In-Memory Database Query

Iason Giannopoulos,* Abhairaj Singh, Manuel Le Gallo, Vara Prasad Jonnalagadda, Said Hamdioui, and Abu Sebastian*

In recent years, several in-memory logic primitives were proposed where bit-wise logical operations are performed in memory by exploiting the physical attributes of memristive devices organized in a crossbar array. However, a convincing real-world application for in-memory logic and its experimental validation are still lacking. Herein, the application of database query where a database is stored in an array of binary memristive devices is presented. The queries are formulated in terms of bulk bit-wise operations and are executed in memory by exploiting Kirchhoff’s current summation law. The concept is experimentally demonstrated by executing error-free queries on a small $4 \times 8$ selector-less phase-change memory crossbar. The impact of crossbar size, resistance of routing wires, and interdevice variability on the accuracy of the logical operations are studied through numerical and circuit-level simulations. Finally, a system for cascaded query is proposed that combines the in-memory logic with conventional digital logic and its functionality is verified on a healthcare-related database. It is estimated that an 11-step long query is executed in 36 ns, consuming 560 $\mu$W, thus achieving an energy efficiency of 166 TOPS/W.

1. Introduction

Today’s computing systems are primarily built based on the principle—known as von Neumann architecture—that data storage and processing units are physically separated. Hence, during the execution of computational tasks, data have to be shuttled back and forth between the processing and memory units, incurring significant costs in latency and energy. The energy and latency associated with accessing data from memory units is a key performance bottleneck for a range of applications, in particular the increasingly prominent data-centric computing tasks.

In-memory computing is an emerging non-von Neumann paradigm, where certain computational tasks are performed in the memory itself by exploiting the physical attributes of the memory devices as well as their array-level organization. It has recently been shown that computational primitives such as bulk bit-wise logical operations and matrix–vector multiply operations can be efficiently realized in memory using an emerging class of memory devices known as resistive memristive devices. Even though the matrix–vector multiply primitive is gaining traction in a range of applications such as deep learning inference, sparse coding, linear solvers, and hyperdimensional computing, there are few experimental demonstrations of end-to-end applications involving in-memory logic operations.

A potential application involving a high percentage of logical operations is data analytics in database management systems. When performing queries on large databases, one of the main challenges is to retrieve the stored data and bring it to the processor that will execute the query. The latency and energy associated with moving data to the processor is a key performance bottleneck. Conventional acceleration methods range from software-oriented schemes that solve queries approximately to hardware-oriented systems that parallelize operations using graphics processing units. However, these approaches do not tackle the main bottleneck of data movement caused by the limited cache capacity of processing units and limited memory bandwidth. This limitation is further enhanced by the constantly increasing size of real-world databases. Near-memory computing, which incorporates complementary metal–oxide–semiconductor (CMOS)-based computing units close to the main memory, is a promising approach to reduce the cost of data movement. However, it does not completely eliminate the separation between memory and processing units and involves additional challenges such as integrating the logic units into the main memory chip. Another tantalizing prospect is that of storing large databases in dense arrays of nonvolatile resistive memory devices (also referred to as memristive devices) and executing the queries directly in memory by exploiting the analog properties of memory devices and circuits. In a previous work, a novel resistive memory-based architecture was proposed to support a variety of...
query functionalities (including bit-wise logical operations) and perform them explicitly in-memory.\textsuperscript{[18]} However, the proposed design, based on a content-addressable memory structure, does not support efficient cascading of multiple in-memory query operations consisting of different logical operands. Moreover, a validation of the feasibility of realizing accurate query operations using existing resistive memory crossbar arrays has been missing.

In this article, we present the realization of database queries based on in-memory read logic primitive. We experimentally demonstrate the concept on a small fabricated phase-change memory (PCM) crossbar array by successfully executing multiple queries in memory involving AND and OR operations. We quantify the impact of interdevice variability and finite wire resistance on the system’s accuracy for different array sizes via circuit simulations. Subsequently, we introduce a design that executes cascaded queries of arbitrary length and complexity by combining in-memory computations with near-memory logic circuits in the crossbar’s periphery. Using the Cleveland heart disease database as a case study, we demonstrate the functionality of the proposed system via accurate HSPICE circuit simulations calibrated on the hardware experiments and evaluate its throughput and energy efficiency.

2. The Concept of In-Memory Database Query

Query operations can be performed on databases that are structured collections of attributes or features, associated with different subject or item entries. The objective of a query is to retrieve the entries from the database that satisfy certain constraints related to the attributes. When databases are represented in a bitmap representation (vectors of the logical “0” and “1”), it is possible to formulate the queries as bulk bit-wise operations on the attribute vectors (see Figure 1a). The key idea of in-memory database query is to store the database entries in arrays of memristive devices using their conductance as the logic state variable. Memristive devices can be scaled down to nanoscale dimensions\textsuperscript{[19,20]} and their nonvolatile storage capability ensures that no energy is spent to retain the stored information unlike in dynamic random access memory or static random access memory. Subsequently, the bulk bit-wise operations associated with the query operations are performed in place in the memristive arrays using in-memory logic.

For in-memory logic, we exploit the nonvolatile binary storage capability of the memristive devices. For example, the 0 and 1s can be represented by the memristive devices’ low and high conductance states, respectively. Several logical operations can be enabled through the interaction between the voltage and conductance state variables.\textsuperscript{[21,22]} One particularly interesting characteristic of certain memristive logic families is statefulness, where both the operands and the result are stored in terms of the conductance state variable.\textsuperscript{[2,23,24]} However, stateful logic operations involve the repeated writing or programming of memristive devices.\textsuperscript{[25,26]} This is highly undesirable given the limited cycling endurance of these devices as well as the large energy footprint associated with writing these devices (typically in the order of pico Joules per write event).

Hence, for the database query problem, we resort to nonstateful logic operations where the logical operands are stored as conductance values, but the result of the logical operation is obtained as a current signal. This method was first implemented in Pinatubo\textsuperscript{[27]} and has inspired the concept of scouting logic.\textsuperscript{[28,29]} The operands stay fixed in the memory array and the devices need not be programmed during the evaluation of the logical operation. Figure 1b shows the realization of bit-wise logical operations using scouting logic. When memristive devices are organized in a crossbar configuration, it is possible to implement logical operations such as AND and OR, by simultaneously activating multiple rows and with the appropriate choice of reference currents ($I_{\text{ref}}$) for the sense amplifiers (SA). This enables us to execute the database query operations. For example, a query that requests the input entries that satisfy attributes “A” AND “D” is performed by biasing simultaneously the crossbar rows that correspond to A and D with a specific read voltage ($V_{\text{Read}}$). The resulting read current ($I_{\text{Read}}$) at each column corresponds to the summed conductance of the memristive devices according to Kirchhoff’s circuit laws. The logical output is obtained from comparison with predefined reference currents via a SA per column. Therefore, the logical result is calculated in place without having to move the contents to an external processing unit. Moreover, the output is a vector with equal length

![Figure 1](image-url)
to the number of crossbar columns and contains the query response for all the entries, thus facilitating the execution of queries with high parallelism. Note that although the concept is not limited to two attributes per operation, only one kind of logical operations can be executed at once. For example, “A” AND “C” AND “D” is executable with appropriate adjustments to the reference current values and the SA’s precision as a function of the given number of operands. It appears that AND, OR, and XOR are the key logical operations needed for database query.[18] XOR is also compatible with scouting logic but would require SA that has three inputs and is designed to output a logical 1 when $I_{\text{Read}}$ is between $I_{\text{ref AND}}$ and $I_{\text{ref OR}}$.

Processors based on memristive devices with logic capabilities have been demonstrated in the past[30–34] and even specifically using the scouting logic concept.[35] However, several aspects related to the functionality of the database application at the array level have been overlooked by previous studies, especially in the case of selector-less crossbars. In this work, we analyze the limitations arising as the crossbar size increases and we set boundaries for successful scouting logic as a function of size and routing wire resistance. Moreover, we develop and calibrate a dynamic circuit model that accurately simulates the read and write operations on PCM crossbars, to derive the maximum selector-less crossbar size that can reliably perform database queries. Another fundamental shortcoming from previous scouting logic demonstrations is the optimal selection of reference currents, which is arguably the most important parameter of scouting logic. It is typically chosen to be the geometrical mean of the expected $I_{\text{Read}}$ values, as shown in Figure 1b, which is not optimal in the presence of interdevice conductance variations. In the following section, we analytically derive the optimal $I_{\text{ref}}$ values, and based on this, an end-to-end experimental demonstration of the database query application on a fabricated selector-less PCM crossbar is presented.

3. Results and Discussion

3.1. Experimental Results on Small-Scale Crossbars

It is possible to fabricate memristive crossbars with or without selector devices. When each memristive device is placed in series with a selector such as a diode or a transistor, it greatly simplifies the operational challenges associated with the crossbar but at the expense of higher fabrication cost and areal footprint.[36–38] Even though, the proposed in-memory database query approach can be implemented using both types of crossbars, here we study the more challenging case of selector-less crossbars.

Experimental results are presented using small-scale crossbars of PCM devices. PCM is one of the most advanced resistive memory technologies[39–41] and is being actively explored for in-memory computing.[42–44] We fabricated a crossbar array using projected PCM devices. A projected PCM device consists of a segment of a noninsulating material, referred to as the projection material, in parallel to the phase-change material segment (see Figure 2a). The essential idea is to decouple the read and write processes in a PCM device by taking advantage of the highly

![Figure 2](image-url)
nonlinear current–voltage (I–V) characteristics of amorphous phase change materials. The sheet resistance of the projection material is such that the current bypasses the highly resistive amorphous phase during the low-field “read” operation and flows through it only during the high-field “write” process. Thus the read signal is marginally affected by the nonideal electronic properties of the amorphous phase, which makes the projected PCM device remarkably immune to conductance variations arising from structural relaxation, 1/f noise, and temperature variations.\textsuperscript{[45,46]} In addition, by tuning the sheet resistance of the projection layer, one can alter the conductance ratio between the high and low conductance states.

The fabricated projected PCM devices were electrically characterized. An appropriate programming pulse in the 100 ns timescale was applied, to provide adequate power to melt the phase-change material. The trailing edge of the pulse is sufficiently short to quench the molten volume and create the amorphous phase. This process is typically referred to as “RESET” and the same term describes the achieved low conductance state. A lower-amplitude pulse crystallizes the amorphous volume in a process referred to as “SET” and leads to a high conductance state. The nonlinear I–V characteristics of the amorphous-to-crystalline phase transition are shown in Figure 2b. At low bias voltage, the PCM devices exhibit a fairly Ohmic behavior in both the SET and RESET states.\textsuperscript{[47]} Moreover, one of the main performance benefits of the projected PCM devices is the notably weaker field dependence over a wider biasing range.\textsuperscript{[46]} The resistor in series ($R_s$) with the device serves for current-control element in a selector-less implementation. Although the series resistor consumes some extra power during scouting logic operations (up to 20% of the total power in our experiments), controlling the current during threshold switching is essential for the proper operation and endurance of the used devices. The threshold voltage $V_{th}$ is a particularly critical parameter for the crossbar operation; therefore, it will be studied thoroughly in Section 3.2. The conductance variability associated with the conductance states upon repeated switching can be mitigated by an iterative programming scheme involving a sequence of program-and-verify routines (see Section 5). With this scheme, the conductance values of the SET and RESET states over multiple cycles were roughly uniformly distributed (see Figure 2c).

A scanning electron micrograph of a fabricated $4 \times 8$ crossbar is shown in Figure 3a. The projected PCM devices are fabricated in planar configuration and hence the design differs from the conventional cross-point structure, where vertical memristive devices are fabricated at the overlapping area of BL and WL wires. Although the areal efficiency may be reduced, there is no functional difference between this crossbar architecture and the conventional cross-point one, because the left and right terminals of this lateral device are connected to column and row wires, respectively. Each cross-point is populated with a metallic resistor in series with a projected PCM device, as shown in Figure 2a. A crossbar without selectors suffers inherently from current sneak-paths that depend on the states of the other devices in the array.\textsuperscript{[48]} Special biasing schemes have to be used for each operation, to limit this effect and make the selector-less crossbars functional (see Figure 3b). To successfully write an individual device through the selected cross-point, pulses with half the programming voltage have to be applied simultaneously to all rows and columns, except the row and column on which the selected device is located. The programming pulse is applied to the selected row, whereas the selected column is grounded. A fundamental prerequisite for this method is that the threshold voltage is higher than half of the programming pulse amplitude; otherwise, the state of the nonselected devices along the selected row and column will be disturbed. On the other hand, precise current measurement is required for both read and scouting logic operations, in which the nonselected rows are grounded so that the contribution of sneak-paths to the selected column

![Figure 3](https://www.advintellsyst.com/2000141/4.png)

Figure 3. Crossbar arrays of projected PCM devices. a) SEM images of the fabricated $4 \times 8$ selector-less crossbar which consists of a projected PCM device (along with a fixed series resistor) at each cross-point. The database values are mapped as conductance states of the PCM devices. b) Biasing schemes used to deal with sneak-paths during write and scouting logic operations. During write, a half-selection rule is used to avoid disturbing the state of nonselected devices. During scouting logic, the nonselected rows are grounded to minimize the effect of sneak currents.
current is minimized. The basic goal is to suppress unwanted voltage differences across nonselected devices by giving them additional grounding paths other than the column read-out node.[49–51]

Prior to presenting the database query experiment, we investigate two primary sources of inaccuracies related to the in-memory read logic. Scouting logic operation involves the simultaneous read operation involving two devices per column. Given that there are three possible combinations, namely, both devices are in a RESET state, both devices are in a SET state, and the two devices are in SET and RESET states, there are three distinct nominal values for the column currents.

The first source of inaccuracy arises from nonideal routing wires. The finite wire resistance causes voltage drops that lead to a reduced effective read voltage across the selected PCM devices. In particular, it becomes crucial for the devices that are positioned the farthest away from the voltage supply node and require the longest routing wires. The effect of this voltage drop is larger for the SET state because the routing wire resistance may become comparable with the SET state as the crossbar size increases. The finite resistance of the routing wires also results in sneak-path currents that tend to increase the effective read current from the RESET states by adding parasitic contributions from alternate devices in the array. Although the row-grounding method is used, sneak-paths cannot be entirely suppressed, because voltage gradients along the row-wires bias devices that otherwise would have grounded terminals. Combined, these intertwined effects diminish the effective read-current ratio, to a point where the three possible column currents cannot be differentiated reliably by an SA.

The routing resistance path is a critical fabrication parameter that depends on both the physical distance between the devices and the sheet resistance of the routing metal. Ultimately, the wire resistance per unit cell is a key factor that determines the maximum size of a functional crossbar. We used a circuit simulator to study the reduction of read-current ratio as a function of wire resistance and crossbar size. We took the worst possible scenario in which 1) all devices are programmed to the SET state to maximize the sneak-current and 2) the selected device is at the outermost corner that has the longest routing path. The selected device is programmed sequentially to both logical states and their read-currents are estimated. We used these values to reconstruct and compare the ratios of the column currents \(I_{00}, I_{01},\) and \(I_{11}\) that correspond to the logical combinations of (0,0), (0,1), and (1,1). The resolution limits were set according to the rule that the SA needs a signal that is at least 20% higher than its reference current, to work reliably against process-related variations and voltage-level imprecision. The failure points for AND and OR operations are met when \(I_{11}/I_{01} \leq 1.2\) and \(I_{01}/I_{00} \leq 1.2\) respectively. AND requires higher precision, hence fails slightly earlier than OR (see Figure 4a). One approach to increase the size of functional crossbars is to use an adaptive SA unit that adjusts

Figure 4. Inaccuracies associated with in-memory logic operations. a) Circuit simulations showing which logical operation fails as the routing-wire resistance and the crossbar size increase. The study was based on the worst-case scenario where read device lies at the longest routing point and the rest of the crossbar is at the SET state maximizing the sneak currents. Resistance/size combinations within the white area correspond to functional systems. AND requires higher precision, thus fails earlier than OR. b) Circuit simulations (similar with (a)) of crossbars with nonunitary aspect ratio and wires with 1 and 5 per unit cell. Reduced number of rows (attributes) results in functional crossbars with larger numbers of columns (entries). c) Probability density function (PDF) of the sum of read currents from two devices for the three possible state combinations. Each device conductance is assumed to be uniformly distributed with the standard deviation expressed as the percentage of their mean value. Column current is calculated, assuming \(V_{\text{read}} = 0.1\, \text{V}\) bias at both devices. The reference currents for AND and OR operations are set according to Equation (1) and (2). d) Numerical simulations of the three possible column currents given by biasing two devices with \(V_{\text{read}} = 0.1\, \text{V}\), each of them being programmed to a conductance state that follows uniform distribution. The same relative standard deviations are used to confirm the statistical analysis shown in (c).
the reference currents proportionally, to compensate for the changes in readout current along the routing path. An alternate approach is to use crossbars with nonunitary aspect ratio (see Figure 4b). Note that the real-world databases have much more columns (entries) than rows (attributes); therefore, the numbers of rows can be reduced to incorporate larger numbers of columns.

The second key challenge for accurate in-memory logic is the interdevice conductance variations. During a logical operation, the column current is proportional to the sum of the two device conductance values representing the operands, for example, a low and a high one would be $I_{\text{Read}} = V_{\text{Read}}(G_0 + G_1)$. Because of the inaccuracies in programming $G_0$ and $G_1$, they will be randomly spread around their respective target values (see Figure 2c). The probability distribution of the sum of two or more independent random variables is the convolution of their individual distributions. The distributions of the three possible column currents were analytically derived (see Supporting Information) and are shown in Figure 4c. The distributions of $G_1$ and $G_0$ were assumed to be uniform with mean values ($\mu$) of 50 $\mu$S and 1 $\mu$S, respectively. We set $V_{\text{Read}} = 0.1$ V and calculated the expected distribution of the three possible column currents for three different values of standard deviation ($\sigma$), expressed as the percentage of their mean conductance values. Based on this observation that the SET state variability has a relatively much higher impact on the column current, it is also possible to derive the optimal current reference levels for AND and OR operations (see Supporting Information)

\[
I_{\text{ref AND}} = I_{00} + \frac{\sigma}{\mu}(I_{11} - I_{00}) \tag{1}
\]

\[
I_{\text{ref OR}} = I_{00} + \frac{\sigma}{\mu}(I_{11} - I_{00}) \tag{2}
\]

The statistical analysis is confirmed by numerical simulations of 20 000 devices (see Figure 4d). Every pair of devices is programmed to conductance states uniformly distributed around the mean values $\mu_{G_1} = 50$ $\mu$S and $\mu_{G_0} = 1$ $\mu$S. Histograms corresponding to $I_{00}$, $I_{01}$, and $I_{11}$ show a perfect match with the analytical derivation.

For the experimental demonstration, we used a database with information about scientific awards presented to some of the worldwide IBM research laboratories. We programmed the devices of a $4 \times 8$ crossbar according to the bitmap representation of the database. SET states follow a uniform distribution around $\mu = 51$ $\mu$S with $\sigma = 3$ $\mu$S and RESET states have $\mu = 0.8$ $\mu$S and $\sigma = 0.1$ $\mu$S (see Figure 5a). Database queries are used to search for laboratories that meet certain criteria involving the geographical location, awards received, etc. We constructed two simple queries and executed them using the crossbar. The appropriate reference currents are set according to the logical operation associated with the query, as shown in Equation (1) and (2). The eight column currents are measured and plotted against the names of the laboratories assigned to each column node (see Figure 5b). They are sufficiently separated for reliable detection by a SA.

3.2. Circuit-Level Simulation Results on Large-Scale Crossbars

The requirement to program the PCM devices at the point of storing the database entries prior to executing the queries introduces an additional limitation to the size of functional crossbars, again unique to selector-less arrays. A consequence of voltage drop along the wires is that the programming voltage should be adequately increased to successfully program the outermost devices. This comes with the penalty of over-biasing the devices nearest to the voltage source potentially beyond $V_{th}$, that inevitably leads to state disturbance. According to the half-selection programming scheme, all the devices along the selected row and column receive a pulse with half the amplitude of the one applied to the selected device (see Figure 3b). To study the influence of programming on the half-selected devices, we developed a circuit model that emulates the behavior of the projected PCM devices based on conventional electronic components such as transistors, capacitors, and resistors (see Figure 6a and Section 5). It is a two-terminal component designed to behave like a projected PCM cell that is initially in the amorphous phase and depending on the voltage level, it may (or may not) switch to the crystalline phase. The device (M) is modeled by an N-MOS transistor ($N_M$) driven by a circuit of switches. The exact values of these

![Figure 5](image-url)

**Figure 5.** Database query experiment. a) The bitmap representation of the IBM Research database as mapped to the $4 \times 8$ crossbar. Depending on 0 or 1, each device is programmed to the respective conductance for the RESET or SET state. b) The measured column currents for two queries performed on the IBM Research database. The two selected rows that correspond to the attributes of each query are biased and the eight column currents are plotted against the names of the laboratories assigned to each column node.
components are finely tuned to match the experimental device characteristics. A set of 64 experimental $I–V$ curves are measured on different projected PCM devices and the emulator is configured to capture their mean behavior (see Figure 6b). Threshold switching is controlled by Schmitt triggers, which match the experimentally measured $V_{th} = 2.1\ \text{V}$. The circuit is designed in a fully symmetric manner, to capture the unipolar nature of PCM devices. Nonvolatility is ensured by the capacitor $C_M$ that retains the gate voltage required for each conductance state during read or scouting logic operations.

We implemented this dynamic model in a Synopsis HSPICE circuit simulator equipped with 65 nm CMOS library, to test the efficacy of the concept on a practical problem with a much larger database. The Cleveland heart disease database, that is available on the UCI machine learning repository, consists of medical metrics obtained from 303 patients with heart-related health problems. After binarization, the database featured 41 attributes requiring a $41 \times 303$ crossbar array, which can be interconnected using a wide range of wire resistances according to Figure 4b. But when the device is at the high conductance state, there is significant voltage drop across the series resistance, that implies a high programming voltage at the row node. A representative programming pulse obtained during the small-scale experiments shows that approximately half of the programming power is dissipated in $R_S$ (see Figure 6c). Given these power requirements, the minimum voltage for reliable programming is set at $V_{\text{Prog}} = 3.6\ \text{V}$. According to the write scheme (see Figure 3b) the voltage across nonselected devices is half the programming one; therefore, the condition for undisturbed programming is $V_{\text{Prog}}/2 < V_{th} = 2.1\ \text{V}$. As a consequence, the maximum level of the node voltage to compensate for the losses is $V_{\text{max \ Prog}} = 4.2\ \text{V}$.

We simulated the system at the worst possible scenario, in which the selected device is the outermost one. The other devices are in the high conductance state, apart from one control device that is at the nearest position to the voltage source (see Figure 6d). We measured the programming power delivered to the selected device using wires with resistance in the range from 0.2 to 1.25 $\Omega$ per unit cell. $V_{\text{max \ Prog}}$ was applied at the selected node, ensuring that the voltage across the control device did not exceed $V_{th}$. Results indicate that for reliable programming under the worst conditions, the $41 \times 303$ crossbar should be split in half and the wire resistance cannot exceed 0.55 $\Omega$ per unit cell (see Figure 6e). The latter implies that as the number of entries increases, the database should expand into more than one crossbar array. In the context of in-memory logic, this is not a particularly detrimental problem, because the crossbar can be split into smaller segments along the rows, without influencing the scouting logic operation at the cost of marginal increase in circuit-level complexity. Nevertheless, it is clear that a crossbar with selector devices would offer significantly better control and flexibility at

![Figure 6.](https://www.advancedsciencenews.com/fig6.png)

**Figure 6.** a) A circuit model that emulates the behavior of the projected PCM device. b) The experimentally obtained $I–V$ measurements from the projected PCM devices are compared with that obtained from the emulator. c) The RESET programming pulse which is 100 ns long with a trailing edge of 10 ns. Approximately half of the total programming power is consumed at series resistance. d) Schematics of a crossbar biased according to the “write scheme.” This combination of stored values and selected device’s position with respect to the voltage node is the most prone to failure. e) The maximum programming power that can be provided to the farthest-positioned device without exceeding $V_{th}$ at any nonselected device, as a function of wire resistance per unit cell.
the initial set of write operations, in addition to the major power consumption benefits associated with not using the “write scheme” at the deselected devices.

We generated SET and RESET conductance states according to the Cleveland heart disease database. The SET (RESET) conductance states were uniformly distributed with a mean and standard deviation of 50 (0.8) and 2 (0.1) μA, for the logical “1” and “0” states, respectively (see Figure 7a). These distributions were mapped to the 41 × 152 and 41 × 151 subcrossbars (see Figure 7b) and replicate the experimental results of Figure 2c in a larger scale. The routing wires’ resistance was set to 0.2 Ω per unit cell. This value corresponds to the sheet resistance of the two lowest-level metals in the 65 nm node, assuming the most compact design (4F² footprint), in which the distance between devices is only twice the minimum width of the wire. Queries were constructed for all possible permutations of the 41 attributes and applied to the crossbar. Statistical analysis of the simulated column currents shows error-free query response for both OR and AND logic (see Figure 7c). To mitigate yield or early-device-failure issues, techniques such as avoiding columns that correspond to failed devices and rewriting the content in redundant ones can be applied. The explicit case of selector-less crossbars with devices that fail into an electrical short would require deactivation of the corresponding row as well, although this failure mode is less common in PCM technology.

3.3. Cascaded Database Query

Real-world database queries consist of a multitude of subqueries with associated logical operations rather than a single query. Solving such a query in the previously demonstrated fashion could yield an inefficient system. The main challenge is that it requires an additional memory unit for temporary storage of intermediate logical results and subsequent fetching for further processing along with the next set of logical outputs. To address this, a configurable computing system is introduced that combines in- and near-memory computations. Note that any query can be expressed as the sum of products (SOP), \((a_1 \times b_1) + (a_2 \times b_2)\), or the product of sums (POS), \((a_1 + b_1) \times (a_2 + b_2)\), where sum and product operators correspond to OR and AND, respectively. However, their occurrence is instructed by the query and is not necessarily alternated OR and AND operations. Hence, an arbitrary query function \(F(A)\) can be expressed as a combination of POS and SOP, as given by

\[
F(A) = F(S_1) * F(S_2) * \ldots * F(S_p)
\]

(3)

where \(F(S_i) = a_i * b_i\), \(*\) is an OR or AND operator, and \(p\) depends on the query length.

The key idea of the proposed cascaded logic computing system is to perform a logical operation both in-memory and near-memory simultaneously. While an in-memory analog computation using scouting logic is executed at the memristive crossbar, a near-memory digital logic operation is conducted at the periphery of the memory array using conventional CMOS-based gates (see Figure 8a). But rather than independently computing in parallel, the system executes the decomposed query, expressed in terms of Equation (3), in a cascaded manner. At a given clock cycle, the control unit selects the crossbar rows that correspond to the questioned attributes and configures the SA by selecting...
the appropriate reference current (switch S1), as instructed by the query operator. Subsequently, the logical results obtained at the SA output nodes are stored in a buffer. This will serve as the first input to the digital gate that can be either an OR or an AND gate, depending on switch S2 that enables the corresponding flip-flop and multiplexer channel. The second input to the digital gate is

Figure 8. Cascaded logic operations. a) Schematic illustration of the cascaded system showing how the analog and digital computations are distributed to the crossbar and periphery, respectively. The digital circuitry design, that locates at each column node, consists of an SA, CMOS logic gates, a multiplexer, and the switches that configure the circuit according to the query-instructed operation. SA converts the analog result from each crossbar column to digital and feeds it to the selected digital gate. There it cascades with the preceding logical products, until the final result is calculated at \( V_o \) after as many cycles as the analog operations. b) An example query with three operations applied to the 41 × 303 crossbar, that is programmed according to the heart disease database. The final operation \( C = A \text{ AND } B \) uses the partial results A, B that correspond to OR operations. A simple illustration of the cascaded logic system in the "POS" configuration that outputs the result C after the end of cycle #2. c) Simulated waveforms of the digital circuitry solving an 11-step cascaded database query. The three nodes correspond to one column periphery and their positions are marked in (a). The final result can be read at \( V_o \) node right after the end of the last clock cycle (#6). d) Simulated computational metrics for solving the 11-step example query on the heart disease-related database.
the accumulated logical result of all the previously executed logical operations. The output of the digital gate will serve as the new intermittent result that gets buffered in a delay circuit. At the subsequent clock cycle this signal will in turn be buffered and will be input to the gate, along with the new crossbar output. In other words, at every cycle, the digital output gets updated with the subsequent result of the logical operation obtained from the crossbar, until the query function gets fully executed.

To demonstrate the concept, we created an example based on the Cleveland heart disease database and the emulator presented earlier. The query comprises the AND of two OR operations (see Figure 8b). At the first clock cycle, rows 3 and 41 are biased with $V_{\text{Read}}$ and each column current is measured by an SA that is configured to perform the OR scouting logic operation. The result of OR is input to the digital AND gate. For the first cycle, the second input to the digital AND gate would be initialized to logical level 1. At the subsequent cycle, rows 1 and 2 are activated and their partial logical result (OR) is input to the digital AND gate along with the buffered OR result from cycle #1. The final query response is the binary vector that consists of the 303 logical results, as obtained from the output nodes of the digital gates right after the end of cycle #2.

The system offers massive parallelism combined with no need for high-power device programming and thus is expected to execute queries with remarkable efficiency. We used our 41 x 303 crossbar emulator to provide computational metrics based on the experimentally measured power consumption, on which we built the cascaded logic digital circuit using the 65 nm CMOS components. Delays related to charging the selected row and column wires and other parasitic capacitors have been well compensated by setting the clock’s period to 6 ns, that is 2.2 x larger than the time required for the accurate operation of the used current-latched SA. The simulated system was clocked at 167 MHz and configured to solve POS. We chose an example query comprising 11 consecutive OR and AND operations, out of which 6 are performed in the analog domain. A maximum of two operations are performed at each clock period which means that the total time required is 36 ns (see Figure 8c). Simulated performance metrics revealed that both the PCMBased crossbar and the digital-logic circuitry (gates, flip-flops, multiplexer) have a very low energy impact compared with the crossbar in two subarrays (see Section 3.2), suppressed almost completely the drawbacks associated with the lack of selectors. The uncontrollable voltage gradients along the wires are kept below 2 V even at the outermost areas of the crossbar. Consequently, only 0.1% of the total power was wasted at the grounded rows as undesired signal and the capacitive currents are considered negligible. Based on these results, using crossbars with selectors as a way to suppress sneak-paths would marginally increase the power efficiency during scouting logic operations. Finally, 18% of the crossbar’s power consumption is owed to the series resistors (3.2 μW). This number is strongly data dependent; given that for each scouting logic operation, the fraction of power that is dissipated on the series resistors is 0.4% when both stored values are 0s and can reach to 20% when both are 1s.

4. Conclusion

We have presented the concept of in-memory database query where the database is stored in a dense array of memristive devices and the queries are performed in place using in-memory logic operations. We fabricated and characterized a 4 x 8 crossbar and experimentally demonstrated the execution of error-free database queries. Moreover, we developed a circuit model that emulates the nonlinear electrical behavior and the nonvolatility of these devices. The emulator was incorporated into a circuit simulator that solved successfully a real-world database query problem in memory. In addition, we studied the operational challenges of selector-less crossbars for this application and provided the array size limitations as a function of wire resistance and the aspect ratio. Finally, we introduced the concept of cascaded logic that combines in-memory computations at resistive crossbars with peripheral CMOS logic and executes queries with arbitrary size and complexity. The proposed system can process database queries with massive parallelism, high throughput, and low energy consumption.

5. Experimental Section

Device Fabrication: Projected PCM devices were fabricated on an isolating 100 nm SiO$_2$ layer, thermally grown on a 525 μm n-type silicon wafer. The projection layer was a metal nitride formed by reactive sputtering. The pure metal target was sputtered in the presence of gaseous N$_2$. Note that in this step the sheet resistance of this film was tuned by adjusting the N$_2$ flow rate in the sputtering chamber. Subsequently, an ultrathin film of phase-change material (Sb$_2$Te$_3$ 10 nm) was sputtered and protected by a 5 nm SiO$_2$ capping layer. These three layers were deposited sequentially in the same sputtering chamber to avoid oxidation of the nitride interface and the PCM material arising from exposure to atmospheric air. Several e-beam lithography routines were executed to 1) pattern the stacked layers via ion milling to the design shown in Figure 2a and open contact areas through the capping layer, 2) define the contact electrodes, the series resistors, and the first-level metal wires by RIE of a sputtered tungsten layer, and finally 3) form the second-level metal and robust top-level pads for the probe card used in the electrical characterization setup.

Electrical Characterization: A four-channel Keithley 2606B source-measure unit (SMU) was used to DC bias and measure the column currents. The nanosecond-long programming pulses were applied by an Agilent 8110A pulse generator and monitored by an Agilent MSO6104A digital oscilloscope. A Celadon T40AF probe card with 25 probes arranged in line with a 100 μm pitch was used to contact the crossbar pads and the
Signals were distributed to the specific probe with a Keithley 707A switching matrix.

Iterative Programming Scheme: The verification condition was met when the device conductance fell within the target range given by $G_{\text{target}} = \pm \Delta G$ according to specifications. If the programming pulse failed to create an appropriate conductance state, the voltage amplitude was readjusted, following the equation $V_{i+1} = V_i + \theta \cdot [\log(G_i) - \log(G_{\text{target}})]$, where $\theta$ is a parameter that has to be set according to the programming characteristics of the devices (in our case $\theta = 0.2$).

Circuit Model and Simulator: Under low electric field, both phases had linear $I$–$V$ characteristics.\(^3\) Because scouting logic is carried out under low-field biasing, the devices were represented by Ohmic resistors. The device model used in Section 3.2 consisted of circuit components drawn from TSMC 90 nm library (see Figure 7a). The N-MOS transistor $N_M$ has a $W/L$ ratio of 30 and the series resistance is $R_s = 5 \, \Omega$. The switching unit consists of two Schmitt triggers to ensure symmetrical behavior independent of the bias node ($V_{\text{source}}$ or $V_{\text{drain}}$). They are supplied with external DC power $V = 4 \, \text{V}$, to keep the state unaffected from the input signals. The switches require additional components which are the resistors $R_{\text{in}}, R_p = 18 \, \text{k}\Omega, R_s, R_p = 11 \, \text{k}\Omega$, and the P-MOS transistors that have $W/L(P_{\text{in}}, P_p, P_s, P_p) = 10$. The gate voltage $V_m$ is built up by charging the capacitor $C_m = 30 \, \text{fF}$ through a resistor $R_m = 300 \, \text{k}\Omega$.

Supporting Information
Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements
This work was supported in part by the European Research Council through the European Union’s Horizon 2020 Research and Innovation Program under grant 682675 and the EU Horizon 2020 project MNEMOSENE under grant 780215.

Conflict of Interest
The authors declare no conflict of interest.

Keywords
database queries, in-memory computing, phase-change memories

Received: June 23, 2020
Revised: August 9, 2020
Published online: October 14, 2020

[1] M. Horowitz, International Solid-State Circuits Conference Digest of Technical Papers, 2014, p. 10.
[2] D. S. Jeong, K. M. Kim, S. Kim, B. J. Choi, C. S. Hwang, Adv. Electron. Mater. 2016, 2, 1600090.
[3] D. Ielmini, H.-S. P. Wong, Nat. Electron. 2018, 1, 333.
[4] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, E. Eleftheriou, Nat. Nanotechnol. 2020, 15, 529.
[5] A. Siemon, T. Breuer, N. Aslam, S. Ferch, W. Kim, J. van den Hurk, V. Rana, S. Hoffmann-Ebert, R. Waser, S. Menzel, E. Linn, Adv. Funct. Mater. 2015, 25, 6414.
[6] N. Talati, S. Gupta, P. Mane, S. Kvatinsky, IEEE Trans. Nanotechnol. 2016, 15, 635.
[7] L. Cheng, Y. Li, K.-S. Yin, S.-Y. Hu, Y.-T. Su, M.-M. Jin, Z.-R. Wang, T.-C. Chang, X.-S. Miao, Adv. Funct. Mater. 2019, 19, 1905660.
[8] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. Le Gallo, K. Moon, J. Woo, H. Hwang, Y. Leblebici, Adv. Phys. 2017, 2, 89.
[9] Q. Xia, J. J. Yang, Nat. Mater. 2019, 18, 309.
[10] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, Q. Xia, J. P. Strachan, Adv. Mater. 2018, 10, 1705914.
[11] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, E. Eleftheriou, Nat. Commun. 2020, 11, 2473.
[12] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, W. D. Lu, Nat. Nanotechnol. 2017, 12, 784.
[13] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, E. Eleftheriou, Nat. Electron. 2018, 1, 246.
[14] G. Karunanatne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, A. Sebastian, Nat. Electron. 2020, 3, 327.
[15] N. Potti, J. M. Patel, Proc. VLSI Endow. 2015, 8, 898.
[16] P. Bakkum, K. Skadron, in Proc. of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units Association for Computing Machinery, Association for Computing Machinery, New York, NY 2010, pp. 94–103.
[17] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, K. Yelick, IEEE Micro. 1997, 17, 34.
[18] M. Imani, S. Gupta, S. Sharma, T. S. Rosing, IEEE Trans. Comput. Aid. Des. Integr. Circuit. Syst. 2019, 38, 628.
[19] F. Xiong, A. D. Liao, D. Estrada, E. Pop, Science 2011, 332, 568.
[20] K.-S. Li, C. Ho, M.-T. Lee, M.-C. Chen, C.-L. Hsu, J. Lu, C. Lin, C. Chen, B. Wu, Y. Hou, C. Yi, C. Lin, Y. Chen, T. Y. Lai, M. Y. Li, L. Yang, C. S. Wu, Yang Fu-Liang, in Proc. Symp. on VLSI Technology, IEEE, Piscataway, NJ 2014, p. 1.
[21] J. J. Yang, D. B. Strukov, D. R. Stewart, Nat. Nanotechnol. 2013, 8, 13.
[22] I. Vourkas, G. C. Sirakoulis, IEEE Circuits Syst. Mag. 2016, 16, 15.
[23] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, R. S. Williams, Nature 2010, 464, 873.
[24] E. Linn, R. Rosezin, S. Tappertzhofen, U. Böttger, R. Waser, Nanotechnology 2012, 23, 305205.
[25] E. Lehtonen, J. H. Poikonen, M. Laiho, in 13th Int. Workshop on Cellular Nanoscale Networks and their Applications, IEEE, Turin, Italy 2012, p. 1.
[26] Y. Sun, Y. Wang, H. Yang, in IEEE 6th Non-Volatile Memory Systems and Applications Symp., IEEE, Hsinchu, Taiwan 2017, p. 1.
[27] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, X. Xie, in Proc. of the Design Automation Conference, ACM, New York, NY 2016, p. 173.
[28] L. Xie, H. A. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFAILakawy, S. Hamdiousi, in Proc. Computer Society Annual Symposium on VLSI, IEEE, Piscataway, NJ 2017, p. 176.
[29] S. Hamdiousi, H. A. Du Nguyen, M. Taouil, A. Sebastian, M. Le Gallo, S. Pande, S. Schaafisma, F. Catthoor, S. Das, F. G. Redondo, G. Karunanatne, A. Rahimi, L. Benini, in Proc. of the Design, Automation and Test in Europe Conf. and Exhibition, IEEE, Piscataway, NJ 2019, p. 486.
[30] S. Gao, F. Zeng, M. Wang, G. Wang, C. Song, F. Pan, Sci. Rep. 2015, 5, 15467.
[31] Y. Liu, Z. Wang, A. Lee, F. Su, C. Lo, Z. Yuan, C. Lin, Q. Wei, Y. Wang, Y. King, C. Lin, P. Khalili, K. Wang, M. Chang, H. Yang, IEEE International Solid-State Circuits Conf., 2016, p. 84.
[32] H. Li, T. F. Wu, A. Rahimi, K. Li, M. Rusch, C. Lin, J. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, W. Chiu, M. Chen, T. Wu, J. Shieh, W. Yeh, J. M. Rabaey, S. Mitra, H. P. Wong, IEEE International Electron Devices Meeting, 2016, p. 16.1.1.
