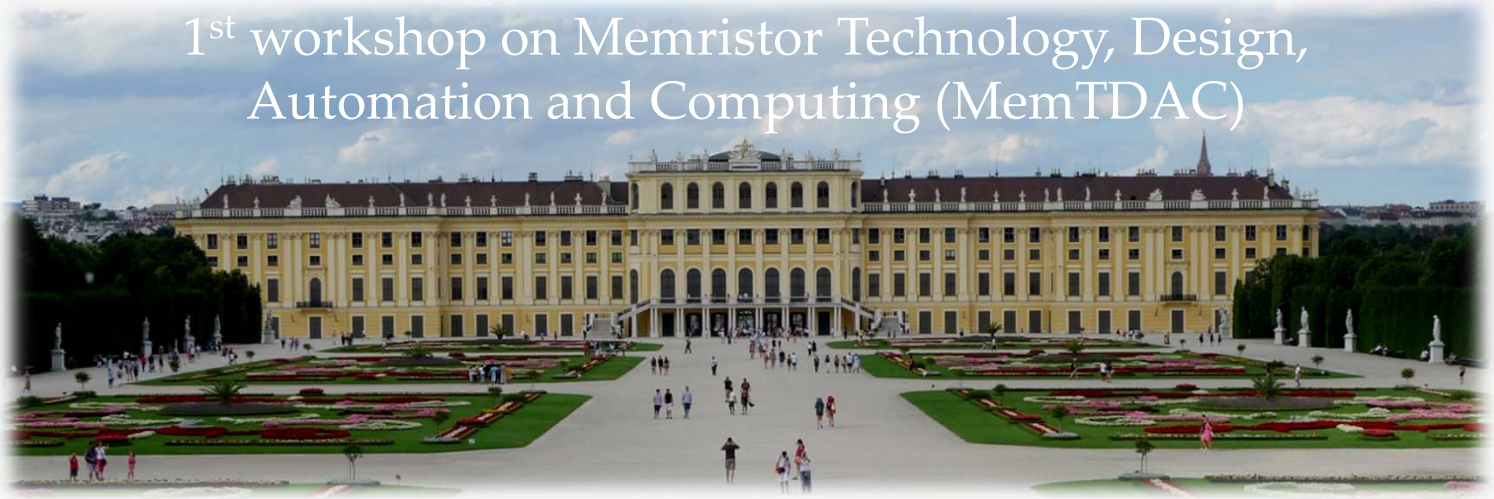
Incorporating Memristors in Currently Established Logic Circuit Architectures: *Design Considerations and Challenges*

Ioannis Vourkas, and Georgios Ch. Sirakoulis



January 22 2014, Vienna, Austria

Democritus University of Thrace
School of Engineering
Department of Electrical & Computer Engineering
Section of Electronics & Information Systems' Technology
Laboratory of Electronics

BODOSSAKI FOUNDATION

HiPEAC COMPILATION ARCHITECTURE

Presentation Outline

1. Memristor – The 4th Element

- Memory Resistors *in brief*
- Characteristic Fingerprints

2. Memristive Logic Circuit Design

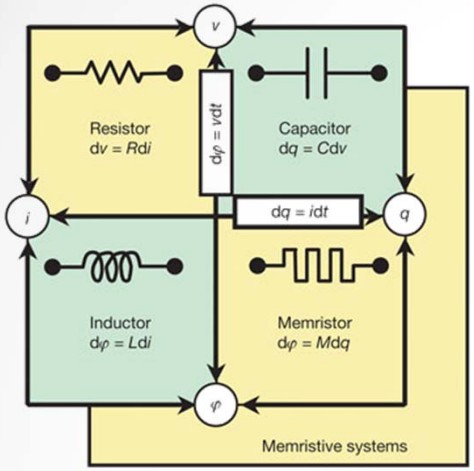
- Exploration of Memristive Dynamics through a Threshold-based Modeling Approach
- Study of Composite Memristive Devices
- Complementary Logic Design Concept for Memristive Circuits

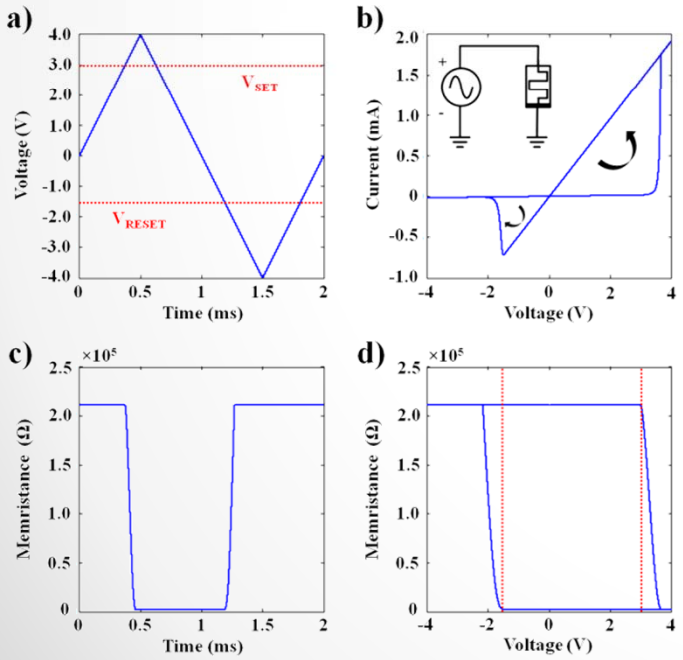
3. Applications of a Novel Circuit Design Paradigm

- Fundamental Properties and Important Characteristics
- Verification through SPICE Level Simulation of Combinational Circuits

4. Conclusions & Remarks

The Memristor

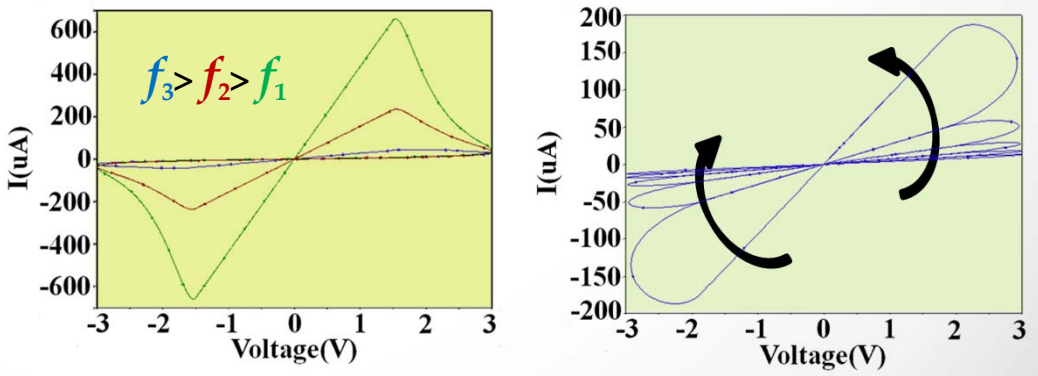




Memory resistors - MEMRISTORS

- Originally postulated by Leon Chua in 1971
- The discovery of the first "modern" memristor by HP Labs in 2008 has drawn great attention of both academia and industry
- To date, memristor represents a technology breakthrough which creates new opportunities for realization of innovative circuits

Characteristic Fingerprints …



Exploration of Memristive Dynamics through *Threshold*-based Modeling

Equations of the Model

$$I(t) = G(L)V_M(t) \quad (1)$$

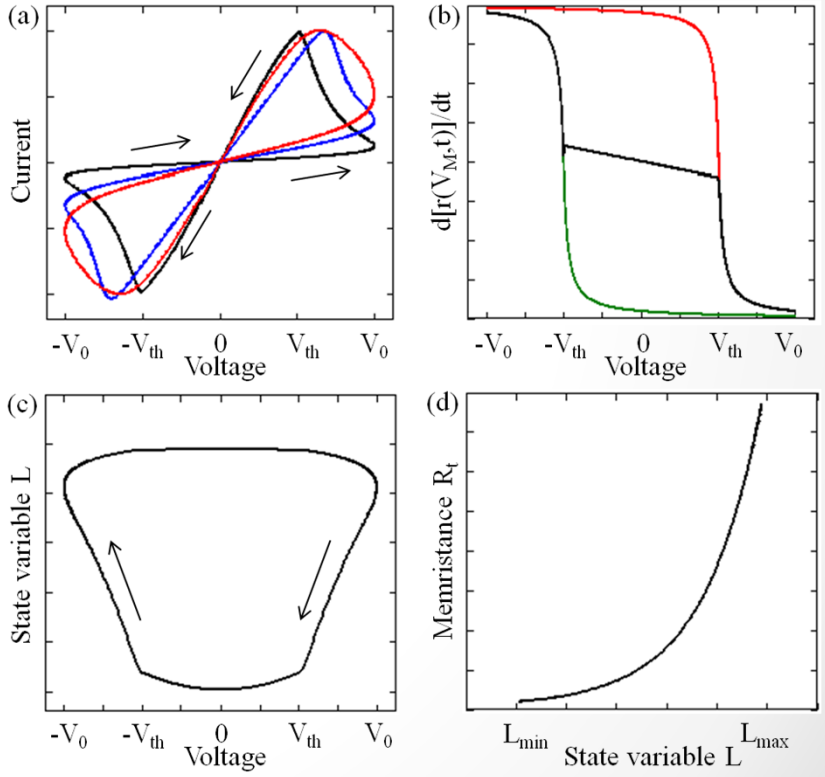
$$\dot{L} = f(V_M) \quad (2)$$

$$Rt(L_{V_M}) = f_0 \cdot \frac{e^{2L_{V_M}}}{L_{V_M}} \quad (3)$$

$$L(V_M) = L_0 \cdot \left(1 - \frac{m}{r(V_M)}\right) \quad (4)$$

$$\dot{r}(V_M) = \begin{cases} a \cdot \dfrac{V_M + V_{th}}{c + |V_M + V_{th}|}, V_M \in [-V_0, V_{RESET}) \\ b \cdot V_M \quad , V_M \in [V_{RESET}, V_{SET}] \\ a \cdot \dfrac{V_M - V_{th}}{c + |V_M - V_{th}|} \quad , V_M \in (V_{SET}, +V_0] \end{cases} \quad (5)$$

Model Response

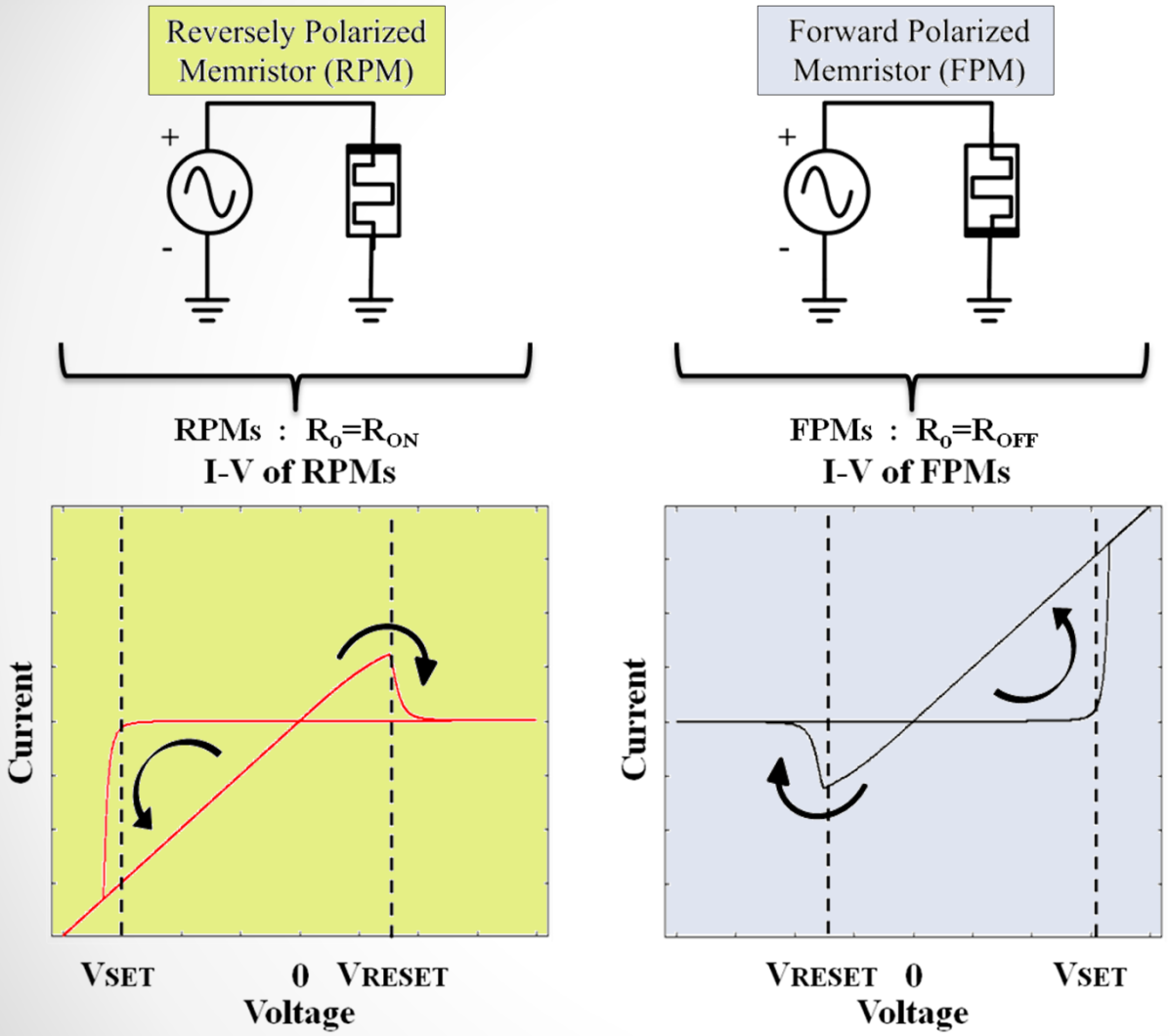


I. Vourkas, and G. Ch. Sirakoulis, "A Novel Design and Modeling Paradigm for Memristor-based Crossbar Circuits," *IEEE Trans. Nanotechnol.*, vol. 11, no. 6, pp. 1151-1159, Nov. 2012

Memristive Logic Circuit Design

- Among many emergent applications that are discussed in the literature, implementation of logic circuits is gaining considerable attention
  - Some basic concepts that allow for logic operations inside passive crossbar arrays have been identified in the recent literature. Among them:
    - … Some rely on the use of programmable memristive interconnects
    - … Others involve using passive crossbar memories as look-up tables
    - … Maybe the most recognized concept so far is a sequential logic concept, the stateful logic
  - Stateful logic is an unconventional computation framework where Boolean functions are computed using material implication and reset operations
  - The key to perform an IMPLY operation with memristors is to understand the conditional toggling property likewise in the operation of resistor–transistor logic
- However, …
  - The relatively lengthy computational sequences involved in memristive implication logic are a major disadvantage
  - No standard design methodology exists up to now
  - It is not immediately clear what kind of computing architectures would in practice benefit the most from the logic computing capability of memristors

Memristors with Opposite Polarities

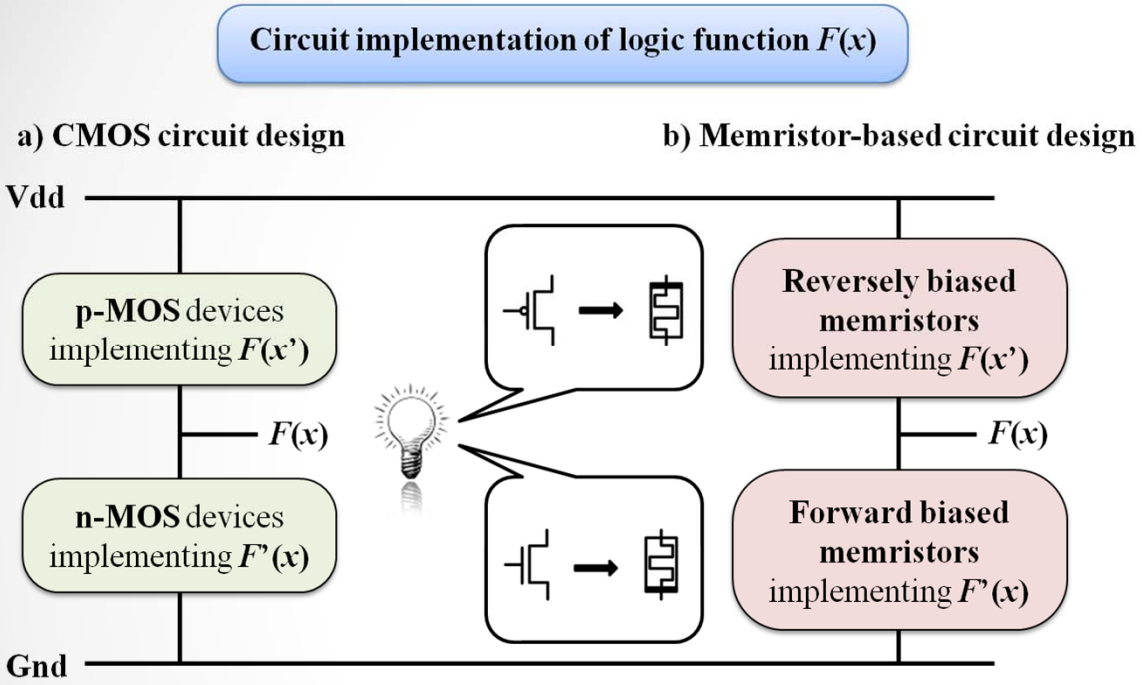


Memristors with opposite polarities present reversed behavior (**flipped *I–V* characteristic**)

The *threshold*-based response of memristors resembles that of two-state switches, "allowing" (i.e. being more conductive) or "preventing" (i.e. being less conductive) the current flow in a circuit branch

Simultaneously biased memristors when arranged in a **complementary** manner will be likely changing their states in a **reciprocal way**

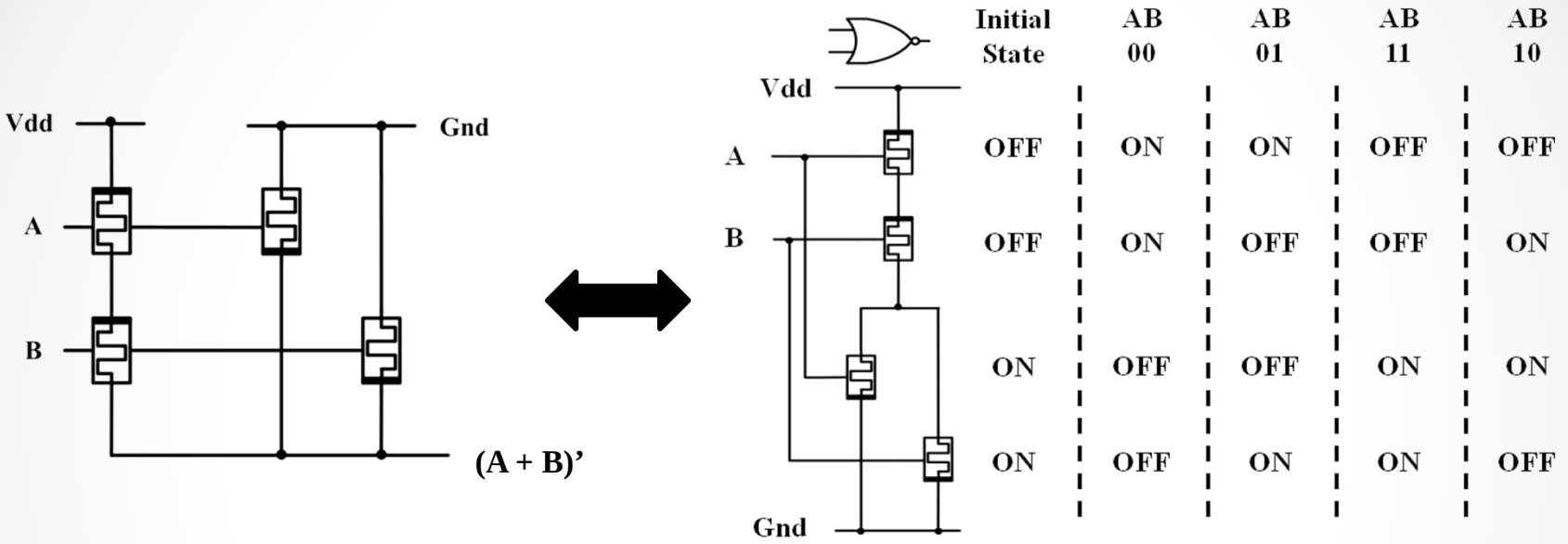
Memristors Enable Complementary Logic



Step 1:	Define inputs and outputs and assign a logic variable to each of them.
Step 2:	Find its complement, e.g. $F'(x)$, and the expression $F(x')$
Step 3a:	Find the circuit implementing : **1.** $F'(x)$ with n-FETs **2.** $F(x')$ with p-FETs
Step 3b:	Find the circuit implementing : **1.** $F'(x)$ with FPMs **2.** $F(x')$ with RPMs
Step 4:	Design the final complementary logic circuit using either FETs or memristors

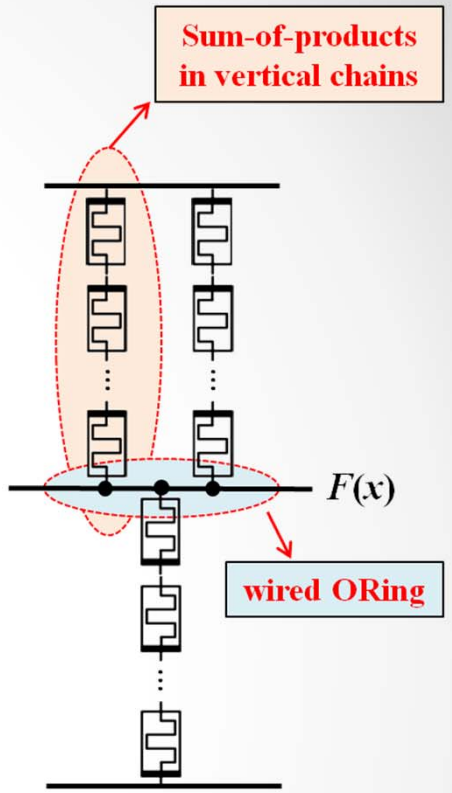
- Transition from CMOS logic to memristive logic can be made in a very coherent way using appropriately polarized memristors instead of FETs, while maintaining the same design methodology
- However, modifications are required to permit exploitation of memristive properties and overcome device limitations…

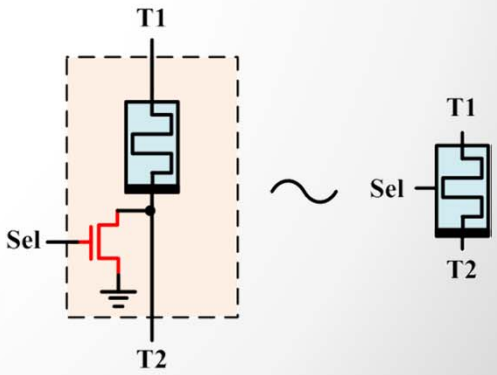
CMOS-*like* Memristive Logic Circuit Operation



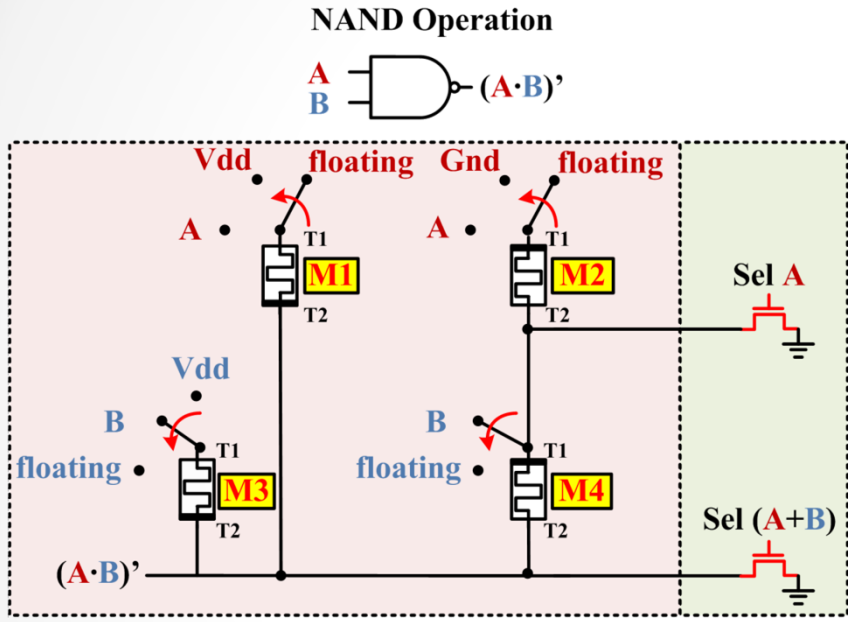
- Input signal correspondence: $-V_o \leftrightarrow$ '0' and $+V_o \leftrightarrow$ '1'
  - The applied input voltages cause reciprocal changing of states in FPMs and RPMs
- Circuit design limitations…
  - Input signals should be applied in a sequential manner
  - memristors have only two terminals compared with the three-terminal FETs
  - memristors receiving the same signal should be appropriately isolated

- Memristors are arranged in array-*like* configuration
  - Devices receiving the same input signals are found in same horizontal lines
- For every logic function, the final circuit consists of an equivalent ohmic resistance for the upper and another one for the lower part (voltage divider)
  - the output voltage is always a fraction of the supply voltage V_{dd}
    - values close to V_{dd} correspond to '1'
    - values close to zero (Gnd) correspond to '0'
  - Output logical state is represented as a voltage, even though resistance is normally used to represent the logical state of individual memristors
  - For sum-of-products function representations, each product term is implemented with a single vertical chain of memristors and the final sum is created by wired-ORing the existing products

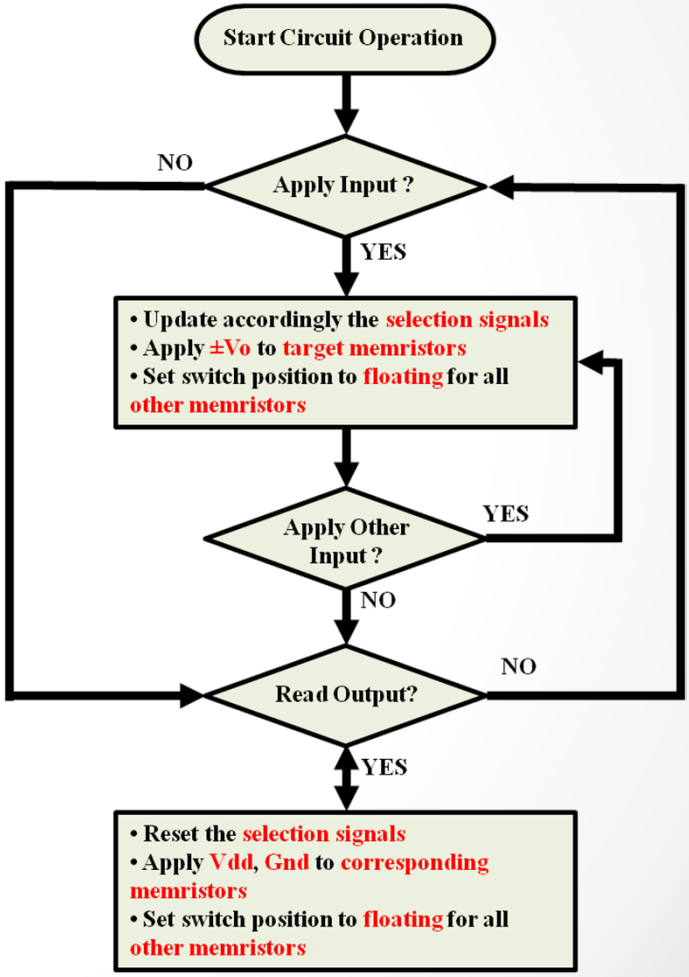


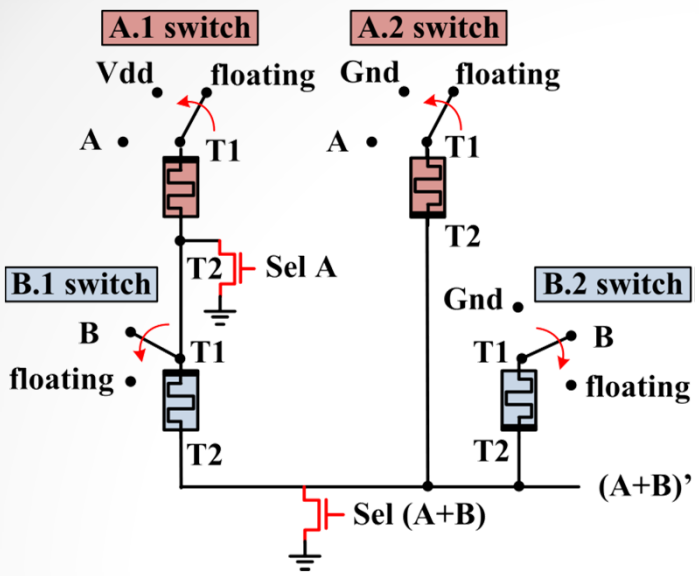


Hybrid memristor–transistor nano/CMOS architecture



	Reading Phase	Apply input A = '0'	Apply input A = '1'	Apply input B = '0'	Apply input B = '1'
Sel A	'0'	'1'	'1'	'0'	'0'
Sel (A+B)	'0'	'1'	'1'	'1'	'1'
Switch M1	Vdd	A = -Vo	A = Vo	floating	floating
Switch M2	Gnd	A = -Vo	A = Vo	floating	floating
Switch M3	Vdd	floating	floating	B = -Vo	B = Vo
Switch M4	floating	floating	floating	B = -Vo	B = Vo





- Auxiliary FETs (CMOS plane) are driven by appropriate selection lines:
  - remove the current sneak-paths
  - facilitate correct access operation to multiple memristors
- The applied voltage sequence ensures no main leakage paths in such memristor-based circuits

Accessing Memristors A.1 & A.2

Switch	Position	Memristor Terminals	
		T1	T2
A.1	A	A	Gnd
A.2	A	A	Gnd
B.1	floating	Gnd	Gnd
B.2	floating	floating	Gnd

Accessing Memristors B.1 & B.2

Switch	Position	Memristor Terminals	
		T1	T2
A.1	floating	floating	B
A.2	floating	floating	Gnd
B.1	B	B	Gnd
B.2	B	B	Gnd

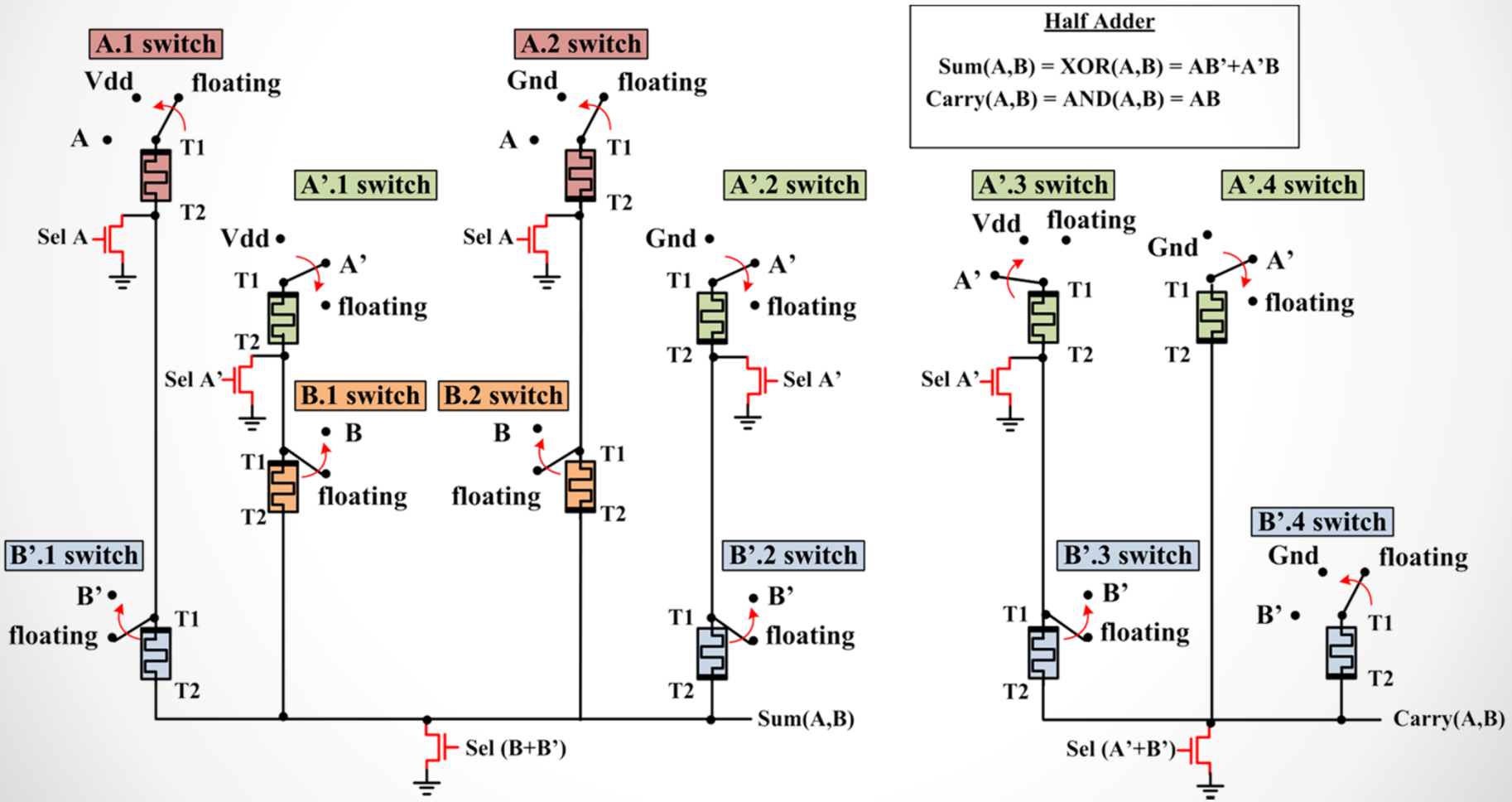
Reading Circuit Output : (A + B)'

Switch	Position	Memristor Terminals	
		T1	T2
A.1	Vdd	Vdd	B.1 – T1
A.2	Gnd	Gnd	(A + B)'
B.1	floating	A.1 – T2	(A + B)'
B.2	Gnd	Gnd	(A + B)'

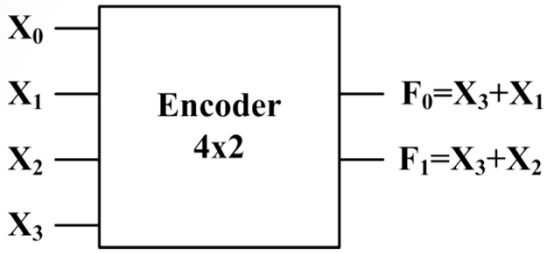
Fundamental properties and pulsing characteristics

- Switches determine the logic gate function
  - application of input (i.e.programming) signals which affect the internal state of memristors
  - reading the circuit output with V_{DD} and Gnd voltages
  - idle operation, where all switches are set to floating position
- Duration and amplitude of input pulses should be selected according to the switching response of individual memristors, so as to exceed V_{SET} and V_{RESET}
- V_{DD} amplitude should be appropriately selected so that the corresponding voltage drop on any invoked memristor never exceeds V_{SET} or V_{RESET}
  - Internal state of memristors remains unaffected during read-out regardless of the current flow from VDD to Gnd.

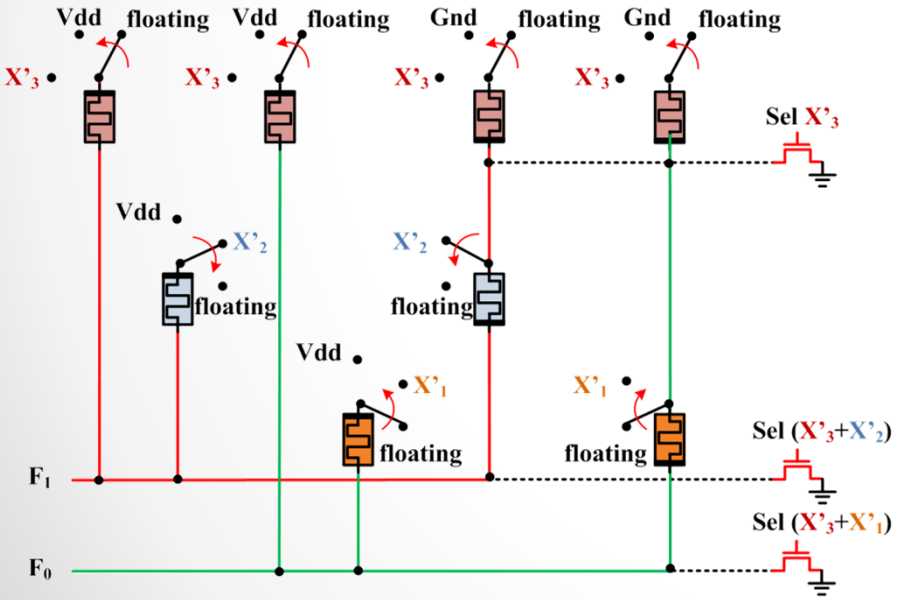
Application of the CMOS-*like* design paradigm in combinational circuits

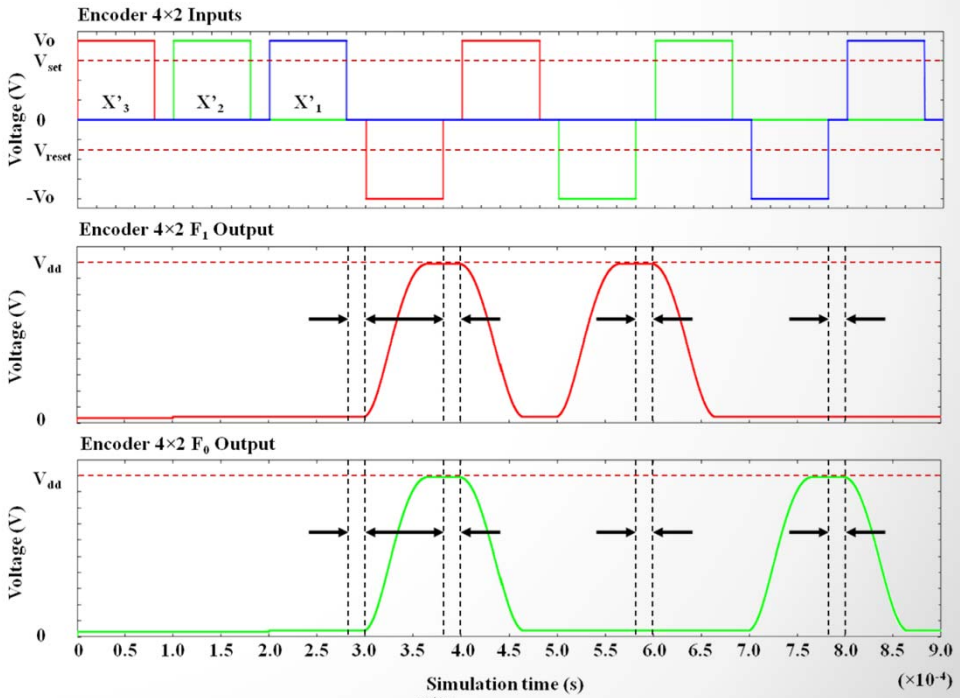


Application of the CMOS-*like* design paradigm in combinational circuits (2)



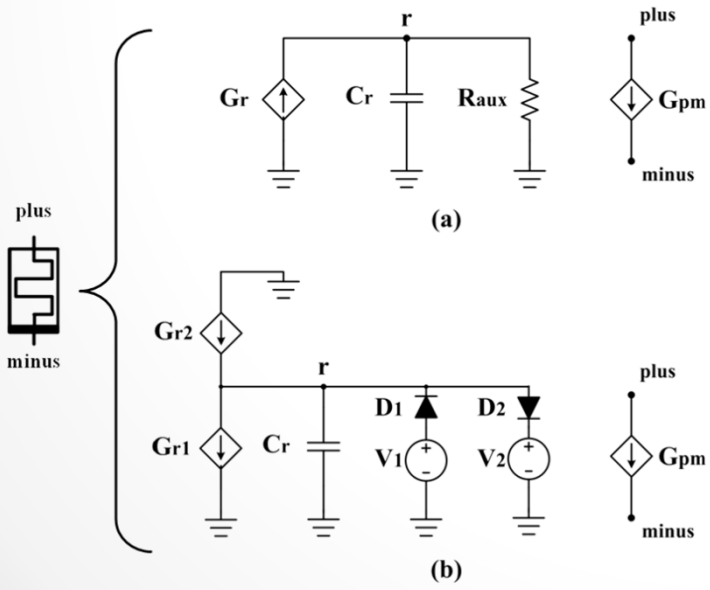
X_0	X_1	X_2	X_3	F_1	F_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

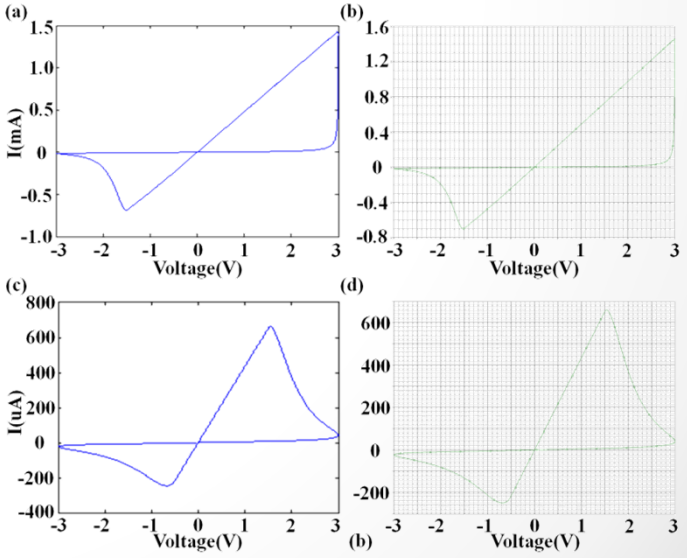




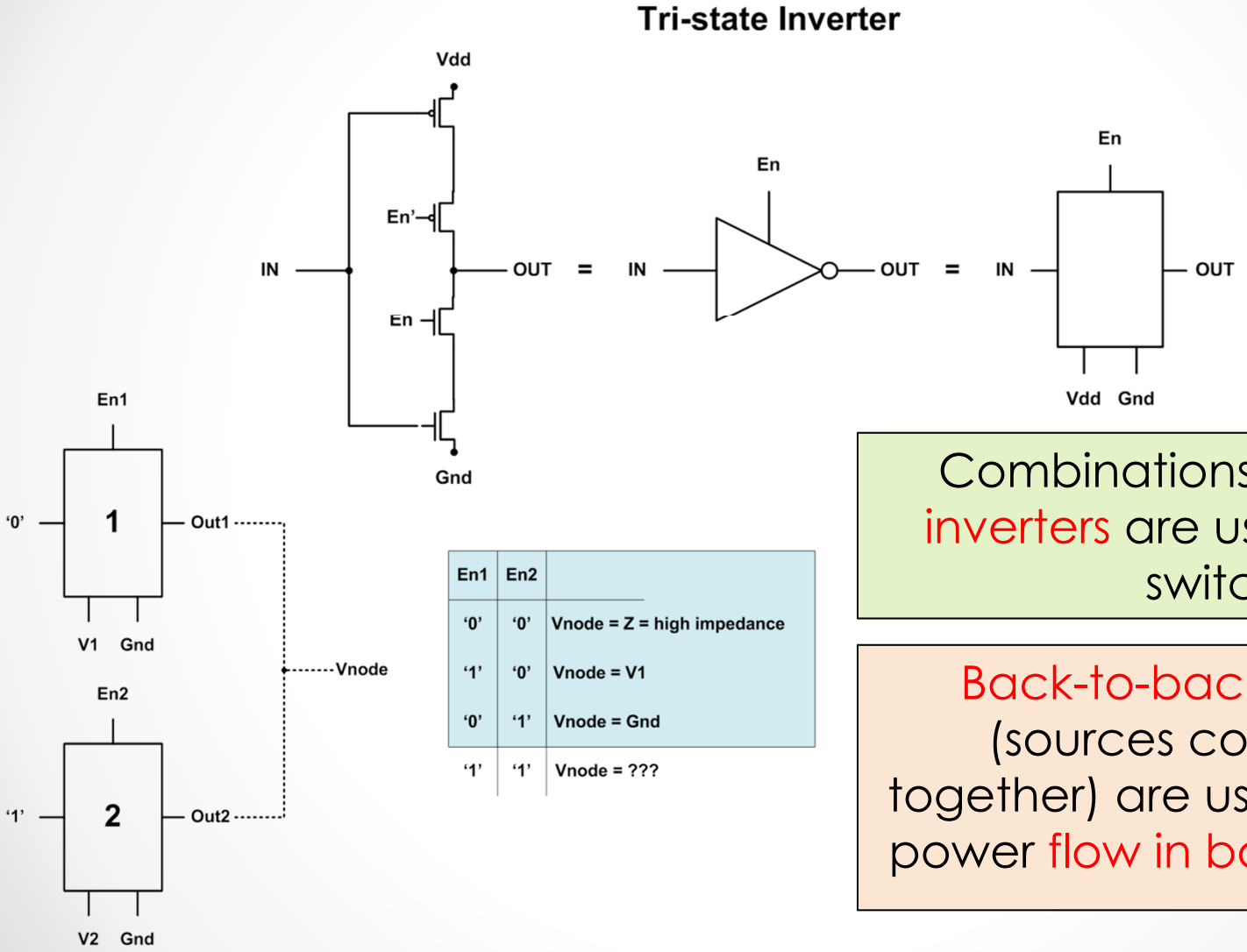
SPICE modeling of memristive behavior

- Using SPICE is common practice in device level simulations and helps in the development of new circuit architectures
- We developed a SPICE implementation of our *threshold*-type modeling solution of a voltage-controlled memristive device
  - the SPICE model complies adequately with the expected fingerprints of all memristors and memristive devices as have been defined in literature
  - simulation results are found in very good qualitative and quantitative agreement with the theoretical model





The necessary driving circuitry



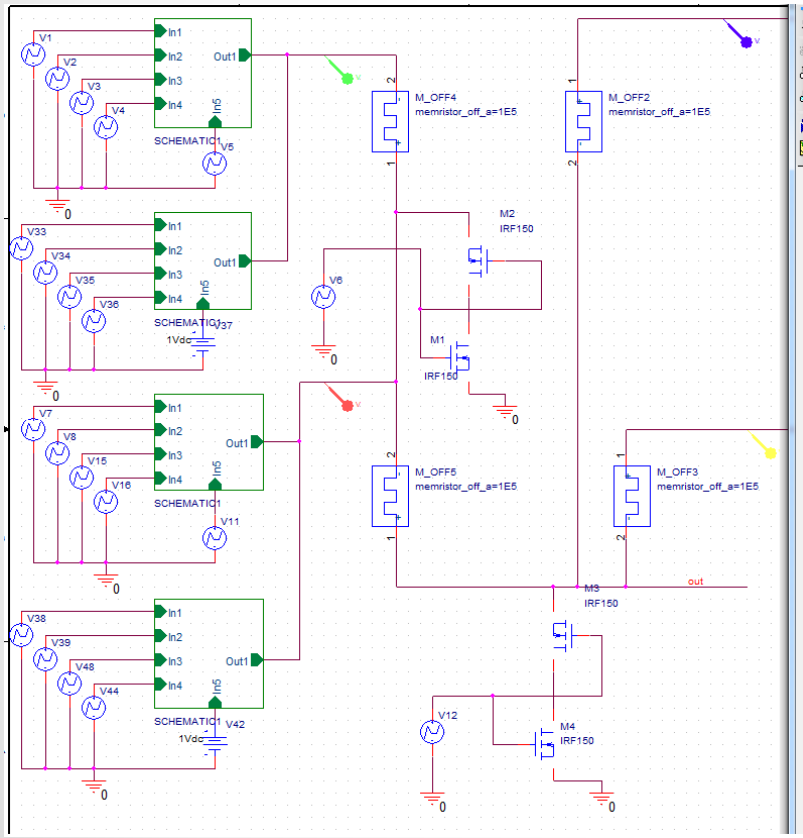
En1	En2	
'0'	'0'	Vnode = Z = high impedance
'1'	'0'	Vnode = V1
'0'	'1'	Vnode = Gnd
'1'	'1'	Vnode = ???

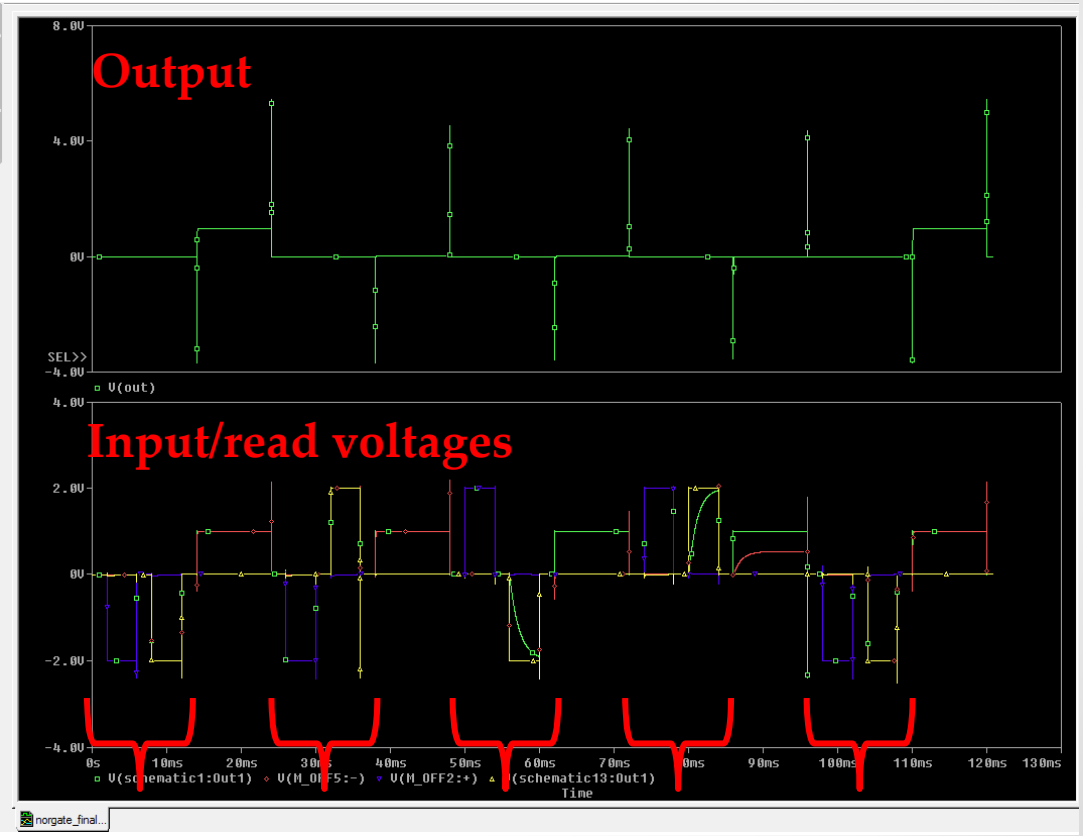
Combinations of tri-state inverters are used in each switch

Back-to-back MOSFETs (sources connected together) are used to control power flow in both directions

SPICE level simulations of CMOS-*like* memristive circuits

Simulation of a memristive NOR logic gate





′0′ ′0′ ′0′ ′1′ ′1′ ′0′ ′1′ ′1′ ′0′ ′0′

Concluding Remarks

- Incorporation of memristors in currently established logic circuit architectures could be accomplished by following the already known logic circuit design principles from the CMOS VLSI technology
- The discussed logic design methodology meets several desirable features for the creation of logic computation
  - It employs a scalable array-based geometry
  - Computation is based on the emergence of collective dynamics by the use of pairs of anti-parallel memristor combinations
  - Provides insensitivity to unavoidable variations of computing components as long as the ratio R_{OFF}/R_{ON} is sufficiently large
  - Does not call for parallel processing of input signals during operation
- Compared to stateful logic, the CMOS-*like* approach:
  - is also sequential but achieves a significant reduction in the computation steps
  - necessary computation steps reach a maximum of $2n$ for an n-input arbitrary Boolean function
  - Significantly simplifies the circuit design procedure

End of presentation

Thank you!

• • •

Questions ???

Georgios Ch. Sirakoulis
Associate Professor
Democritus University of Thrace
Dept. of Electrical and Computer Engineering
http://gsirak.ee.duth.gr