Multiplier Architectures: Challenges and Opportunities with Plasmonic-based Logic

(Special Session Paper)

Eleonora Testa*, Samantha Lubaba Noor†, Odysseas Zografos‡,
Mathias Soeken*, Francky Catthoor‡§, Azad Naeemi†, and Giovanni De Micheli*
*EPFL, Lausanne, Switzerland
† Georgia Institute of Technology, Atlanta, USA
‡IMEC, Leuven, Belgium
§KU Leuven, Leuven, Belgium

***Abstract*—Emerging technologies such as plasmonics and photonics are promising alternatives to CMOS for high throughput applications, thanks to their waveguide's low power consumption and high speed of computation. Besides these qualities, these novel technologies also implement logic functionalities uncommon to traditional technologies that can be beneficial to existing CMOS architectures. In this work, we study how plasmonic-based devices can complement CMOS technology to achieve a more efficient implementation of multiplier architectures, which are the core of state-of-the-art data- and signal-processing circuits. A critical part of modern multipliers is the partial-product reduction step, used to reduce the partial product tree into a 2-input addition. In CMOS technology, this step is achieved by using compact and fast counters. On the other hand, the proposed plasmonic cells naturally implement counters of 3-, 9- and 27-inputs within a few logic levels at ultra-high speed. Thus, we present novel multiplier architectures, which take advantage of large plasmonic-based counters to reduce the number of cells and logic levels in the partial product reduction step of the multiplication. Our experimental results show that 3 levels and 30 counters are needed when 27-input cells are used. On the other side, 6 levels and 72 counters are employed with 9-input cells. Finally, we present various 16×16 multiplier implementations mixing 9- and 27-input cells, focusing on the trade-off in the number of counters, levels, and area of each architecture.**

I. Introduction

To overcome the intrinsic scaling limitations of CMOS, emerging technologies are going to play a key role in the near future [1]. Novel emerging technologies such as plasmonics and photonics devices are promising alternatives to CMOS, because of the low propagation losses and their high speed of computation [2], [3]. In particular, plasmonic devices based on *surface plasmon polaritons* (SPP, [4]) overcome the limits of nanoscale photonics devices and efficiently implement Boolean functions uncommon to traditional CMOS technologies such as majority [3], [5] and threshold-based logic functions [6]. Furthermore, the integrated electric field at the output of a single plasmonic device is proportional to the number of input bits that are equal to 1 [3]. Thus, each plasmonic device intrinsically implements a circuit able to count the number of 1s of an input stream within one logic cell. Such circuit is known as digital counter, and, to our knowledge, this capability of plasmonics logic has never been exploited before in state-of-the-art literature.

Digital counters are generally important in the implementation of modern arithmetic architectures, and they play a key role in the implementation of multipliers. Multiplier architectures are fundamental in the design of microprocessors, digital signal processors, and integrated data-processing [7]. Furthermore, many computations and algorithms also involve multipliers; as an example, efficient architecture implementations can be beneficial for highly computational intensive algorithms, such as *convolutional neural network* (CNN, [8]). While there are many and diverse approaches for the design of multipliers trading-off area and delay, we are interested in parallel multipliers [9], [10]; in particular, in n-bit 2-input $n \times n$ Wallace-tree multipliers [9]. The key component of a Wallace-tree multiplier is the partial products reduction step used to reduce the sum of the partial products into a 2-input addition. Many and diverse solutions to speed up the partial-product reduction array have been proposed [7], [11], [12], [13]. In CMOS technology, this is usually achieved by using compact and fast counters over 3 inputs (also called carry-save adders). Here, we propose a plasmonic-based device that naturally implements counters up to 27 inputs within few logic levels, retaining a quasi-constant cell delay for different number of inputs.

In this work, we study how plasmonic-based logic can complement CMOS technology to achieve a more efficient implementation of a multiplier architecture. We present a novel 16×16 multiplier architecture, which takes advantage of large plasmonic-based counters to reduce the number of cells and logic levels in the partial product reduction step of the Wallace-tree multiplication architecture. As different plasmonic cells are available (3-, 9-, and 27-input cells) we propose various 16×16 multiplier architectures trading-off the number of counters and logic levels. The experimental results show that, when using 27-input plasmonic counters, 3 levels and 30 counters are sufficient to reduce the height of the partial product tree; when 9-input counters are involved, instead, 72 counters and 6 levels are necessary. Hybrid solutions (i.e., using both 27- and 9-input counters) have a number of levels between 3 and 6, with a number of counters up to 53. We also present an area estimation of each plasmonic logic counter assuming a structure similar to

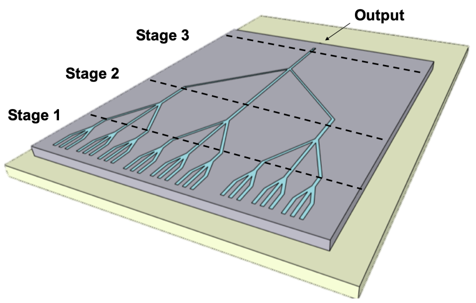


Fig. 1: Structure of the 27-input primitive from [3]. The gate structure is simulated using 3-D simulation in the finite difference time domain (FDTD) solver of Lumerical Solution [14].

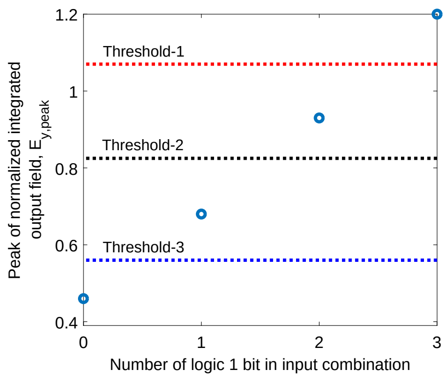


Fig. 2: Peak of the normalized integrated electric field at the output of the 3-input primitive. The output value is proportional to the number of 1s in the input combination.

the one presented in [3]. Each plasmonic cell is simulated using 3-D simulation in the finite difference time domain (FDTD) solver of Lumerical Solution [14]. We describe results over two different layouts, focusing on the trade-off between area and latency for the different 16×16 multiplier architectures.

The remainder of the paper is organized as follows. Section II introduces the preliminaries on plasmonic technology and the functionality of the cells, while Section III explains how to use large plasmonic-based counters for the partial-product reduction step of Wallace-tree multipliers. Section IV illustrates the results over a 16×16 multiplier together with an area evaluation of the plasmonic cells, and Section V concludes the paper.

II. Plasmonic Gates

In this section, we detail the plasmonic devices. First, we present the structure of a single plasmonic gate and give an example of a 27-input cell. Then, we illustrate the functionality of 3-, 9- and 27-input plasmonic devices.

A plasmonic-based device as implemented in [3], [6] is based on the propagation of *surface plasmon polaritons* (SPP, [4]), which are electromagnetic waves propagating at the interface between a dielectric and a metal. It thus consists of a waveguide used to transmit the information, built with *metal-insulator-metal* (MIM) configuration (e.g., Ag-SiO_2-Ag). The 3-, 9- and 27-input plasmonic devices have already been discussed in [3], [5]. We consider the 27-input structure, shown in Fig. 1, as an example. It consists of three stages: each stage having the 3-input primitives, a combiner, and an output region. This is the largest building block that can be implemented today using the mentioned plasmonic technology. This is due to the fact that the propagation losses of SPP put a limitation on the maximum number of cascaded stages (i.e., the number of levels of the circuits). Currently, it is not efficient to have more than three stages, which means that, after the third stage, either an amplifier or a converter to voltage domain is necessary [3].

The important feature of plasmonic logic and wave computing is the ability to efficiently implement functionalities that are complex to realize in CMOS, e.g., large majority and threshold gates [3], [5]. For the logic operation, the phase of the plasmonic wave is considered as the computational state variable, i.e., phase $180°$ for logic 0 and $0°$ for logic 1. The 27-input gate can be used to implement not only a majority-of-twenty-seven-input, but also threshold functions [6]. The same holds for the 3- and 9-input primitives.

In this work, we exploit each plasmonic device as a counter. The peak of the normalized integrated electric field at the output of each 3-input primitive is depicted in Fig. 2. The output is proportional to the number of input signals equal to 1. In other words, the plasmonic cell implements a circuit that counts the number of 1s of the input bit-stream, i.e., a digital counter. As the 9- and 27-input gates present the same functionality, it means that plasmonic cells implement counters up to 27 inputs within 3 plasmonic levels. Note that plasmonic circuits operate at a frequency of around $1THz$ and ultra-low energy levels that cannot be achieved by their CMOS counterparts at this frequency [6]. In the next section, we will explain how to use these properties to speed up the partial product reduction step in modern multipliers.

It is worth mentioning that we envision each plasmonic device to work as proposed in [6]. It means that at the output of each logic primitive, a plasmonic waveguide carrying a reference signal merges with the output signal [3], [6] to convert the phase information into amplitude information, and thus to avoid THz phase detection. A plasmonic waveguide photodetector [15] can be tightly integrated with the output plasmonic waveguide, converting the amplitude information to the electrical domain.

III. Plasmonic Devices for Partial Product Reduction

Plasmonic devices efficiently implement digital counters up to 27-input. In this section, we take advantage of such property to build a more compact and faster multiplier architecture, by reducing the number of counters and logic levels in the partial product reduction step of Wallace-tree multipliers.

Given two n bitstreams a and b, a Wallace-tree architecture for the multiplication between a and b comprises three steps: (i) the array of n^2 AND gates to compute the partial products between each bit of a and b; (ii) the step made of counters to reduce the sum of the partial products into a 2-input addition;

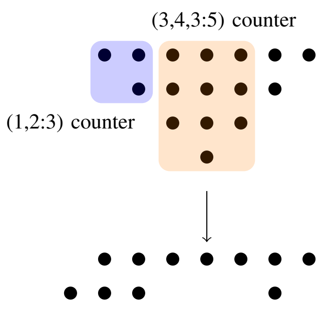


Fig. 3: 4×4 multiplier product array, reduced to 2-input addition by 2 counters: (3,4,3:5) and (1,2:3).

(iii) the final 2-input addition [9]. Note that, in the following, we envision step (i) and step (iii) to be implemented using standard CMOS technology; we thus focus on the implementation of the second step with only plasmonic counters. In general, in a Wallace-tree implementation, step (ii) is obtained using carry-save adders. The number of levels l required in a Wallace-tree to reduce the height h of the tree to 2 is given by [16]:

$$l = \lceil log_{1.5}(\frac{h}{2}) \rceil \tag{1}$$

As an example, for a 4-input multiplier, $h = 4$, and 2 levels of carry-save adders are needed.

In the following discussion, we refer to $(x : y)$ counters [10] as circuits that count the number of 1 bits over x input bits of *equal weights*, and output of y bits of increasing weight. For instance, a carry-save adder is a (3:2) counter. The output is a y-bit word whose value is the sum of the inputs bit, and it is given by:

$$v = \sum_{i=0}^{x-1} b_i \tag{2}$$

where b_i is the binary value of the i^{th} input bit. This class of counters is used in multiplier architectures [9], [10] to reduce the partial product tree by dividing it into columns of three partial products each. Many and diverse optimizations using CMOS implementation of (7:3), and (5:3) fast counters have been proposed [13], [16]; a (4:2) counter has also been investigated [17].

In this paper, we extend the definition of counters [13] to consider successively weighted input columns. The sum of the columns is produced by taking the weights into account. We refer to these counters [13] as $(x_{k-1}, x_{k-2}, \ldots, x_0 : y)$, where k is the number of inputs columns, x_i is the number of bits in the column of weight 2^i. The output is a y-bit word whose value is the sum of the inputs bit, and it is given by:

$$v = \sum_{i=0}^{k-1} \sum_{j=0}^{c_i-1} b_{ij} 2^i \tag{3}$$

where b_{ij} is the value of bit j in column i. As an example, a (5,5:4) counter has been presented in [13].

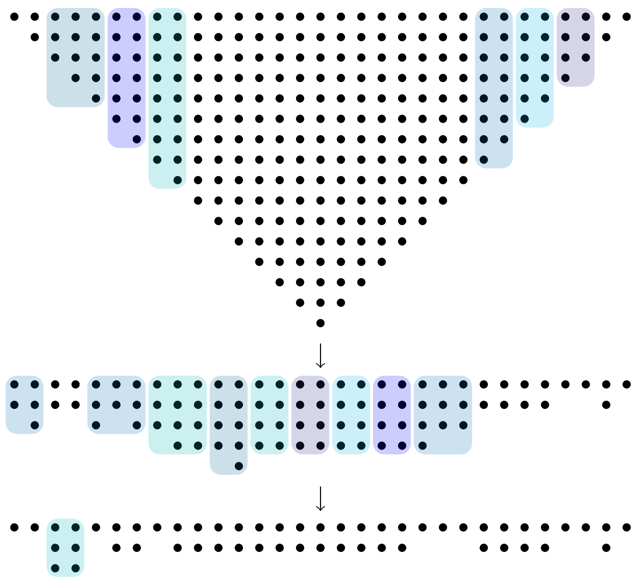


Fig. 4: Reduction of the partial product tree of a 16×16 multiplier using 27-input plasmonic devices. Counters over more than one column are represented in blue; while counters over one column are not represented to ease the notation.

It follows that (i) a 9-input plasmonic device implements the following counters: (9:4), (3,3:4); (ii) while a 27-input plasmonic gate can be used as: (27:5), (4,4,3:5), (9,9:5). Note that, while complete utilization of the inputs is desirable, this is not a necessary condition. It thus follows that all "underutilized" arrangements of the counters are possible.

The proposed plasmonic counters are employed here to reduce the partial product tree of multipliers. Consider as an example the 4×4 multiplier product array of Fig. 3; note that the partial product terms are represented as $\bullet$ to ease the notation. The partial product tree consists of 16 terms distributed over 7 columns of various sizes. Only one level and two counters are needed in order to reduce the height of the tree. The blue counter is a (3,4,3:5) counter, while the red one has $2 + 1 \times 2 = 4$ inputs distributed over 2 columns. In the next section, we present results in terms of number of counters, levels, and area over a 16×16 multiplier.

IV. Results

In this section, we present the experimental results. We use as running example a 16×16 multiplier as this is commonly used in CNN hardware accelerators [18]. First, we illustrate different architectures: (i) one that use only 27-input devices, (ii) one with 9-input plasmonic, and (iii) a blend of those (called "hybrid" mapping in the following discussion). Then, we conclude with an area estimation of the plasmonic cells obtained by simulation and present various architectures trading-off area, number of cells, and logic levels.

A. *Architectures for the* 16×16 *Multiplier*

As discussed in Section III, the large number of inputs of plasmonic counters allows us to reduce the number of counters and levels in the reduction of the partial product tree. Furthermore, we consider more than one column at the same time, by taking into account different weights. We study here a 16×16 multiplier architecture, whose tree is shown in the top part of Fig. 4. It consists of a partial product tree of $16 \times 16 = 256$ terms, over 31 columns of different sizes. We consider plasmonic cells having 3, 9, and 27 inputs, thus counters: (27:5), (4,4,3:5), (9,9:5), (9:4), (3,3:4) and (3:2). First, we present (together with an example from Fig. 4) an architecture that uses 27-input cells. It means that, when possible, the first choice is using 27-input cells.

The steps for the partial product reduction tree using 27-input cells are presented in Fig. 4. The first level of plasmonic counters consists of 20 parallel counters, with number of inputs ranging from 9 to 25. Note that counters over one column are not reported. The small number of counters is achieved due to (i) the large size of the counters, and (ii) by considering different weighted columns. The second level of counters is depicted in the second step of Fig. 4, and consists of 9 counters, from 25 to 7 inputs. The last level only needs one more counter (3,3:4). It means that, when using mostly 27-input counters, the architecture can be implemented using 30 counters, over 3 levels. This result is reported in the first columns of Table II for the '27-input cells' architecture. Note that the number of levels refers to the number of counters on the topological critical path. It is also worth mentioning that this example represents one of the many and diverse architectures that can be implemented with the plasmonic counters.

The same process can be achieved when only counters up to 9-input are allowed, i.e., 27-input cells are not involved in the reduction step. In this case, the architecture needs 6 levels, and 72 counters (see '9-input cells' architecture from Table II). It is worth mentioning that while this could seem worse than the previously described architecture, since the area of a 9-input cell is up to $10\times$ smaller than the 27-input one, this results in a smaller overall area. As a consequence, complex global trade-offs exist in the overall search space. Some solutions with a blend of 27- and 9- input plasmonic counters (called *hybrid solutions*) can also be built. These architectures range between 6 and 3 levels, with a number of counters from 39 to 53. Results for '# counters' (number of counters) and '# levels' (number of counters on the topological critical path) for 10 hybrid architectures are summarized in the first two columns of Table II.

B. Area Estimation and Results

In this section, we propose a layout and area evaluation of plasmonic cells when used as counters based on simulative models developed at GeorgiaTech. We also present the trade-off in the number of levels, counters, and area of various 16×16 multiplier architectures, when built using plasmonic-based logic cells.

To use the plasmonic waveguides as counters, different building blocks need to be considered. Each logic counter cell should include (i) a plasmonic logic waveguide (described in Section II), (ii) a plasmonic power splitter, (iii) reference waveguides, and (iv) integrated detectors (see Fig. 5).

The structure of the logic gate has already been explained in Section II. In order to obtain a counter-like behavior, it is possible to count the number of logic `1`s in the input data from the integrated peak output of the logic gate using different threshold levels as shown in Fig. 2. For example, in the 3-input primitive of Fig. 2, 'Threshold-2' decides the *most significant bit* (MSB) of the counter output. If the peak of the output is greater than 'Threshold-2', MSB has value equal to `1`; otherwise, MSB will be `0`. According to the value of the MSB, 'Threshold-1' and 'Threshold-3' are employed to obtain the value of the next significant bit. In other words, if the MSB is `1`, 'Threshold-1' decides the next significant bit, while 'Threshold-3' is used if the MSB is `0`. As an example, for the (3:2) counter, the output of the plasmonic gate needs to be split into two signals to decide the MSB and the *least significant bit* (LSB). The mentioned threshold functionalities can be implemented in plasmonic by *reference waveguides*, as described in [6]. In order to split the output of the logic gate, instead, we introduce plasmonic *power splitters*.

A $1 \times n$ power splitter consists of Y cascaded branches dividing the output optical power of the main plasmonic gate among n plasmonic waveguides. The number of split outputs n depends on the number of output bits of the $(x : y)$ counter. As n increases, the SPP waves have to travel longer paths and the transmission efficiency of the splitter decreases due to losses in metal. The transmission efficiency is defined as follows:

$$T = \sum_{k=1}^{n} P_{out}^{k} / P_{in} \tag{4}$$

where P_{in} denotes the optical power into the splitter and P_{out}^{k} denotes the power flowing out of the k^{th} split output waveguide. The splitter has been simulated in the FDTD solver of Lumerical Solution [14]: T decreases from 80% to 40.4% as n increases from 2 to 5. A reference waveguide and an integrated photodetector are associated with each output waveguide of the power splitter. For the photodetector, a high-speed metal-semiconductor-metal plasmonic photodetector with a narrow Ge slot as an active region can be used [15].

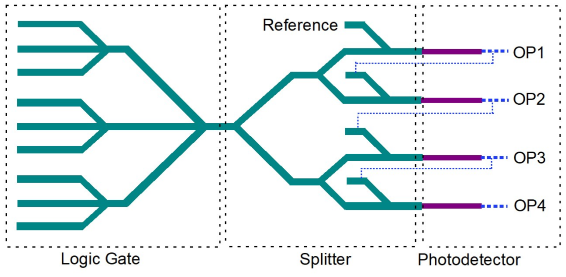
We used the proposed blocks to estimate the area of each plasmonic-based counter. First, the area of the plasmonic gate depends on the number of stages or number of inputs, with 3-, 9-, and 27-input gates having areas of $0.84\mu m^2$, $9.13\mu m^2$, and $83.4\mu m^2$, respectively. Cross-sectional dimensions of the input side waveguides, splitter waveguides, and photodetectors are $60nm \times 100nm$, $120nm \times 100nm$, and $90nm \times 100nm$, respectively. To maintain a low crosstalk noise, we assumed that the separation of the SiO_2 slots is $p = 300nm$ for the entire plasmonic waveguide configuration. The bending angle of the bent waveguides in both logic gate and Y branches is chosen to be $\theta = 35.7^o$ as in [3]. For the power splitter, the width (W_s) can be calculated from the number of splitted outputs n, the slot width w, and the slots separation p:

$$W_s = 2nw + (2n-1)p \tag{5}$$

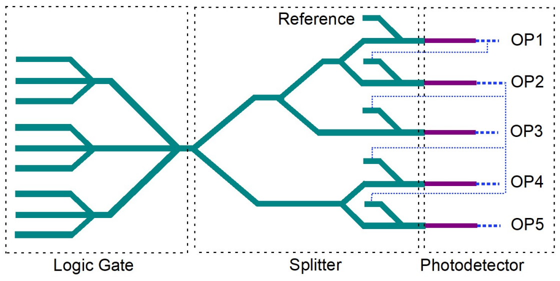
The length of the splitter (L_s) depends on n which determines the number of required Y branches $(n-1)$ as well as the length associated with the longest bent waveguide $(0.5n(p + w)/\tan(\theta))$. For instance, $L_s = 0.6\mu m$, $1.6\mu m$, and $1.9\mu m$ for the 1×2, 1×3, and 1×4 splitters, respectively. The

TABLE I: Combined area of the splitter, reference waveguide, and photodetector

# Output in splitter	# Stages in splitter	Combined length [μm]	Combined width [μm]	Combined area [μm^2]
2	1	2.1	1.4	2.9
3	2	3.1	2.2	6.8
4	2	3.4	3.1	7.2
5	3	4.5	3.9	17.5
6	3	5.2	4.7	24.7
7	3	5.6	5.6	31.5
8	3	5.8	6.4	37.2
9	4	7	7.3	50.7
10	4	7.9	8.1	64.2
11	4	8.73	8.94	78
12	4	9.39	9.78	91.8
13	4	9.9	10.62	105.1
14	4	10.26	11.46	117.6
15	4	10.48	12.3	129



(a)



(b)

Fig. 5: Different layout for the plasmonic counter. (a) layout 1: minimum area, larger latency; (b) layout 2: larger area, minimum latency.

reference waveguide adds an additional length of $500nm$ while the length of the photodetector is $L_p = 1\mu m$. The combined areas of the splitter, reference waveguide, and photodetector are summarized in Table I for number of outputs ('# Outputs') from 2 to 15. This covers all possible counters configurations, from 3 to 27 inputs.

Using the building blocks presented so far, we propose two layouts for a specific $(x : y)$ counter, as shown in Fig. 5. We consider a (9:4) counter as running example. Layout 1 (Fig. 5 (a)) has 4 outputs: each output evaluates one bit of the counter output $(y_0y_1y_2y_3)$. First, MSB (y_0) is computed from OP1; from the value of the MSB, CMOS circuits set

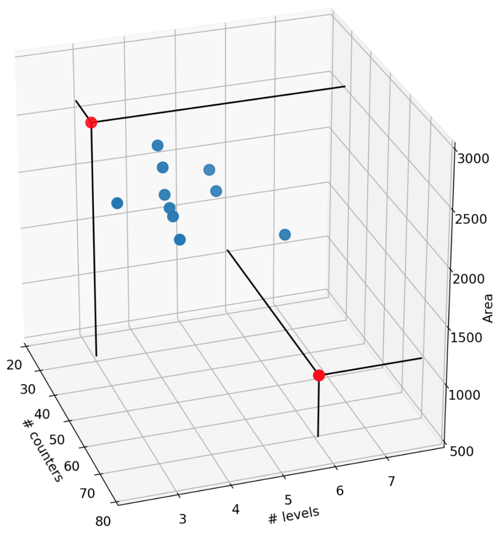


Fig. 6: Trade-off: number of counters, number of levels and Area1. Hybrid solutions are in blue.

the reference level for OP2 which calculates the next bit. The process continues until LSB (y_3) is decided from OP4. Layout 2 differs from Layout 1 in the evaluation of y_2 and y_3. As a consequence, Layout 1 is more compact (one splitter 1×4 vs 1×5) but has higher latency as compared to Layout 2. While in Layout 1 all the 4 outputs are sequentially computed one by one, only 3 sequential stages are needed in Layout 2.

We used the two presented layouts and the data from Table I to evaluate the area of each plasmonic counter. The total area of each plasmonic counter-cell is given by the area of the plasmonic cell (described before) plus the combined area of splitter, photodetector, and references (A_{SRP}) as proposed in Table I:

$$A_{\text{counter}} = A_{\text{plasmonic\_cell}} + A_{SRP} \tag{6}$$

For each cell, two total area estimations are evaluated, corresponding to using Layout 1 or Layout 2. As an example, the (9:4) counter has Area1 (using Layout 1) equal to $9.13 + 7.2 = 16.33\ \mu m^2$, while Area2 (using Layout 2) is $9.13 + 17.5 = 26.63\ \mu m^2$. It is worth mentioning that counters over multiple columns (e.g., (3,3:4)) have the same area as their one-column counterparts (e.g., (9:4)). Furthermore, in case some inputs of the counters are not used, the area is decreased as smaller splitters can be employed. For example, the area of an (8:4) counter is different from the area of a (9:4), even though they are implemented using the same plasmonic cell.

The results for the area evaluation are presented in the second part of Table II. The table lists different implementations of the reduction step of a 16×16 multiplier when mapped using various plasmonic cells. The first row shows results for the 27-input architecture presented in Section IV-A, with 3 levels and 30 counters. The first area estimation (column 'Area1') is the one using Layout1, while the second area estimation (column 'Area2') is the one obtained from Layout2. The results for 'Area3' and 'Area4' compare to a hypothetical future downscale of Layout1 by $10\times$ and $20\times$, respectively. The same results are

TABLE II: Experimental results for 16×16 multiplier

Mapping	# Counters	# Levels	Area1 [μm^2]	Area2 [μm^2]	Area3 [μm^2]	Area4 [μm^2]
27-input cells	30	3	2577	3869	258	138
9-input cells	72	6	1040	1240	124	121
hybrid1	46	3	2196	3155	229	122
hybrid2	52	4	1936	2790	201	136
hybrid3	43	5	2106	3301	218	115
hybrid4	46	4	2087	3078	211	150
hybrid5	53	6	1861	2791	191	149
hybrid6	48	4	2055	3068	208	148
hybrid7	42	4	2357	3536	236	158
hybrid8	43	4	2141	3416	223	115
hybrid9	39	5	2212	3486	226	119
hybrid10	39	4	2487	3723	253	135

presented for the 9-input architecture, having 6 levels and 72 counters. The "hybrid" rows describe the number of counters, levels, and area estimate for other 10 architectures having 27-input cells interleaved with 9-input counters.

Our results show that while the 9-input architecture has twice as many counters and levels compared to 27-input one, it results in areas which are more than $2\times$ smaller. Furthermore, the difference between 'Area1' and 'Area2' for the 9-input cells (1040 vs 1240) is much smaller than the one of 27-input ones (2577 vs 3869). The solution with 9-cells presents the smaller areas overall. Fig 6 shows the trade-off results in the number of counters, number of counters on the critical path (levels), and area (evaluated as Area1). The solution with 27- and 9- input cells are highlighted in red. As previously described, even if the 9-input solution has a larger number of counters and levels, it results in a much smaller area. Hybrid solutions present the number of levels, counters, and area in between the two corners cases.

V. Conclusion

In this work, we propose a novel plasmonic-based implementation of digital counters. Plasmonic logic can efficiently implement counters up to 27-input, within few logic levels. We thus take advantage of this property to successfully reduce the height of the partial product tree of multipliers. Our results show that for a 16×16 multiplier, 3 logic levels and 30 counters are needed when 27-input cells are used. We also present an area evaluation and different layouts of the plasmonic cells, trading-off area and latency.

Acknowledgment

This research was supported by the EPFL Open Science Fund, by the Swiss National Science Foundation (200021-169084 MAJesty) and by the ERC project H2020-ERC-2014-ADG 669354 CyberCare.

References

[1] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2498–2533, 2013.

[2] H. J. Caulfield and S. Dolev, "Why future supercomputing requires optics," *Nature Photonics*, vol. 4, no. 5, p. 261, 2010.

[3] S. Dutta, O. Zografos, S. Gurunarayanan, I. Radu, B. Soree *et al.*, "Proposal for nanoscale cascaded plasmonic majority gates for non-Boolean computation," *Scientific reports*, vol. 7, no. 1, p. 17866, 2017.

[4] W. L. Barnes, A. Dereux, and T. W. Ebbesen, "Surface plasmon subwavelength optics," *Nature*, vol. 424, no. 6950, pp. 824–830, 2003.

[5] E. Testa, M. Soeken, L. Amarù, and G. De Micheli, "Logic synthesis for established and emerging computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 165–184, 2018.

[6] O. Zografos, F. Catthoor, S. Dutta, and A. Naeemi, *US Patent Application US20190064438A1*, 2019.

[7] N. Sureka, R. Porselvi, and K. Kumuthapriya, "An efficient high speed Wallace tree multiplier," in *International Conference on Information Communication and Embedded Systems*, 2013, pp. 1023–1026.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[9] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964.

[10] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[11] J. Fadavi-Ardekani, "M*N Booth encoded multiplier generator using optimized Wallace trees," *IEEE Trans. VLSI Syst.*, vol. 1, no. 2, pp. 120–125, June 1993.

[12] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. on Computers*, vol. 45, no. 3, pp. 294–306, 1996.

[13] W. J. Stenzel, W. J. Kubitz, and G. H. Garcia, "A compact high-speed parallel multiplication scheme," *IEEE Trans. on Computers*, no. 10, pp. 948–957, 1977.

[14] "Lumerical, F. Solutions. Web source [https://www.lumerical.com/tcad-products/fdtd/] ," 2013.

[15] Y. Salamin, P. Ma, B. Baeuerle, A. Emboras, Y. Fedoryshyn *et al.*, "100 GHz plasmonic photodetector," *ACS photonics*, vol. 5, no. 8, pp. 3291–3297, 2018.

[16] S. Waser and M. J. Flynn, "Introduction to arithmetic for digital systems designers," 1982.

[17] D. Shen and A. Weinberger, "4-2 carry-save adder implementation using send circuits," *IBM Technical Disclosure Bulletin*, vol. 20, no. 9, pp. 3594–3597, 1978.

[18] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 644–656, 2018.