Low-Overhead Implementation of Binarized Neural Networks Employing Robust 2T2R Resistive RAM Bridges

M. Ezzadeen, A. Majumdar, Marc Bocquet, B. Giraud, J.-P. Noel, F. Andrieu, D. Querlioz, Jean-Michel Portal

Low-Overhead Implementation of Binarized Neural Networks Employing Robust 2T2R Resistive RAM Bridges

M. Ezzadeen[1,3], A. Majumdar[2], M. Bocquet[3], B. Giraud[1], J.-P. Noël[1], F. Andrieu[1], D. Querlioz[2], J.-M. Portal[3]
[1]CEA-List/Leti, Univ. Grenoble Alpes, Grenoble, France, mona.ezzadeen@cea.fr
[2]Université Paris-Saclay, CNRS, C2N, 91120 Palaiseau, France
[3]Aix-Marseille Univ., Université de Toulon, CNRS, IM2NP, Marseille, France

***Abstract*—The energy consumption associated with data movement between memory and processing units is the main roadblock for the massive deployment of edge Artificial Intelligence. To overcome this challenge, Binarized Neural Networks (BNN) coupled with RRAM-based in- or near-memory computing constitute an appealing solution. However, proposals from the literature tend to involve significant periphery circuit overheads. In this work, we propose and demonstrate experimentally, on a fabricated hybrid CMOS-RRAM integrated circuit, a robust in-memory XOR operation based on a 2T2R cell used in a resistive bridge manner. With this architecture, the RRAM read operation and the BNN multiplication operation can be achieved simultaneously, requiring only inverters connected to each Source Line of the memory array, and the BNN POPCOUNT operation can be realized with an analog capacitive neuron. Based on our measurements and extensive Monte Carlo simulations, we validate that this approach is suitable for large neurons with a low error rate (3.12% of error considering the full range of POPCOUNT values). Based on the circuit simulation results, we highlight the resilience of this approach at the network level, with a minimal accuracy degradation on the MNIST (0.07%) and CIFAR-10 (0.35%) tasks with regards to software solutions.**

Keywords—RRAM memory, In-memory computing, Binary Neural Network (BNN), XNOR, POPCOUNT

I. INTRODUCTION

With the increase of computational resources, Deep Neural Networks have achieved record-breaking performance in numerous tasks such as speech and image recognition. However, these results come at the price of high power consumption, mainly due to the data movement between memories and processing cores [1]. This challenge, the von Neumann bottleneck, is particularly critical for edge Artificial Intelligence (AI), where the power budget is very restricted. In this context, Binarized Neural Networks (BNNs) [2] (or the closely related XNOR-NETs [3]) are outstanding candidates. BNNs can perform high accuracy AI while relying on binary weights and activations. This simplicity limits the memory capacity requirements of BNNs and their computational complexity: in BNNs, the multiplications between real numbers required by conventional neural networks are replaced by one-bit XNOR operations, and the sums of real numbers are replaced by bit counting operations (POPCOUNTs).

The most appealing implementations of BNNs rely on emerging nonvolatile memories such as Resistive RAMs (RRAMs). Such memories, which can be embedded at the core of modern CMOS processes, allow fully associating logic and memory operations, eliminating the von Neumann bottleneck. In the literature, two main approaches are used for implementing BNNs with RRAMs. In the fully In-Memory Computing approach, the equations of BNNs are implemented in an analog fashion using Ohms' law and the Kirchhoff current law [4]. This approach is especially elegant but requires significant overhead to avoid voltage offsets and high-area and high-energy consuming Analog-to-Digital converters. It is also strongly affected by RRAM device variability. On the other hand, it is possible to read the RRAM states using standard memory circuits, and to compute the BNN equations by external circuitry near the RRAM array. For example, [5] is using a 2 Transistors – 2 RRAMs (2T2R) differential cells structure to reduce the number of bit errors – and therefore offer a large immunity to device variation – and a standard precharge sense amplifier augmented by transistors that perform the XNOR operation at the same time as the sense. The use of a sense amplifier is still an important energy and area cost, however.

In this work, we propose and validate in silicon, on a fabricated hybrid CMOS/RRAM integrated circuit, a simpler alternative using minimal periphery circuitry: a CMOS inverter is enough to read the state of the RRAMs and to perform the XNOR operation of the BNN at the same time. Additionally, our solution is compatible with a simple capacitor-based neuron capable of doing the POPCOUNT operation in a very energy-efficient fashion. We also show experimentally and theoretically that our approach has extremely high immunity to RRAM variation issues.

Related research papers have also recently proposed low-variability charge-based computation for BNN applications, but based on SRAM [6], [7]. Using capacitive dividers to perform the POPCOUNT operation, these solutions do not suffer from variability issues thanks to the high stability of MOM capacitors, and can therefore enhance parallelism by using very large neurons. However, SRAM-based solutions suffer from static consumption for always-on applications and

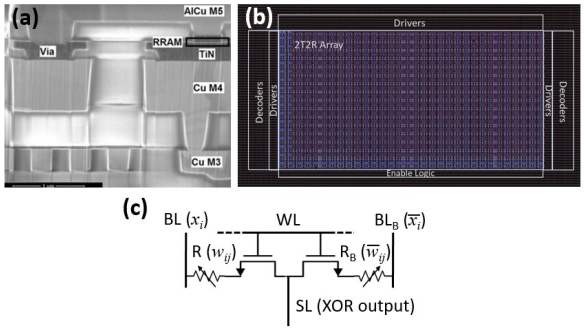

Fig. 1. (a) Electron microscopy image of a hafnium oxide resistive memory cell (RRAM) integrated in the backend-of-line of a 130nm CMOS process. (b) Photograph of the test chip using this process (c) 2T2R bitcell scheme characterized in this work

This work was funded by the European Research Council (ERC) through the My-CUBE and NANOINFER (715872) projects.

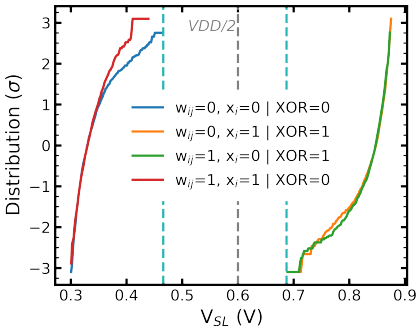

Fig. 2. Measurements, on test chip of Fig. 1, of XOR (≡SL) distributions with V_{LOW}=0.3V and V_{HIGH}=0.9V for the four cases of the XOR truth table, measured on 1,024 2T2R RRAM cells (programming conditions are V_G(set) = 1.9V, V(set) = 2V} and {V_G(reset) = 3.3V, V(reset) = 2.5V})

need to be reprogrammed after a circuit shutdown. Therefore, nonvolatile approaches such as RRAM-based solutions seem to be a better compromise for edge AI, as they avoid power-hungry transfers of constant weight to volatile memories.

The contributions of this paper are as follows:

- The 2T2R cell used as a bridge divider to perform XNOR operation is measured on a 1kbit 2T2R memory array, assessing the margin of the XOR response experimentally and thus the possibility to generate proper digital XNOR with a single inverter (sec. II)
- We introduce, for the first time, an RRAM-based capacitive neuron. The neuron activation function is supported by capacitive bridge voltage differentiation between POPCOUNT and threshold (sec. III).
- Based on XNOR 2T2R measurements, we validate the capacitive bridge voltages differentiation through extensive Monte Carlo simulations (sec. IV).
- We carry simulations at the neural network level to show the impact of the circuit implementation, taking into account error rate due to variability for the very large neurons on the MNIST and CIFAR-10 tasks and evidence the error resilience of our approach (sec. V).

II. 2T2R CELL FOR XNOR OPERATION

BNNs function by computing neuron activations a_j, calculating the XNOR between weights w_{ij}, fixed during inference, and neural activation inputs x_i:

$$a_j = sign\left(POPCOUNT_i\left(XNOR(w_{ij}, x_i)\right) - t_j\right) \quad (1)$$

where t_j is a threshold associated with each neuron.

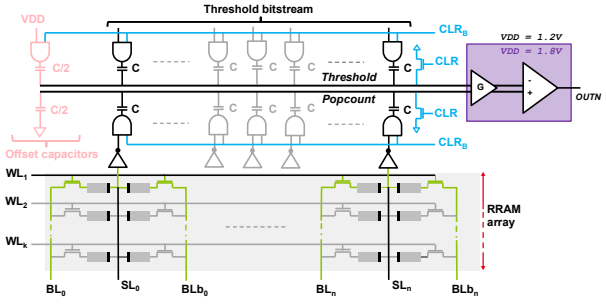

Fig. 3. Circuit of the proposed RRAM-based capacitive neuron. . It consists of (1) a POPCOUNT capacitive divider connected to a 2T2R-RRAM array via an inverter row – the latter being used to perform our proposed RRAM-based XNOR, (2) a threshold capacitive divider and (3) a latch-based comparator.

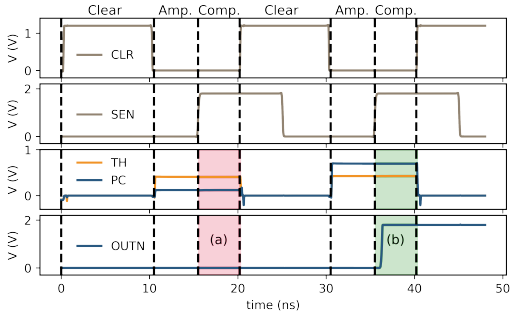

Fig. 4. Simulation of the proposed neuron's operation, for (a) the case of threshold (TH) > POPCOUNT (PC), and (b) the case of TH < PC , with three operation phases in each case : first the capacitive divider charges are removed (Clear), then the TH and PC voltage are preamplified (Amp.), and finally the two resulting signals are compared (Comp.).

The main idea of our work is to program the weights using 2T2R RRAM memory cells in a complementary state fashion, and to apply the neural activation inputs on their respective Bit Lines (BLs) in a complementary way also. We will see that the 2T2R structure then acts as a resistive bridge, whose middle point is the XOR output. The use of nonvolatile memory, such as RRAM, for programming weights, is very advantageous, as the weights have to be programmed only once or a few times. When programming the weights, we can use write-verify algorithms or strong programming conditions to obtain a high ratio between the High Resistive and Low Resistive State resistance values (HRS and LRS).

Fig. 1 presents the test chip used to experimentally demonstrate that a 2T2R cell can be used as a resistive bridge to perform XOR operations in an IMC fashion. This test chip co-integrates 130 nm foundry CMOS and resistive memory in the back-end-of-line, between levels four and five of metal. The resistive memory cells are based on 10 nm thick hafnium oxide (see Fig.1(a)). The test chip is a 32x32 array of 2T2R cells (see die photograph, Fig.1(b)). Each cell is directly accessible from characterization pads that control its word line (WL), bit lines (BL and BLB), and source line (SL) independently. The 2T2R bit cell is illustrated in Fig.1(c).

TABLE I. RRAM-BASED XNOR TRUTH TABLE

WEIGHT			NEURON INPUT			XOR (SL)		XNOR
bin	*R*	R_B	*bin*	*BL*	BL_B	(V)	Bin	
0	LRS	HRS	0	V_{LOW}	V_{HIGH}	<V_{DD}/2	0	V_{DD}
0	LRS	HRS	1	V_{HIGH}	V_{LOW}	> V_{DD}/2	1	gnd
1	HRS	LRS	0	V_{LOW}	V_{HIGH}	> V_{DD}/2	1	gnd
1	HRS	LRS	1	V_{HIGH}	V_{LOW}	< V_{DD}/2	0	V_{DD}

Table I lists the complementary RRAM resistance values R and R_B used to code the binary value (*bin*) of the weight w_{ij}, and the voltage level applied on *BL* and BL_B to code neural activation inputs x_i. It is straightforward to see that, as the 2T2R cell is programmed in a complementary fashion to code the weight, it acts as a resistive bridge. Therefore, the *SL* voltage (XOR) solely depends on the resistance ratio of R and R_B and on the applied voltage V_{BL} and V_{BL_B}, following

$$V_{SL} = \left(V_{BL} - V_{BL_B}\right).\frac{R_B}{(R+R_B)} \text{ when } V_{BL} > V_{BL_B}\ (x_i = 1) \quad (2)$$

$$V_{SL} = \left(V_{BL_B} - V_{BL}\right).\frac{R}{(R+R_B)} \text{ when } V_{BL} < V_{BL_B}\ (x_i = 0). \quad (3)$$

The voltages V_{HIGH} and V_{LOW}, applied on *BL* and BL_B depending on the x_i value, are chosen carefully to be symmetric with respect to V_{DD}/2 (V_{DD} being the biasing

voltage of the inverter connected to the *SL*) and to ensure that the RRAM state is not disturbed during XNOR computation. The resulting *SL* voltage is either higher or lower than $V_{DD}/2$, and corresponds to the XOR value. The HRS resistance (*R* or R_B), indeed, systematically takes the largest voltage drop, and thus the *SL* voltage remains close to the LRS resistance bit line voltage (V_{HIGH} or V_{LOW}). A simple CMOS inverter can be used on the bottom of *SL* to complement and amplify the *SL* voltage to obtain a proper digital XNOR output.

In our experiment, we use V_{LOW}=0.3V and V_{HIGH} =0.9V. Fig. 2 reports the *SL* voltage distribution for the various conditions of operation given in Table I, for the 1,024 cells of our test chip. These measurements demonstrate that the V_{SL} voltage of the 2T2R cells follows the expected XOR function with a margin of 133 mV and 87 mV between the two *SL* states and $V_{DD}/2$. This experiment shows that the 2T2R approach is robust against RRAM variability and allows the generation of XOR output voltage with a large margin. This feature allows obtaining a proper XNOR digital level by using a simple inverter on the *SL* as a sensing solution. Moreover, this solution functions with low power consumption, as one of the RRAM devices is always in High Resistance State.

III. RRAM-Based Capacitive Neuron Description

The scheme of our new RRAM-based capacitive neuron is illustrated in Fig. 3. The neuron is mainly composed of three blocks:

- A 2T2R RRAM array with XNOR outputs (section II). The 2T2R cells on a given row implement all the synaptic input weights of the neuron.
- Two capacitor bridge dividers to compute the POPCOUNT and the threshold in an analog way.
- A latch-based comparator. This circuit compares the POPCOUNT voltage and the threshold voltage to generate the binary neuron output activation.

The core of an RRAM-based capacitive neuron with *n* inputs is, therefore, composed of a selected row of *n* weights in a 2T2R RRAM array, which performs *n* RRAM-based XNOR operations, available at the bottom of the array. The XNOR outputs are connected to *n* identical capacitors of value *C* –through AND gates to force the clear signals– that form a capacitive divider. In our work, we use 3.9fF MOS capacitors, which have limited area compared to, e.g., MOM capacitors. The POPCOUNT resulting voltage is given by

$$V_{PC} = \frac{m}{n}.V_{DD}, \quad (4)$$

with *m* the number of XNOR outputs equal to one. We reuse the same capacitive divider structure for the threshold part using a simple bitstream, whose number of ones determines the threshold value. This differential approach, where the POPCOUNT and the threshold are handled similarly, simplifies the threshold generation considerably compared to state-of-the-art solutions, as we do not need Digital-to-Analog converters to generate the threshold voltage [7], or to add weighted biases on the capacitive divider as in [6].

To avoid a comparison between two identical values when the POPCOUNT and threshold values are equal, an offset capacitor of value *C/2* is added on each capacitive divider. The offset capacitor is grounded for the POPCOUNT divider, whereas it is connected to V_{DD} for the threshold divider. The POPCOUNT resulting voltage V_{PC} is then given by

$$V_{PC} = \frac{m}{n+0.5}.V_{DD}, \quad (5)$$

whereas the threshold resulting voltage V_{TH} is given by

$$V_{TH} = \frac{k+0.5}{n+0.5}.V_{DD}, \quad (6)$$

where *k* is the number of 1s in the threshold bitstream. Therefore, for *m*=*k*, the threshold and POPCOUNT voltages are not equal and have an offset given by

$$|\Delta V| = \frac{0.5}{n+0.5}.V_{DD}, \quad (7)$$

thus helping the comparator decision.

The threshold and POPCOUNT voltages are then passed through an isolation preamplifier before being compared using a classical strongARM comparator [8] followed by a latch. The preamplifier stage mainly helps isolate the capacitive divider from the kickback noise generated by the comparator activation signal. To ensure a high input overdrive for all threshold and POPCOUNT values, as they can range from ground to V_{DD}, we use a higher voltage domain for the preamplifier and comparator stage (with a supply voltage of 1.8V instead of 1.2V).

Fig. 4 reports the electrical simulation results of the proposed neuron, obtained once for a POPCOUNT value below the threshold, followed by a POPCOUNT value above the threshold. The comparison process starts by removing all charges from the capacitors. This operation is made by activating CLEAR signals (*CLR* and CLR_B) to ground capacitors bottom electrodes and top electrodes, respectively, through AND gate for the top and a pull-down NMOS transistor for the bottom. In parallel, the inputs neurons activation signals are applied on all *BL* and BL_B to perform the XNOR operations on the selected row. The threshold bitstream is applied at the same time on the threshold capacitive divider inputs. The XNORs and the threshold bitstream values do not disturb the CLEAR operation, as they are gated by the AND gate with CLR_B signal grounded. Thus, the AND gates enable the parallelization of the CLEAR operation, the XNOR computation, and the threshold bitstream value application.

When the CLEAR operation is finished, the XNORs and threshold bitstream values are directly applied to their respective capacitors through the AND gates, settling the POPCOUNT and threshold capacitive divider voltages given by (5) and (6). Then, the strongARM comparator is activated (i.e., the *SEN* signal is tied to V_{DD}), and the two preamplified voltages are compared, leading to a neuron output (*OUTN* signal). The output signal is set to 1 if the POPCOUNT is above the threshold and set to 0 in the other case.

IV. RRAM-Based Capacitive Neuron Validation

To fully validate our RRAM-based capacitive neuron circuit, we performed extensive Monte Carlo circuit simulations for various neuron sizes, ranging from 8 to 512 inputs, for the full range of possible POPCOUNT/threshold combinations. We consider global and local sources of variability at -3σ/+3σ, including mismatch. For all simulated cases, 1,000 runs are performed. To be as close as possible to the silicon results, the SL measured distributions are directly injected at the XNOR inverter's inputs.

The Monte Carlo simulations confirm that, whatever the *SL* level, the XNOR inverter output is a proper ground or V_{DD} level. From the Monte Carlo simulations results, we also extract the number of errors for all neuron sizes, as a function

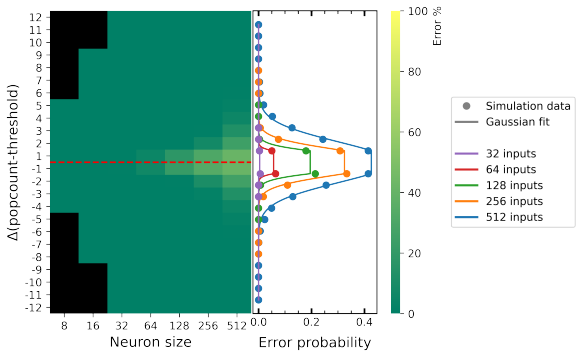

Fig. 5. Error profiles of the proposed neuron with 8 up to 512 inputs with regards to their middle threshold values. Note that the black color in the heatmap corresponds to non-existent cases for small neuron sizes.

of the difference between the POPCOUNT value and the threshold value. This extraction is performed for the full range of POPCOUNT value around the threshold, until it is error-free. From this extraction, we fit the error distribution with a Gaussian law, when necessary. Fig. 5 illustrates some of these extractions. More precisely, this figure presents the results with a threshold value set to half of the different neuron sizes (8 to 512 inputs). To study the evolution of the errors, we have plotted a heat map of the error expressed in percentage together with the Gaussian law.

From Fig. 5, it is evident that for small neurons (up to 32 inputs), the output presents roughly no error, since the voltage difference between two adjacent POPCOUNT and threshold levels is sufficient, and as given in (7), is around 18 mV for 32 inputs. As the voltage difference is divided by a factor of two each time the number n of inputs is doubled, only one mV difference is available between two adjacent levels when considering 512 inputs neurons. This evolution explains that the error value and range increase with the number of inputs. However, as presented in Fig. 5, the standard deviation of the Gaussian error distribution remains tight, with only a range ±8 POPCOUNT values around the threshold that may present errors for the 512 inputs case. This result can be normalized to only 3.12% of the total range values that may presents errors. To increase the neuron inputs size above 512 while reducing the errors, one straightforward solution would be to adapt the biasing level of the capacitive bridge by overdriving the V_{DD} value.

V. Neural Network Demonstration

To determine the impact of these errors on BNN inference accuracy on practical AI tasks, we perform simulations using a fully connected architecture for the handwritten digit recognition (MNIST) task and a convolutional architecture for an object classification (CIFAR-10) task. We train the networks with no errors in the neurons and only introduce the errors during the inference step. The erroneous thresholding was not implemented for the first layer of both networks, as the input to a BNN is not binarized, and thus we cannot resort to such circuits. The error probability model used in the comparison of XNOR POPCOUNT and threshold is as shown in Fig. 5.

The fully connected network used for the MNIST task had a single hidden layer with 3,000 neurons, and the inference accuracy showed degradation of 0.07 percentage points, from a no-error accuracy of 98%. For the more difficult CIFAR-10 task, six convolutional layers followed by three fully connected layers were used, as is typical of convolutional neural networks. The accuracy degradation for this task was 0.35 percentage points, wherein the no-error accuracy had been 90.09%.

VI. Conclusion

In this work, we propose for the first time a capacitive RRAM-based BNN neuron with XNOR operation performed directly in memory using classical 2T2R cells programmed in complementary states as resistive bridge divider; activation neuron inputs are also coded in a complementary fashion and applied on *BL* and BL_B. We demonstrate the robustness of this approach experimentally by characterizing the *S*L voltage for the full XOR truth table on 1,024 fabricated 2T2R cells. With this approach, only a single inverter is needed on *SL* to generate the XNOR output at the bottom of the array. The experimental results demonstrate the robustness of the approach. Our proposed associated neuron uses an analog POPCOUNT computation based on a capacitive divider, and a similar approach is adopted for the threshold computation. Finally, the sign function is implemented by comparing the POPCOUNT and threshold values with a differential latch-based comparator. Based on *SL* voltage measurements and extensive Monte Carlo simulation, we have extracted error probability for input neuron sizes ranging from 8 to 512. Errors are negligible below 32 inputs and remain low for 512 inputs, with 3.12% of error considering the full range of POPCOUNT values. Errors are included only in a range of ±8 POPCOUNT values around any threshold value. Finally, extracted errors are introduced in neural network simulation, showing a minimal accuracy reduction of 0.07% using our circuit on the MNIST task, and only 0.35% on the CIFAR-10 task.

References

[1] M. Horowitz, '1.1 Computing's energy problem (and what we can do about it)', in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, pp. 10–14.

[2] M. Courbariaux, Y. Bengio, and J.-P. David, 'BinaryConnect: Training Deep Neural Networks with binary weights during propagations', *arXiv:1511.00363 [cs]*, Apr. 2016.

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, 'XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks', *arXiv:1603.05279 [cs]*, Aug. 2016.

[4] C.-X. Xue *et al.*, '24.1 A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing Time for CNN Based AI Edge Processors', in *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, Feb. 2019, pp. 388–390.

[5] M. Bocquet *et al.*, 'In-Memory and Error-Immune Differential RRAM Implementation of Binarized Deep Neural Networks', in *2018 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, Dec. 2018, pp. 20.6.1-20.6.4,

[6] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, 'An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS', in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb. 2018, pp. 222–224.

[7] 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, Jun. 2019.

[8] T. Kobayashi, K. Nogami, T. Shirotori, Y. Fujimoto, and O. Watanabe, 'A current-mode latch sense amplifier and a static power saving input buffer for low-power architecture', in *1992 Symposium on VLSI Circuits Digest of Technical Papers*, Jun. 1992, pp. 28–29.