DESIGN, AUTOMATION & TEST IN EUROPE
27 – 31 March, 2017 · STCC · Lausanne · Switzerland
The European Event for Electronic System Design & Test

Hybrid VC-MTJ/CMOS Non-volatile Stochastic Logic for Efficient Computing

Shaodi Wang, **Saptadeep Pal, Tianmu Li, Andrew Pan, Cecile Grezes, Pedram Khalili-Amiri, Kang L. Wang, and Puneet Gupta**

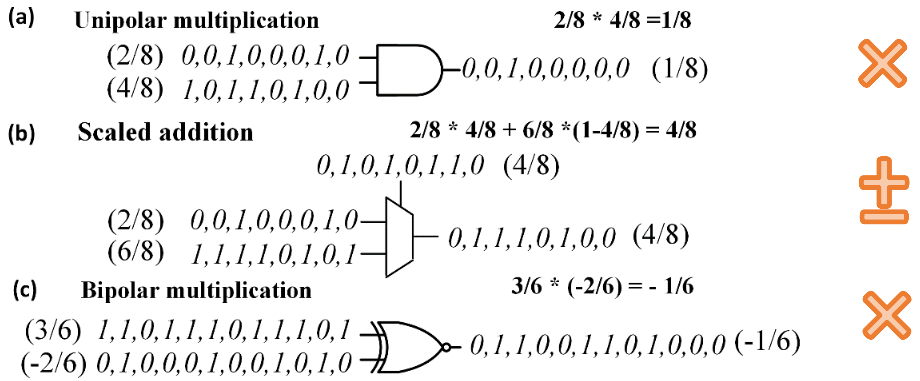
Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA

Guidelines

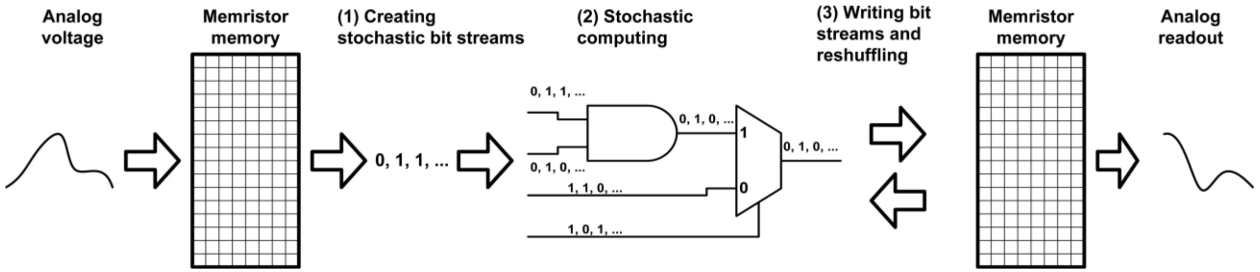
- Introduction of stochastic computing (SC)
- Introduction of voltage-controlled magnetic tunnel junctions (VC-MTJ) and negative differential resistance (NDR)
- Stochastic logic gates designed by VC-MTJ and NDR
- Stochastic bitstream (SBS) generation design
- Design evaluation

Stochastic computing (SC)

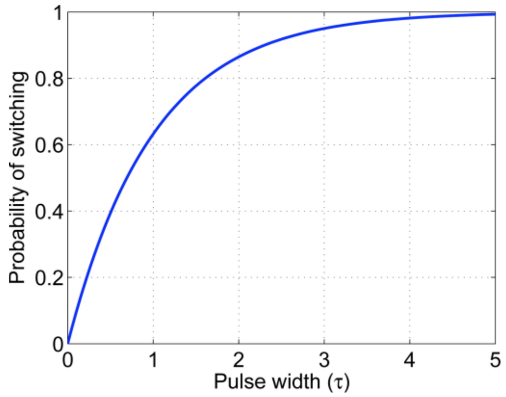
- **Advantages**
  - **Efficiency in additions and multiplications**
  - **Parallelism**
- **Disadvantages**
  - **High leakage from massive registers**
  - **Inefficiency for high-precision applications**
  - **Ineffiecient pseudo stochastic bitstream generation**



Non-volatile SC with STT-MTJ and memristor



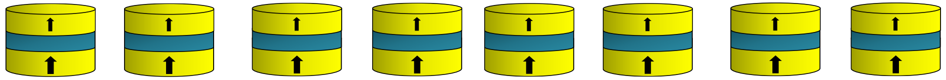
- **Advantages**
  - **Low leakage**
  - **Truly random SBS generation**
- **Disadvantages**
  - **SBS generation is limited by precision**
  - **Correlation caused by process variation**
  - **Data copy between CMOS and NVM**
  - **Amended by the proposed work**



Figures from Knag, Phil, Wei Lu, and Zhengya Zhang. "A native stochastic computing architecture enabled by memristors." *IEEE Transactions on Nanotechnology* 13.2 (2014): 283-293.

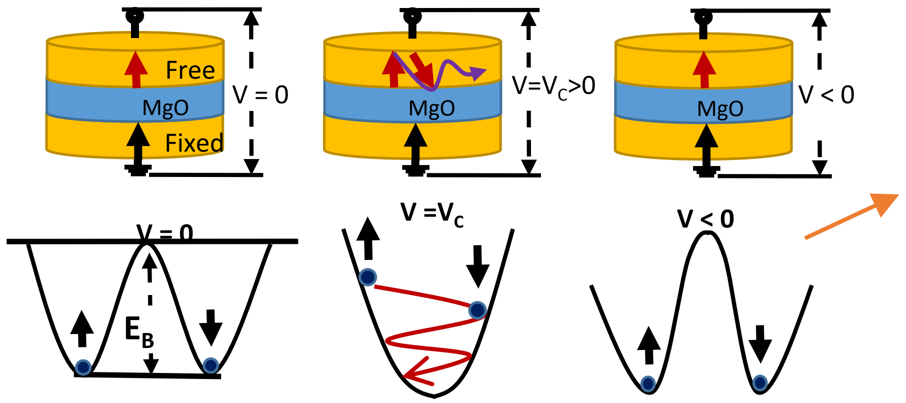
Highlights

- **The proposed voltage-controlled magnetic tunnel junctions (VC-MTJ) based SC**
  - **VC-MTJ based NV SBS registers**

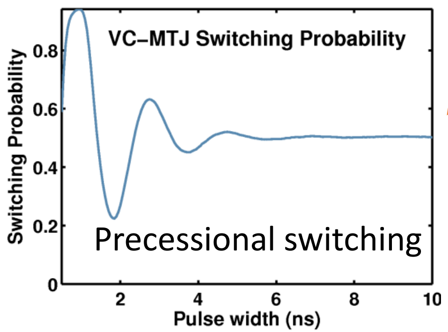


  - **Computations on VC-MTJs**
    - **Skipping the data copy between non-volatile memory and CMOS logics**
  - **Energy-efficient computation**
  - **Efficient and reliable truly stochastic bitstream (SBS) generator**
    - **Free from process and temperature variation impact**

Utilizing Voltage-controlled magnetic tunnel junctions (VC-MTJ) for SBS generation and storing



- Zero read disturbance
- **But need a simple hardware to enable the use of register**

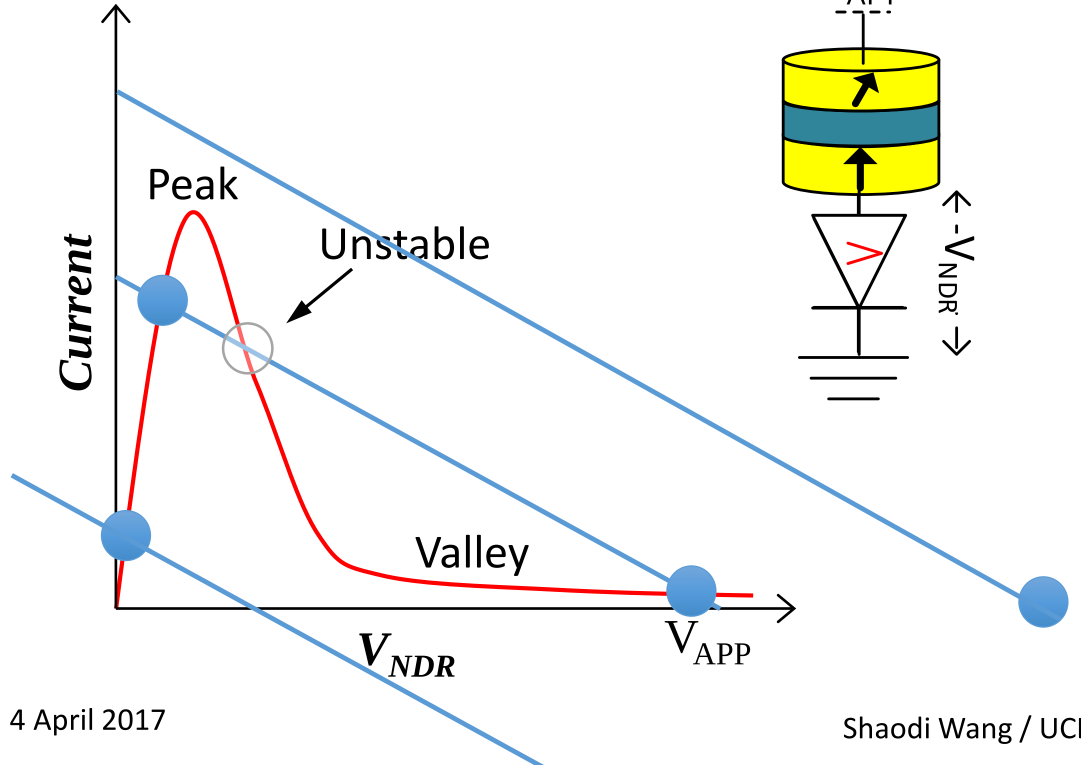


- Anti-parallel (AP) and parallel (P) resistance >50k Ω → low switching energy
- Non-deterministic switching → **need a simple hardware to enable deterministic switching**

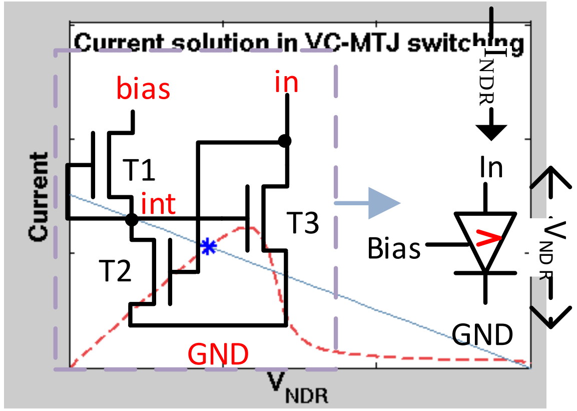
Introduction – Negative differential resistance (NDR)

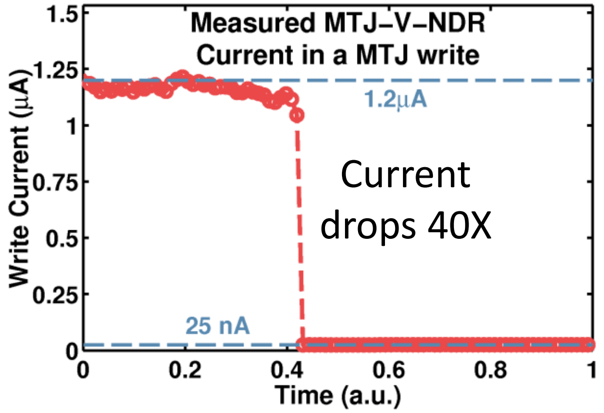
Increase V_{APP} from 0

Decrease V_{APP} from high to low



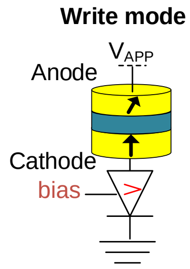
RRAM resistance from HRS to LRS

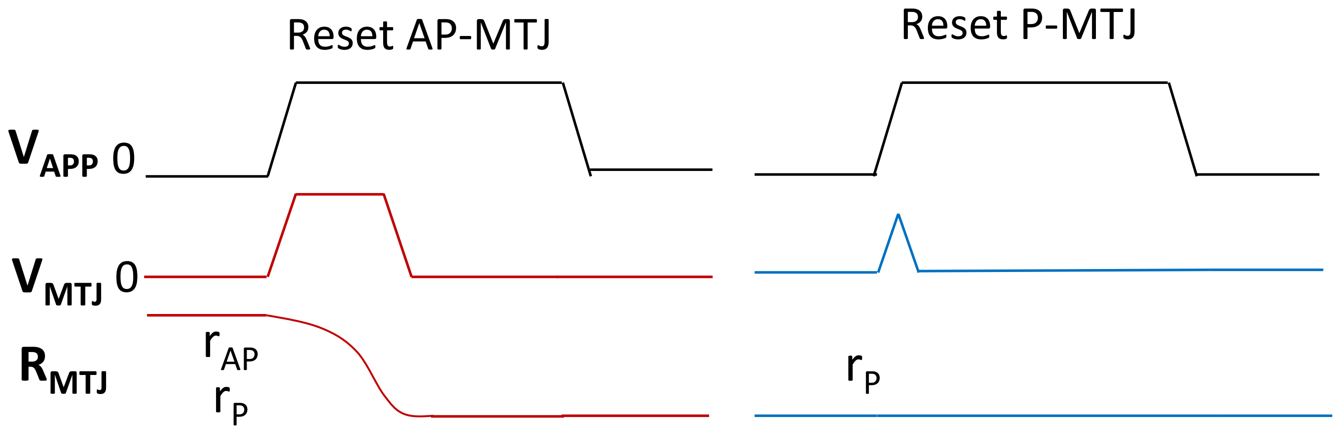




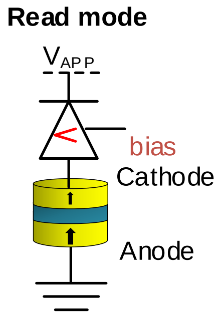
Bitwise operation using MTJ and NDR

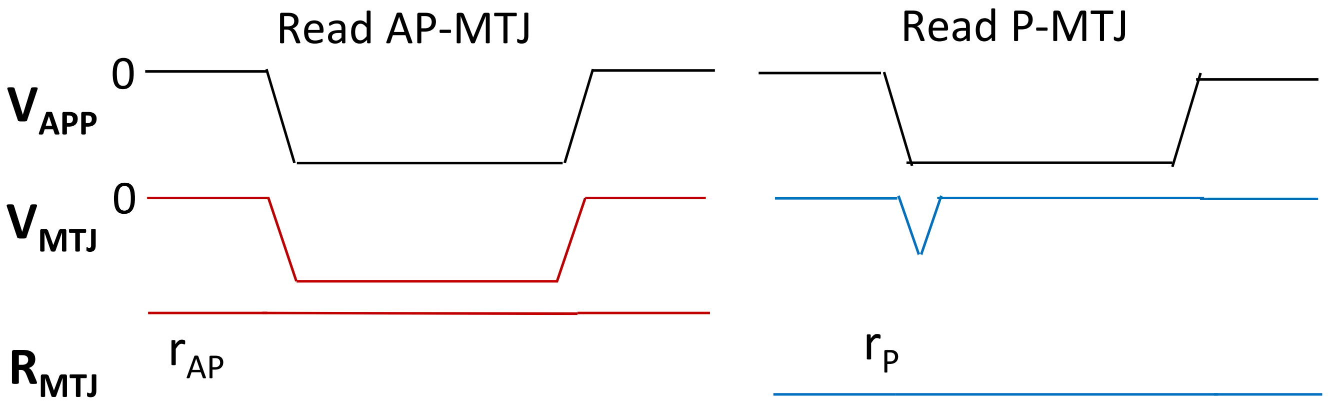
- **Reset operation**





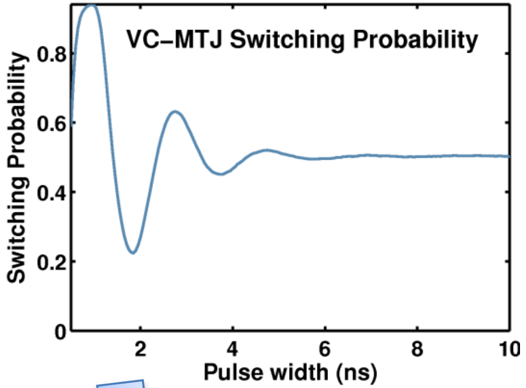
- **Read operation**





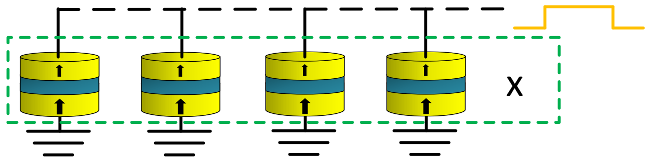
Bitwise operation (cont'd)

VC-MTJ switching probability



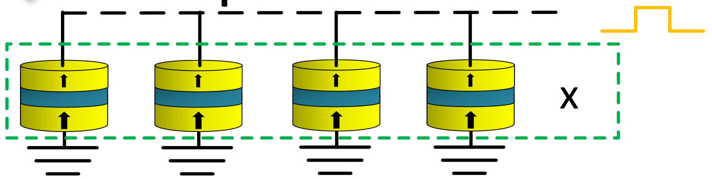
Randomization

Randomization x→0.5



Flip MTJ data

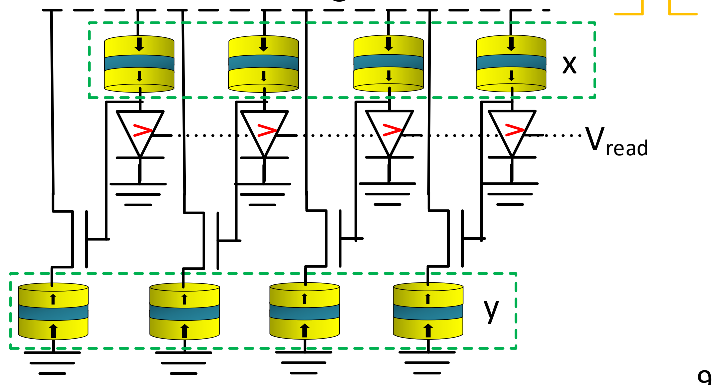
Flip x→$\overline{x}$



Reset + Flip = Write-1 (AP)

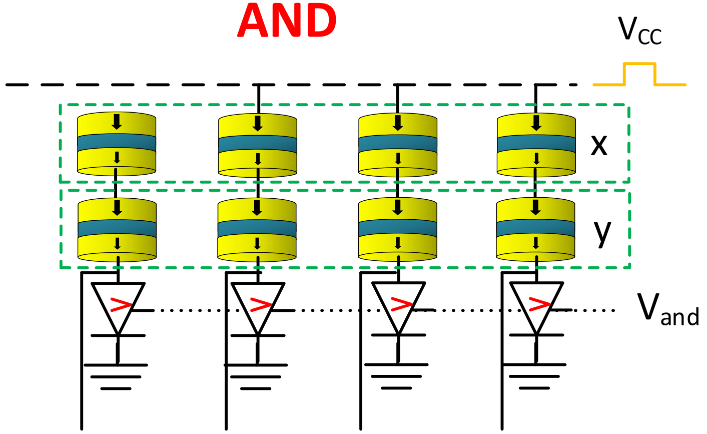
XNOR

XNOR y→$\overline{x \oplus y}$



Bitwise operation (cont'd)

AND

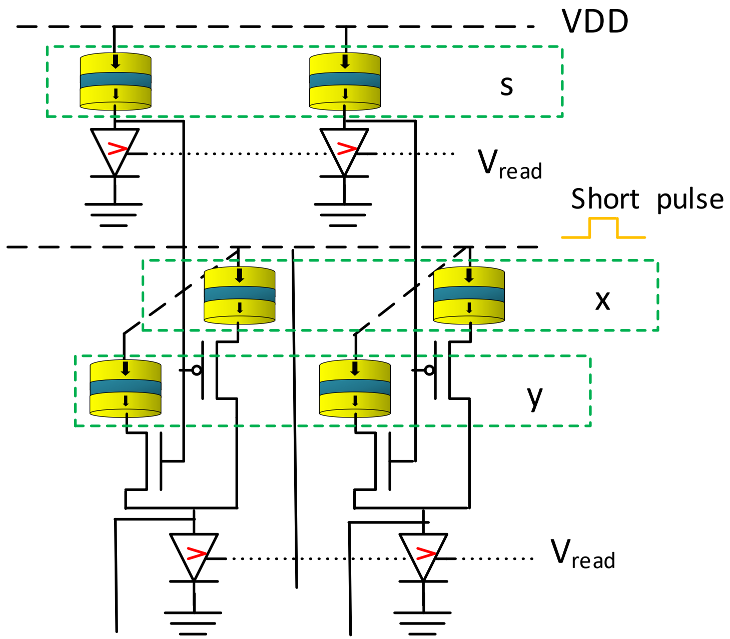


x•y

NDR peak current design prerequisite:

$$V_{CC} / (2r_{AP}) < I_{peak} < V_{CC} / (r_{AP} + r_P)$$

Scaled addition



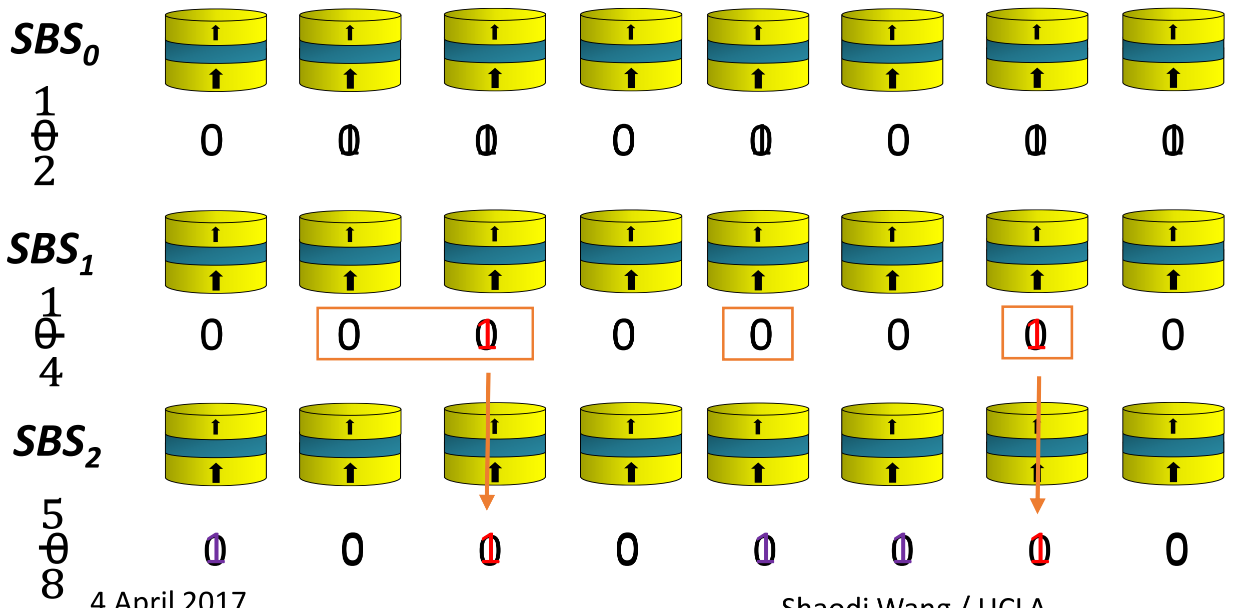
s·x+(1-s)·y

Stochastic bitstream generation

Binary fixed point 0.101

- Generation can be pipelined

$$=1\cdot\left(\frac{1}{2}\right)^1+0\cdot\left(\frac{1}{2}\right)^2+1\cdot\left(\frac{1}{2}\right)^3=\left(1+\boxed{\left(0+\boxed{1\cdot\frac{1}{2}}\right)\cdot\frac{1}{2}}\right)\cdot\frac{1}{2}$$



Randomization→ $\frac{1}{2}$

Scaled copy → x/2
(if $SBS_0[i]$==1 then Randomize $SBS_1[i]$)

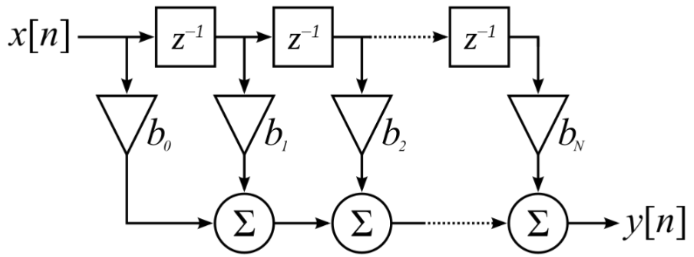
copy and rand → ***(x+1)/2***
(if $SBS_1[i]$==1, then flip $SBS_2[i]$, else randomize $SBS_1[i]$)

Design simulation

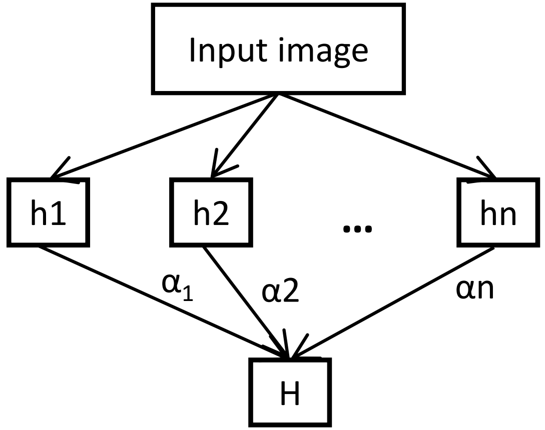
- SBS generator
  - HSPICE simulation with experimentally verified 50nm VC-MTJ Verilog-A model
  - 55X lower energy than CMOS linear-feedback shift register (LFSR)
- Standard adder and multiplier
  - HSPICE simulation to extract power and delay
  - Multiplier: 8 transistors per bit (this work) vs. 160 transistor per bit for a multiplier (CMOS binary)
  - Register: 3 transistors + 1 VC-MTJ (this work) vs 16 transistors (CMOS binary)

Evaluation benchmarks

- **Two representative design benchmarks**
  - **Finite impulse response (FIR) filter**
  - **Adaboost machine learning accelerator**
- **Ratio of SBS generation to multiplication and additions**
  - **FIR > Adaboost**



FIR filter



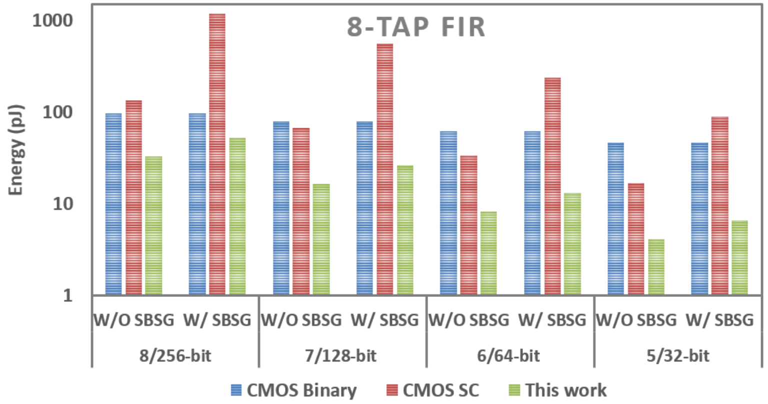
Adaboost

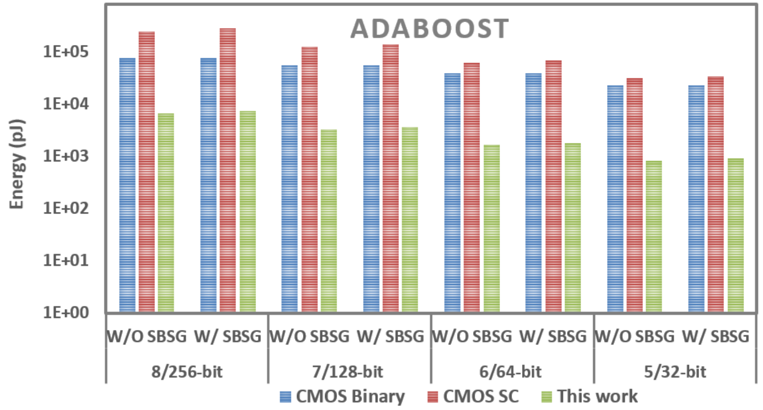
Evaluation (cont’d)

- Experimental setup
  - Precision: 32-bit SBS to 256-bit SBS corresponding to 5-bit to 8-bit binary fixed-point number
  - Energy is accounted for computation and SBS generation separately
  - Energy/output is the comparison metric
    - SC is easily pipelined, so there is no delay comparison
- Evaluation procedure
  - CMOS binary and CMOS SC implementation
    - Synthesis and place-route with 45nm commercials library
  - This work
    - HSPICE with 45nm CMOS commercial library and VC-MTJ experimentally verified model

Evaluation (cont'd)

- **Energy efficiency comparisons**
  - **FIR: 3~7X (256~32-bit precision) lower energy than CMOS binary design**
  - **Adaboost: 12~25X (256~32-bit precision) lower energy than CMOS binary design**
  - **This work < CMOS binary < CMOS SC in terms of energy**
  - **SC is more advantageous in low-precision applications**



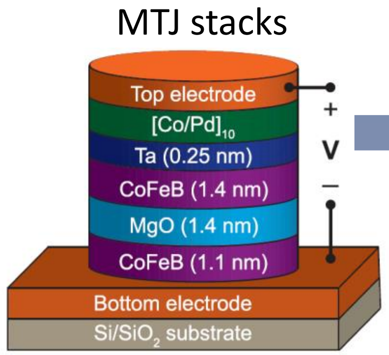


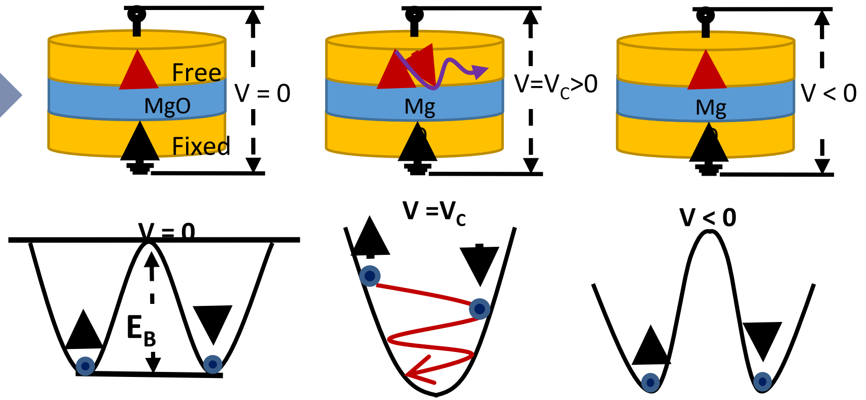
Thank you!

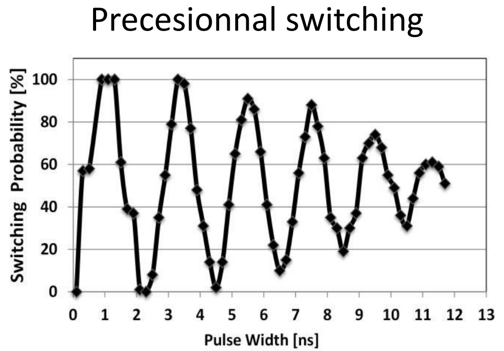
Q & A

Backup slides

Voltage-controlled magnetic tunnel junctions (VC-MTJ)

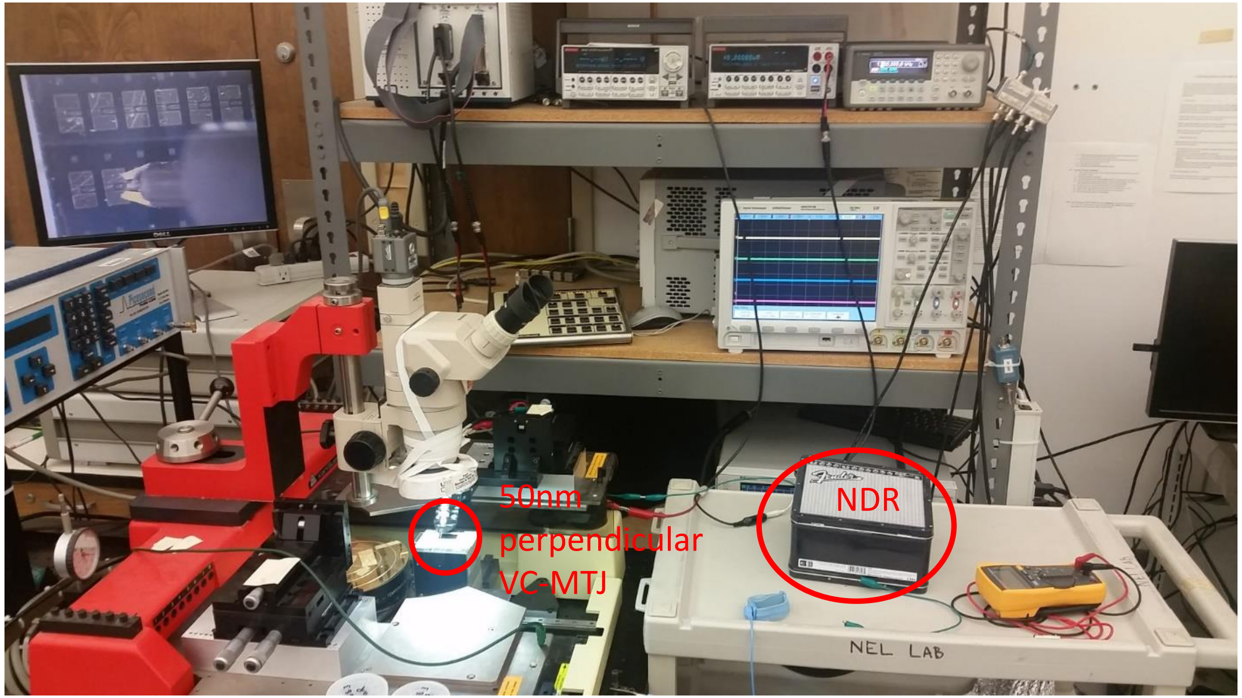




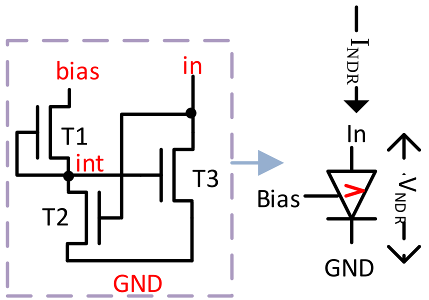


- Zero read disturbance
- Anti-parallel (AP) and parallel (P) resistance (>50k Ω) → low switching energy
- Efficient switching (computing): ~fJ per bit operation

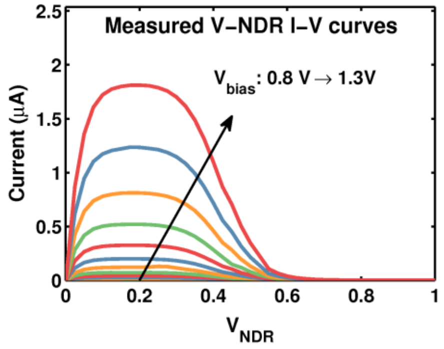
Experimental Setup for NDR and MRAM



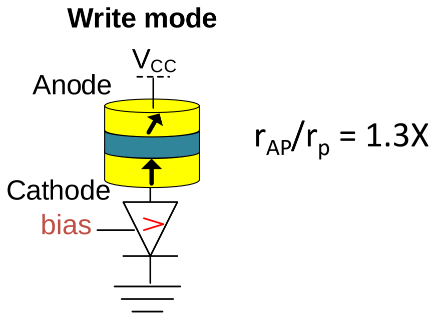
Experimental NDR and NDR-assisted write and read demonstration



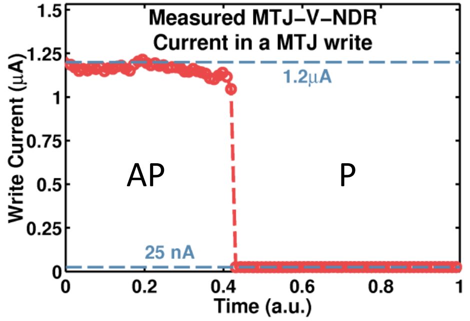
NDR implementation



NDR I-V curves vs. bias
Peak-to-valley current ratio of 10~1000



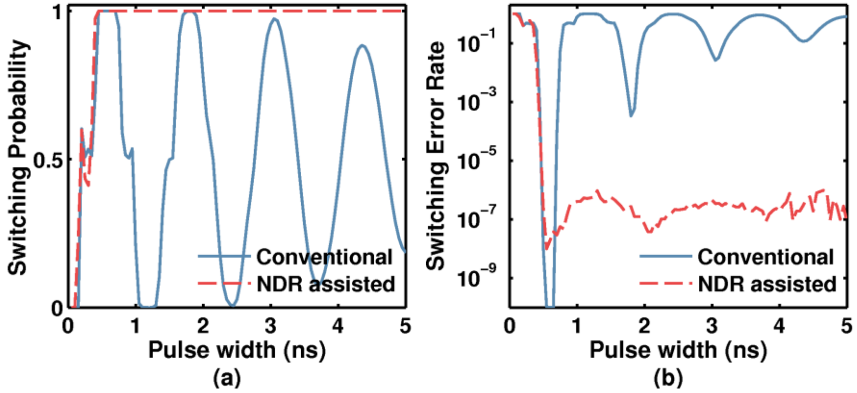
NDR assisted VC-MTJ write



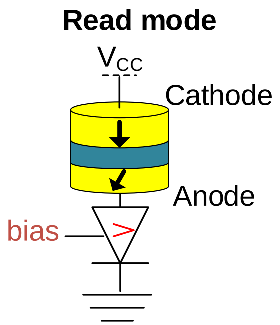
Write current drops 40X, while write voltage drops 50X

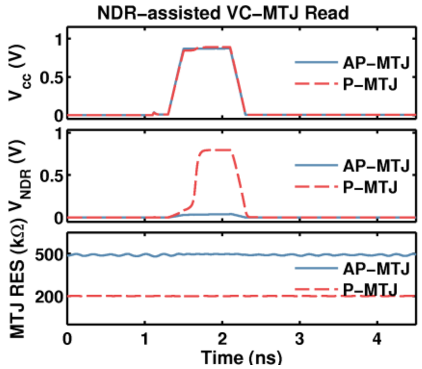
Experimental NDR and NDR-assisted write and read demonstration

- "Reset" VC-MTJ to P state (0 state) w/ NDR
  - Reset error rate < 10^{-6}



- NDR-assisted non-destructive VC-MTJ read
  - 0 read disturbance
  - Full voltage output swing through NDR to detect MTJ states



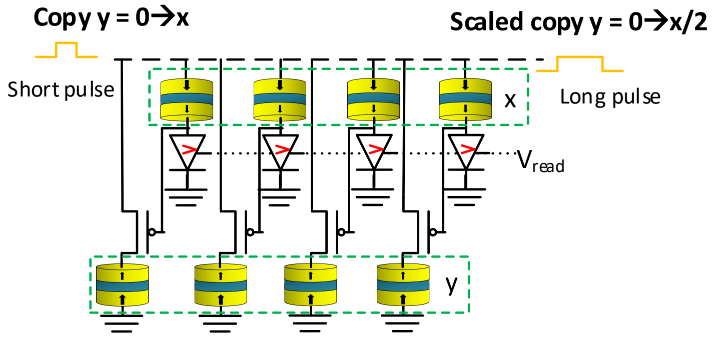


Stochastic bitstream generation

- Conversion from binary input to fraction
  - $0.n_2n_1n_0(binary) = n_2 \cdot \left(\frac{1}{2}\right)^1 + n_1 \cdot \left(\frac{1}{2}\right)^2 + n_0 \cdot \left(\frac{1}{2}\right)^3$
- Conversion from faction to stochastic bitstream (SBS)
  - $n_2\left(\frac{1}{2}\right)^1 + n_1\left(\frac{1}{2}\right)^2 + n_0\left(\frac{1}{2}\right)^3 = \underline{\left(n_2 + \left(n_1 + (n_0) \cdot \frac{1}{2}\right) \cdot \frac{1}{2}\right)} \cdot \frac{1}{2}$
  - x_i is 0 or 1
  - $(1 + x) \cdot \frac{1}{2}$ and $(0 + x) \cdot \frac{1}{2}$, where x is a SBS
  - With long pulse, VC-MTJ switching creates perfect $\frac{1}{2}$

Stochastic bitstream generation (cont'd)

- Copy (*$y=x$*)
  - Reset SBS y to 0
  - For each $x_i=1$, flip y_i *to 1*
- Scaled copy (*$y=x/2$*)
  - Reset SBS y to 0
  - For each $x_i=1$, random y_i
- Copy and rand (*$y=(1+x)/2$*)
  - Reset SBS y to 0
  - For each $x_i=1$, flip y_i *to 1*
  - For each $x_i=0$, random y_i



Stochastic bitstream generation (cont'd)

	$IN_2IN_1IN_0$
Binary input[1]:	0.1 0 1
Binary input[2]:	0.0 1 0
Binary input[3]:	0.1 1 1

Rules:

IN_i==1: copy and rand → *$y=(x+1)/2$*

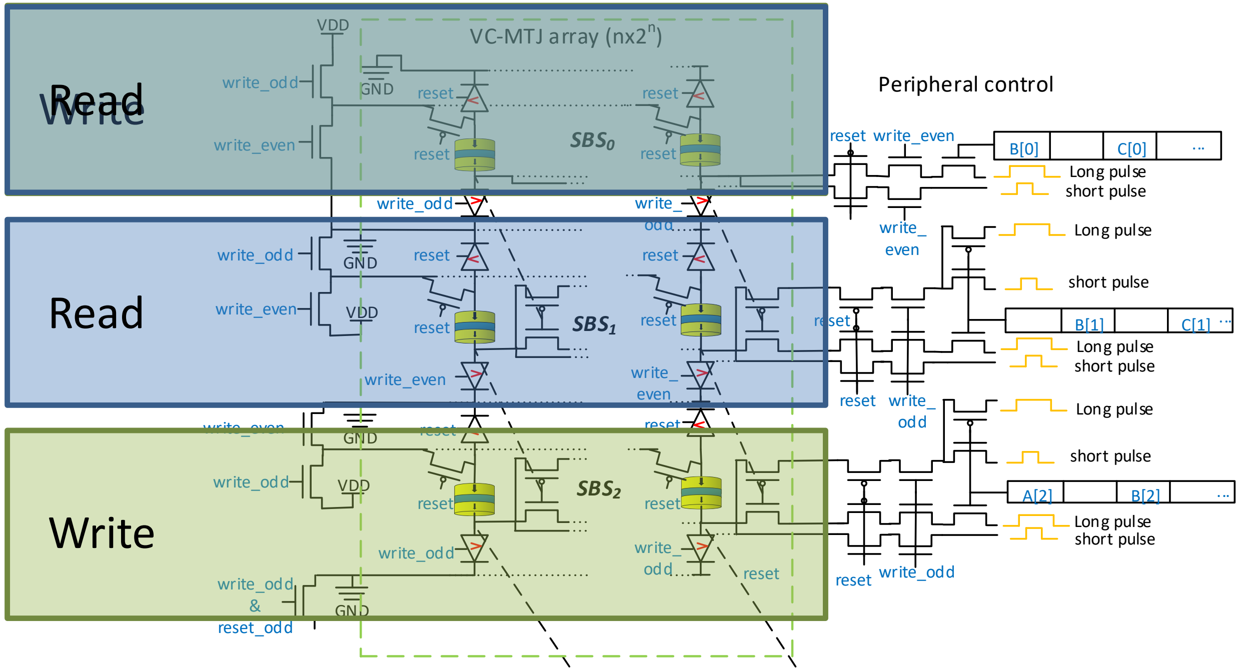
IN_i==0: scaled copy → *$y=x/2$*

STEP	SBS_0	IN_0	SBS_1	IN_1	SBS_2	IN_2
1	01001000					
2	00000000		01101011			
3	10101100		01010000			
4			11101101			

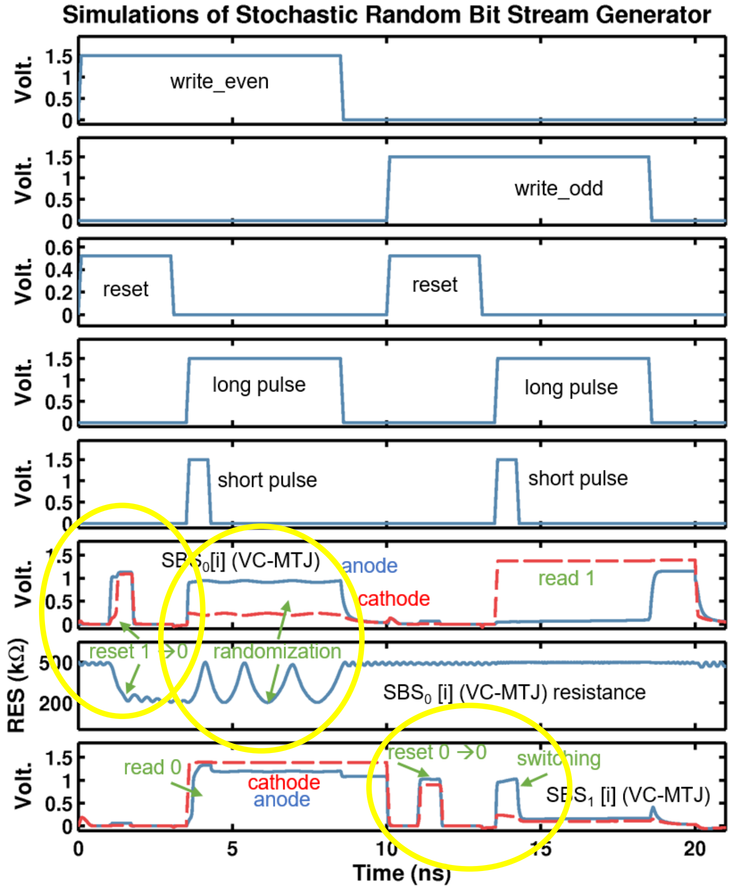
Pipelined SBS generation

Stochastic bitstream generation (cont’d)

- Hardware designs:
  - SBS_{odd} is written while SBS_{even} is being read
  - Every clock cycle half of SBS is changed



Stochastic bitstream generation (cont'd)



Reset SBS_0

Randomize SBS_0

Reset SBS_1 and copy from SBS_0

On-going works – automatic synthesis

- Target: synthesizing algorithm description to circuit
- Optimizing targets
  - Correlation, area and latency minimization
- Proposed solutions
  - Factorization and computing depth minimization
  - Examples:
    - $F = abg + acg + adf + aef + afg + bd + ce + be + cd \longrightarrow (b+c)(d+e+ag) + (d+e+g)af$