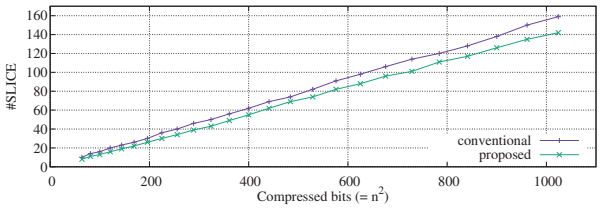
Our method uses constraints C1, C2, C3″, C4′, C5, C6, C7, and C8.

IV. Experimental Results

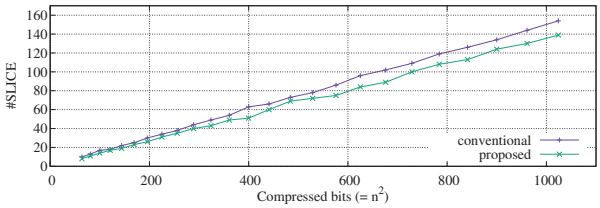
We implemented multi-input adders based on the proposed method and conducted experiments comparing them with the optimal compressor tree-based approach in [7]. For $n = 8$ to 32, we constructed two types of circuits: an n-bit multiplier and an adder that sums n binary integers of n bits. The GPC set used is shown in TABLE I. Integer linear programming was solved using CPLEX 22.1.0.0 with a time limit of 7200 seconds on a Ryzen 9 3900X. The resulting multi-input adders were implemented in Verilog HDL, and logic synthesis and place-and-route were performed using Vivado 2023.2 targeting the Xilinx Artix-7 (xc7a100tcsg324-3).

Fig. 4 (a) and (b) show the circuit area for an n-bit multiplier and an adder summing n binary integers of n bits, respectively. The horizontal axis represents the total number of input bits ($= n^2$), and the vertical axis shows the number of slices. Using the proposed method, the circuit size was reduced by an average of 13.06% for multipliers and 11.08% for multi-input adders. While the previous method used a 12-input, 5-output GPC as the most efficient option for single-slice implementation, our method enables the use of a 13-input, 5-output GPC within a single slice. This improvement is considered a key factor in the area reduction.

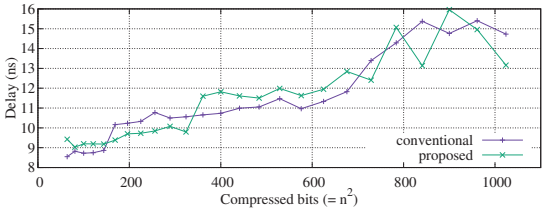
Fig. 4 (c) and (d) show the delay of the n-bit multiplier and the adder summing n binary integers of n bits. The horizontal axis represents the total number of input bits ($= n^2$), and the vertical axis shows the delay time (ns). The delay of the adder based on the proposed method is nearly identical to that of the conventional method.



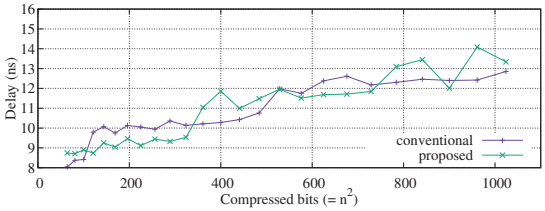
(a) Slice count of n-bit multiplier



(b) Slice count of adder for n integers of n bits



(c) Critical path delay of n-bit multiplier



(d) Critical path delay of adder for n integers of n bits

Fig. 4: Experimental results

V. Conclusion

This paper has proposed an efficient FPGA implementation of a multi-input adder using chained GPCs. The proposed method reduces the circuit size of both multipliers and adders while maintaining a delay comparable to conventional compressor trees. This design is expected to be applicable to FPGA-based neural network implementations [13]. Extending the approach to FPGAs beyond the Xilinx Artix-7 series remains a subject for future work.

Acknowledgment

Authors would like to express their appreciation to Dr. Hiroyuki Kanbara of ASTEM/RI, Prof. Hiroyuki Tomiyama of Ritsumeikan Univ., and Mr. Takayuki Nakatani (formerly with Ritsumeikan Univ.) for their valuable advises. We would also like to thank to the members of Ishiura Lab. of Kwansei Gakuin University for their discussion.

REFERENCES

[1] S. C. Wallace: "A Suggestion for a Fast Multiplier," in *IEEE Trans. on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17 (Feb. 1964).

[2] L. Dadda: "Some Schemes for Parallel Multipliers," in *Alta Frequenza*, vol. 34, pp. 349–356 (May 1965).

[3] T. Matsunaga, S. Kimura, and Y. Matsunaga: "Multi-Operand Adder Synthesis on FPGAs Using Generalized Parallel Counters," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC 2010)*, pp. 337–342 (Feb. 2010).

[4] B. Khurshid and R. N. Mir: "High Efficiency Generalized Parallel Counters for Xilinx FPGAs," in *Proc. International Conference on High Performance Computing (HiPC 2015)*, pp. 40–46 (Dec. 2015).

[5] M. Kumm and P. Zipf: "Efficient High Speed Compression Trees on Xilinx FPGAs," in *Proc. Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV 2014)*, pp. 1–12 (Jan. 2014).

[6] T. B. Preußer: "Generic and Universal Parallel Matrix Summation with a Flexible Compression Goal for Xilinx FPGAs," in *Proc. International Conference on Field Programmable Logic and Application (FPL 2017)*, pp. 1–7 (Sept. 2017).

[7] M. Kumm and J. Kappauf: "Advanced Compressor Tree Synthesis for FPGAs," in *IEEE Trans. on Computers*, vol. 67, no. 8, pp. 1078–1091 (Jan. 2018).

[8] Xilinx, Inc. : 7 Series FPGAs Configurable Logic Block User Guide (UG474) (Sept. 2016), https://docs.amd.com/v/u/en-US/ug474_7Series_CLB (accessed in Jan. 2025).

[9] M. Kumm and P. Zipf: "Pipelined Compressor Tree Optimization using Integer Linear Programming," in *Proc. International Conference on Field Programmable Logic and Applications (FPL 2014)*, pp. 1–8 (Sept. 2014).

[10] Y. Yuan, L. Tu, K. Huang, X. Zhang, T. Zhang, D. Chen, and Z. Wang: "Area Optimized Synthesis of Compressor Trees on Xilinx FPGAs Using Generalized Parallel Counters," *IEEE Access*, vol. 7, pp. 134815–134827 (Sept. 2019).

[11] M. Noda and N. Ishiura: "Enumeration of Generalized Parallel Counters for Multi-Input Adder Synthesis for FPGAs," in *Proc. Asia Pacific Conference on Circuits and Systems (APCCAS 2024)*, pp. 64–68 (Nov. 2024).

[12] Mugi Noda, Ryo Kanai, and Nagisa Ishiura: "Eficitent FPGA Implementation of Multiple-Input Adders Using Generalized Parallel Counter (6,0,7;5)" (in Japanese), in *Proc. IEICE Society Confenrece 2018*, A-6-2 (Sept. 2024).

[13] T. Tanigawa, M. Noda, and N. Ishiura: "Efficient FPGA Implementation of Binarized Neural Networks Based on Generalized Parallel Counter Tree," in *Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2024)*, pp. 32–37 (Mar. 2024).