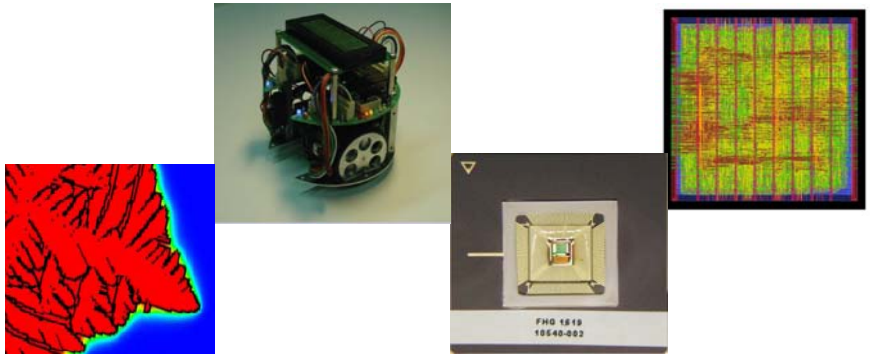
Using the Multi-bit Feature of Memristors for Register Files in Signed-Digit Arithmetic Units



Dietmar Fey
Department Computer Science –
Chair for Computer Architecture
Friedrich-Alexander-Universität Erlangen-Nürnberg

Outline

- Multi-bit feature of memristors
- Signed-digit (SD) arithmetic
- Memristor-based SD arithmetic unit
- Conclusion

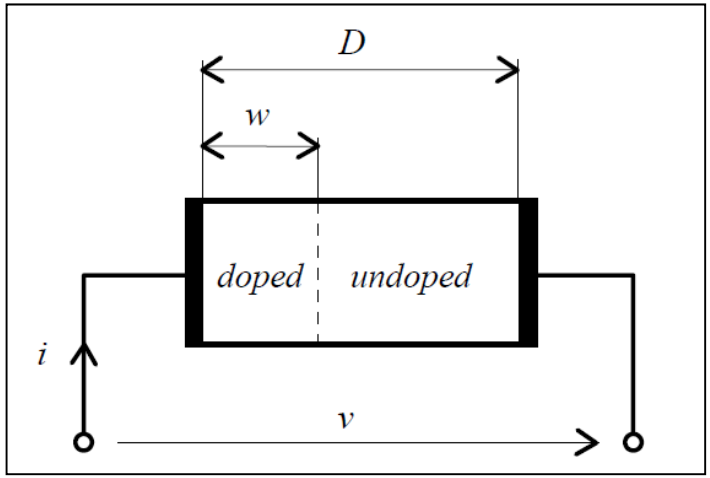
Multi-bit feature of memristors

- Promising features of memristors
  - High-dense and fast memory element
  - CMOS compatible
  - Configurable logic applications
  - Neuropmorphic architectures
- Focus here in this work
  - Feature to store multiple values in a single cell

Multi-bit feature of memristors

- Promising feature of memristors

Yalcin Yilmaz and Pinaki Mazumder
Threshold Read Method for Multi-bit Memristive Crossbar Memory
Proc. 2011 International Symposium on Electronic System Design, IEE CS, pp. 217-222, 2011.



$$R_{MEM}(x) = R_{ON}x + R_{OFF}(1-x),$$

$$\text{where } x = \frac{w}{D} \in (0,1)$$

$$\frac{dw(t)}{dt} = \mu_v \frac{R_{ON}}{D} i(t)$$

Multi-bit feature of memristors

- Modelling multi-bit feature

  - Used in a SPICE simulation

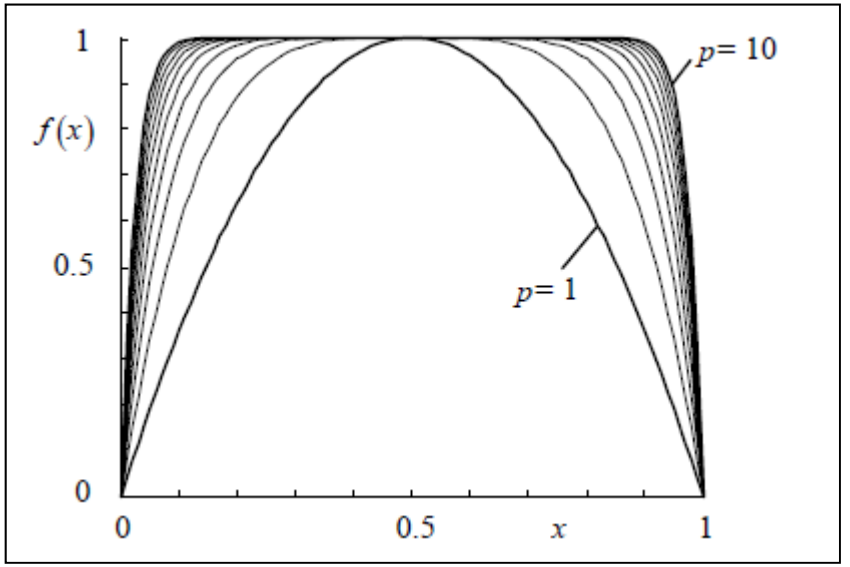
Zdeněk BIOLEK, Dalibor BIOLEK Viera BIOLKOVÁ
SPICE Model of Memristor with Nonlinear Dopant Drift
RADIOENGINEERING, VOL. 18, NO. 2, JUNE 2009

$$\frac{dx}{dt} = k\, i(t) f(x), \quad k = \frac{\mu_v R_{ON}}{D^2}$$

$$v(t) = R_{MEM}(w)\, i(t).$$

Used window function

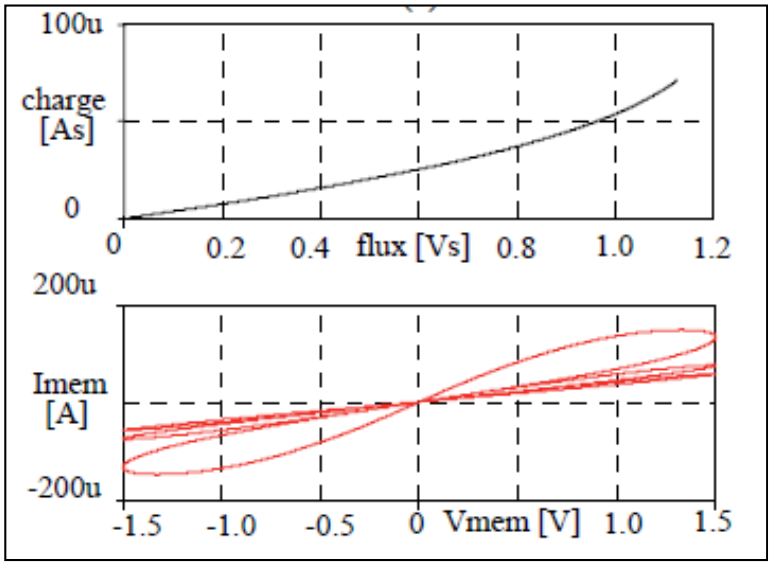
$$f(x) = 1 - (2x-1)^{2p}$$

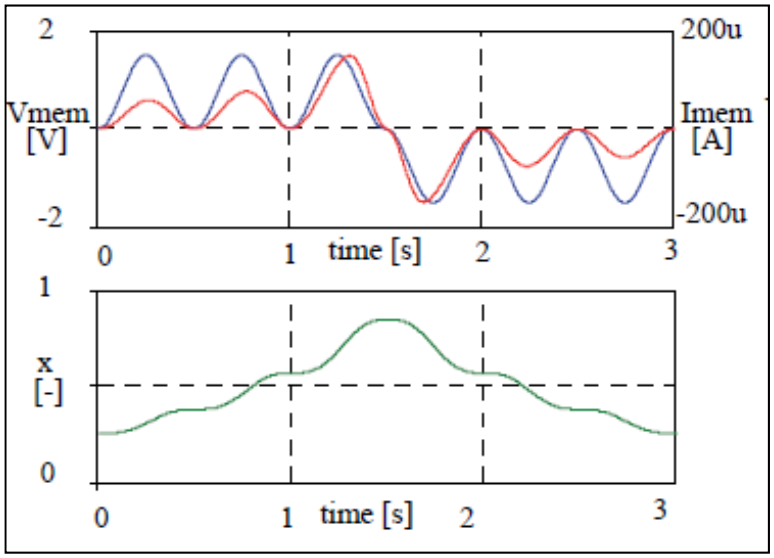


  - Using a model for a non-linear dopant drift (window function)

Multi-bit feature of memristors

- Promising feature of memristors
  - Multi-stable states for x and R_{MEM}

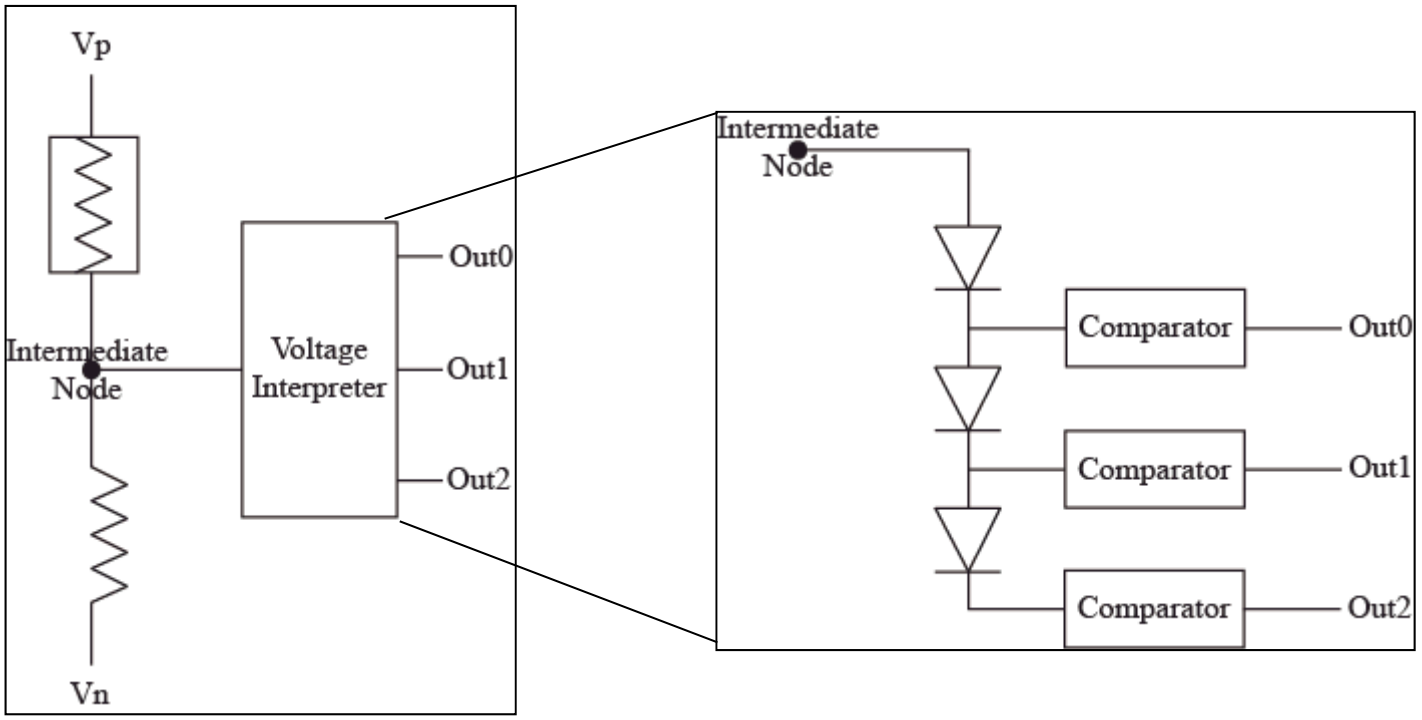




Multi-bit feature of memristors

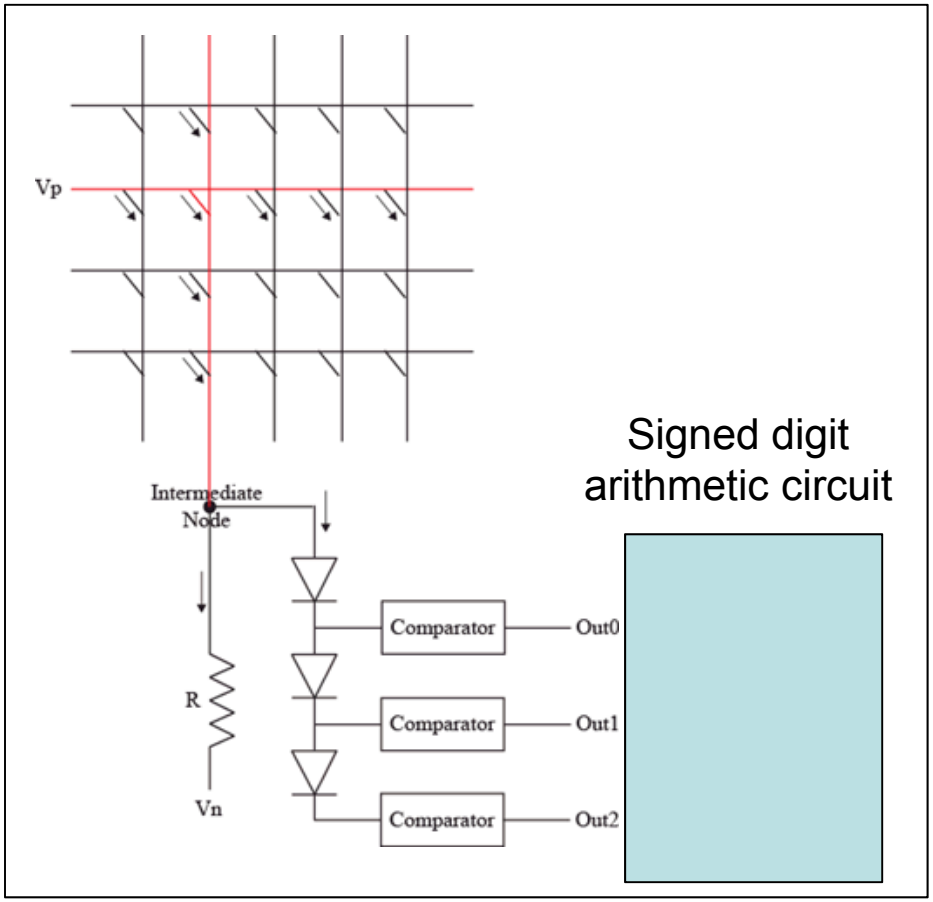
- Promising memristor features
  - Depending on the stored resistance value one of the outputs Out0, Out1 or Out2 is becoming HIGH

Yalcin Yilmaz and Pinaki Mazumder
Threshold Read Method for Multi-bit Memristive Crossbar Memory
Proc. 2011 International Symposium on Electronic System Design, IEE CS, pp. 217-222, 2011.



Multi-bit feature of memristors

- Coupling with crossbar memristor memory and SD arithmetic



Signed-digit (SD) arithmetic

- Signed-digit number representation to base 2

$$w(a) = \sum_{i=0}^{n} a_i \cdot 2^i \qquad a = (a_{n-1},\ldots,a_0),\ a_i \in \{-1,0,1\}$$

Be $a = (a_{n-1},\ldots,a_0)$ a SD number to base 2

$$a^+ = (a^+_{n-1},\ldots,a^+_0) \quad \text{and} \quad a^+_i = 1 \Leftrightarrow a_i = 1 \wedge a^+_i = 0 \Leftrightarrow a_i \neq 1$$

$$a^- = (a^-_{n-1},\ldots,a^-_0) \quad \text{and} \quad a^-_i = 1 \Leftrightarrow a_i = \overline{1} \wedge a^-_i = 0 \Leftrightarrow a_i \neq \overline{1}$$

Signed-digit (SD) arithmetic

- Benefits of SD arithmetic

Binary addition

		1	0	1	1
+		0	1	1	1
C	0	0	1	1	–
S	–	1	1	0	0
C	0	1	0	0	–
S	–	1	0	1	0
C	1	0	0	0	–
S	0	0	0	1	0
S	1	0	0	1	0

O(*n*)
Best case: *log(n)*

Signed digit addition

		1	0	1	1
+		0	1	1	1
C	1	1	1	1	0
Z	0	$\bar{1}$	$\bar{1}$	0	0
S	1	0	0	1	0

O(*1*)

		1	0
+		0	1
C	1	1	
Z		$\bar{1}$	$\bar{1}$

Signed-digit (SD) arithmetic

- Different codings are possible
  - Storing in tri-state representation
  - Calculations shall still happen using digital processing binary numbers
  - Used a coding proposed by Muller/Duprat, 1991

a_i^+	a_i^-	a
0	0	0
0	1	$\bar{1}$
1	0	1
1	1	X

X: not defined

Signed-digit (SD) arithmetic

- Addition / subtraction of (i) a SD number *a* and a binary number *B* and (ii) two SD numbers *c* and *z*
  - (i) *a* + *B* *(subtraction later)*

a	a_{n-1}^{+}	...	a_1^{+}	a_0^{+}
	a_{n-1}^{-}	...	a_1^{-}	a_0^{-}
B	B_{n-1}	...	B_1	B_0
c	c_{n-2}^{+}	...	c_0^{+}	
z	z_{n-1}^{-}	...	z_1^{-}	z_0^{-}
s	s_{n-1}^{+}	...	s_1^{+}	s_0^{+}
	s_{n-1}^{-}	...	s_1^{-}	s_0^{-}

a_i^+	a_i^-	B_i	c_i^+	c_i^-	z_i^+	z_i^-
1	0	1	1	0	0	0
1	0	0	1	0	0	1
0	0	1	1	0	0	1
0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	1

$$c_i^+ = a_i^+ \vee \left(B_i \wedge \overline{a_i^-} \right) \qquad \wedge\text{: and} \quad \vee\text{: or} \qquad (4.3)$$

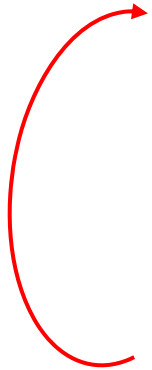
$$z_i^- = \left(a_i^+ \vee a_i^- \right) \oplus B_i \qquad \oplus\text{: exor} \qquad (4.4)$$

Signed-digit (SD) arithmetic

- (ii) Two SD numbers: *c* + *z*

a	a_{n-1}^{+}	...	a_1^{+}	a_0^{+}
	a_{n-1}^{-}	...	a_1^{-}	a_0^{-}
B	B_{n-1}	...	B_1	B_0
c	c_{n-2}^{+}	...	c_0^{+}	
z	z_{n-1}^{-}	...	z_1^{-}	z_0^{-}
s	s_{n-1}^{+}	...	s_1^{+}	s_0^{+}
	s_{n-1}^{-}	...	s_1^{-}	s_0^{-}

a_i^+	a_i^-	B_i	c_i^+	c_i^-	z_i^+	z_i^-
1	0	1	1	0	0	0
1	0	0	1	0	0	1
0	0	1	1	0	0	1
0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	1



?

Signed-digit (SD) arithmetic

- Truth table and simplified Boolean equation since only c_{i-1}^{+} and z_i^{-} have to be considered

z_i^+	z_i^-	c_{i-1}^+	c_{i-1}^-	s_i^+	s_i^-
0	0	0	0	0	0
0	0	1	0	1	0
0	0	0	1	x	x
1	0	0	0	x	x
1	0	1	0	x	x
1	0	0	1	x	x
0	1	0	0	0	1
0	1	1	0	0	0
0	1	0	1	x	x

x: don't care

$$s_i^+ = \overline{z_i^-} \wedge c_{i-1}^+ \qquad (4.5)$$

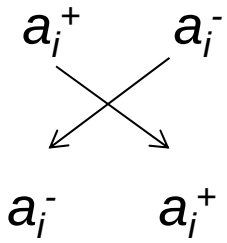
$$s_i^- = \overline{c_{i-1}^+} \wedge z_i^- \qquad (4.6)$$

Signed-digit (SD) arithmetic

- Subtraction can be simply reduced to addition

$$a - B = (-1) \cdot ((-1) \cdot a + B)$$

  - Negative *a* can be generated by exchanging positive and negative part of SD number *a*
    - We will need an exchange/bypass circuit



$$c_i^+ = a_i^- \vee \left(B_i \wedge \overline{a_i^+}\right) \qquad z_i^- = \left(a_i^- \vee a_i^+\right) \oplus B_i$$

$$s_i^+ = \overline{c_{i-1}^+} \wedge z_i^- \qquad s_i^- = \overline{z_i^-} \wedge c_{i-1}^+$$

Same equations as (4.3) and (4.4)

Only a_i^+ and a_i^- are exchanged

Signed-digit (SD) arithmetic

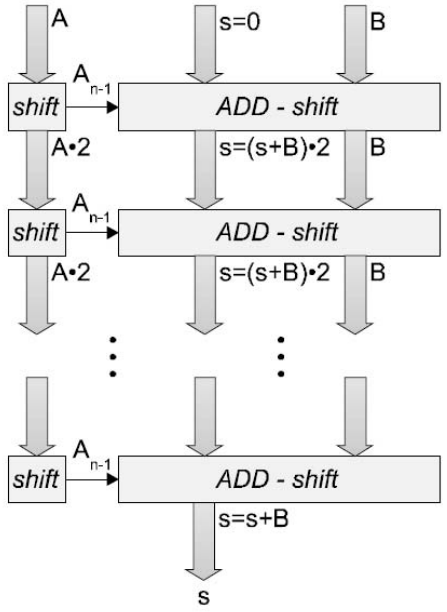
- Multiplication
  - As usual reduction to subsequent (sigend-digit) additions
  - Most significant bit of A determines if addition has to be carried out or not
  - A and s have to be shifted

$A_{n-1}\ A_{n-2}\ \dots\ A_0 \quad x \quad B_{n-1}\ B_{n-2}\ \dots\ B_0$

$A_{n-1} \cdot \quad B_{n-1}\ B_{n-2}\ \dots\ B_0 \ +$
$A_{n-2} \cdot \quad B_{n-1}\ B_{n-2}\ \dots\ B_0 \ +$
$\dots \quad \dots \quad \dots$
$A_0 \cdot \quad B_{n-1}\ B_{n-2}\ \dots\ B_0$

$p_{n-1}\ p_{n-2} \qquad\qquad p_0$

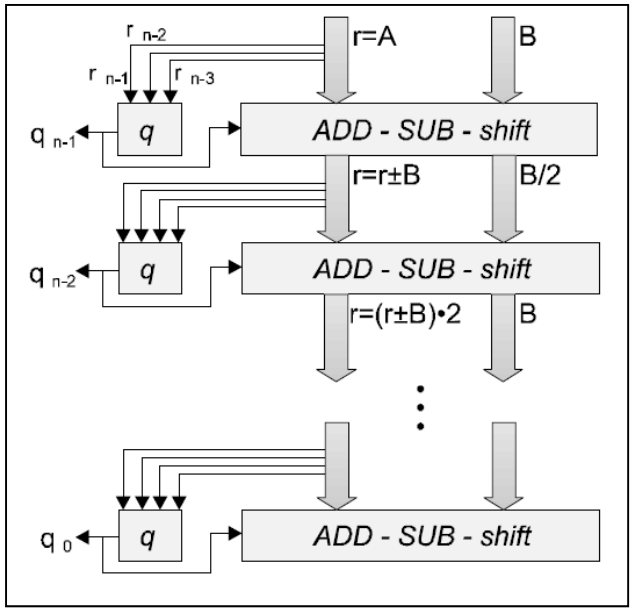
```
s = 0
for i = 1 to n
      if (A_{n-1} == 1) s = s+B
      if (A_{n-1} == 0) s = s+0
      shift left A
      shift left s
```



Signed-digit (SD) arithmetic

- (Integer) Division
  - As usual reduction to subsequent (sigend-digit) additions / subtractions
  - Most significant bit of A determines if addition has to be carried out or not
  - Quotient bits are iteratively determined

$$
\begin{aligned}
&r = A \\
&\text{for } i = 1 \text{ to } n \\
&\qquad \text{if } (r_{n-1}, r_{n-2}, r_{n-3}) < 0 \text{ then } q_{n-i} = \bar{1} \\
&\qquad \text{if } (r_{n-1}, r_{n-2}, r_{n-3}) = 0 \text{ then } q_{n-i} = 0 \\
&\qquad \text{if } (r_{n-1}, r_{n-2}, r_{n-3}) > 0 \text{ then } q_{n-i} = 1 \\
&\qquad r = 2 \cdot (r - q_{n-i} \cdot B) \\
&\qquad \text{if } (i == 1) \text{ then shift right B}
\end{aligned}
$$



Signed-digit (SD) arithmetic

- Back conversion to binary representation, e.g. two's complement
  - On-the-fly Algorithm from Ercegovac, Lang $O(n)$

$$A[n-1] = 1 \qquad B[n-1] = 0$$

$$A[k-1] = \begin{cases} 2 \cdot A[k] + 1 & if\ q_{k-1} = 1 \\ 2 \cdot A[k] & if\ q_{k-1} = 0 \\ 2 \cdot B[k] + 1 & if\ q_{k-1} = \bar{1} \end{cases} \qquad B[k-1] = \begin{cases} 2 \cdot A[k] & if\ q_{k-1} = 1 \\ 2 \cdot B[k] + 1 & if\ q_{k-1} = 0 \\ 2 \cdot B[k] & if\ q_{k-1} = \bar{1} \end{cases}$$

$$Q_k = 1\ \bar{1}\ 010$$

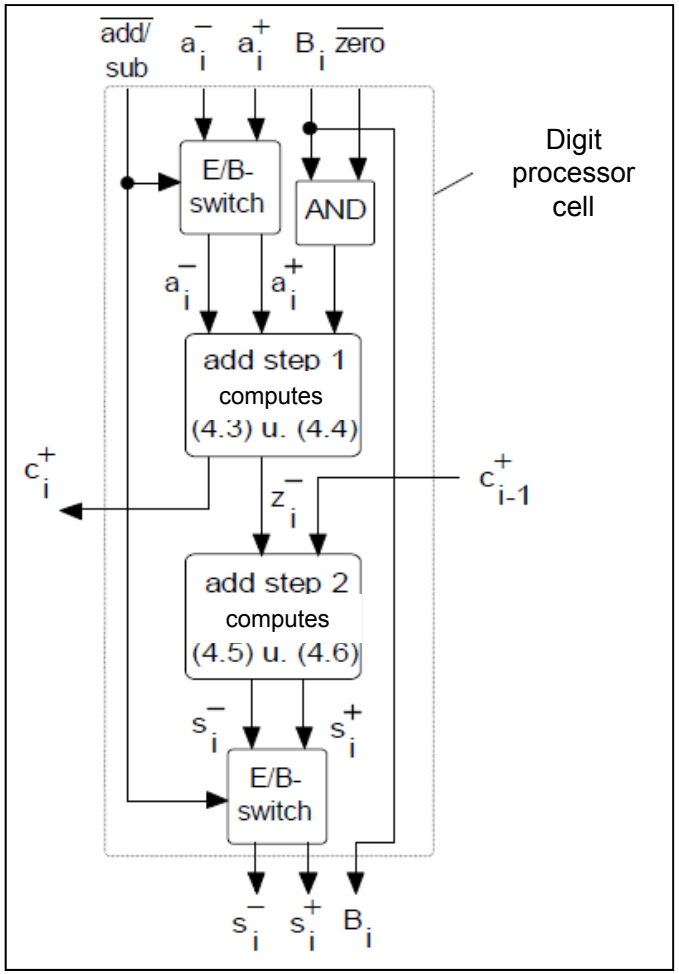
k	Q_k	A[k]	B[k]
4	1	1	0
3	$\bar{1}$	01	00
2	0	010	001
1	1	0101	0100
0	0	01010	01001

Signed-digit (SD) arithmetic

- Schematic of a digit processor cell

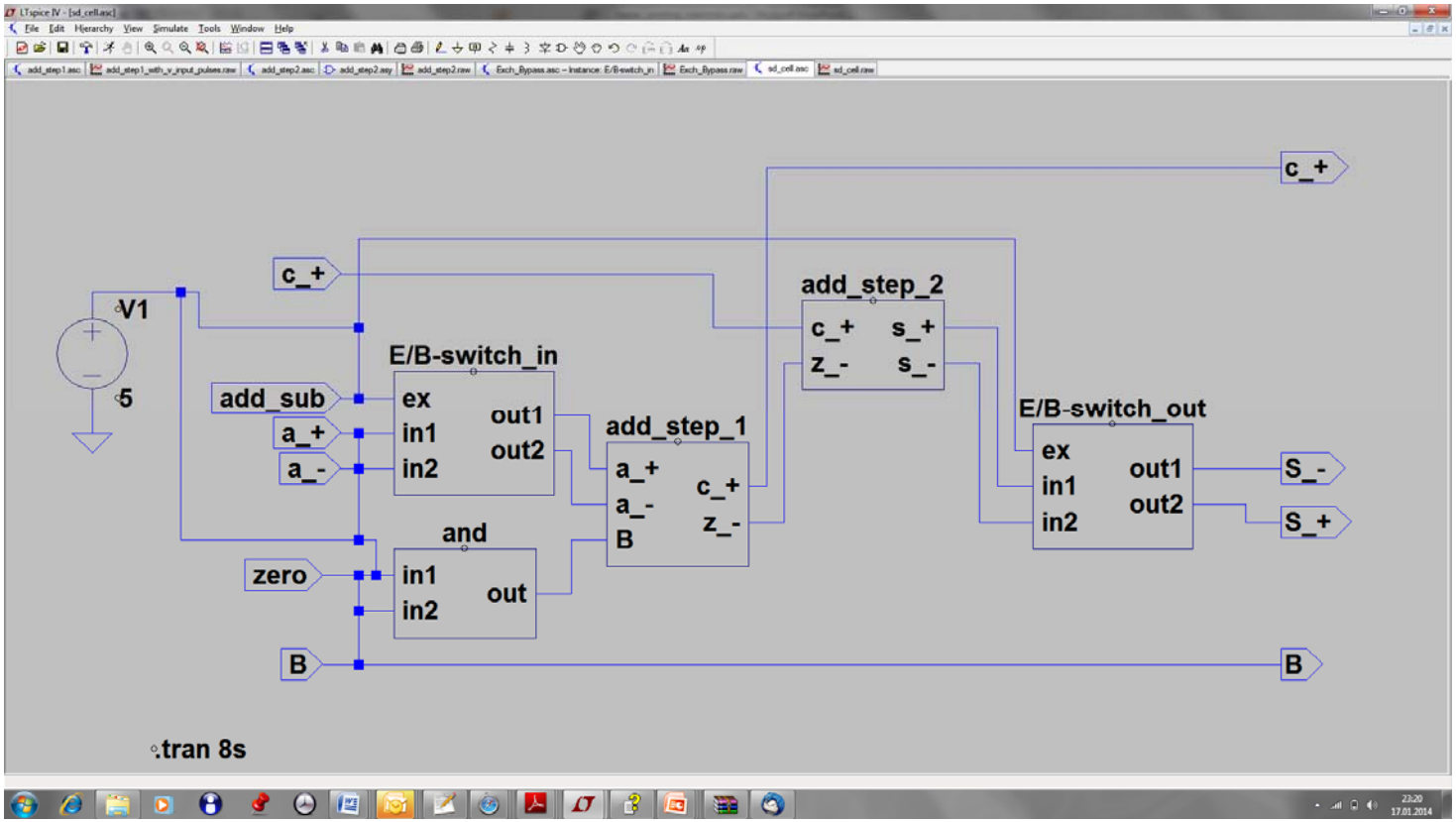
  - Several cells are connected side-by-side to a row

  - Multiple rows are arranged below each other to a pipeline



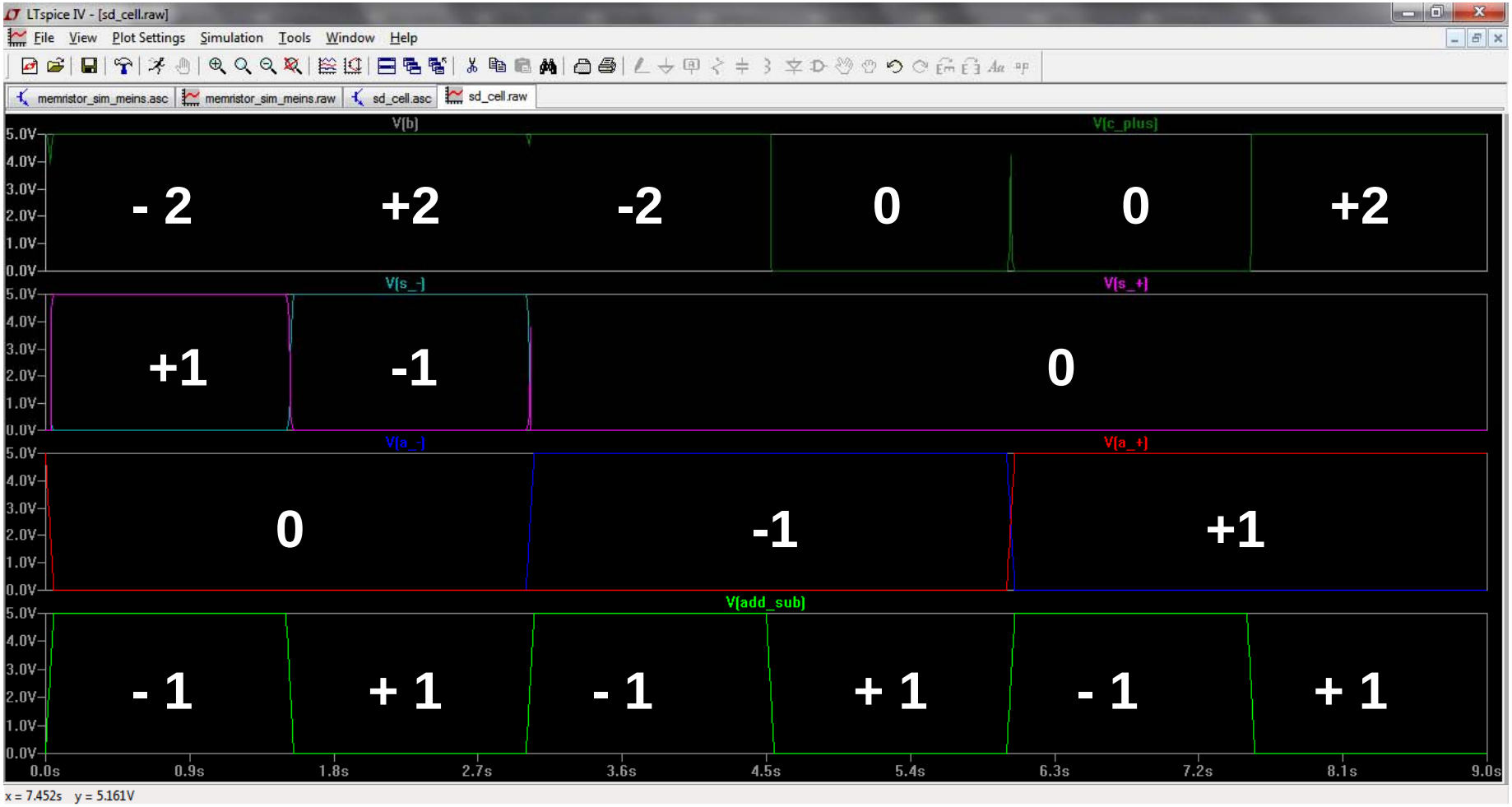
Signed-digit (SD) arithmetic

- Corresponding gate logic for an SD adder / subtractor cell
  - Completely implemented in SPICE



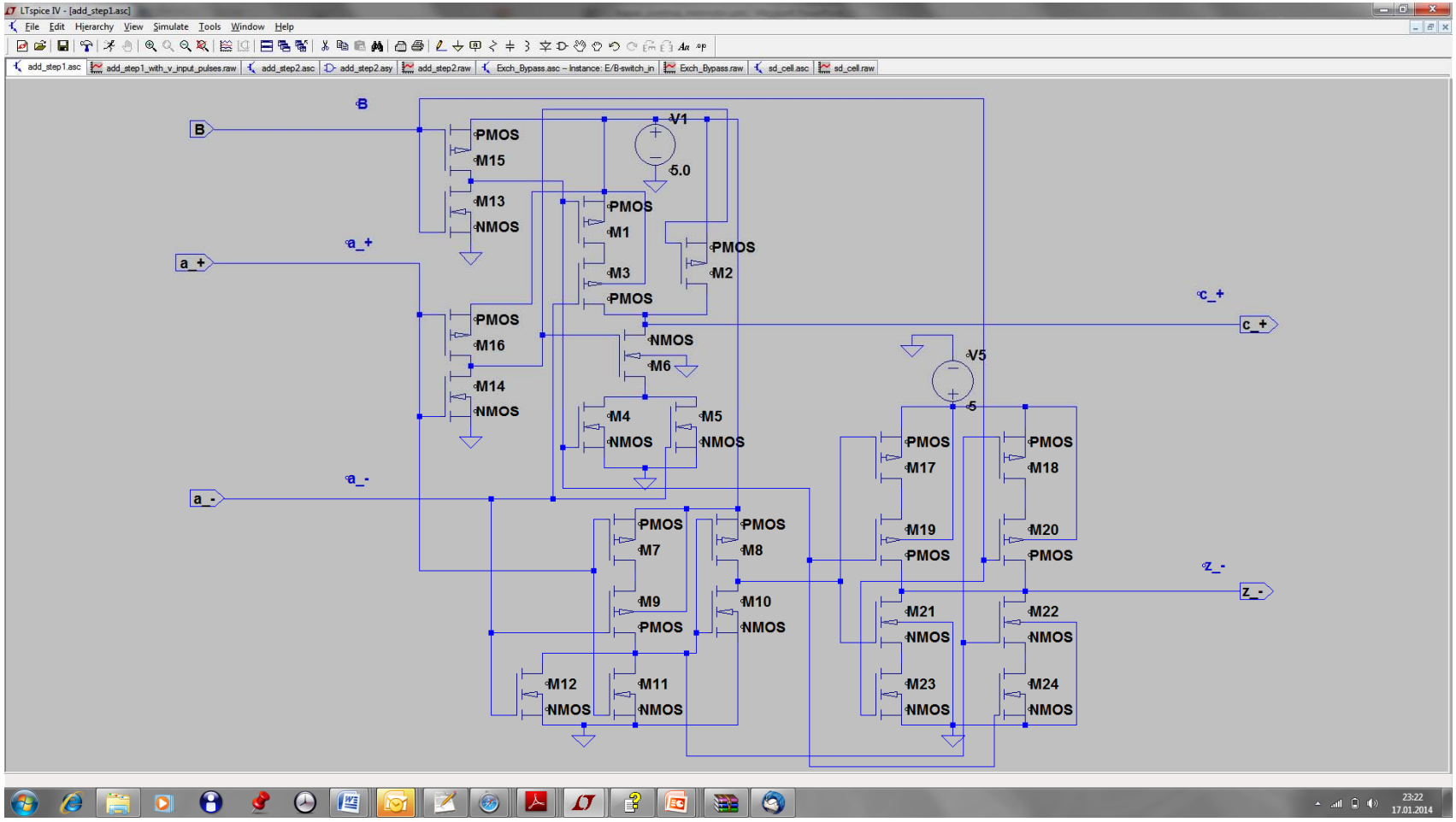
Signed-digit (SD) arithmetic

- Simulation result



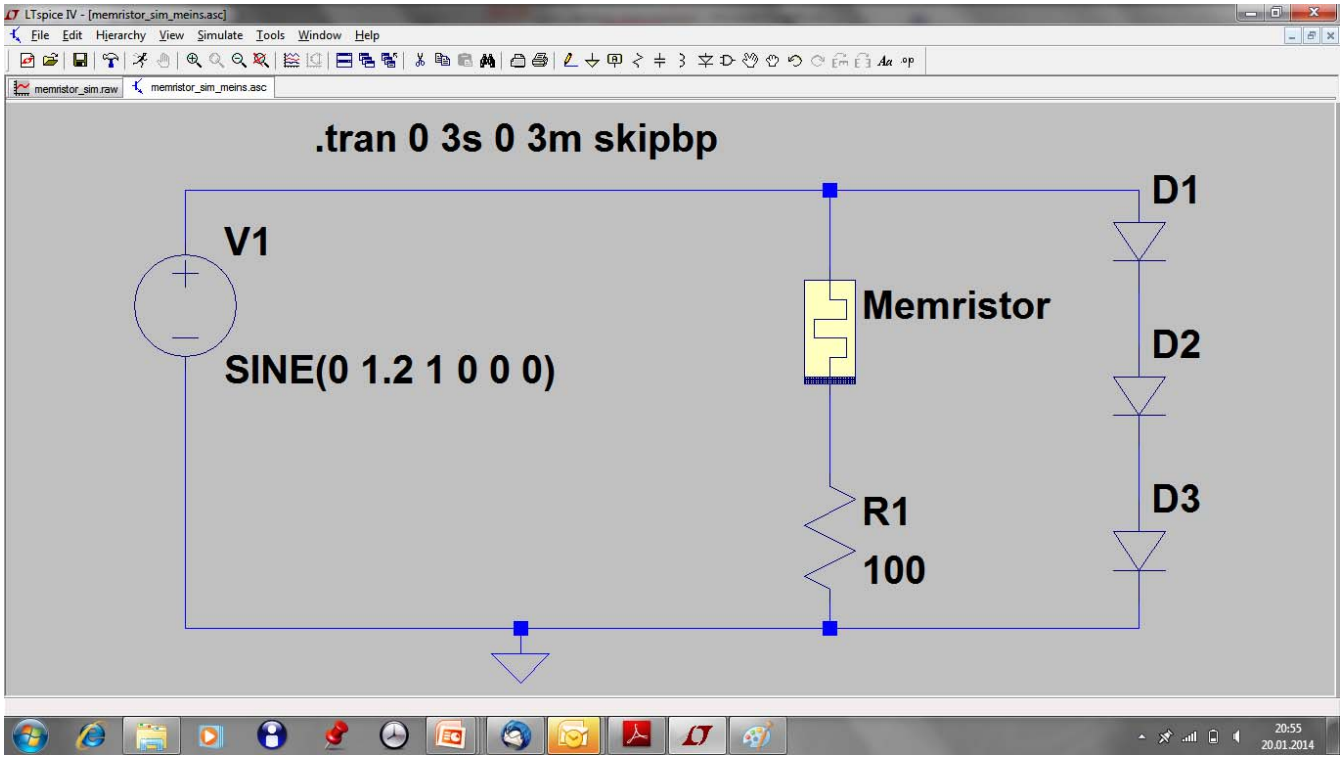
Signed-digit (SD) arithmetic

- Add_step_1 cell implementing equations (4.3) and (4.4)



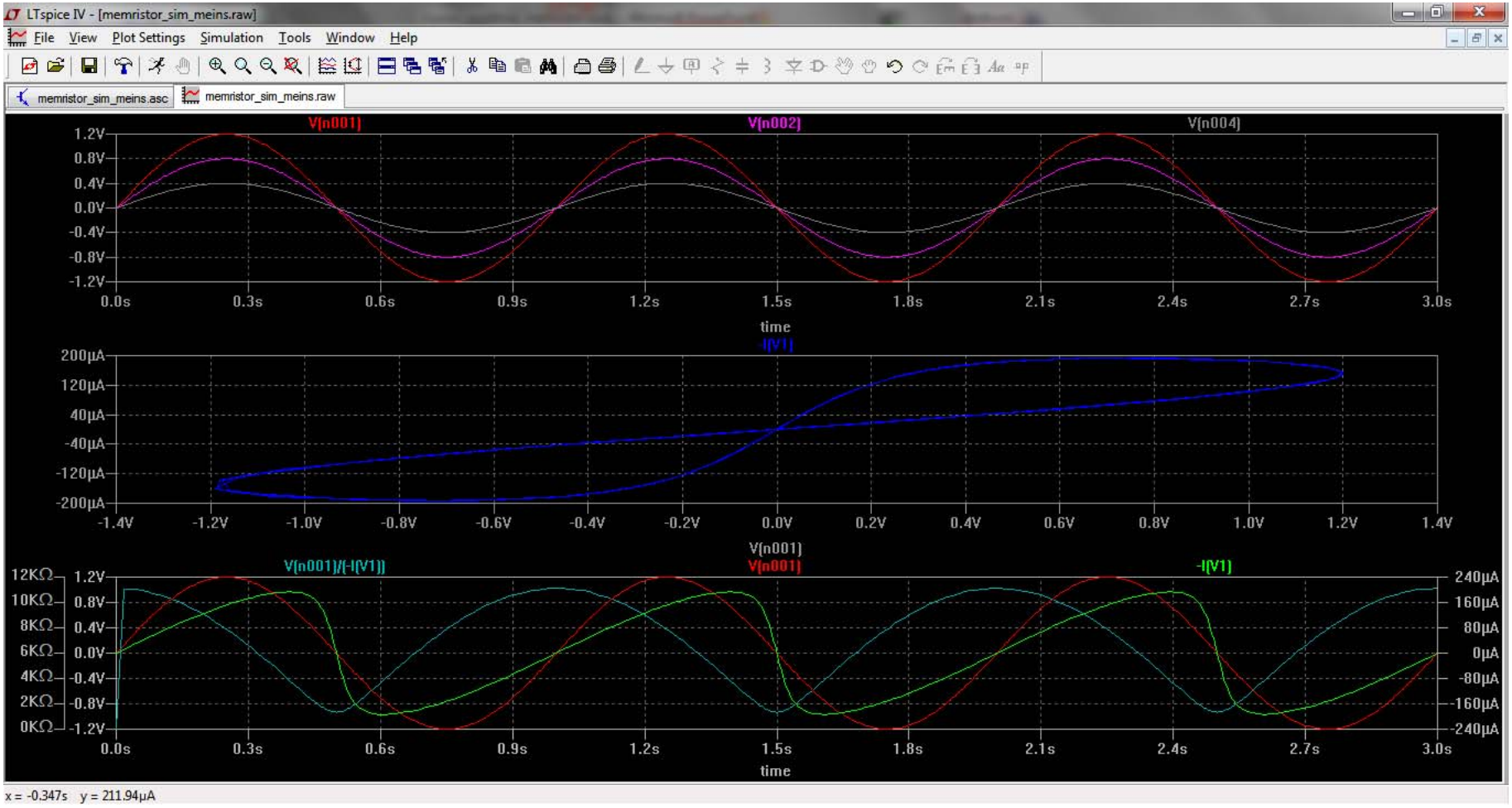
Memristor-based SD arithmetic unit

- Work-in-progress for marriage with memristor SPICE model
  - Used SPICE modelling of multi-bit memristor cell according to Yalcin Yilmaz and Pinaki Mazumder



Memristor-based SD arithmetic unit

- Simulation result



Memristor-based SD arithmetic unit

- Comparison SD adder using multi bit memristors (pipeline register) vs. conventional binary adders
  - Assuming an *n* stage linear pipeline
  - No parallel partial product matrix calculation

Used Adder	Latency	Bandwidth by pipeline	#number of storing cells
SD adder with memristors	N × O(1)	1 cycle time	$2n + n^2$
Ripple Carry Adder	N × O(N)	1 / N cycle time	$2n + n^2$
SD adder without memristors	N × O(1)	1 cycle time	$2n + 2n^2 + (n^2+n)$

for on-the-fly back conversion

Conclusion

- Proposal for using multi-bit storing feature of memristor
- Realisation of SD arithmetic unit
- Transistor netlist for corresponding SD cell available
- Interfacing to memristor netlist
- Looking for co-operation on device modelling