

# Design of a 5-bit Signed SRAM-based In-Memory Computing Cell for Deep Learning Models

Ó. Pereira-Rial, D. García-Lesta, V.M. Brea, P. López, D. Cabello

*Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)*

*Universidade de Santiago de Compostela*

Santiago de Compostela, Spain

email: victor.brea@usc.es

**Abstract**—Neural network mixed-mode hardware accelerators for deep convolutional neural networks (CNN) strive to cope with a high number of input feature maps and increasing bit depths for both weights and inputs. As an example of this need, the ResNet model for image classification comprises  $512\ 3 \times 3$  feature filters in its conv5 layer. This would lead to 4068 multipliers driving a summing node for actual concurrent processing of all the input feature maps, which makes up a challenge in mixed-mode. This paper addresses the design of a 5-bit signed SRAM-based in-memory computing cell in 180 nm 3.3 V CMOS technology, dealing with the impact of increasing the number of input feature maps. The data presented in the paper are based on electrical and post layout simulations.

**Index Terms**—In-memory computing, convnets, neural network hardware accelerator, mixed-mode, artificial intelligence on the edge

## I. INTRODUCTION

Today the number of applications addressed by deep learning in computer vision is virtually endless. State-of-the-art deep learning networks to this end are usually designed as convolutional neural networks, also known as convnets, of many layers, each of them with as many feature kernels or convolutional filters run on as many input feature maps. This poses a great challenge to cope with real-time operation and low power consumption with conventional architectures, which, in turn, has triggered the coming of neural network hardware accelerators.

Differently from the conventional Von Neumann approach, neural network hardware accelerators in conventional CMOS or new emerging memory technologies like FeRAM adopt in-or close-to memory architectures, where the different elements of a convolutional kernel or filter, also known as weights, are stored next to their corresponding multiplier to circumvent the communication and power burden from separated memory and processing blocks, usually laid down further away from each other [1]. Custom made neural network hardware accelerators try to leverage the advantages of mixed-mode design to cut area and power, and increase throughput at the cost of less

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101016734. This work was also funded by Spanish Ministry of Science, Innovation and Universities under grant RTI2018-097088-B-C32, from Xunta de Galicia-Consellería de Cultura, Educación e Ordenación Universitaria Accreditation 2019-2022 ED431G-2019/04 and Reference Competitive Group Accreditation 2021-2024 ED431C2021/048, co-funded by (ERDF/FEDER programme).

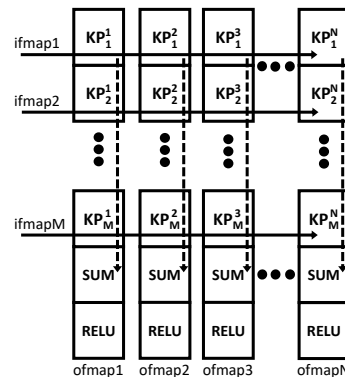


Fig. 1: Baseline architecture for a convnet engine.

accurate weights, and bit depth in images. This is usually the cost to bring artificial intelligence to the edge in the paradigm of internet of things (IoT).

State-of-the-art neural network hardware accelerators in the mixed-mode domain [2]–[5] strive to cope with the decreasing signal to noise ratio (SNR) caused by an increasing number of input feature maps or filters and increasing weights and inputs bit depths [1]. As an example, the ResNet deep learning model [6] for image classification comprises  $512\ 3 \times 3$  feature filters, i.e., input feature maps, in its conv5 layer. This would lead to 4068 multipliers driving a summing node for actual concurrent processing of all the input feature maps, which makes up a challenge in mixed-mode.

This paper addresses the design of a 5-bit signed SRAM-based in-memory computing cell in 180 nm 3.3 V CMOS technology, dealing with the impact of increasing the number of input feature maps or convolutional filters. The data presented in the paper are based on electrical and post layout simulations.

## II. CONVNET ENGINE BASELINE ARCHITECTURE

Fig. 1 shows the floorplan of a baseline convnet engine architecture for computer vision applications. In an in-memory computing architecture, every kernel processor (KP) includes all the elements required for the inner product of the feature kernel or filter with either the input image in the first layer or a given input feature map in a hidden layer of the network. The

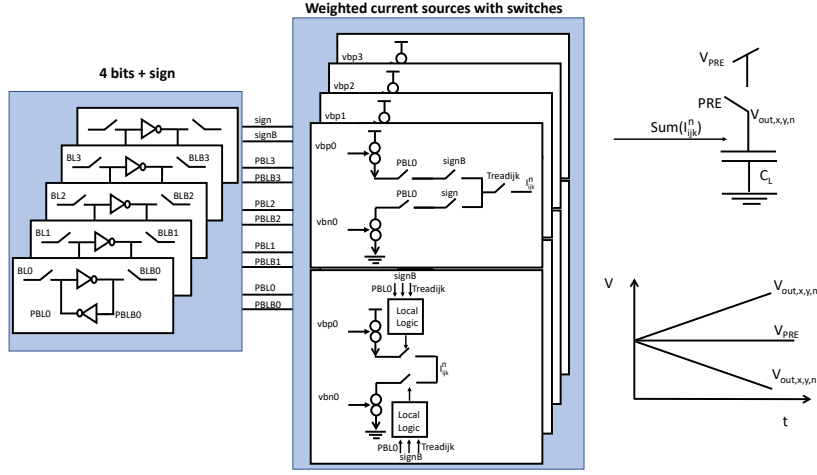


Fig. 2: 5-bit signed SRAM-based in-memory multiplier concept.

circuitry of the KP encompasses the storage of the feature filter components, the so-called weights, and the multipliers next to them, making up an in-memory computing approach.

In the baseline convnet engine architecture shown in Fig. 1 every convnet layer is implemented with one column of KP's. The input image patch to every KP in a column comes from a different input channel, also known as input feature map, thus, the number of input feature maps  $M$  equals the number of KP's in a column. The number of columns in the convnet engine array is the number of output feature channels  $N$  provided by a given layer, thus, all the KP's across different columns in a row are driven by the same input image patch. Every KP has their own weights,  $KP_z^n$ , with indexes  $z$  and  $n$  representing a given input and output feature map, ( $ifmap_z$  and  $ofmap_n$  in Fig. 1). The output of every inner product of weights and inputs are summed throughout the column, typically in the charge or current domain [3]–[5], before going through a RELU function [7]. This is formulated by the next equation:

$$Out_{x,y,n} = \sum_{i,j,k} W_{i,j,k}^n * Inp_{x+i,y+j,z+k} \quad (1)$$

where  $Out_{x,y,n}$  is the output of a given pixel with coordinates  $x$  and  $y$  in a given output feature map  $n$ ,  $W_{i,j,k}^n$  means the weights of a given feature filter or kernel for said output feature map  $n$ ,  $Inp_{x+i,y+j,z+k}$  is the input patch that matches the size of a given feature convolutional filter or kernel across the different input feature maps. Index  $k$  runs across the different input feature maps  $ifmap$ , while indexes  $i$  and  $j$  go over an image patch of typically  $3 \times 3$  pixels around the pixel of interest with coordinates  $x$  and  $y$ .

### III. 5-BIT SIGNED SRAM-BASED IN-MEMORY MULTIPLIER CONCEPT

Fig. 2 shows the concept of our 5-bit signed SRAM-based in-memory multiplier. We take this circuit as an use case to study scaling issues on the way to concurrently run hundreds

of convolutional kernels in a deep convnet. Its design is more demanding than that of binary neural networks, or deep learning models with unsigned weights [3], [5].

Our in-memory 5-bit signed SRAM-based multiplier implements weights  $W_{ijk}^n$  with single transistor current sources biased with current mirrors deployed outside of the array of KP's to provide  $vbpr$  and  $vbnr$  bias voltages. PMOS and NMOS transistors yield positive and negative weights, respectively. All the transistors of the same type feature the same dimensions, thus their bias voltages change with their position in the weight word- index  $r$  in Fig. 2- and with their sign, either positive-  $signB$ - or negative-  $sign$ . This permits a very regular and easier layout than that with only one bias voltage per transistor type, either n- or p-, and different W/L ratios among transistors of the same type. Input feature maps are provided by PWM signals through  $Tread_{ijk}$ . The outputs from all the multipliers along a column  $I_{ijk}^n \times Tread_{ijk}$  are summed as charges and turned into voltages in a load capacitor  $C_L$  per column of KP's. This capacitor is precharged to a  $V_{PRE}$  voltage, which can be made programmable to make room for the different working drain to source  $V_{ds}$  voltage regions in the P- and NMOS current sources due to their different  $V_{TH}$  voltages.

The voltage vs time plot of Fig. 2 illustrates the evolution of voltage  $V_{out}$  at capacitor  $C_L$  along time, either increasing when the column current is drawn into the capacitor or decreasing when the opposite. This behavior is conveyed in this equation:

$$V_{out,x,y,n} = \frac{1}{C_L} \sum_{i,j,k} I_{i,j,k}^n \cdot Tread_{x+i,y+j,z+k} \quad (2)$$

which is the same as Eq. (1) but replacing  $Out$  with  $V_{out}$ ,  $W$  with  $I$ , where  $I$  is the current from a given multiplier, and  $Inp$  with  $Tread$ . It should be noted that it is convenient to make  $C_L$ ,  $Tread_{ijk}$  and the least significant bit current programmable in order to reuse hardware for different layers, and eventually for on-chip training.

The in-memory multiplier described in this paper is conceived as compatible with 6T SRAM cells, but differently from standard 6T SRAM cells, read and write ports are decoupled from each other, which allows for memory cells with minimum sized transistors. Also, our in-memory computing multiplier works with binary outputs  $PBLr$  and  $PBLBr$  provided by the SRAM cells.

In comparison with recent in-memory computing solutions, as the approach in [5], our approach also decouples read and write operations with an SRAM circuit as memory cell. Nevertheless, differently from their implementation, the bias voltages  $vbpr$  and  $vbnr$  of our current sources are always fixed, the outputs of the SRAM cells are always driving high impedance nodes, and the summing is done in a capacitor instead of in a virtual ground of a transimpedance amplifier. Our approach is more similar to that of [3]. Nevertheless, our solution features signed weights and we resort to SRAM cells, rendering binary voltages  $PBLr$  and  $PBLBr$  instead of analog voltages from DRAM cells.

Finally, as seen in Fig. 2, different versions of this 5-bit signed SRAM-based in-memory multiplier can be designed according to the number of switches in cascade to drive the current through the current mirrors driven by  $vbpr$  or  $vbnr$  up to the column line and load capacitors  $C_L$ . The impact of both of them on the number of input feature maps, i.e., number of KP's per column, is studied in the next section.

#### IV. 5-BIT SIGNED SRAM-BASED IN-MEMORY MULTIPLIER DESIGN

Our approach is designed in high voltage 180 nm CMOS technology to ease compatibility with emerging devices like FeRAM technology. We have designed our in-memory computing cell with  $V_{dd} = 3.3$  V.

As said above, the swing or the dynamic range of the summing node in a mixed-mode neural network hardware accelerator is a hard constraint when increasing the number of KP's per column, i.e. the number of input feature maps, which becomes even harder with multi-bit input and weight convnets. This leads to transmission gates instead of single transistors as switches in our design in order not to lose  $V_{TH}$  volts of swing or dynamic range at the summing node  $C_L$ , and not to add non-linearities derived from the dependence of  $V_{TH}$  with  $V_{BS}$ .

The issue of the finite output impedance of the current sources of Fig. 2 is tackled with long transistors. The width of these transistors is a balance between a voltage  $V_{GS}$  close to  $V_{TH}$  to increase dynamic range and as large a  $V_{GS}$  voltage as possible, which leads to lower mismatch. Our solution is based on PWM through the  $Tread_{ijk}$  signals, which means that our summation is done in a capacitor, so that intrinsic capacitances do not only increase settling time but, which is worse, they might also degrade the accuracy of our in-memory computing multiplier. On this basis, the number of transistors and their sizing in the data path of the current provided by the weighted current sources affect the accuracy. In this line, the

alternative of one switch vs several switches connected to the weighted current sources along the data path must be studied.

Fig. 3 shows the signal data path for one bit  $BLr$  for positive and negative signs, and for the case of one and three switches. The switches are implemented with transmission gates with minimum sizes, namely,  $W = 220$  nm,  $L = 350$  nm, and  $W = 220$  nm,  $L = 300$  nm for NMOS and PMOS transistors, respectively. Every switch adds an intrinsic capacitance  $C_{SW}$  along with that of the current source transistor  $C_{Mbias}$  on the signal data path, causing non-linearities that make Eq. (2) deviate from its ideal response. In turn, such non-linearities are worsened due to the finite output impedance of the current sources. In the case of the conv5 of ResNet implemented with 5-bit signed weights, there might be up to 4068 multipliers with up to 4 bits of the same sign ON connected to a summing node. These data make it clear the importance of minimizing both the area of every switch and its number along the signal data path. It should also be noted that parasitic capacitances as the line capacitance across a column of KP's can be accounted for beforehand as they are non-signal dependent.

The non-linearity errors coming from intrinsic capacitances can be compensated for with a large enough load capacitance  $C_L$  per column of KP's. Quantitatively, this can be calculated from the voltage across a capacitor  $V_{outBLr}$  resultant from a particular bit line  $BLr$  through Eq. (3), where  $I_{BLr}$  is the current of a given bit line of the multiplier under study,  $Tread_{ijk}$  is the integration time for a given multiplier  $ijk$ , and  $C_{TBLr}$  is the total capacitance seen by a bit line  $BLr$ . Capacitance  $C_{TBLr}$  includes both, the designer defined capacitance,  $C_{LBLr}$ , and the intrinsic capacitances from the switches,  $C_{addBLr}$ . These capacitances set a minimum summing capacitor value  $C_{LBLrmin}$  that maintains the relative error, calculated with Eq. (4), below a given threshold. This value can be estimated with Eq. (5).

$$V_{outBLr} = \frac{1}{C_{TBLr}} \cdot I_{BLr} \cdot Tread_{ijk} \quad (3)$$

$$\eta_r = \frac{V_{outBLrideal} - V_{outBLr}}{V_{outBLrideal}} = \frac{C_{addBLr}}{C_{LBLrmin} + C_{addBLr}} \quad (4)$$

$$C_{LBLrmin} = C_{addBLr} \frac{1 - \eta_r}{\eta_r} \quad (5)$$

In our implementation, the values of the intrinsic capacitances have been estimated by simulation as  $C_{SW} \simeq 2$  fF and  $C_{Mbias} \simeq 2.2$  fF. An error in the multiplier below 5% leads to a minimum summing capacitance per bit of  $19 \times C_{add}$ , so the minimum capacitance per 1-bit multiplier will be approximately  $C_{LBLrmin} \simeq 80$  fF when the single switch approach is used, raising up to  $C_{LBLrmin} \simeq 155$  fF for the solution with three switches. In the case of the conv5 in the ResNet model, 512  $3 \times 3$  filters with 5-bit signed weights would amount to  $C_{LBLrmin} \simeq 1.5$   $\mu$ F for the multiplier with one switch, and approximately twice this value for the case with three switches. Still, with MiM capacitors made up of

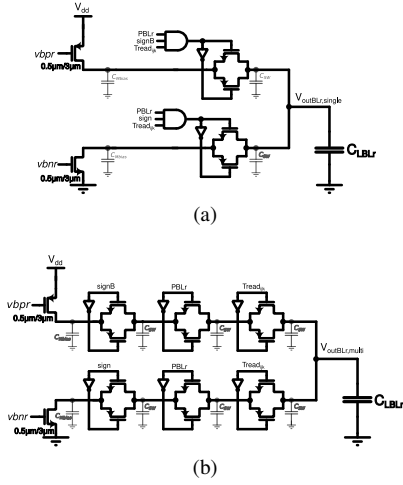


Fig. 3: Signal path for a 1-bit signed multiplier with (a) one and (b) three switches.

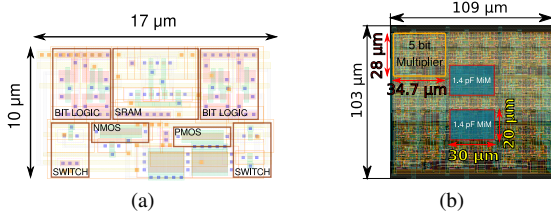


Fig. 4: Layout of a 1-bit signed multiplier with one switch (a), and its  $3 \times 3$  KP layout (b) with 9 5-bit signed multipliers.

high metal layers that allow for circuitry underneath, the area of the KP is the limiting factor to concurrently run 512  $3 \times 3$  filters with 5-bit signed weights. As an example, in the 180 nm CMOS technology, a KP of  $3 \times 3$  multipliers with 5-bit signed weights occupies over  $100 \times 100 \mu\text{m}^2$ , as seen in Fig. 4b, while the minimum load capacitance  $C_L$  per KP to keep the error below 5% would correspond to approximately 3 pF, which can be laid down with MiM capacitors with the top two metal layers in about  $40 \times 30 \mu\text{m}^2$ , as highlighted in blue in the center of Fig. 4b, which is a significantly less area than that of one KP. This leads to the important conclusion that for MiM technologies with top metal layers that allow for circuitry underneath, and for architectures provided with a column capacitor to collect the contributions from KP's along a given column (see Fig.1), said capacitor can be increased to enhance accuracy without area penalty for a restrained number of bits, in our case up to 5-bits with sign.

To probe further, we illustrate the effect of the intrinsic capacitances along the signal data path with the number of switches through Fig. 5 with post-layout simulations for two examples in the case of 2-bit multipliers with  $I_{BLr0} = 50$  nA, a summing capacitor  $C_L = 150$  fF, and an integration time  $T_{read_{ijk}} = 600$  ns. The relative deviation of the simulated values from the ideal ones are 1.2% and 1.8% with the single switch multiplier, and 5.8% and 6.8% for a multiplier with three switches.

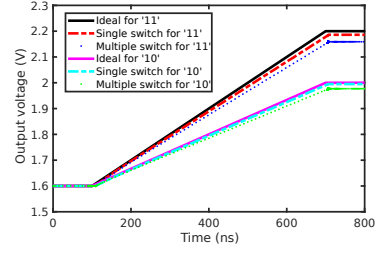


Fig. 5: Post-layout simulations with the ideal response at capacitor  $C_L$  for multipliers with one and three switches.

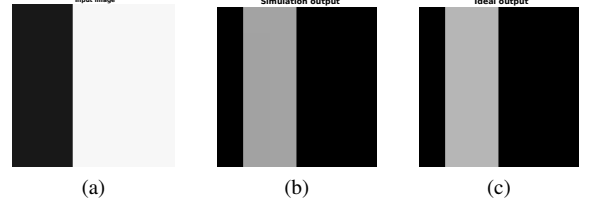


Fig. 6: Edge detection on an  $8 \times 8$  image (a) with the  $3 \times 3$  5-bit proposed KP of one switch (b) and the ideal output (c).

Post-layout simulations running a  $3 \times 3$  edge detection filter with weights  $(-15 \ -15 \ -15 \ 0 \ 0 \ 15 \ 15 \ 15)$  on the  $8 \times 8$  input of Fig. 6a illustrate the functionality of the KP. The lack of padding produces a resultant  $6 \times 6$  output image (enlarged up to the size of the input image for a better comparison).

The aforementioned filter was written in the SRAM corresponding to the 9 weights of a KP with  $3 \times 3$  multipliers. The  $T_{read_{ijk}}$  values of each input pixel are calculated with a Verilog circuit that implements a 5-bit digital to time converter. The least significant input pixel value, i.e, black pixel, was set to 50 ns. To obtain the resultant image pixels in Fig. 6b, the output voltage at the end of each integration time in the pixel under study was sampled. As it can be seen, the results from post-layout simulations are in line with those expected by calculation (see Fig. 6c).

## V. OUTLOOK AND CONCLUSIONS

This paper has addressed the design of a 5-bit signed SRAM-based in-memory computing multiplier for deep learning models in computer vision. This circuit has been a use case to study the effect of an increasing number of input feature maps, or KP's on the computation accuracy, a timely topic in mixed-mode neural network hardware accelerators. In our approach, the output feature maps are voltages resultant from integrating currents from all the input feature maps for a given layer into a load capacitor along time. Our work has shown that an approach with less switches in the multiplier favors accuracy, leading to smaller capacitances. Still, for weights with a restrained bit width, in our case 5-bit signed, the required load capacitor to keep reasonable error levels ( $<5\%$ ) does not increase the area of a KP. The design of our multiplier has been included on a die submitted to fabrication. If available, experimental results will be shown at the conference.

## REFERENCES

- [1] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.
- [2] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.
- [3] Z. Chen, X. Chen, and J. Gu, "A 65 nm 3t dynamic analog ram-based computing-in-memory macro and cnn accelerator with retention enhancement adaptive analog sparsity and 44tops/w system energy efficiency," in *IEEE Int. Solid-State Circuits Conf.(ISSCC) Dig. Tech. Papers*, pp. 240–241, 2021.
- [4] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.
- [5] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8t sram cell as a multibit dot-product engine for beyond von neumann computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2556–2567, 2019.
- [6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.