Design Flow for Hybrid CMOS/Memristor Systems—Part II: Circuit Schematics and Layout

Sachin Maheshwari, *Member, IEEE*, Spyros Stathopoulos, Jiaqi Wang, *Graduate Student Member, IEEE*, Alexander Serb, *Senior Member, IEEE*, Yihan Pan, *Graduate Student Member, IEEE*, Andrea Mifsud, Lieuwe B. Leene, Jiawei Shen, Christos Papavassiliou, *Senior Member, IEEE*, Timothy G. Constandinou, *Senior Member, IEEE*, and Themistoklis Prodromakis, *Senior Member, IEEE*

***Abstract*— The capability of in-memory computation, reconfigurability, low power operation as well as multi-state operation of the memristive device deems them a suitable candidate for designing electronic circuits with a broad range of applications. Besides, the integrability of memristor with CMOS enables it to use in logic circuits too. In this work, we demonstrate with examples the design flow for memristor-based electronics, after the custom memristor model already being integrated and validated into our chosen Computer-Aided Design (CAD) tool to performing layout-versus-schematic and post-layout checks including the memristive device. We envisage that this step-by-step guide to introducing memristor into the standard integrated circuit design flow will be a useful reference document for both device developers who wish to benchmark their technologies and circuit designers who wish to experiment with memristive-enhanced systems.**

***Index Terms*— CAD tool, circuit design, hybrid CMOS/memristor, in-memory computation, low-power, RRAM.**

I. Introduction

MEMRISTOR is a two-terminal passive device where the resistance can be altered by allowing electrical current to flow. An important characteristic feature of the device is its non-volatility i.e it remembers the resistance value when the power is switched off (memory function). In addition, this resistance value can also be programmed i.e increased or decreased depending on the amount of current flow and its direction. Chua's theory [1] was overlooked for decades due to the lack of technological advancement at that time. Nonetheless, after nearly four decades, the connection of Chua' famous work was proven physically by observing resistive switching in a titanium dioxide device developed by Strukov *et al.* [2] at Hewlett-Packard Laboratory.

The primary application of a memristor has been an energy-efficient and scalable memory element, where the data is stored in terms of the resistance value. For high-density memory, crossbar array architecture [3], where memristors are connected across each junction, is the most suitable topology. Other unique properties of the memristor are its multi-bit storage capability and state retention with no power applied [4].

Memristors are generally used as a memory but they can be used to construct circuits (as in logic gates) and also be integrated with CMOS for many applications. The non-linear circuit and system theory plays an important role in the memristor-based circuit design [5], [6]. Memristor-based design is an emerging concept fuelled by the continually growing need for energy-efficient computation. The memristor property of a variable resistive state due to the applied voltage (amplitude, number of pulses) has lured digital designers to use them for representing ON and OFF logic as low and high resistance states respectively. Thus, numerous works on memristor-based logic, where the memory operation of the device is combined with the Boolean function have been demonstrated including MAGIC [7], IMPLY logic [8], Stateful memristor gates [9], Complementary resistive switches [10], Akers logic array [11], Memristor rationed logic [12].

Manuscript received March 30, 2021; revised July 20, 2021; accepted October 15, 2021. This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) Programme under Functional Oxide Reconfigurable Technologies (FORTE) Grant EP/R024642/1, in part by a SYnaptically connected brain-silicon Neural Closed-loop Hybrid system (SYNCH) under Grant H2020-FETPROACT-2018-01, and in part by the RAEng Chair in Emerging Technologies under Grant CiET1819/2/93. This article was recommended by Associate Editor Y. Zhang. *(Sachin Maheshwari, Spyros Stathopoulos, Jiaqi Wang, Alexander Serb, and Yihan Pan contributed equally to this work.) (Corresponding author: Sachin Maheshwari.)*

Sachin Maheshwari, Spyros Stathopoulos, Jiaqi Wang, Alexander Serb, Yihan Pan, and Themistoklis Prodromakis are with the Centre for Electronics Frontiers, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: s.maheshwari@soton.ac.uk; s.stathopoulos@soton.ac.uk; jw9y17@soton.ac.uk; a.serb@soton.ac.uk; y.pan@soton.ac.uk; t.prodromakis@soton.ac.uk).

Andrea Mifsud and Christos Papavassiliou are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: a.mifsud@imperial.ac.uk; c.papavas@imperial.ac.uk).

Lieuwe B. Leene is with Novelda AS, 0484 Oslo, Norway (e-mail: lieuwe.leene@novelda.no).

Jiawei Shen is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: jiawei.shen17@imperial.ac.uk).

Timothy G. Constandinou is with the Department of Electrical and Electronic Engineering and the Care Research and Technology Centre, U.K. Dementia Research Institute, Imperial College London, London SW7 2AZ, U.K. (e-mail: t.constandinou@imperial.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2021.3122381.

Digital Object Identifier 10.1109/TCSI.2021.3122381

The salient feature of memristive devices has gained popularity in crossbar arrays [13], [14] for implementing brain-inspired computing [14]–[17]. Here the variable resistance of the memristor is used to mimic the function of a synapse in a neural network. The threshold logic gate based on such a function is configured to implement both the neuromorphic and Boolean logic [18]. However, since as a standalone device the memristor suffers from significant signal degradation in cascaded logic gates [12] and sneak current paths [19], thus, it was observed that to implement a large neural network using a threshold function, memristors integrated with CMOS were found suitable. Most of the memristor-based logic gate approaches are exemplified within the structure of crossbar array such that it performs in-memory computation [20]. As demonstrated through the extensive work on circuits, the memristor-based logic design holds great potential for high density and energy-efficient computing. Recently, a hybrid CMOS/memristor chip for neuromorphic computing has been fabricated with a memristor crossbar array directly integrated with custom-designed CMOS circuits that contain mixed-signal blocks and a digital processor for re-programmable computing [21]. This successful integration practically demonstrates efficient hardware implementation in terms of area and energy consumption and also gives an add-on property of re-configurability [16].

As memristive devices are prominently being researched and are finding their application in the next generation nano-electronics, it is important to build a step-by-step design flow for primary researchers to design and validate memristor-based circuits before it is laid onto the wafer and integrated with CMOS. Although this manuscript uses the VCM memristor model (validated in Part I [22]), which binds together as much of the commonalities that exist across the broadest possible sample of memristive devices, the given methodology can be easily utilized with alternative memristive models in designing circuits.

In Part II of this design flow, the behavioral model that was validated in Part I [22] is adopted for hybrid CMOS/memristor design using the Cadence Virtuoso Design Environment for schematic capture, layout design, Verilog-A/Spectre for circuit simulation, and Mentor Graphics Calibre for physical verification including DRC, LVS, and PEX. The work of this Part II is organized as: Section II focuses on using the resistive switching memory (RRAM)[1] to build circuit blocks for crossbar arrays and to handle high voltage. Later in this section, the design of a re-configurable gate is demonstrated using RRAM. This section ends with a CMOS-memristor design workflow that takes into account the reconfigurability and uncertainty of RRAM. Section III, introduces a potential fully customised layout of a single memristor followed by physical verification using Calibre. Besides, a memristor array in a 16×16 crossbar structure is also illustrated in this section. The electrical and physical verification in this work has been done using a commercially available 0.18μm CMOS technology.[2] The paper is concluded with a discussion note in section IV.

[1]This work uses RRAM and Memristor interchangeably.

II. Designing With Memristors

All of the effort described in Part I [22] sets the groundwork for the circuit design using memristive devices (RRAM). Myriads of circuits using RRAM have been conceived and implemented thus far with a broad space of possibilities open for the future. In this section we shall show by examples how design with RRAM can be carried out, pointing out any pitfalls or points of particular interest for the design engineer.

A. Primitive Cells for Controlling Memristive Devices

Once a device model has been settled upon, the next logical step is to start developing primitive CMOS-RRAM cells for practical use as building blocks for larger circuits. At this stage a number of operational, implementation and non-ideal parameters have to be considered. Operational parameters include supported voltages and polarities, pulse durations and in general the waveform parameters of all signals involved in operating the RRAM (which may not necessarily be pulses). Implementation parameters include the sizing of the RRAM and associated MOSFETs, the widths of the back-end lines connecting the circuit and the overall layout topology (for example, do we want the cell to be tile-able into an array?). Layout issues are covered specifically in the next section, but these should be considered during the design phase as well. Finally, non-idealities include the series resistance of transistors, line resistances, maximum voltage tolerances and parasitic capacitances.

In this section of Part II, we will see a couple of examples of fundamental RRAM-based circuit modules; namely the 1T1R structure that represents the combination of a single RRAM element with a single switch (which can be interpreted as an access control device). This very same structure is used to build crossbar arrays with selectors. Next, we move on to a 2T1R topology used to handle high-voltage electroforming in a CMOS technology that does not easily support high voltages on-chip.

1) The 1T1R Primitive: 1T1R topology whereby a transistor is connected in series with a RRAM cell is shown in fig. 1. Despite this being a fundamental building block by itself we shall discuss it within the more complex operating environment of a crossbar array for completeness; this will be reflected in the labelling of the block's terminals, which will correspond to their typical connectivity within the crossbar context. The first issue that arises immediately is whether a pMOS or nMOS transistor should be used as the switching element. This will depend on the importance of the differentiating characteristics of each device respective to the design at hand. nMOS transistors have higher mobility and so feature higher current drive capability than their equivalent, same-width pMOS. Additionally they do not require an N-well substrate

[2]The given design flow can be adapted to different technologies and other tool vendors, e.g. Synopsys, Tanner, and or modules e.g. Hspice, Assura, Pyxis, Eldo, etc. The tools specified here are not exhaustive but intend to give you an idea of other possibilities.

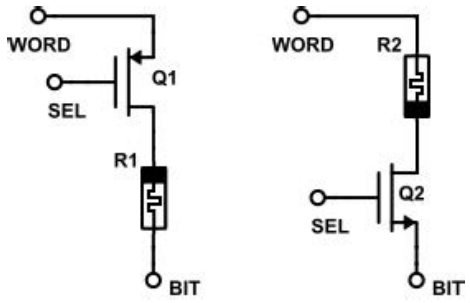

Fig. 1. Examples of 1T1R topologies. Left: pMOS-based design. Right: nMOS-based design. Note: at this stage we have said nothing about the operation of the 1T1R, so the labels 'WORD' and 'BIT' are there only as visual guides. Similarly the polarity of the RRAM device is only indicative; in principle the RRAM devices can be arranged or indeed interpreted as being in any direction. The same applies to all schematics in this section.

structure as a pMOS design would need both N-well taps as well as substrate taps in order to ensure good quality bulk biasing. This makes nMOS devices faster overall, however it also precludes the possibility of using the bulk terminal for anything other than simple grounding. In most cases this will suffice to render nMOS the polarity of choice.

In our example design we have chosen a pMOS-based design for experimental purposes: we wish to be able to control the pMOS transistor bulk voltages column wise in the array in order to understand how it can influence device operation. We shall seek to connect the N-wells of the pMOS devices column-wise in our array (which will have implications for layout and so cell size).

Next, we must consider whether we require bipolar operation and if so, whether this should be symmetrical (i.e. we want to pass similar voltage/current magnitudes in both directions). In either case, the required operation voltages/currents will determine the type of transistor that we use; most notably the allowable V_{GS} and V_{DS} (although each MOSFET's V_{GS}, V_{GB}, V_{DS}, etc. all need to be checked thoroughly for suitability). These voltage requirements allow us to exclude any unsuitable transistor types. For example if experimentally a RRAM cell requires 2V to switch, using a MOSFET that will suffer gate dielectric breakdown if either V_{GS} or V_{GD} exceeds 1V would be ill advised. Once a list of suitable transistors is found, the selection can proceed on the basis of 'positive attributes', i.e. finding the smallest/easiest to operate/most robust option, depending on project requirements.

In our example we will consider the symmetrical case. The RRAM devices will be modelled as having a minimum static resistance of $1\,k\Omega$, fixed for all voltage biases across the device, and require $1.5\,mA$ to operate (switch) in both directions; in other words, we always consider the device to be at its absolute worst case (see toy example in Fig. 2).[3] This translates to 1.5V dropped across the worst-case memristor in both directions. This means that the transistors used in

[3]This is a very conservative design approach and the level to which it is secure depends on the confidence level that the actual worst case is known. In a technology characterised by uncertainty, determining the confidence interval is an activity that must be undertaken very carefully. Furthermore, linearising the I-V to worst case is useful for setting design constraints. It should not be used for precise design as the estimated and actual currents may vary quite significantly.

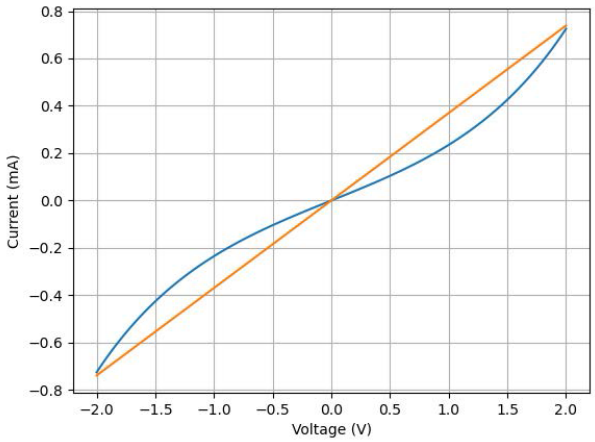

Fig. 2. Constructing a worst-case (current) boundary of our device's behaviour: Visual example of typical -but fictive- double-exponential, static (i.e. in the absence of switching) I-V characteristic of single, independent RRAM device (blue) assumed to be at hard LRS (i.e. the lowest Low Resistive State of interest, requiring the highest operating currents) together with worst-case linearised I-V (orange) within the voltage operating range of interest; here $[-2, +2]V$. In this example our test device reaches a minimum static resistance of approx. $2.85k\Omega$ at $V_{bias} = 2V$, which is then taken to be the resistance/slope of the linear IV marking the absolute worst-case bound. I.e., by using the orange I-V we always assume the device will require more current than it actually does at any given bias voltage, including if in practice it might switch to a higher resistive state if stimulated with some of the higher voltage magnitudes in the range of interest. In the examples used throughout this text we simplify further by using a worst-case resistance/slope of $1k\Omega$.

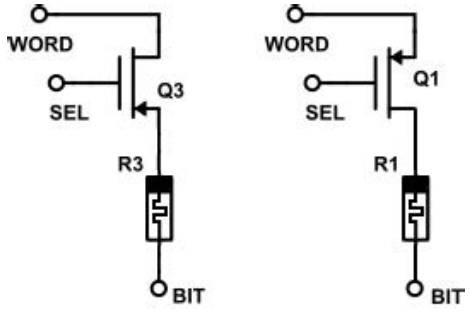

Fig. 3. Circuit diagrams for 'source-to-RRAM' (left) and 'drain-to-RRAM' (right) connectivity configurations. These are not identical if the transistor is not symmetrical (i.e. its drain and source terminals are not interchangeable).

the 1T1R scheme will have to be rated for at least 1.5V. In practice the required rating may become substantially higher if the series resistance of the FET device is comparable to that of the worst-case RRAM resistance. Note: we can always trade off transistor aspect ratio for FET series resistance, but there are practical limits to how wide we are willing to design our transistors. In our case this means we limit ourselves to using $3.3V$ or $5V$ devices; excluding all lower voltage variants offered by the technology (for the required transistor width becomes exceedingly large if our power supply headroom is reduced to, say, 2V).

Once a suitable transistor has been selected we need to identify whether is it symmetrical or not, i.e. whether source and drain can be used interchangeably. Should that be the case, the 1T1R design becomes symmetric up to relabelling. However, should that not be the case (e.g. in the case of

an extended drain transistor,[4] the V_{GD} required to admit some drain current i_x will differ from the corresponding V_{GS}. In this case the natural first instinct would be to 'point the source' of the transistor towards the RRAM as shown in fig. 3 (left schematic) so that when current flows first through the FET device and then through the RRAM, the FET drops approximately its V_{GS}' worth of bias voltage, leaving the rest to drop across the RRAM (saturation mode assumed). Then, when the polarity of the current is reversed, V_{GD} (now acting as the effective V_{GS}) is directly controlled by V_G and one of the biased lines (wordline or bitline), i.e. it no longer depends on the voltage between FET and RRAM. This allows the FET to now consume $V_{DS} \geq (V_{GD} - V_{th,D})$ bias voltage while remaining in saturation, where $V_{th,D}$ is the effective threshold voltage when the FET is used 'in the wrong direction', with V_{GD} acting as V_{GS}. Generally $V_{th,D}$ can be expected to be substantially larger than the 'normal' V_{th}. Thus, the 'source-to-RRAM' case can be expressed mathematically as:

$$V_{bias,f} = V_{mem} + V_{DS} \approx V_{mem} + (V_{DG} - |V_{th,D}|) \quad (1)$$

for the forward case (i.e. $V(WORD) > V(BIT)$), where $V_{bias,f}$ is the bias voltage in the forward direction. The gate voltage of Q3 and BIT are both treated as GND. The approximation corresponds to Q3 being around the onset of saturation.

$$V_{bias,r} = V_{mem} + V_{SG} \approx V_{mem} + |V_{th}| \quad (2)$$

where $V_{bias,r}$ is the bias voltage in the reverse direction. Here gate of Q3 and WORD are at GND. The approximation corresponds to the case where enough current passes through Q3 to enforce $V_{SG} \approx V_{th}$.

The 'drain-to-RRAM', on the other hand is expressed as follows:

$$V'_{bias,f} = V_{mem} + V_{SD} \approx V_{mem} + (V_{SG} - |V_{th}|) \quad (3)$$

i.e. similar to $V_{bias,f}$, but now the $V_{th,D}$ penalty has been reduced to V_{th}, allowing for more current, all approximations holding and all other things being equal. Meanwhile in the reverse direction:

$$V'_{bias,r} = V_{mem} + V_{DG} \approx V_{mem} + |V_{th,D}| \quad (4)$$

so again similar to $V_{bias,r}$, but V_{th} has now been replaced by $V_{th,D}$, increasing the penalty and reducing the voltage drop (and current) across the device (usual assumptions hold).

As a result, the 'point the source towards the RRAM' topology seems to allow for a more balanced application of voltage/current across the RRAM device all else being equal by using the fact that $|V_{th,D}| > |V_{th}|$ to mitigate the differences between forward and reverse currents.[5] This analysis assumes that the effective voltage threshold is lower when the devices are operated with the source and drain terminals playing the intended roles, rather than being reversed - i.e. that the devices are asymmetric. Table I illustrates this situation by showing numbers from a simulation of both configurations' behaviour under 5V power supply and $1k\Omega$ fixed resistive load. Currents are given in the forward (current flows from 'WORD' to 'BIT' terminals) and reverse voltage bias regimes. It is clear that the 'source-to-RRAM' configuration is more balanced and thus more suitable for RRAM devices with symmetric current requirements, such as we investigate in this example. Asymmetric devices were used for this test.

TABLE I

CURRENT PASSED THROUGH 1T1R CONFIGURATIONS FROM FIG. 3 FOR THE 'SOURCE-TO-RRAM' AND 'DRAIN-TO-RRAM' CIRCUIT CONFIGURATIONS FOR AN ASYMMETRIC PMOS TRANSISTOR FEEDING A $1k\Omega$ (RESISTIVE WORST-CASE) LOAD UNDER A 5V SUPPLY. WHEN THE VOLTAGE AT THE WORD TERMINAL IS HIGHER THAN A THE BIT TERMINAL, WE ARE IN 'FORWARD CONFIGURATION'

	Forward	Reverse
Source-to-RRAM	2.5mA	1.7mA
Drain-to-RRAM	4.5mA	1.3mA

However, the drain-to-RRAM configuration could be useful for cases where the RRAM devices have highly asymmetric current requirements. The reason is that the drain-to-RRAM configuration supports substantially higher maximum currents in the forward bias case (i.e. where current flows from 'WORD' to 'BIT' terminals). Thus, the conclusion is that unless the situation has a very obvious solution (e.g. there are obvious symmetries or extremely accentuated asymmetries), it is advisable that simulations are carried out for both topologies and both bias polarities before the most acceptable 1T1R configuration can be selected.

The simulations used to generate the corner cases summarised in Table I are very simple: They are DC operating point analyses of the circuits illustrated in fig. 3 taken at the corner cases where the voltage across the 1T1R stack (using resistive approximation of worst-case for the role of 'R') is the nominal power supply (typ. close to the maximum voltage tolerated by the transistor) and the gates of the transistors are fully closed (V_G = VDD for nMOS devices and GND for pMOS[6]). A specific configuration and transistor sizing will in general pass the basic performance criteria if: a) It can successfully pass sufficient current in both directions onto the worst-case load and b) It falls within the rated voltages of the devices supported by the technology. Note: in some technologies this is indicated by the presence or absence of safe operating area check (SOAC) errors.

Thus far we have covered the signal voltage and polarity support. The pulse duration then introduces an additional set of considerations, mostly in terms of heating. Continuous stress, as might arise due to extremely long pulses (possibly even ms or s) means that the calculations for FET, and even more pertinently metal line current-carrying capacity, need to be adjusted accordingly. In general, however, this becomes

[4] An extended drain transistor is one where the drain terminal is placed at an increased distance from the gate, as opposed to the source terminal, which is immediately adjacent to the gate. This allows extended drain transistors to handle much higher source-drain voltages than regular transistors, but in exchange they become asymmetric with the threshold voltage between source and gate being smaller than the threshold between drain and gate (i.e. when the current flow polarity changes and the drain is forced to play the role of source).

[5] Of course, in both cases significant channel resistance (the V_{DS} effect) will further complicate the equations.

[6] Note: in the more general case where the drain terminal of the device can exceed the power supply rails the formula is for V_{GS} = VDD, so that the allowable values of V_G depend on the current value of V_S.

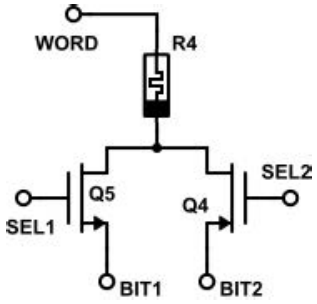

Fig. 4. Schematic of the 2T1R system with asymmetrical devices, that we use to bidirectionally apply 'high voltages' (above the supported max. V_{GS} of the transistors) to the test RRAM device. BIT1 is grounded whilst BIT2 is fixed to the highest voltage we wish to apply (here 10V). When we need to apply 'positive' voltage to the device, Q5 is ON and WORD is at voltage $V_w > 0V$ as appropriate. When applying 'negative' voltage to the device, Q4 is ON and WORD is at $V_w < 10V$ as appropriate. Q4 and Q5 are never concurrently ON.

a layout issue for the back-end lines as transistor sizes are adjusted to keep current densities (and as a result channel resistance and local power dissipation) under check.

Finally, we note that just like in any other array design, aiming for good layout compactness, matching, low parasitic capacitance and tileability farther down the design cycle helps improve the performance of the 1T1R cell. This is particularly pertinent in 1T1R designs that are very frequently used as the building blocks for large crossbar arrays.

Please note that frequently in practical designs, the assumption of symmetrical FETs can be made, which greatly simplifies the design.

Additional, crossbar array-specific points: As alluded at the beginning of this section, if the 1T1R primitive is to be used in a crossbar array, additional design considerations arise, most notably the interconnection of the 1T1R elements. Whilst this lies outside the scope of this section, we will note that the typical connectivity of a crossbar stipulates (canonically) that the wordline (connecting all 'WORD' terminals) runs perpendicular to the bitlines and selector lines (shorting BIT and SEL terminals column-wise respectively). This allows us to use combinations of (WORD, BIT) and (WORD, SEL) to isolate individual RRAM devices within the array with the aid of their corresponding 'selector transistor'. Other approaches are possible, but in general maximum versatility is ensured if one set of lines runs perpendicular to the other two.

Additionally, the connectivity of the transistor bulks needs to be specified. In a vanilla nMOS-based array, the bulk terminal will be the substrate and so there is no further decision point. However, if pMOS devices, triple-well nMOS or e.g. FDSOI technologies are used the connectivity pattern of the bulk terminals needs to be also specified. In general, there are only a handful principal connectivity combinations that satisfy the restriction that for an $N \times N$ array we have a maximum of N bulk lines servicing all the bulks. First, column-wise connected bulks would allow crosspoint combinations between bulk and wordline bias in a similar fashion as is used to isolate individual cross points for normal word-line/bit-line operation. Next, we could have row-level connectivity. Finally, there is the option of connecting all bulk terminals to a common supply.

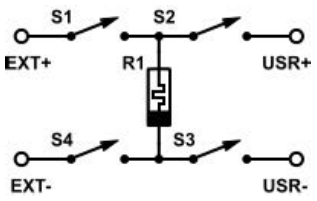

Fig. 5. Schematic of double transmission-gated RRAM primitive cell. In normal operation either the pair S1,4 or S2,3 are closed, connecting the RRAM cell to the 'external' (EXT terminals) or 'in-circuit' environment (USR terminals). This is very convenient in particular for circuits that only require the RRAM devices to maintain a fixed resistive state during circuit operation.

2) The 2T1R Primitive: Next, we consider a more complicated primitive cell example. This is an illustration of how primitive cell design can quickly become much more complicated if the RRAM technology used requires voltages higher than what the underlying CMOS technology supports as a standard.

In this scenario we have a technology that supports up to X volts as a standard, but the RRAM requires $Y \gg X$ volts bidirectionally for guaranteeing successful electroforming [4]. In our scenario the CMOS technology does not feature transistors that support Y across all possible terminal pairings, but does feature extended drain FETs that support V_{DS}, V_{GD} and V_{DB} at the level required by Y. Furthermore, the resistance of the RRAM device is assumed to be able to take any value, including single digit kΩ and MΩ. For the example that follows we arbitrarily set X=3V and Y=10V.

By the assumptions of our scenario we cannot guarantee a sufficiently low voltage drop across the RRAM cell. As a result, a 1T1R solution is unworkable even using the extended drain devices because we would be forced to assume that the transistor's V_{DS} can swing between $\pm 10V$. Instead, we opt for a 2T1R approach as shown in fig. 4, with extended drain (and therefore asymmetrical) devices. The circuit operates on the basis that the two bitlines are fixed at 0V and 10V while the wordline can move freely between 0-10V. This might be the case if the wordline is directly connected to a pad (with appropriate ESD voltage limits), for instance. We note that both FETs have their sources 'pointing away' from the RRAM, which is done in order to guarantee that by controlling V_G we can control the 'normal' V_{GS} in both cases (since V_{GS} must be <3V). Then, when we wish to apply +10V to the RRAM, we apply 10V on the wordline and sink the current through the nMOS Q5. Similarly, for applying −10V on the RRAM, we ground the wordline and turn on the pMOS (Q4). For intermediate voltages we set the wordline voltage correspondingly and choose according to the desired polarity which transistor will be responsible for applying the voltage. Naturally, how much of the voltage drop actually does occur across the target device is something that needs to be checked thoroughly and simulated.

In the example above, the pMOS and nMOS are never operated simultaneously. Furthermore, the gate voltages swing between [0,3]V for the nMOS and [7,10]V for the pMOS, therefore appropriate circuitry guaranteeing that no V_G can exceed those limits needs to be designed. Whilst this is no longer part of the primitive design, it is an example of a key primitive core decision that significantly impacts the

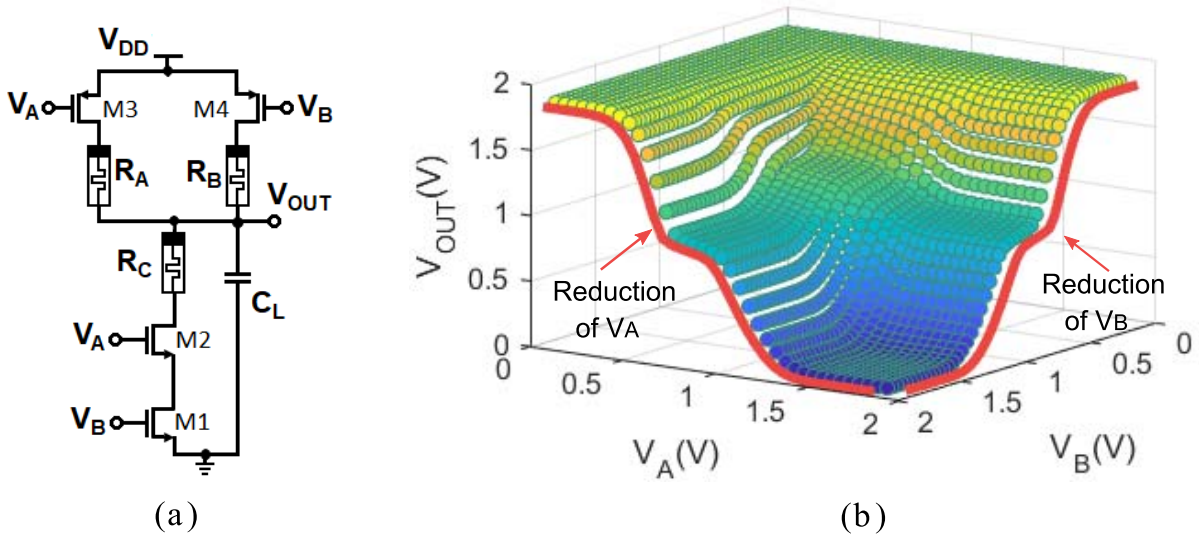

Fig. 6. RRAM-enhanced NAND gate. (a): Schematic of the RRAM-enhanced gate. (b): Input/output transfer function for a number of (analogue) input voltages at V_A and V_B. Changing the values of R_A, R_B and R_C changes the shape of the transfer function. Corner simulations (with extreme values of $R_{A,B,C}$) can reveal the extent to which the transfer function surface can be manipulated whilst simulations with $R_{A,B,C}$ at mid-range provide a good indication of the transfer function shape when the RRAM devices are at their most flexible. Reproduced from [23].

design of the rest of the system. Transistor selection and other implementation and parasitic considerations are similar to the 1T1R case with appropriate modifications.

Finally, we note that in this example we chose to keep both bitlines at fixed voltages (GND and 10V).This naturally extends the 1T1R 'forward' scenario where the bitline is fixed to GND. There may, however, be interesting control options in varying those voltages as well.

3) Other Primitives: Naturally, there are other significant RRAM-CMOS topologies that can be developed, such as a dual transmission gate-accessible RRAM cell (fig. 5) that can be used to switch the RRAM cell between 'in-circuit' and 'out-of-circuit' operation. This could be used for example to operate the RRAM cell normally in-circuit and then change its resistive state in a very controlled fashion using the out-of-circuit terminals.

B. Simple Design Example: A Reconfigurable Gate

The use of our primitive components is exemplified very clearly in the design of a reconfigurable logic gate, as originally described in [23]. The schematic for the reconfigurable NAND gate is shown in fig. 6a. We notice that the NAND gate can be decomposed into a set of 2x pMOS-type 1T1R cells plus a single nMOS-type 1T1R with an extra transistor for enforcing the pull-down path. Naturally the 1T1R structures can be implemented such that the places of the RRAM and the transistors are swapped, as required by the problem specifications.

1) Nominal Circuit Design: Once the basic architecture of the gate is decided, the circuit needs to be fully specified (e.g. transistor sizes and RRAM nominal resistive state ranges). Importantly, we talk about RRAM 'resistive state ranges', as opposed to simply 'states' because in the general use case the reconfigurable gate will operate in a 'lifelong reconfiguration' mode, as opposed to what one might call a 'configure & forget' modality. Thus, in the general case the design will require that the I/O transfer characteristic can cover a multitude of states within certain bounds, as opposed to simply taking a single nominal shape.

The procedures for elaborating 1T1R-based circuits are still developing, but a good starting point is to set all CMOS devices to a suitably chosen sizing[7] and all RRAM devices to the middle of their expected operating range and then extract a 3D plot of the gate output voltage against the voltages at the inputs (A,B). If the resulting performance is acceptable, the corners can be analysed next: pull-up RRAM devices are set to their maximum (minimum) allowable resistance values and pull-downs to their minimum (maximum) values and the shape of the input/output (I/O) function is examined. Thereafter, an iterative process (either manually, or using machine learning techniques) can be carried out for altering the RRAM resistive state ranges until the I/O function meets all specifications. This may imply running the iterative process once for every 'corner' case of the I/O transfer characteristic, where 'corner' implies that the chosen I/O characteristic places a unique restriction on the RRAM resistive range of at least 1x RRAM device. The union of all resistive ranges demanded of RRAM device X by all corner cases yields the set of resistive states that device X should cover to guarantee proper operation. Should the I/O function specifications require RRAM resistive ranges lying outside the operating ranges supported by the initial choice of RRAM device designs (i.e. a solution cannot be found), either the RRAM devices can be redesigned or transistor sizing can be changed.[8]

[7]This may be such that the nominal circuit with all RRAM devices at the middle of their resistive ranges directly yields the central-case nominal I/O transfer characteristic, or set to minimise overall circuit size, or anything else as determined by the application specifications.

[8]Note: In order to develop better intuition of the problem, the following may also be carried out in aid of the designer: For both pMOS and nMOS 1T1R types, plot the current through the 1T1R vs. the voltage applied at the gate of the transistor and the voltage at the output node of the gate. This creates plots that can be interpreted as 3D load-lines.

Once the gate is complete, the usual battery of tests used to check circuit functionality and reliability can be applied to it: i) DC sweeps to check the I/O transfer characteristic, ii) transient analysis to check rise/fall times and assess power dissipation, iii) Monte Carlo (MC) and corner analysis to assess the full performance profile, iv) temperature sweeps etc. CAUTION: When carrying out any of the above tests it is imperative that the limitations of the RRAM model are taken into account: whilst the CMOS component of the circuit will account for all of the above behaviours, not all RRAM models will include e.g. MC or temperature data (almost none do at the time of writing).

2) Dealing With Uncertainty: Importantly, during MC and corner analysis the RRAM devices are treated slightly differently than ordinary CMOS devices because of the reconfigurability of RRAM. When designing our nominal circuit we explained how the requirements of the set of desired I/O characteristics in general lead to a resistive *range* requirement for each RRAM device. There is no guarantee that the device's actual operating range can cover the desired range, however. Furthermore, in general, a correction factor needs to be included to account for uncertainty in the devices' actual resistive state ranges. Thus, if our circuit analysis has resulted in a desired RRAM range of [A,B] (derived from all sources of uncertainty except the RRAM device itself), the nominal RRAM resistive range is [X,Y] and the α percentiles of the RRAM resistive range extrema lie at $X + \Delta x$ and $Y - \Delta y$ (i.e. $\alpha\%$ of the devices can swing as low as $X + \Delta x$ and $\alpha\%$ of devices can swing as high as $Y - \Delta y$) then satisfying the following condition:

$$[A, B] \in [X + \Delta x, Y - \Delta y] \tag{5}$$

guarantees that at least $(\alpha\%)^2$ of devices will feature enough swing to satisfy the $[A, B]$ requirement.

For this task, RRAM models including a description of (practically reachable) resistive range variability become necessary. As per standard practice, if the specs are not met under uncertainty, the system needs to be adjusted accordingly.

The exact same approach holds when we introduce operating temperature as a parameter in determining corners. This approach can, in principle, be automated and integrated into current industry-standard CAD tools such as Cadence; the principle of operation would be similar to the current criterion-based testing tools available for pure CMOS designs. In that respect designing with RRAM is no different than designing with pure CMOS.

Finally, if noise is likely to cause the effective value of the RRAM devices to fluctuate, then the design range of $[A, B]$ might need to be reduced even further. This might be the case, for example, in a circuit where we wish to guarantee an absolute minimum impedance A over e.g. a set of $1ns$ time windows. Especially if oversampling and other noise-reduction techniques are not applicable this extra effect needs to be considered separately.

In summary, a good design flow taking into account the reconfigurability and uncertainty of RRAM can be presented as per the flow chart of fig. 7.

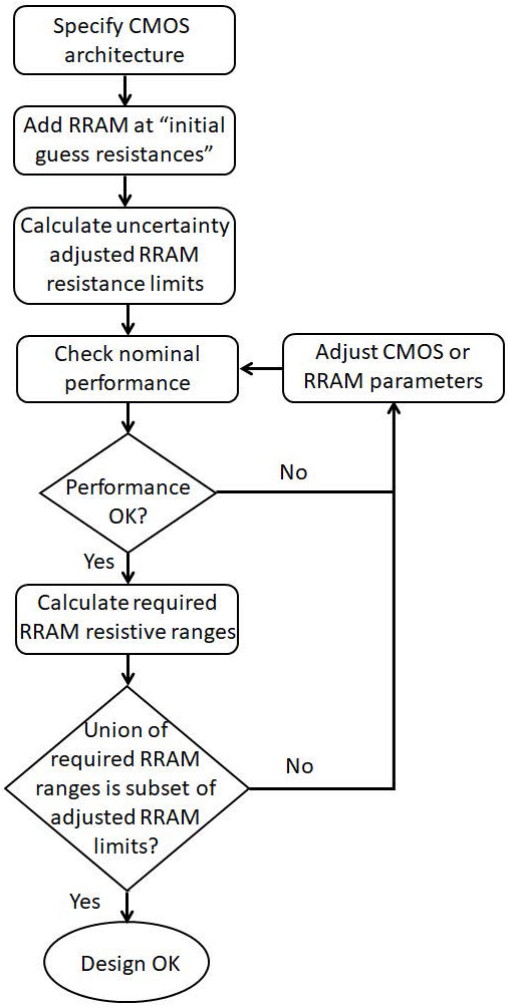

Fig. 7. CMOS design with RRAM indicative workflow. The preparatory work entails specification of the overall circuit architecture, adding RRAM devices at some nominal, initial guess states and computing the uncertainty-adjusted limits of the RRAM technology in use. Then the iterative part of the design begins. First, nominal performance is checked and adjustments made if necessary. Then, the required ranges for each individual RRAM device are computed such that the circuit can cover all its performance corners. If the union of required RRAM ranges is a subset of the originally calculated uncertainty-adjusted RRAM limits, then the design can proceed. Otherwise, further adjustments need to be made.

III. Physical Design - Layout

After we have designed our CMOS-RRAM primitive cells and larger systems, this section moves to the physical design of memristor devices and systems, referring to the design flow presented in Fig. 5 of Part-I [22]. First, the layout of a single memristor is introduced and mapped to its physical structure. Here, the layout approach is in conjunction with the existing CMOS based layout rules. The versatility of the approach is its compatibility with CMOS rule set, thus the memristor layers can be retrofitted with the existing process. The CMOS routes are linked to the memristor layers through customised vias allowing the memristor to be regarded as a standard component in the circuit. Moreover, several layout examples for memristor primitives depicted in Section II are demonstrated. A further step into layout design involving memristor array is provided. Finally, this section demonstrates memristor layer mapping description for exporting layout to GDSII stream. The layout

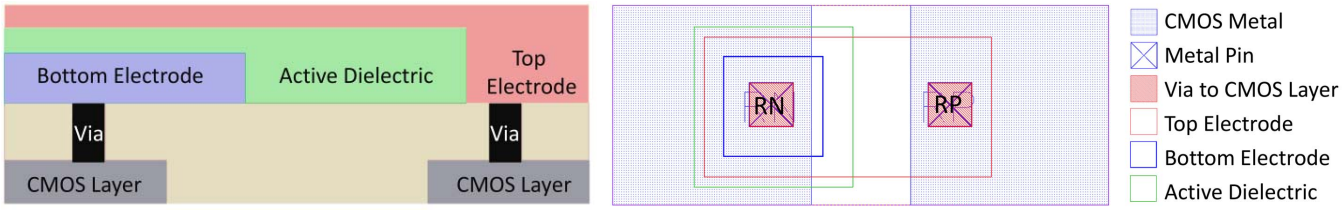

Fig. 8. Layout of a memristor (right) with its equivalent cross section view (left). Layers are annotated corresponding to its MIM physical structure. Vias to CMOS layer create the link between memristor and CMOS metal routing.

examples provided in this section are constructed in 0.18μ m CMOS technology and the physical verification is performed with Calibre.

A. Standard Layout Cell

Memristors, as a new designed devices, requires a fully customised layout design by engineers to specify the layers of different functionalities. In order to separate them from the CMOS process, the model should be built on layers that differ from CMOS manufacturing layers. Fig.8 shows the layout of a single memristor with its layers labelled corresponding to its physical structure. An equivalent cross section view is provided on the left.

Fig. 8 on the right displays the layout of our in-house memristor implementation, including three memristor specified layers and one metal layer from CMOS. This memristor has a dimension of 5 μ m $\times$ 2 μ m. The layout also has a 2.9 μ m $\times$ 1.4 μ m top electrode and a square bottom electrode with a size of 1μ m^2. Meanwhile, the active dielectric lies in-between both electrodes, covering a slightly larger square than the bottom electrode (1.6 μ m $\times$ 1.6 μ m). There are two vias between memristor metal layers to higher CMOS metal layer, enabling the access to top and bottom memristor electrodes in CMOS process. Finally, as a link to CMOS layers, two CMOS metal pins labelled as RP for memristor top electrode and RN for its bottom one are also placed. As a result, the memristor can be linked to CMOS design as a standard cell by routing the wanted memristor port to CMOS metal wires.

The memristor layout in Fig. 8 will be used as a standard layout cell in the following examples. In this cell, the CMOS metal layer, metal 4, is used to provide connections between memristor and CMOS routes. However, this layout configuration is not the only way to design a memristor; other variations are also possible. The dimensions of each layer can be adjusted depending on the technology that is used to fabricate the memristors, as well as other factors such as design purposes. At the same time, the distance between the two vias is also configurable. Situations can be highly variable according to different design.

B. Layout Instruction and Calibre Environment Setup

1) Calibre Design Rule Check: To fabricate the design successfully, DRC files are provided by the foundry that help the designers to check design violations. It is the first step before running the Layout versus Schematic (LVS) check. Performing DRC checking regularly can avoid accumulated errors.

2) Calibre Layout Versus Schematic: Calibre LVS compares electrical circuits from the specified source netlist and layout geometry. LVS applications establish a one-to-one mapping between the elements of one circuit (instances, nets, ports, and instance pins) in the source netlist to the layout circuit. This matching is completed when a one-to-one mapping between the elements is established. However, the standard pdk is unable to recognise the memristor device as this is not a built-in device in the design kit. Thus, when the LVS application attempts to map the source netlist to the layout netlist errors are reported.

In order to generate an equivalent matched circuit, the description in the text box below is required to be appended into the original "calibre.lvs" file located in the design kit. Additionally, the design kit contains another file named "source.added" that contains the information of all the sub-circuits. Here, the declaration of the memristor device used in the design is utmost important as otherwise LVS will show an error.

calibre.lvs/calibre.rcx

```
LAYER MEMRESLYR          450
//layer to form memristor
LAYER MAP 215~DATATYPE 21 450
MEMRESLYRT = MEMRESLYR AND M4
MEMRESLYRZ = MEMRESLYR NOT M4
CONNECT metal4~MEMRESLYRT
DEVICE memristor MEMRESLYRZ MEMRESLYRT(RN)
MEMRESLYRT(RP) netlist model memristor
```

source.added

```
.SUBCKT memristor RN RP
.ENDS
****************************************
```

calview.cellmap

```
(memristor
  (std_memristors memristor symbol)
  (
    (RN RN)
    (RP RP)
  )
  (
    (nil multi 1)
    (nil m 1)
  )
)
```

3) Calibre Netlist Extraction: After successfully running DRC and LVS, Calibre xRC (PEX) is required to generate

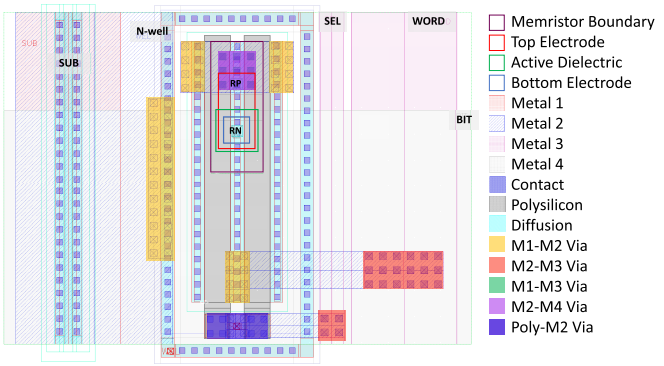

Fig. 9. The 1T1R primitive Layout with a focus on the memristor-linked routing. Its schematic is provided in fig.3 (left). The net names corresponding to the schematic are labelled on the layout. The memristor is placed above the pMOS with metal M2-M4 via connected to its top electrode. Memristor bottom electrode RN is only attached to metal 4 layer to the BIT line. The WORD and SEL lines are routed by metal 3 vertically through Poly-M2 and M2-M3 vias, providing necessary joints to transistor terminals. This graph only displays mostly the routing layers.

extracted view with all the parasitics. For creating PEX (xRC) rule files, same regulations (as for "calibre.lvs") should also be appended to the "calibre.rcx" file. Moreover, in order to extract the memristor netlist in the calibre extracted view, another description needs to be written and appended in the original "calview.cellmap" file present in the design kit. NOTE: The presented model does not incorporate the parasitic model hence PEX will only extract the netlist for the memristor without any parasitic.

C. Layout of Primitive Cells - 1T1R and 2T1R

Once we have designed a standard cell for memristor, systems involve memristors can then be normally routed as they are in CMOS process. The standard layout cell used in this paper uses CMOS metal 4 layer as a connection layer to the RRAM. Normal Layout routing tips for CMOS fabrication still apply to the systems here, for example, minimising noise coupling on critical signals, matching differential signals and widening power rail.

Layout of memristor system varies as this is dependent on the circuits being designed, and the layout of the cell. To illustrate the differences, layout examples are given based on the designed 1T1R and 2T1R primitives discussed in section II. These systems only include a single memristor whose layout design can be simple as long as only one memristor is used in a design. A more complicated case, the crossbar array, is discussed later.

Fig.9 shows the 1T1R layout structure in detail. Its corresponding schematic is given in Fig.3 (left) in section II. The three nets (BIT, WORD, SEL) corresponding to the schematics are labelled on the layout. The layers of memristor are provided for references. To make the design more visible, only selected CMOS layers are displayed, mostly the routing layers. In the 1T1R structure, the layout is made up by only a pMOS and a memristor. The pMOS has a much wider width than the memristor, so it is split into two fingers with a shared drain terminal. The memristor is placed above the split transistor that shares the drain terminal. According to the schematic, the memristor is connected to the pMOS source terminal. This connection is achieved by metal via M2-M4 (3×3) on the memristor RP port and metal via M1-M2 (4×2) on both sides of the pMOS source terminals. The via-to-via route (M2-M4 via to M1-M2 via) are connected by metal 2 layer. Moreover, the bottom electrode RN of this memristor remains its connection to the only CMOS layer, metal 4 layer. This metal 4 layer extends horizontally through the pixel connecting RN to BIT line. SEL and WORD lines are routed vertically in metal 3, coupling to the transistor gate and drain through Poly-M2 via and M2-M3 vias.

The 2T1R primitive layout is illustrated in Fig.10 with its schematic given in Fig.4 in section II. As explained in section II, the 2T1R structure requires the circuit operating up to higher voltages with a pMOS and a nMOS. The layout is highlighted by the net names that correspond to the signals in the schematic. In the layout, the two high voltage transistors are labelled respectively and one memristor placed in between the transistors (highlighted in the blue rectangle).

Just like in the 1T1R layout, the transistors are split into fingers with equal length. The high voltage pMOS has six fingers, while the nMOS is split into two. Each two fingers share one drain terminal for pMOS whereas nMOS has a shared source terminal. There are three rows of metal via M1-M2 on the pMOS. The three vias on the middle row are connected together to the memristor and the other vias link the fingered sources to the BIT2 line through metal 2 layer. In the close look to the memristor connection, the memristor has its top electrode connected to metal M2-M4 vias. These vias are further linked to metal 2 route horizontally, extending to both transistors. The route reaches the transistors' drain fingers through M1-M2 vias. The memristor bottom layer, connected to the word line, keeps its only connection to metal 4 layer, however, with a wider wire placing across the length of this whole cell. This is because the signal applied to the bottom electrode is a powered signal.The SEL1, SEL2, BIT1 and BIT2 are all routed in metal 3 layer which connect to transistors through various vias.

It can be noticed that all the port signals are placed to the edge of the cell with lengths equal to the cell width or length. This is designed on purpose considering to implement the cells in an array, section III-D will demonstrate more in details. The layout still follow the CMOS routing strategies for creating a professional design. For instance, the metal layers are placed in one direction as possible. Metal 3 are in vertical direction, whereas metal 2 and metal 4 are routed horizontally. Meanwhile, the ground signal is been placed at the lowest metal layer for minimising noises.

More importantly, the layout cells shown above are for illustrative purposes, more compact layout are also possible by shrinking the memristor sizes and keep distances at minimum.

D. Memristor Array Layout

Memristor takes advantage for compact layout cells since it has a simple structure. It has been widely used in array structures among the field such as image processing [24], memcomputing [25], neuromorphic systems and other in-memory computing [26]–[28]. It is possible to build high density memories by using the memristor array. A crossbar

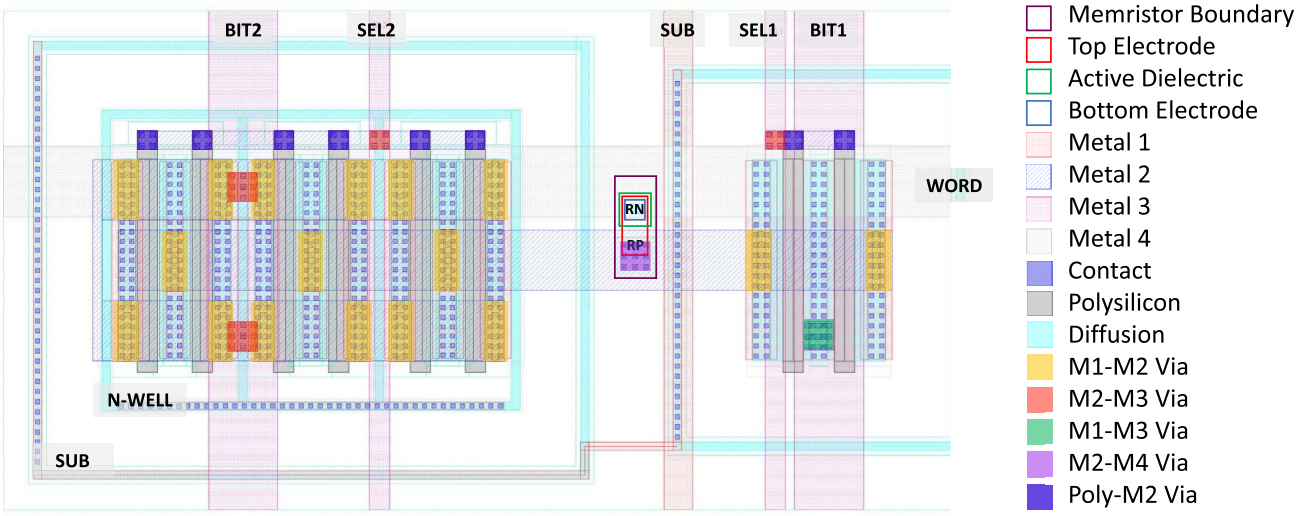

Fig. 10. The 2T1R primitive Layout highlighted with M1-M2 via on transistors. The schematic is given in fig.4 with the same net names labelled. A zoomed view of memristor area showing metal via M2-M4 on the top electrode. All the signals for ports are placed till the edge of the cell bringing benefits for array routing. This graph only displays selected layers for a clearer view.

array typically consists of bit-lines and word-lines controlling the unit memory cell. Each of the cross point has a memristive device. This configuration maximises the memory area density.

When designing an array with memristor, one should take care of sneak current path issue from design point of view [13]. As from layout aspect, main focus has been placed on the line resistance challenge that may occur when a large-scale array is designed. Typically, the wire resistance for a small array can be negligible. However, when it comes to scaling up the array, the wire resistance will result in a considerable voltage drop leading to a signal delivered lower than expected at the memristive devices. High wire resistance blocks the devices from receiving sufficient voltages for operation. Although this can be advantageous for some use-cases, in the other situation, it may breaks the functionality of the circuit, making the design potentially unusable in the case where not enough voltage can be applied across the memristor device, resulting in no suitable change in its resistance. Line resistance can never be eliminated but it can be kept at a minimum to reduce its negative impact to the circuit. At the same time, we also desire a compact memory cell which allows us to fit more devices in the same area.

Therefore, parasitic analysis is non-avoidable before implementing an array. For a reliable and high density crossbar array, it is suggested to use wider wire width as parasitic resistance usually decreases at a lower rate than parasitic capacitance increase. A trade-off should be considered between the wire width and the size of the layout. The wider the line width, the smaller the wire resistance. At the same time, the area of the layout increases as an increase in line width. For a small memory pitch, the limitation becomes the cell size itself. Moreover, for more complicated design, additional resistance also exist for the array with selectors when the switching components such as transmission gates are included. The introduce of transistors further weaken voltages to the memristive devices. Additionally it is important to look at the digital signals; as the RC constant will affect the minimum pulse width at which the line can be driven.

The 1T1R and 2T1R primitive layout cells demonstrated in Section III-C can be used into crossbar structures whose maximum array size depends on the line resistance. The overall line resistances can be approximated by measuring the width and length of the corresponding lines. These are then divided to find the number of squares in each line, which is then multiplied by the resistance value in the technology documentation provided by the foundry (usually in Ohms per square). In our examples, for the 1T1R primitive cell, the analogue signals which are routed with wide widths have a resistance less than 0.25 Ohms. The digital signals have a line resistance slightly larger but tolerable value of about 1 Ohms. Whether these are good enough or not depends on the application for which these memory cells are used.

Fig 11 shows a 16 × 16 array designed using 1T1R standard cell given in fig 9. The wires at the boundary extend to the signal pads, which are not shown in the figure. The gates (SEL<1:16>) of the array are connected vertically where each column has one signal to the pads at the bottom. The n-wells (NWELL<1:16>) for each cell are also controlled column-wised to the pads at the left side. They are built to control the bulk voltage for every column (discussed in section II). The bottom plates of the memristive devices (BIT<1:16>) in each row are routed together and accessed by the pads located at the right side. The right bottom graph indicates that the memristor bottom electrode RN is shared by one row connecting by metal 4 layer (only memristor-related connection is routed in this layer). The remaining top pads provide the external connections to the pMOS transistors' drains (WORD<1:16>) for all columns. An additional wire is routed for the array substrates (SUB). There are several ways of improving the standard cell layout for this array, including placing the substrate connection under the n-well. Optimising the layout of the cell is important but not within the scope of this work.

In array layout, the standard cells can be carefully designed with special techniques as used in other arrays found in image sensors and SRAM or DRAM memories. As noticed

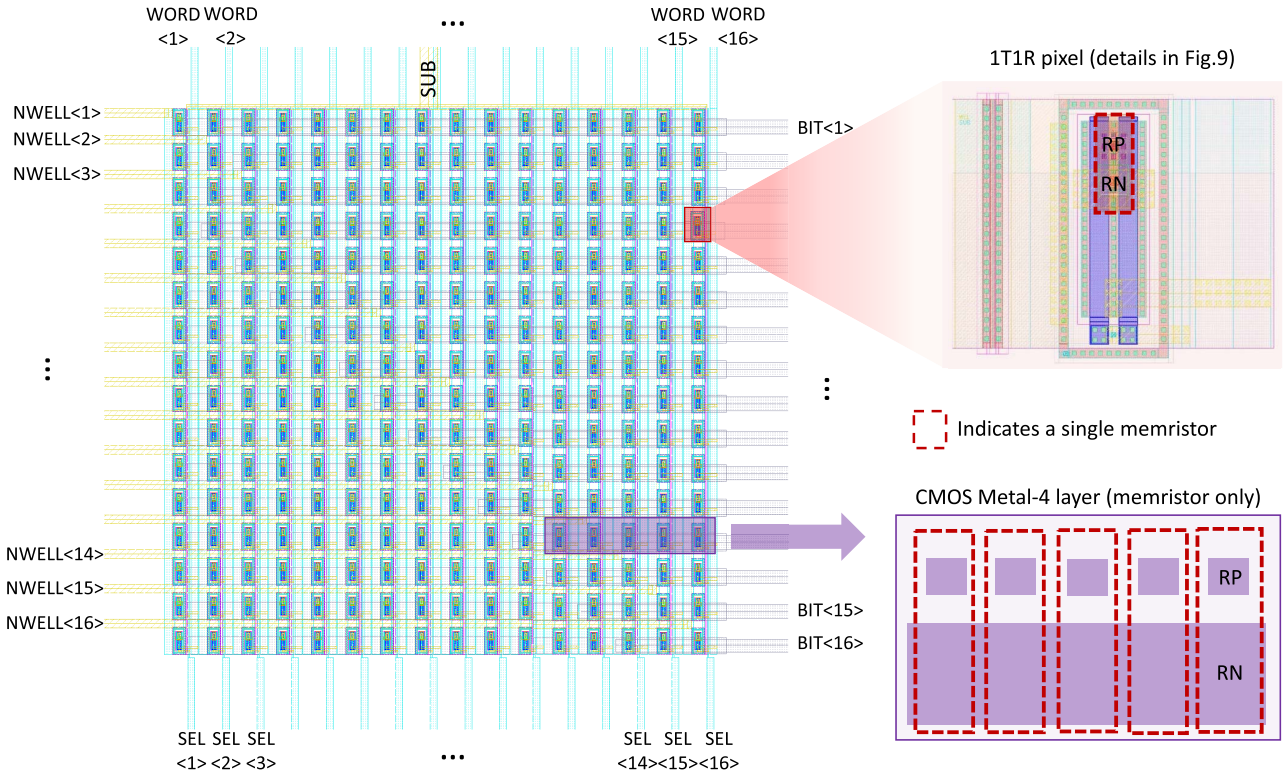

Fig. 11. Memristors array in a 16 × 16 1T1R crossbar structure with standard cells shown in Fig.9. The schematic corresponding net names are labelled on the layout. Memristive devices in a row share one bottom plate RN which is linked by metal 4. They are tied horizontally naming the port BIT<1:16>, whereas the remaining signals (WORD<1:16>, SEL<1:16>, NWELL<1:16>) are respectively controlled column-wise. The array substrate is connected as a whole by SUB.

on the 2T1R layout cell previously in Fig. 10, the port signals are routed till the edge of the cell boundary covering the lengths or the widths of their standard cell. In this way, the implementation of a array is achieved by simply aligning the boundaries of each cell. The wires are auto-connected since they are at the edge if handled by care. This alignment brings convenience to routing an array and keeps the standard cells dense and tight.

E. Exporting Layout to GDSII

The GDSII stream format is a file format which is the *de facto* industry standard for data exchange of integrated circuit layouts. It is a binary file containing a database of planar objects including geometric shapes, text labels, etc in a hierarchical form. This file alone can be used to transfer designs between different tools or to create masks used in the fabrication process.

Designs that are created using the Cadence Design Systems Virtuoso tool suite are typically exported to GDSII for submission to CMOS foundries.

However, to create a GDSII file containing custom design layers, e.g. for back-end RRAM processing, the designer needs some further insight into the GDSII format itself. The GDSII file format does not use layer names but instead, geometry exists on a numbered layer and datatype. Typically, the layer number and datatype can be in the range 0-255. There is therefore the need to translate the native design layers/purposes to GDSII layer numbers/data types. This is typically achieved using a "layer mapping table" and can be used for both importing and exporting GDS files. For CMOS layers, the default setting specifies this through the technology file that is internally linked to the design kit, using the process listed above. For custom post processing layers, this needs to be specified in a custom layer mapping table. This can be achieved in one of two ways: (1) by extending the technology file using the graphical user interface, or (2) manually through a plain text file. Most CMOS technologies provide both a technology file and layer mapping table, so it is also possible to extend the standard layer mapping table that is provided by the foundry.

However as extending the technology file, means the generated GDSII file will contain both the CMOS design layers and post-processing layers, this may be undesirable if submitting the CMOS design to a foundry. This may be desirable should we wish to contain the entire design (CMOS + RRAM layers) within a single file. The description below therefore focuses on a manual layer mapping description through a plain text file. An example is shown below. Key features are as follows:

- The Cadence layer purposes are listed in the LSW (layer window). Examples: dg = drawing, d1=drawing1, pn=pin.
- Layers that are not listed in the layer mapping table are not exported/imported.
- Stream layer numbers must be integers between 0 and 255. Numbers 1 through 127 are user defined, and 128 to 255 are reserved for system definition.

- Generally, only Stream data type 0 should be used for drawing layers.

postprocesslayer.map

```
#Cadence      Cadence      Stream      Stream
#layer        layer        layer       data
#name         purpose      number      type
#
via2rram      drawing        1           0
electrode1    drawing        2           0
foxide        drawing        3           0
Electrode2    drawing        4           0
```

Finally to export a GDSII stream using the custom layer mapping table, this needs to be specified in the File/Export/Stream form i.e. if a custom layer mapping file is specified this is used, otherwise if left blank, the technology file is used.

IV. Discussion and Conclusion

As we have seen, design with RRAM introduces a unique set of challenges ranging from the definition of the device model, to its incorporation into the CAD toolchain, the actual schematic-level design and all the way to the layout. In this guide we have presented a basic procedure that can take the designer through the entire flow from a CAD tool, to layout design. This is an essential first step before RRAM devices can start entering the mainstream, allowing the community to start embedding RRAM into CMOS and beginning the long effort required for characterising the technology in full and building a solid foundation for eventual incorporation of the technology into the standard toolkit available to the engineer.

Next steps towards the incorporation of RRAM into standard CMOS would include:

- Upgrading current RRAM models with e.g. variability data for Monte Carlo analysis and a parasitics model, among others.
- Creating a parametrisable cell (p-cell) for schematic design, much akin to the corresponding transistor models.
- Building a design rule check (DRC) deck so that layout tools can automatically check the correctness and manufacturability of RRAM device layouts.
- Upgrading the LVS deck for automatically recognising RRAM devices.
- Providing a small, basic library of fundamental designs (e.g. 1T1R) ready for use by designers, much like logic gates are provided for typical commercial CMOS technologies.
- Creating macros for generating e.g. memory blocks using RRAM devices using hardware description languages (HDLs).

These capabilities are expected to slowly mature over time as RRAM technology is tightly integrated with CMOS becoming an increasingly standardised and well-supported part of the CMOS fabric.

References

[1] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[3] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, and A. Rubio, "Crossbar-based memristive logic-in-memory architecture," *IEEE Trans. Nanotechnol.*, vol. 16, no. 3, pp. 491–501, May 2017.

[4] S. Stathopoulos *et al.*, "Multibit memory operation of metal-oxide bi-layer memristors," *Sci. Rep.*, vol. 7, no. 1, p. 17532, Dec. 2017.

[5] A. Ascoli, R. Tetzlaff, S.-M. Kang, and L. Chua, "System-theoretic methods for designing bio-inspired mem-computing memristor cellular nonlinear networks," *Frontiers Neurosci.*, vol. 3, May 2021, Art. no. 633026, doi: 10.3389/fnano.2021.633026.

[6] A. Ascoli, A. S. Demirkol, R. Tetzlaff, S. Slesazeck, T. Mikolajick, and L. O. Chua, "On local activity and edge of chaos in a NaMLab memristor," *Frontiers Neurosci.*, vol. 15, Apr. 2021, Art. no. 651452, doi: 10.3389/fnins.2021.651452.

[7] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[8] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.

[9] N. Wald and S. Kvatinsky, "Design methodology for stateful memristive logic gates," in *Proc. IEEE Int. Conf. Sci. Electr. Eng. (ICSEE)*, Nov. 2016, pp. 1–5.

[10] E. Linn, R. Rosezin, C. Kügeler, and R. Waser, "Complementary resistive switches for passive nanocrossbar memories," *Nature Mater.*, vol. 9, pp. 403–406, May 2010.

[11] Y. Levy *et al.*, "Logic operations in memory using a memristive Akers array," *Microelectron. J.*, vol. 45, no. 11, pp. 1429–1437, 2014.

[12] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—Memristor ratioed logic," in *Proc. 13th Int. Workshop Cellular Nanosc. Netw. Their Appl.*, 2012, pp. 1–6.

[13] J. Rajendran, H. Manem, R. Karri, and G. S. Rose, "Memristor based programmable threshold logic array," in *Proc. IEEE/ACM Int. Symp. Nanosc. Architectures*, Jun. 2010, pp. 5–10.

[14] G. Rose, J. Rajendran, H. Manem, R. Karri, and R. Pino, "Leveraging memristive systems in the construction of digital logic circuits," *Proc. IEEE*, vol. 100, no. 6, pp. 2033–2049, Jun. 2012.

[15] L. Gao, F. Alibart, and D. B. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, Mar. 2013.

[16] M. A. Zidan and W. D. Lu, "RRAM fabric for neuromorphic and reconfigurable compute-in-memory systems," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–8.

[17] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Mater.*, vol. 18, no. 4, pp. 309–323, 2019.

[18] G. Papandroulidakis, A. Khiat, A. Serb, S. Stathopoulos, L. Michalas, and T. Prodromakis, "Metal oxide-enabled reconfigurable memristive threshold logic gates," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[19] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," *Microelectron. J.*, vol. 44, no. 2, pp. 176–183, 2013.

[20] B. Chen, F. Cai, J. Zhou, W. Ma, P. Sheridan, and W. D. Lu, "Efficient in-memory computing architecture based on crossbar arrays," in *IEDM Tech. Dig.*, Dec. 2015, pp. 17.5.1–17.5.4.

[21] F. Cai *et al.*, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019.

[22] S. Maheshwari *et al.*, "Design flow for hybrid CMOS/memristor systems—Part I: Modelling and verification steps," *IEEE Trans. Circuits Syst. I, Reg. Papers*, 2021.

[23] A. Serb, A. Khiat, and T. Prodromakis, "Seamlessly fused digital-analogue reconfigurable computing using memristors," *Nature Commun.*, vol. 9, no. 1, pp. 1–7, Dec. 2018.

[24] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, pp. 52–59, 2018.

[25] R. Tetzlaff, A. Ascoli, I. Messaris, and L. O. Chua, "Theoretical foundations of memristor cellular nonlinear networks: Memcomputing with bistable-like memristors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 502–515, Oct. 2020.

[26] M. Laiho, E. Lehtonen, A. Russell, and P. Dudek, "Memristive synapses are becoming reality," in *The Neuromorphic Engineer*. College Park, MD, USA: Institute of Neuromorphic Engineering, 2010, doi: 10.2417/1201011.003396.

[27] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[28] W. Wan *et al.*, "A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and *in-situ* transposable weights for probabilistic graphical models," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 498–500.

Sachin Maheshwari (Member, IEEE) received the bachelor's degree in electrical and electronic engineering from ICFAI University, India, the master's degree in microelectronics from the Birla Institute of Technology and Science, Pilani, India, and the Ph.D. degree in electronics engineering from the University of Westminster, London, U.K. He is currently a Research Fellow with the Centre of Electronics Frontiers, University of Southampton, U.K. His research interests include energy recovery logic and neural network design with memristive crossbar arrays.

Spyros Stathopoulos received the Diploma degree in applied physics, the M.Sc. degree in microelectronics and nanodevices, and the Ph.D. degree in applied physics researching on shallow junction engineering in silicon and germanium from the National Technical University of Athens, Athens, Greece. He is currently with the NanoGroup, School of Electronics and Computer Science, University of Southampton, U.K., researching on the fabrication and characterization of memristive devices.

Jiaqi Wang (Graduate Student Member, IEEE) received the bachelor's degree in microelectronic science and engineering from Shenzhen University, China, in 2017, and the M.Sc. degree in microelectronics systems design from the University of Southampton, U.K., in 2018, where she is currently pursuing the Ph.D. degree with the Zepler Institute. Her research interests include memristor-based hardware design, and analogue and mixed-signal integrated circuit design for biosignal processing.

Alexander Serb (Senior Member, IEEE) received the degree in biomedical engineering and the Ph.D. degree in electrical and electronics engineering from Imperial College London in 2009 and 2013, respectively. He is currently a Research Fellow with the Zepler Institute (ZI), University of Southampton, U.K. His research interests include cognitive computing, neuro-inspired engineering, and algorithms and applications using RRAM, RRAM device modeling, and instrumentation design.

Yihan Pan (Graduate Student Member, IEEE) received the B.Eng. degree in electronic engineering with The University of Manchester, Manchester, U.K., in 2019, and the M.Sc. degree in analogue and digital integrated circuit design from Imperial College London, London, U.K., in 2020. She is currently pursuing the Ph.D. degree with the Zepler Institute, University of Southampton, Southampton, U.K. Her research interests include hardware topologies for symbolic processing and RRAM-based memory architectures.

Andrea Mifsud received the B.Eng. degree from the University of Malta in 2016 and the M.Sc. specializing in full-custom integrated circuit design from Imperial College London in 2017. He then joined the Science and Technology Facilities Council as a CMOS Sensor Design Engineer within the CMOS Sensor Design Group. He joined the Next Generation Neural Interfaces (NGNI) Lab and the Centre for Bio-Inspired Technology (CBIT), Imperial College London, in 2020, where he is currently a Research Associate. His work focused on the design and testing of CMOS image sensors, including both pixel and periphery circuit design.

Lieuwe B. Leene received the B.Eng. degree in electronic engineering from The Hong Kong University of Science and Technology in 2011, and the M.Sc. and Ph.D. degrees in electronic engineering from Imperial College London in 2012 and 2016, respectively. He is currently a Senior IC Design Engineer at Novelda AS. Previously, he was a Research Associate with the Centre for Bio-inspired Technology, Department of Electrical and Electronic Engineering, Imperial College London. His current research interests include low-noise instrumentation systems, brain–machine interfaces, data converters, and mixed signal circuits.

Jiawei Shen received the B.Eng. degree in electronic engineering from the University of Nottingham and the M.Sc. degree in analogue and digital integrated circuit design from the Imperial College London, London, where he is currently pursuing the Ph.D. degree. His current research interests include developing CMOS-memristor technologies, periphery circuits for on-chip memristor array, and analogue applications of memristor.

Christos Papavassiliou (Senior Member, IEEE) received the B.Sc. degree in physics from the Massachusetts Institute of Technology and the Ph.D. degree in applied physics from Yale University. He is currently with the Department of Electrical Engineering, Imperial College London. He has contributed to over 70 publications on weak localization, GaAs MMICs, and RFIC. He also works on memristor applications, sensor devices, and systems and antenna array technology.

Timothy G. Constandinou (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electronic engineering from Imperial College London in 2001 and 2005, respectively. He is currently a Reader of neural microsystems within the Circuits and Systems Group, Department of Electrical and Electronic Engineering, and also the Deputy Director of the Centre for Bio-Inspired Technology, Imperial College London. His research interests include neural microsystems, neural prosthetics, brain–machine interfaces, implantable devices, and low-power microelectronics. He is a fellow of the Institution of Engineering and Technology, a Chartered Engineer, and a member of the Institute of Physics. He serves on the IEEE Circuits and Systems (CAS) Society BioCAS and sensory systems technical committees and the IEEE Brain Initiative Steering Committee. He was the Technical Program Co-Chair of the 2010, 2011, and 2018 IEEE BioCAS Conferences, and the General Chair of the BrainCAS 2016 and NeuroCAS 2018 Workshops. He is the Associate Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS.

Themistoklis Prodromakis (Senior Member, IEEE) received the bachelor's degree in electrical and electronic engineering from the University of Lincoln, U.K., the M.Sc. degree in microelectronics and telecommunications from the University of Liverpool, U.K., and the Ph.D. degree in electrical and electronic engineering from Imperial College London, U.K. He then held a Corrigan Fellowship in nanoscale technology and science with the Centre for Bio-inspired Technology, Imperial College London, U.K., and a Lindemann Trust Visiting Fellowship with EECS UC Berkeley, USA. He is currently a Professor of nanotechnology and the Director of the Centre for Electronics Frontiers, University of Southampton, U.K. His background is in electron devices and nanofabrication techniques. His current research interests include memristive technologies for advanced computing architectures and biomedical applications. He is a fellow of the Royal Society of Chemistry, the British Computer Society, the IET, and the Institute of Physics. He holds a Royal Academy of Engineering Chair in emerging technologies and a Royal Society Industry Fellowship.