

## Article

# A Low-Power Area-Efficient Precision Scalable Multiplier with an Input Vector Systolic Structure

Xiqin Tang <sup>1,2,3</sup> , Yang Li <sup>2,3</sup>, Chenxiao Lin <sup>1,2,3</sup>  and Delong Shang <sup>1,3,\*</sup><sup>1</sup> Research and Development Center for Intelligence and Perception, Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China<sup>2</sup> College of Integrated Circuits, University of Chinese Academy of Sciences, Beijing 100049, China<sup>3</sup> Research and Development Center for Brain-like Supercomputing, Nanjing Institute of Intelligent Technology IMECAS, Nanjing 211135, China

\* Correspondence: shangdelong@ime.ac.cn

**Abstract:** In this paper, a small-area low-power 64-bit integer multiplier is presented, which is suitable for portable devices or wireless applications. To save the area cost and power consumption, an input vector systolic (IVS) structure is proposed based on four 16-bit radix-8 Booth multipliers and a data input scheme is proposed to reduce the number of signal transitions. This structure is similar to a systolic array in matrix multiply units of a Convolutional Neural Network (CNN), but it reduces the number of processing elements by 3/4 concerning the same vector systolic accelerator in reference. The comparison results prove that the IVS multiplier reduces at least 61.9% of the area and 45.18% of the power over its counterparts. To increase the hardware resource utilization, a Transverse Carry Array (TCA) structure for Partial Products Accumulation (PPA) was designed by replacing the 32-bit adders with 3/17-bit adders in the 16-bit multipliers. The experiment results show that the optimization could lead to at least a 6.32% and 13.65% reduction in power consumption and area cost, respectively, compared to the standard 16-bit radix-8 Booth multiplier. In the end, the precise scale of the proposed IVS multiplier is discussed. Benefiting from the modular design, the IVS multiplier can be configured to support sixteen different kinds of multiplications at a step of 16 bits [16b, 32b, 48b, 64b] × [16b, 32b, 48b, 64b].

**Keywords:** multiplier architecture; low-power; area-efficient; iterative computing; systolic array

**Citation:** Tang, X.; Li, Y.; Lin, C.; Shang, D. A Low-Power Area-Efficient Precision Scalable Multiplier with an Input Vector Systolic Structure. *Electronics* **2022**, *11*, 2685. <https://doi.org/10.3390/electronics11172685>

Academic Editor: Athanasios Kakarountas

Received: 15 July 2022

Accepted: 24 August 2022

Published: 27 August 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Currently, deep learning in embedded applications is becoming extensively popular due to the large demand from the markets. The deep learning algorithm requires millions of multiplications and data movements, which is always power-consuming. However, most of the embedded devices such as biomedical signal processing devices, wireless sensors, smart cards, etc. [1], are powered by a battery. For these smart products, the requirements of low-power and area-efficient are more important than rapid computing speed [2]. Therefore, investigating the hardware dedicated computing circuits with less area and power consumption is of great importance [3]. Considering that multipliers are the most frequently used, any reduction in the power consumption and area overhead of the multiplier will bring great benefits.

The major power savings can be obtained by optimizing the architecture of a multiplier. Normally, the multipliers are designed with three kinds of architectures: tree-based, array-based, and shift-and-add structures, and many low-power designs are implemented based on one of the three architectures [4–12]. For example, a bypass zero feed A directly (BZ-FAD) architecture for a shift-and-add multiplier was proposed [4], which removed the shift operation of the Partial Products (PPs) and bypassed the adder whenever possible, thus reducing the total switching activity up to 76% and the power consumption up to 30% compared with the conventional architecture. In [5], a novel Multi-Precision

(MP) multiplier architecture was presented; all blocks could either work as independent lower-precision multipliers or work in parallel to perform higher-precision multiplications. The MP architecture brought a 28.2% and 15.8% reduction in area and power consumption compared with the conventional counterpart. In [6], the authors proposed a 16-bit iterative multiplier with an asynchronous architecture, which can offer a 74% power reduction with a small area cost compared with the synchronous one.

Apart from the circuit structure, the configurability of a multiplier is also very important for power reduction because the length of instructions and data in a digital system is different. For example, if we use a 64-bit multiplier to deal with the  $32 \times 32$ -bit multiplications, the energy and hardware resources are wasted. Therefore, the modular design method is adopted to implement a configurable multiplier [13,14], and many low-power techniques such as the power block, power switch, and multi-voltage technology [15–17] have been applied to modularized multipliers to reduce the overall power consumption of the circuit. This design method normally uses multiple short-width multipliers to build a large-width configurable multiplier, then the results of multiple short-width multiplications are summed together to obtain the final result. It has a great ability to maximize module reuse, with the least modules to meet more personalized needs.

Therefore, the modular design method was used in our proposed 64-bit integer multiplier and the reused processing element was a 16-bit radix-8 Booth multiplier. Besides, the data input scheme was also analyzed to further reduce the dynamic power consumption. As demonstrated in Equation (1), the dynamic power consumption of an integrated circuit mainly comes from four parts: the switching activity factor  $\alpha$ , the load capacitance  $C_L$ , the clock frequency  $f$ , and the supply voltage  $V_{DD}$ .

$$P_{\text{dyn}} = \alpha C_L V_{DD}^2 f \quad (1)$$

The load capacitance  $C_L$  can be reduced by decreasing the circuit area, while the switching activity can be reduced by avoiding unnecessary data transitions. Therefore, we studied the data input scheme of the 64-bit IVS multiplier and arranged the data input sequence to minimize the data transition rate. One of the inputs in the 16-bit multiplier remained fixed, and the data input sequence of another port was arranged according to the algorithm. The novelty of this work can be listed as follows:

- A novel IVS structure for a 64-bit integer multiplier was designed based on the modular design method. During the computation, input-A of each 16-bit multiplier was stationary, while input-B is the systolic input. As the input data move like a systolic array, it was named an “input vector systolic” architecture. The comparison results prove that the IVS multiplier reduces at least 61.9% of the area and 45.18% of the power over its counterparts.
- For the inner 16-bit sub-multiplier, a TCA structure for the PPA was designed to increase the utilization of hardware resources. The original full 32-bit adders were replaced with separate 3/17-bit adders to reduce the adder’s area and eliminate redundant bits. The experiment in an FPGA showed a 39.8% reduction in the number of LUTs.
- The precision scale ability of the IVS multiplier is discussed, which can support sixteen different kinds of  $[16b, 32b, 48b, 64b] \times [16b, 32b, 48b, 64b]$  multiplications.

The area cost is largely saved by reusing the same function modules, and the power is reduced by the IVS structure as it cuts down the number of operating units (only four 16-bit multipliers are used in parallel) and avoids unnecessary switching activities (only one input port of 16-bit multiplier is changed) and the modular design also makes configurable multiplication possible. The experiment results show that the IVS-based configurable multiplier can achieve a great balance among power, area, and speed.

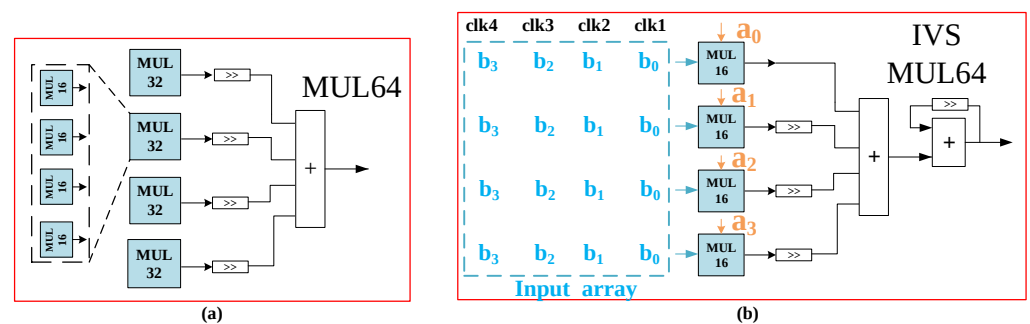
The rest of this paper is organized as follows: Section 2 presents the implementation methods of the 64-bit IVS multiplier and the optimized low-power 16-bit radix-8 Booth

sub-multiplier. Section 3 gives the comparison results and discussions. Section 4 draws the conclusions.

## 2. Methods

### 2.1. The Design of the Proposed 64-Bit IVS Multiplier

To describe the characteristic of the proposed 64-bit IVS multiplier more clearly, the structure of a 64-bit high-performance multiplier in [18] is presented in Figure 1a for comparison, which also uses the modular design method. As shown in Figure 1a, the 64-bit multiplier (MUL64) in [18] is constructed by four parallel-working 32-bit sub-multipliers (MUL32), while each MUL32 contains four 16-bit multipliers (MUL16), so the total number of MUL16 is sixteen. It generates all PPs in one clock cycle and reduces PPs in the adder array concurrently, achieving high speed, but bringing sizable hardware overhead due to the abundant hardware resources.



**Figure 1.** (a) The structure of a full-parallel 64-bit multiplier in [18]; (b) The IVS structure of the proposed 64-bit multiplier.

However, for smart products, which care little about the operating speed, but more about the area cost and power consumption, this fully parallel structure seems no longer beneficial. Therefore, we propose a more area-efficient IVS structure for smart applications.

Figure 1b presents the structure of the proposed IVS multiplier, which only consists of four MUL16 and two adder modules. It generates sixteen 32-bit intermediate products with four iterations and outputs the 128-bit final product every four clock cycles. Assume the binary representations of A and B are two 64-bit integers, which are decomposed into four 16-bit equally sized parts and denoted as (a<sub>0</sub>, a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>) and (b<sub>0</sub>, b<sub>1</sub>, b<sub>2</sub>, b<sub>3</sub>), respectively. Note that all the a<sub>i</sub> inputs of each MUL16 are fixed, while the b<sub>i</sub> inputs are changed during each cycle from b<sub>0</sub> to b<sub>3</sub>. The data-flow forms an input systolic array, which can reduce the switching activity as only one input of MUL16 changes at each clock edge. Compared to [18], the number of MUL16 is reduced by 3/4. Section 2.1.1 gives a detailed algorithm description of the 64-bit IVS multiplier.

#### 2.1.1. Algorithm Description

As we mentioned before, two 64-bit integers A and B were decomposed into four 16-bit equally sized parts (a<sub>0</sub>, a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>) and (b<sub>0</sub>, b<sub>1</sub>, b<sub>2</sub>, b<sub>3</sub>) and can be represented as Equations (2) and (3), respectively.

$$A = a_0 + a_1 2^{16} + a_2 2^{32} + a_3 2^{48} \quad (2)$$

$$B = b_0 + b_1 2^{16} + b_2 2^{32} + b_3 2^{48} \quad (3)$$

The product of A \* B is:

$$\begin{aligned}
& (a_0 + a_1 2^{16} + a_2 2^{32} + a_3 2^{48}) (b_0 + b_1 2^{16} + b_2 2^{32} + b_3 2^{48}) \\
&= a_0 b_0 + (a_0 b_1 + a_1 b_0) 2^{16} + (a_0 b_2 + a_1 b_1 + a_2 b_0) 2^{32} + (a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0) 2^{48} \\
&+ (a_2 b_2 + a_3 b_1 + a_1 b_3) 2^{64} + (a_2 b_3 + a_3 b_2) 2^{80} + a_3 b_3 2^{96}
\end{aligned} \quad (4)$$

Equation (4) shows that the  $64 \times 64$ -bit multiplication is divided into sixteen  $16 \times 16$ -bit multiplications and fifteen additions.  $M_{00}$  to  $M_{33}$  are used to represent the sixteen intermediate products ( $a_0 b_0$  to  $a_3 b_3$ ).

Then, a question is raised about how to obtain the sixteen  $16 \times 16$ -bit intermediate products and deal with the addition. In [19], the authors investigated different design strategies based on iterative multipliers; the specific experiment results showed that the implementation overhead for an iteration degree higher than four does not provide a smaller multiplier and will significantly increase the computation time due to exponential increase in PPs. Therefore, we assumed the iteration degree is four-times and used four  $16$ -bit multipliers in parallel to complete the  $64 \times 64$ -bit multiplication.

To understand the proposed input systolic array in Figure 1 easily, the coefficient matrix of the  $64 \times 64$ -bit multiplication is presented in Equation (5). The A matrix (1 by 4 dimension) is only composed of four segments in the multiplicand A, while the B matrix (4 by 4 dimension) is composed of sixteen vectors, which are the copies of the four segments in the multiplicand B. From this matrix multiplication, the sixteen coefficients in Equation (4) are obtained.

$$[a_0 \ a_1 \ a_2 \ a_3] \times \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ b_0 & b_1 & b_2 & b_3 \\ b_0 & b_1 & b_2 & b_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0 b_0 & a_0 b_1 & a_0 b_2 & a_0 b_3 \\ a_1 b_0 & a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_0 & a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_0 & a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix} \quad (5)$$

Figure 2 depicts the simple physical implementation of the IVS structure. The right four multiply symbols represent the four  $16$ -bit multipliers. Note that all the  $b$  inputs of each multiplier are fixed once two  $64$ -bit operands are input, while the  $b$  inputs are connected together and input one by one from  $b_0$  to  $b_3$  by a 4-to-1 multiplexer. The left sixteen blocks correspond to sixteen PPs generated by  $16$ -bit multipliers with four clock cycles ( $\text{clk1}$  to  $\text{clk4}$ ). Here, we classified four groups (Group1 to Group4) according to their weight coefficients  $2^i$ ; the PPs with the same weight were divided into one group. When performing the computation, each group was summed and compressed with the sum of the previous group in the next clock. Computations in different groups are not temporally related to each other, so they can be executed concurrently.

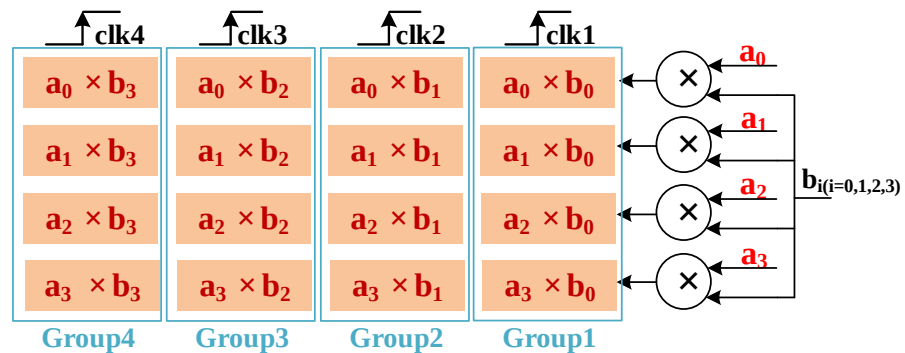


Figure 2. The multiplication process of the 64-bit IVS multiplier.

### 2.1.2. The Iterative PPA Process

To save the hardware resources, the accumulation process of the IVS multiplier is designed to be iterative with a  $64$ -bit carry-save adder (CSA) and an  $80$ -bit ripple-carry

adder (RCA). The critical path length is largely shortened compared to the traditional tree-based multiplier [20], which needs at least four levels of a 4-2 compressor and a 128-bit adder, so the operation frequency of the 64-bit IVS multiplier is larger.

The PPA process is shown in Figure 3. The “L” blocks indicate the lower 16 bits of each PP, while the “H” blocks indicate the higher 16 bits. The PPs blocks with the same color will be produced simultaneously and divided into one group. The sixteen PPs are aligned with the corresponding weights and then compressed into a 128-bit result. Only one group of PPs is calculated during each clock cycle. Once a group of PPs is produced, it is transferred to the PPA module and compressed with the sum of the previous group immediately without waiting for the generation of the next group of PPs.

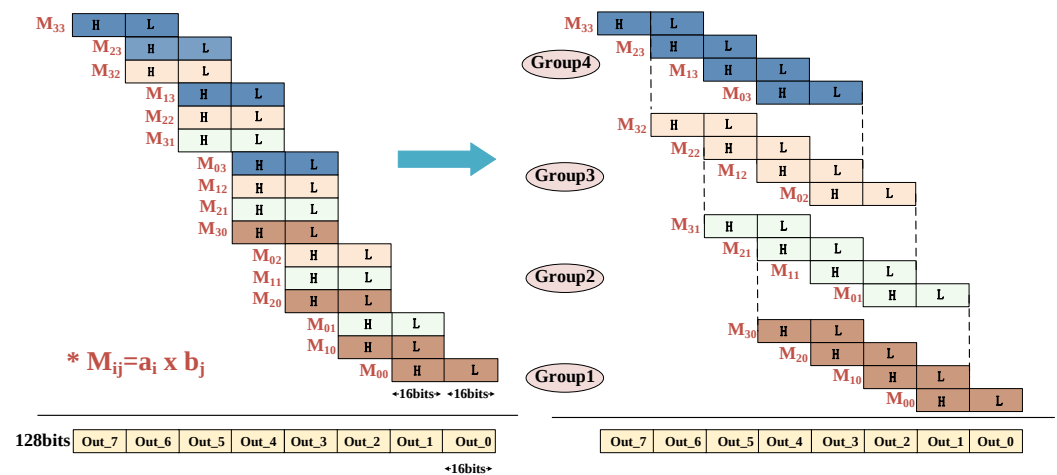


Figure 3. The PPA process of MUL16.

A detailed PPA process at each clock edge is demonstrated in Figure 4. There are three intermediate registers for data storing (Temp1, Temp2, Temp3). Assuming the first group of PPs is generated at clk1, during the period of clk1, the higher and lower 16 bits of each PP in Group1 are reordered and stored in Temp1 and Temp2 according to the shift position, as shown in Figure 3, then it will be accumulated at clk2. The accumulation processes of Group2, Group3, and Group4 are the same.

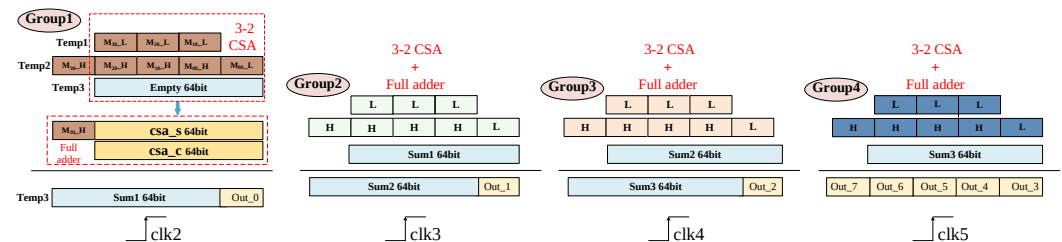


Figure 4. Four steps of the PPA with four clock cycles (Group1 is produced at clk1, and the PPA begins at clk2).

At the edge of clk2, Temp3 is reset with 64-bit initial values “0”, and PPs in Group1 are compressed into an 80-bit intermediate result by a 64-bit CSA and an 80-bit RCA. The lower 16 bits of the intermediate result are output directly, while the higher 64 bits are stored in Temp3 and summed with Group2 during the next clock cycle (clk3). The addition processes at the edge of clk3, clk4, and clk5 are the same as clk2, the only difference being that the initial value of Temp3 is the higher 64 bits of the summation of the previous group rather than “0”.

## 2.2. The Design of MUL16

As a key functional component in the proposed 64-bit multiplier, the 16-bit multiplier has a great influence on the overall power consumption and area cost. The Booth multiplier is renowned for being able to offset the shortcoming of a low throughput rate in the iterative structure. For a 16-bit Booth multiplier, the radix-8 encoding algorithm is advantageous in that it has a shorter delay, smaller power consumption, and less area overhead compared with the radix-4 encoding one due to a simpler addition circuit [21]. Therefore, the 16-bit radix-8 Booth multiplier was adopted in this work.

The hardware implementation of the PPA in MUL16 is shown in Figure 5, which mainly consists of two parts: PPs' generation and PPs' reduction. In the PPs' generation part, the multiplicand B is encoded according to the modified radix-8 Booth algorithm, as shown in Table 1, and the PPs' generators pre-produce the positive versions of PPs (a, 2a, 3a, 4a) and signs ( $S_0, S_1, S_2, S_3, S_4, S_5$ ). Then, the pre-produced PPs are selectively inverted according to the sign  $S_i$ . Finally, the PPs are extended and sent to the transverse carry array (TCA) structure for accumulation. Note that, as there are only five carry-in ports for sign signals, the  $S_0$  should be added with  $I_0$  in a half adder for the two's complement operation in advance.

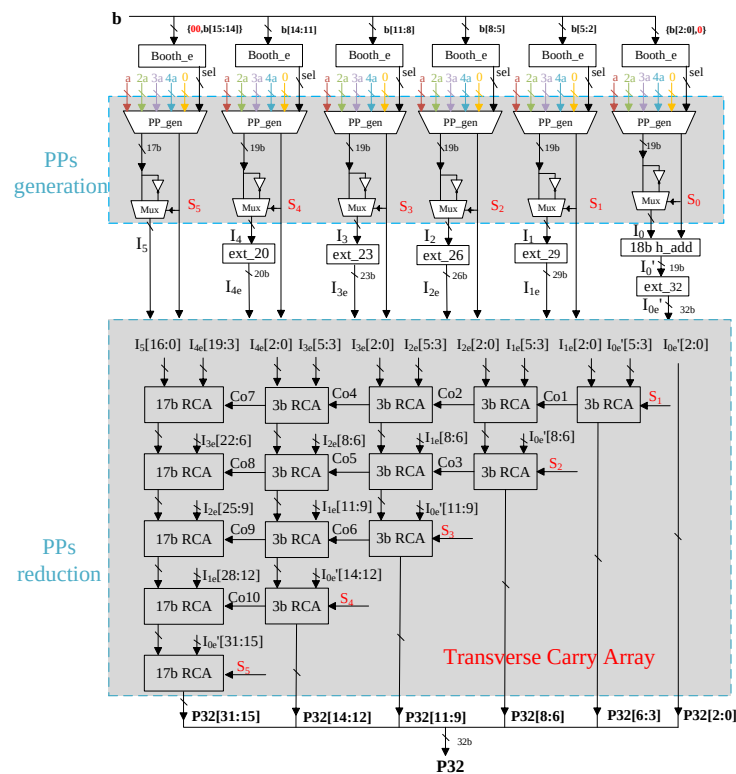


Figure 5. The 16-bit radix-8 Booth multiplier.

Table 1. Modified radix-8 Booth encoding.

| Group Bits | PP <sub>i</sub> | S <sub>i</sub> | Group Bits | PP <sub>i</sub> | S <sub>i</sub> |
|------------|-----------------|----------------|------------|-----------------|----------------|
| 0000       | 0               | 0              | 1000       | 4a              | 1              |
| 0001       | a               | 0              | 1001       | 3a              | 1              |
| 0010       | a               | 0              | 1010       | 3a              | 1              |
| 0011       | 2a              | 0              | 1011       | 2a              | 1              |
| 0100       | 2a              | 0              | 1100       | 2a              | 1              |
| 0101       | 3a              | 0              | 1101       | a               | 1              |
| 0110       | 3a              | 0              | 1110       | a               | 1              |
| 0111       | 4a              | 0              | 1111       | 0               | 0              |

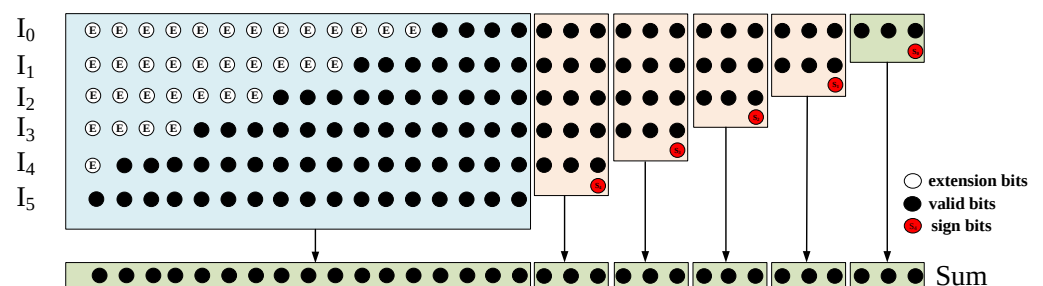
### 2.2.1. Modified Radix-8 Booth Encoding

In the traditional radix-8 encoding method, two operations (“inverse” and “plus 1”) are required to generate the complementary code of the negative PPs. However, the “plus 1” operation brings carry propagation, which will affect the speed and additional power consumption. Therefore, we propose a new method to deal with the negative PPs more efficiently and minimize the hardware resources.

Unlike the traditional Booth encoding, which represents negative and positive PPs separately, the proposed method encodes all PPs with their positive versions and a sign  $S_i$ . If the sign  $S_i$  is 1, indicating  $PP_i$  is negative, then  $PP_i$  is inverted before being input into the TCA. At the same time, as the signs  $S_i$  are connected to the carry-in ports of the adders, the “plus 1” operations can be inserted into the PPA process, thus avoiding the use of additional adders for the complement operation and saving the area cost and power consumption. The modified radix-8 encoding method encodes all PPs with their positive versions and signs  $S_i$ , as shown in Table 1.

### 2.2.2. Transverse Carry Array Structure for PPA

Figure 6 demonstrates the PPA process after the radix-8 Booth encoding. There are six PPs ( $I_0$ – $I_5$ ) that need to be added. The red dots are signs  $S_i$ , which are taken as carry-in signals and added with the PPs. The black dots are effective bits, and the white dots with the letter “E” are extension bits. Normally, those PPs are shifted and extended into 32-bit operands and, then, added together with five 32-bit adders. However, in this conventional method, only a part of the ports in the 32-bit adder are used effectively with effective digits, and the rest of the ports are used for shifted bits. For example, the lower 15 bits of  $I_5$  are “0”, which do not participate in the calculation, but reduce the rate of port utilization. Therefore, instead of using five 32-bit full-adders, the PPs are reduced with smaller bit-width adders such as 3 bits and 17 bits to save the hardware resources.



**Figure 6.** The PPA process of the sixteen 32-bit PPs.

The TCA adder-tree and the full 32-bit adder-tree were implemented in an FPGA at the synthesis level. Table 2 demonstrates the comparison of the hardware resources, delays, and power consumption after synthesizing. A thousand pairs of randomly generated input vectors were input to test the power consumption.

**Table 2.** Comparison results in the FPGA implementation.

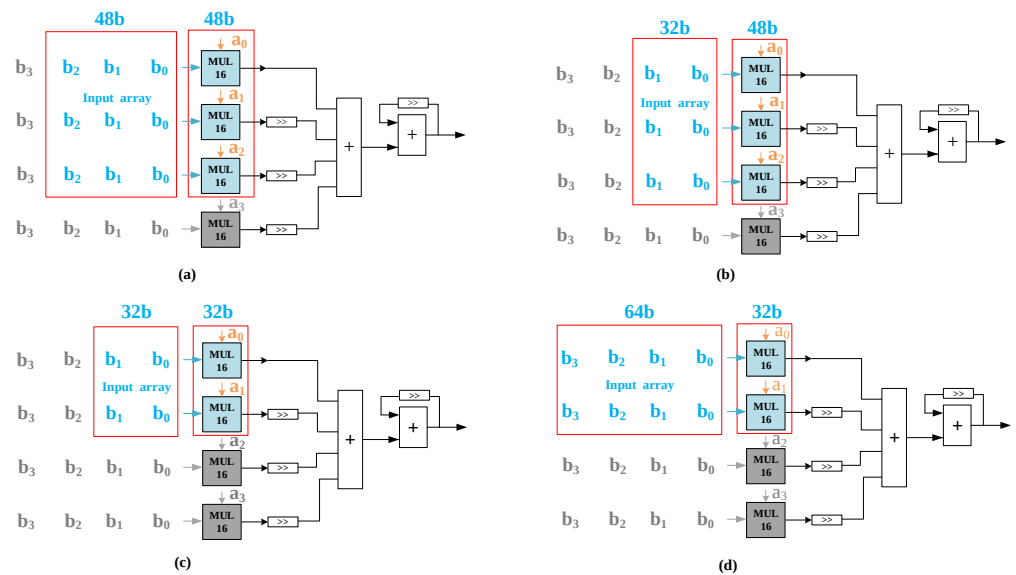
| Adders       | LUT | IO Buffer | Logic Delay (ns) | Route Delay (ns) | Power (W) |
|--------------|-----|-----------|------------------|------------------|-----------|
| full-32-bit  | 349 | 224       | 2.426            | 11.234           | 0.323     |
| 3/17-bit TCA | 210 | 148       | 2.62             | 14.384           | 0.319     |

As shown in Table 2, the number of LUTs and the power consumption were saved by 39.8% and 1.2%, respectively, compared to the full 32-bit adder-tree structure. However, the logic delay of the TCA structure increased by 7.9%, as the worst-case signal propagation delay path is from the upper right corner of the array to the high-order product bit output

at the bottom left corner of the TCA array. Therefore, the TCA structure is more suitable in a system that requires a smaller area, but cares little about speed.

### 2.3. Configurability of the IVS Multiplier

Benefiting from the modularized design, the MUL16 block can be selectively activated to save energy, and the IVS multiplier can be configured for sixteen different kinds of input conditions (e.g.,  $[16b, 32b, 48b, 64b] \times [16b, 32b, 48b, 64b]$ ). Note that since the basic element is a 16-bit multiplier, the operands are scaled with a limited step of 16 bits. Figure 7 demonstrates four different multiply cases:  $48 \times 48$  bits,  $32 \times 48$  bits,  $32 \times 32$  bits, and  $64 \times 32$  bits. The number of MUL16 used simultaneously is dependent on the bit-width of input-A, and the number of computing cycles is dependent on the bit-width of input-B. For example, in Figure 7b, input-A is 48 bits, while input-B is 32 bits. Therefore, only three MUL16 are used and the last MUL16 is closed. At the same time, the input array is turned into a 3 by 3 dimension matrix. The total calculation period requires three clock cycles.



**Figure 7.** Four different multiply cases: (a)  $48 \times 48$  bits; (b)  $32 \times 48$  bits; (c)  $32 \times 32$  bits; (d)  $64 \times 32$  bits.

Therefore, for faster computation speed, the longer operand should be taken as input-A, while for less area utilization and fewer operating units, the shorter operand should be taken as input-A.

The configurability of the IVS multiplier avoids unnecessary computing cycles and dynamic energy once the operands are not full-64 bits, which can significantly reduce the glitching activity, but it requires additional control circuitry to generate the gated clocks for various registers in the implementation.

## 3. Results and Discussions

The proposed 64-bit IVS multiplier and the MUL16 were simulated by Synopsys VCS and synthesized by Synopsys Design Compiler. IC Compiler was utilized for place-and-route with SMIC 40 nm CMOS technology, and Prime Time was used for STA and power analysis, while the voltage and temperature were set to 0.99 V and 125 °C.

### 3.1. Comparison of MUL16

The proposed MUL16 was compared with the standard 16-bit radix-8 Booth multiplier with two types of adder structures in the same conditions and technology. The clock of top-level designs was constrained to 20 ns. The experiment results are presented in Table 3.

**Table 3.** Comparison of 16-bit multipliers (clock frequency = 50 MHz).

| Types of MUL16                     | Delay (ns) | Area ( $\mu\text{m}^2$ ) | Power ( $\mu\text{W}$ ) |
|------------------------------------|------------|--------------------------|-------------------------|
| Standard radix-8 with array-adder  | 12.33      | 2530.28                  | 85.4                    |
| Standard radix-8 with Wallace-tree | 9.22       | 2471.01                  | 93.6                    |
| radix-8 with TCA                   | 12.99      | 2133.58                  | 80.22                   |

The results show that the proposed MUL16 has an approximate 6.32% and 14.29% improvement over the other two counterparts in power consumption and has a 15.67% and 13.65% reduction in area cost. However, the latency increased 5.35% and 40.88%, respectively, compared to the two baselines. The area reduction mainly benefits from the increased area utilization in the TCA structure, because all input ports of the 3/17-bit adders are used effectively without redundant ports. The power is also reduced due to the simpler circuit structure of MUL16.

### 3.2. Comparison of the 64-Bit IVS Multiplier

#### 3.2.1. Compared with the Traditional Multipliers

In the experiment of the 64-bit IVS multiplier, three conventional multipliers: the 64-bit Wallace-tree-based pipeline Booth multiplier, the traditional 64-bit array-based multiplier, and the 64-bit multiplier based on the architecture in [18], were implemented with the same technology for comparison. The maximum delay, area cost, power, number of clock cycles required during one computation, and energy per multiplication were collected. The energy per multiplication was calculated by Equation (6), where  $T_{\text{clk}}$  is the clock period, Power is the average power consumption of the multiplier, and  $N_{\text{clk}}$  is the number of clock cycles required for one multiplication, which is also the iteration period. The comparison results are listed in Table 4 and displayed clearly with bar graphs in Figure 8.

$$\text{Energy} = T_{\text{clk}} \times \text{Power} \times N_{\text{clk}} \quad (6)$$

**Table 4.** Comparison of 64-bit multipliers with different architectures (clock frequency = 13 MHz).

| Multiplier                    | Delay (ns) | Area ( $\mu\text{m}^2$ ) | Power (mW) | Number of Clocks | Energy (pJ/Mul.) |
|-------------------------------|------------|--------------------------|------------|------------------|------------------|
| Booth multiplier              | 13.36      | 55,007.52                | 0.448      | 1                | 34.49            |
| Array multiplier              | 49.21      | 28,914.95                | 0.155      | 1                | 11.96            |
| Full-parallel multiplier [18] | 25.35      | 35,402.95                | 0.0903     | 1                | 6.953            |
| IVS                           | 14.52      | 11,007.18                | 0.0495     | 4                | 15.58            |

In general, the area cost was roughly reduced by 79.9%, 61.9%, and 68.9% with respect to the conventional Booth multiplier, the array multiplier, and [18] because of the resource-sharing strategy, and the average power consumption was decreased by 88.95%, 68.06%, and 45.18%, respectively. This low-power property is especially beneficial for portable devices and wireless smart systems, where power is supplied by a small-sized battery with low capacity [22], because the lower discharge current increases the effective battery energy conversion ratio and the battery lifetime.

Figure 8 highlights the advantages of the proposed IVS multiplier in terms of area cost and power consumption. However, the maximum delay of the proposed multiplier is not the smallest when compared to the Booth multiplier, which uses a high-performance Wallace-tree for PPs' reduction, but 70.49% and 42.72% faster than the array multiplier and [18]. Besides, as the IVS multiplier spends four clock cycles to complete one  $64 \times 64$ -bit multiplication, the energy per multiplication is larger than that of the array multiplier and [18].

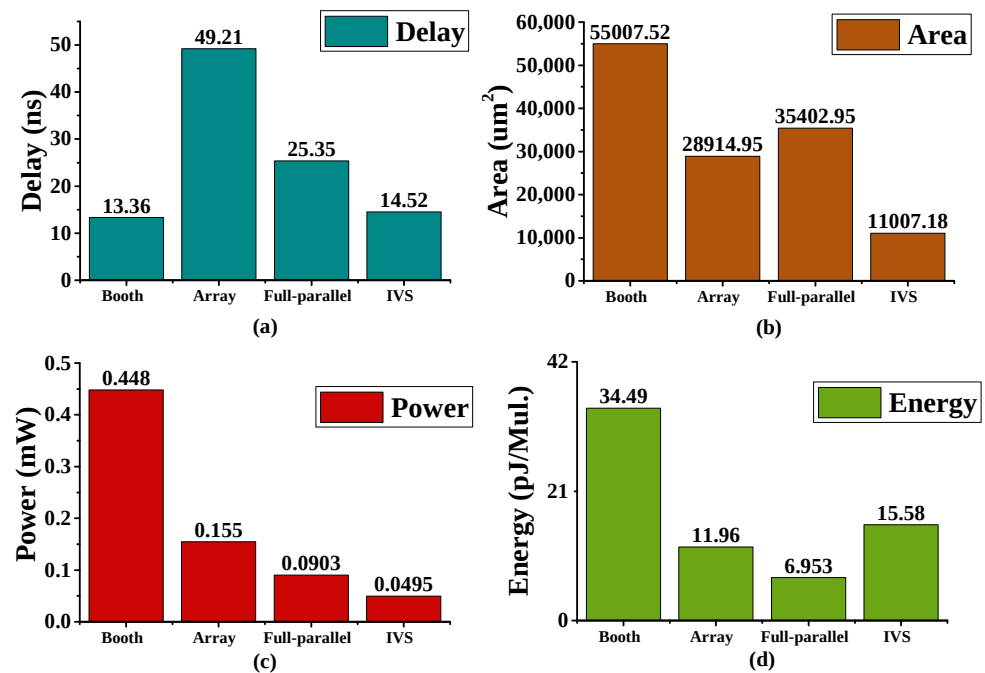


Figure 8. Comparisons in bar graphs. (a) Delay. (b) Area. (c) Power. (d) Energy.

### 3.2.2. Comparison with Some Low-Power Multipliers in the Literature

Table 5 tabulates the comparison results of power consumption and area cost in the proposed multiplier and some other popular multipliers with different architectures, including the tree-based parallel Booth multiplier, the array-based parallel Booth multiplier, the shift-and-add interactive multiplier, the logarithmic iterative multiplier, and the asynchronous iterative multiplier. For comparison purposes, the energy consumption and area cost were normalized with the formula in [23,24]. The results reveal that the IVS multiplier is qualified to be low-power and area-efficient.

$$\text{Norm.energy} = \frac{\text{Energy}}{\left(\frac{\text{Tech}}{40\text{nm}}\right) \times 15.58} \quad \text{Norm.area} = \frac{\text{Area}}{\left(\frac{\text{Tech}}{40\text{nm}}\right)^2 \times 11,007} \quad (7)$$

Table 5. Comparison of different multipliers in terms of energy per multiplication.

| Design                  | [3]        | [25]        | [4]       | [26]   | [6]    | This Work    |
|-------------------------|------------|-------------|-----------|--------|--------|--------------|
| Tech.                   | 45 nm      | 40 nm       | 130 nm    | 32 nm  | 55 nm  | 40 nm        |
| Feature                 | tree-based | array-based | shift-add | logar. | asyn.  | IVS          |
| Type                    | paral.     | paral.      | itera.    | itera. | itera. | paral-itera. |
| Width                   | 64b        | 16b         | 16b       | 32b    | 16b    | 64b          |
| Area (um <sup>2</sup> ) | 77,841     | 6436(tr)    | 3903      | 3102   | 1618   | 11,007       |
| Energy (pJ/Mul.)        | 26.9       | 1.08        | 48.5      | 13.7   | 3.7    | 15.58        |
| Norm. energy            | 2.11       | 1.53        | 21.148    | 6.06   | 2.59   | 1            |
| Norm. area              | 5.38       | N.A         | 0.54      | 1.42   | 1.243  | 1            |

tr: based on the number of transistors.

#### 4. Conclusions

In this paper, a novel low-power and area-efficient precision configurable 64-bit IVS multiplier was proposed. The IVS architecture reuses four 16-bit radix8 Booth multipliers and allows the input data to move like a systolic array, reducing the area and power consumption by 61.9% and 45.18% with respect to its counterparts. For the 16-bit sub-multiplier, a PPA structure called the TCA was designed to increase the hardware resource utilization, which eliminates redundant ports. The FPGA implementation results show that the TCA structure could save 39.8% of the LUTs and 1.2% of the power dissipation when compared to the conventional full-32-bit adders. The precision of the proposed multiplier can also be scaled owing to the modularized design with a step of 16 bits. Compared to some up-to-date low-power multipliers with different structures in the literature, the IVS multiplier also shows great priority in energy and area savings. The proposed IVS multiplier can be applied in wireless smart products, which consider low-power and small-area more than high throughput.

In the future, we plan to apply the proposed 64-bit IVS multiplier to portable devices such as smart cards to verify its effectiveness in a real application. Furthermore, this work will be taped out and tested on a chip.

**Author Contributions:** Conceptualization, X.T. and D.S.; formal analysis, X.T.; funding acquisition, D.S.; investigation, X.T. and Y.L.; methodology, X.T. and C.L.; project administration, X.T. and D.S.; resources, X.T.; software, Y.L.; supervision, D.S.; writing—original draft, X.T.; writing—review and editing, X.T. and D.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the project of the Ministry of Science and Technology of the People's Republic of China Grant Number 2020AAA0109102.

**Institutional Review Board Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors are highly grateful to the Nanjing Institute of Intelligent Technology (NIIT), the Institute of Microelectronics of the Chinese Academy of Sciences (IMECAS), and the University of the Chinese Academy Research of Sciences for hosting the research team.

**Conflicts of Interest:** The authors declare no conflict of interest.

#### Abbreviations

The following abbreviations are used in this manuscript:

|        |                               |
|--------|-------------------------------|
| IVS    | Input Vector Systolic         |
| CNN    | Convolutional Neural Network  |
| TCA    | Transverse Carry Array        |
| PPA    | Partial Products Accumulation |
| BZ-FAD | Bypass Zero Feed A Directly   |
| PPs    | Partial Products              |
| MP     | Multi-Precision               |
| MUL16  | 16-bit Integer Multiplier     |
| CSA    | Carry-Save Adder              |
| RCA    | Ripple-Carry Adder            |
| IC     | Integrated Circuits           |
| VCS    | Verilog Compile Simulator     |
| STA    | Static Timing Analysis        |

#### References

1. Dhem, J.F. Design of an Efficient Public-Key Cryptographic Library for RISC-Based Smart Cards. Ph.D. Thesis, UCL-Université Catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium, 1998.
2. Newell, D.; Duffy, M. Review of Power Conversion and Energy Management for Low-Power, Low-Voltage Energy Harvesting Powered Wireless Sensors. *IEEE Trans. Power Electron.* **2019**, *34*, 9794–9805. [[CrossRef](#)]

3. Chaithra, T.; PradeepKumar, S.; Nandini, K.; Harshitha, S.; Ankitha, K. ASIC realization and performance evaluation of  $64 \times 64$  bit high speed multiplier in CMOS 45 nm using Wallace Tree. In Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 19–20 May 2017; pp. 1115–1119. [\[CrossRef\]](#)
4. Mottaghi, M.D.; Ali, A.K.; Massoud, P. BZ-FAD: A low-power low-area multiplier based on shift-and-add architecture. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2009**, *17*, 302–306. [\[CrossRef\]](#)
5. Zhang, X.; Boussaid, F.; Bermak, A. 32 Bit  $\times$  32 Bit Multiprecision Razor-Based Dynamic Voltage Scaling Multiplier With Operands Scheduler. *IEEE Trans. Very Large Scale Integr. Syst.* **2014**, *22*, 759–770. [\[CrossRef\]](#)
6. You, H.; Hei, Y.; Yuan, J.; Tang, W.; Bai, X.; Qiao, S. Design of low-power low-area asynchronous iterative multiplier. *IEICE Electron. Express* **2019**, *16*, 20190212. [\[CrossRef\]](#)
7. Nandan, D.; Kanungo, J.; Mahajan, A. An efficient VLSI architecture for iterative logarithmic multiplier. In Proceedings of the 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2–3 February 2017; pp. 419–423. [\[CrossRef\]](#)
8. Lee, C.Y.; Horng, J.S.; Jou, C.I.; Lu, E. Low-complexity bit-parallel systolic Montgomery multipliers for special classes of GF(2/sup m/). *IEEE Trans. Comput.* **2005**, *54*, 1061–1070. [\[CrossRef\]](#)
9. Yong, K.M.; Hussin, R.; Kamarudin, A.; Ismail, R.C.; Isa, M.N.M.; Naziri, S.Z.M. Design and Analysis of 32-Bit Signed and Unsigned Multiplier Using Booth, Vedic and Wallace Architecture. *J. Phys. Conf. Ser.* **2021**, *1755*, 012008. [\[CrossRef\]](#)
10. Booth, A.D. A signed binary multiplication technique. *Q. J. Mech. Appl. Math.* **1951**, *4*, 236. [\[CrossRef\]](#)
11. Chang, W.Y.; Jen, C.W. High-speed Booth encoded parallel multiplier design. *IEEE Trans. Comput.* **2000**, *49*, 692–701. [\[CrossRef\]](#)
12. Huang, Z.; Ercegovac, M. High-performance low-power left-to-right array multiplier design. *IEEE Trans. Comput.* **2005**, *54*, 272–283. [\[CrossRef\]](#)
13. Wey, C.; Li, J. Design of reconfigurable array multipliers and multiplier-accumulators. In Proceedings of the 2004 IEEE Asia-Pacific Conference on Circuits and Systems, Tainan, Taiwan, 6–9 December 2004; Volume 1, pp. 37–40. [\[CrossRef\]](#)
14. Praveen Kumar, M.; Sivanantham, S.; Balamurugan, S.; Mallick, P. Low power reconfigurable multiplier with reordering of partial products. In Proceedings of the 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies, Thuckalay, India, 21–22 July 2011; pp. 532–536. [\[CrossRef\]](#)
15. Tu, J.; Van, L. Power-efficient pipelined reconfigurable fixed-width Baugh-Wooley multipliers. *IEEE Trans. Comput.* **2009**, *58*, 1346–1355. [\[CrossRef\]](#)
16. Kuang, S.; Wang, J. Design of Power-Efficient Configurable Booth Multiplier. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 568–580. [\[CrossRef\]](#)
17. Sjalander, M.; Drazdziulis, M.; Larsson-Edefors, P.; Eriksson, H. A low-leakage twin-precision multiplier using reconfigurable power gating. In Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, 23–26 May 2005; Volume 2, pp. 1654–1657. [\[CrossRef\]](#)
18. Li, K.; Mao, W.; Zhou, J.; Li, B.; Yang, Z.; Yang, S.; Du, L.; Huang, S.; Yu, H. A Vector Systolic Accelerator for Multi-Precision Floating-Point High-Performance Computing. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *99*, 1. [\[CrossRef\]](#)
19. Christoph, N.; Michael, M.; Frank, K. Evaluation of the back-end design overhead for ASIC implementations of large-operand multipliers targeting resource-constrained environments. In Proceedings of the 22nd Austrian Workshop on Microelectronics (Austrochip), Graz, Austria, 9 October 2014; pp. 1–6. [\[CrossRef\]](#)
20. Fried, R. Minimizing energy dissipation in high-speed multipliers. In Proceedings of the 1997 International Symposium on Low Power Electronics and Design, Monterey, CA, USA, 18–20 August 1997; p. 214. [\[CrossRef\]](#)
21. Pallavi, C.; Rajani, C. Comparative Analysis of 16-Bit Booth Multiplier Using Radix-4 and Radix-8 Encoding Technique. *Int. J. Adv. Sci. Technol. (IJAST)* **2020**, *29*, 62–75.
22. Mikhaylov, K.; Tervonen, J. Experimental Evaluation of Alkaline Batteries's Capacity for Low Power Consuming Applications. In Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 26–29 March 2012; pp. 331–337. [\[CrossRef\]](#)
23. Lin, Y.W.; Liu, H.Y.; Lee, C.Y. A dynamic scaling FFT processor for DVB-T applications. *IEEE J. Solid-State Circuits* **2004**, *39*, 2005–2013. [\[CrossRef\]](#)
24. Chen, Y.; Lin, Y.; Tsao, Y.; Lee, C. A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems. *IEEE J. Solid-State Circuits* **2008**, *43*, 1260–1273. [\[CrossRef\]](#)
25. Chang, Y.J.; Cheng, Y.C.; Liao, S.C.; Hsiao, C.H. A low power radix-4 Booth multiplier with pre-encoded mechanism. *IEEE Access* **2020**, *8*, 114842–114853. [\[CrossRef\]](#)
26. Kim, H.; Kim, M.S.; Barrio, A.A.D.; Bagherzadeh, N. A Cost-Efficient Iterative Truncated Logarithmic Multiplication for Convolutional Neural Networks. In Proceedings of the 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), Kyoto, Japan, 10–12 June 2019; pp. 108–111. [\[CrossRef\]](#)