Variation Tolerant Logic Mapping for Crossbar Array Nano Architectures

Cihan Tunc and Mehdi B. Tahoori

Karlsruhe Institute of Technology, Germany

Northeastern University, Boston, USA

Outline

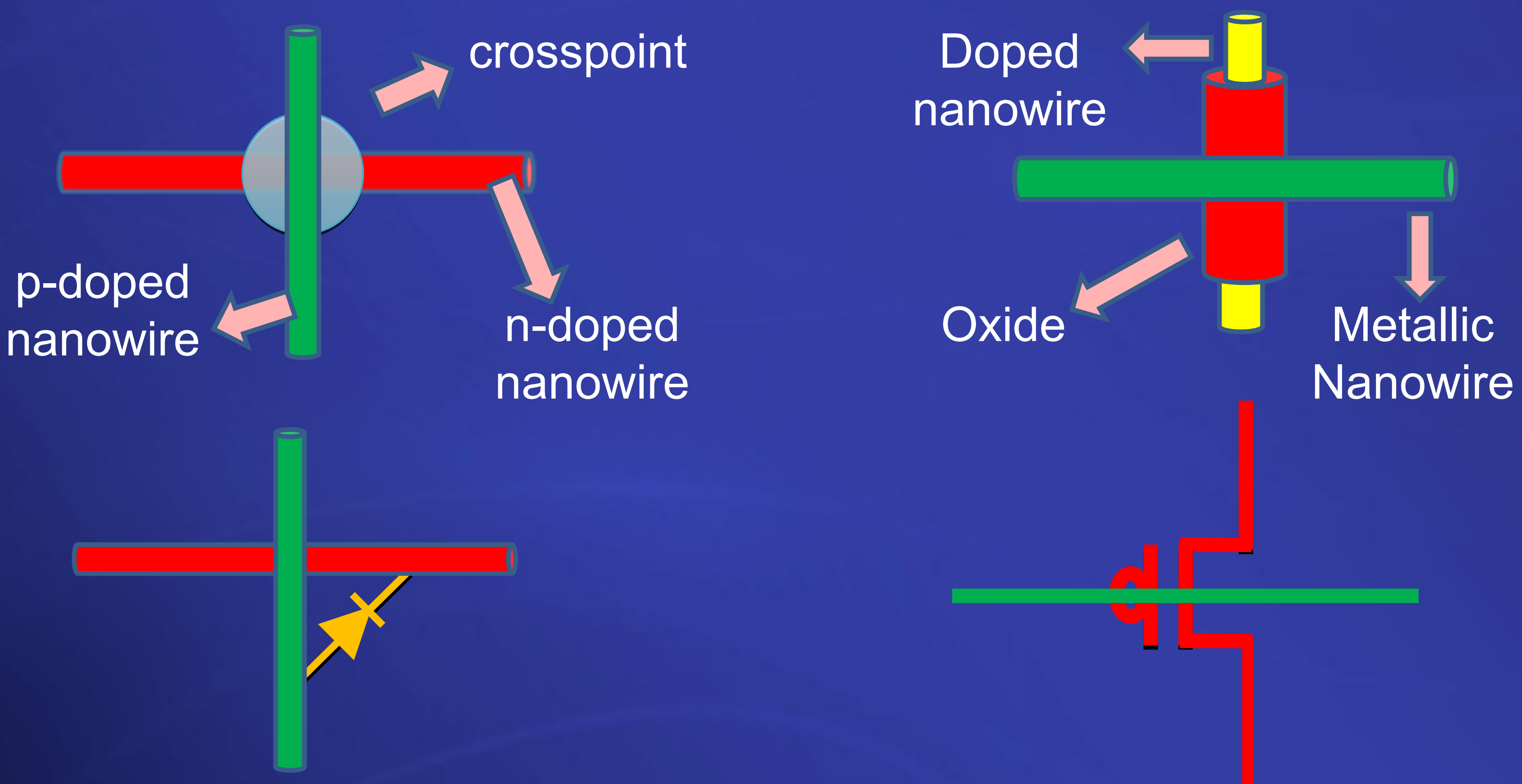
- Introduction and Motivation
- Definitions
- Algorithms
- Defect Tolerance
- Experimental Results
- Conclusion

Introduction

- Increasing challenges in CMOS downscaling
  - More power dissipation
  - Parasitic issues
  - Direct tunneling
  - More complex tools resulting in higher costs
- Alternative: Emerging Nanotechnologies
  - Higher device density
  - Less expensive in manufacturing
  - Manufacturing with bottom-up stochastic self assembly, nanoimprinting, etc.

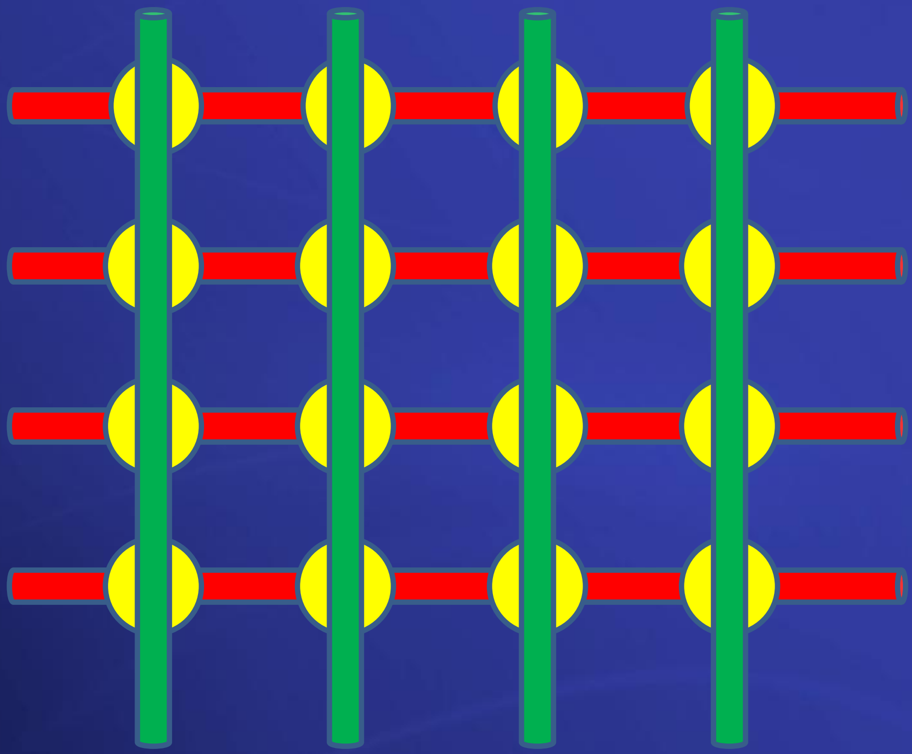
Emerging Nanotechnologies

- Nanowires for p-n diode rectifiers and FETs



Emerging Nanotechnologies

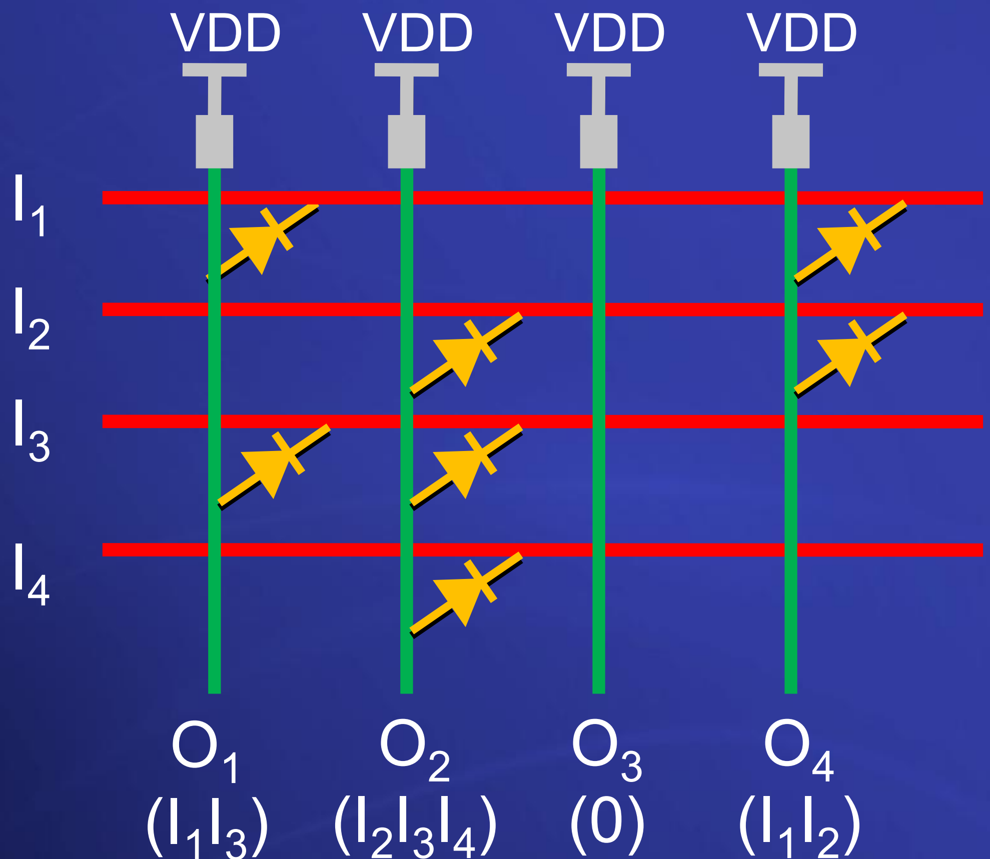
- Crossbars are implemented with two perpendicular nanowire sets

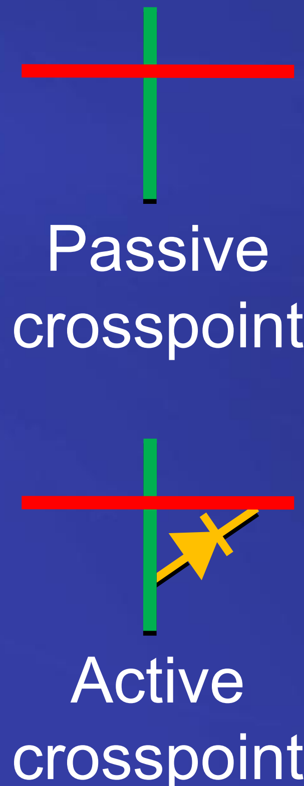


A crosspoint

Emerging Nanotechnologies

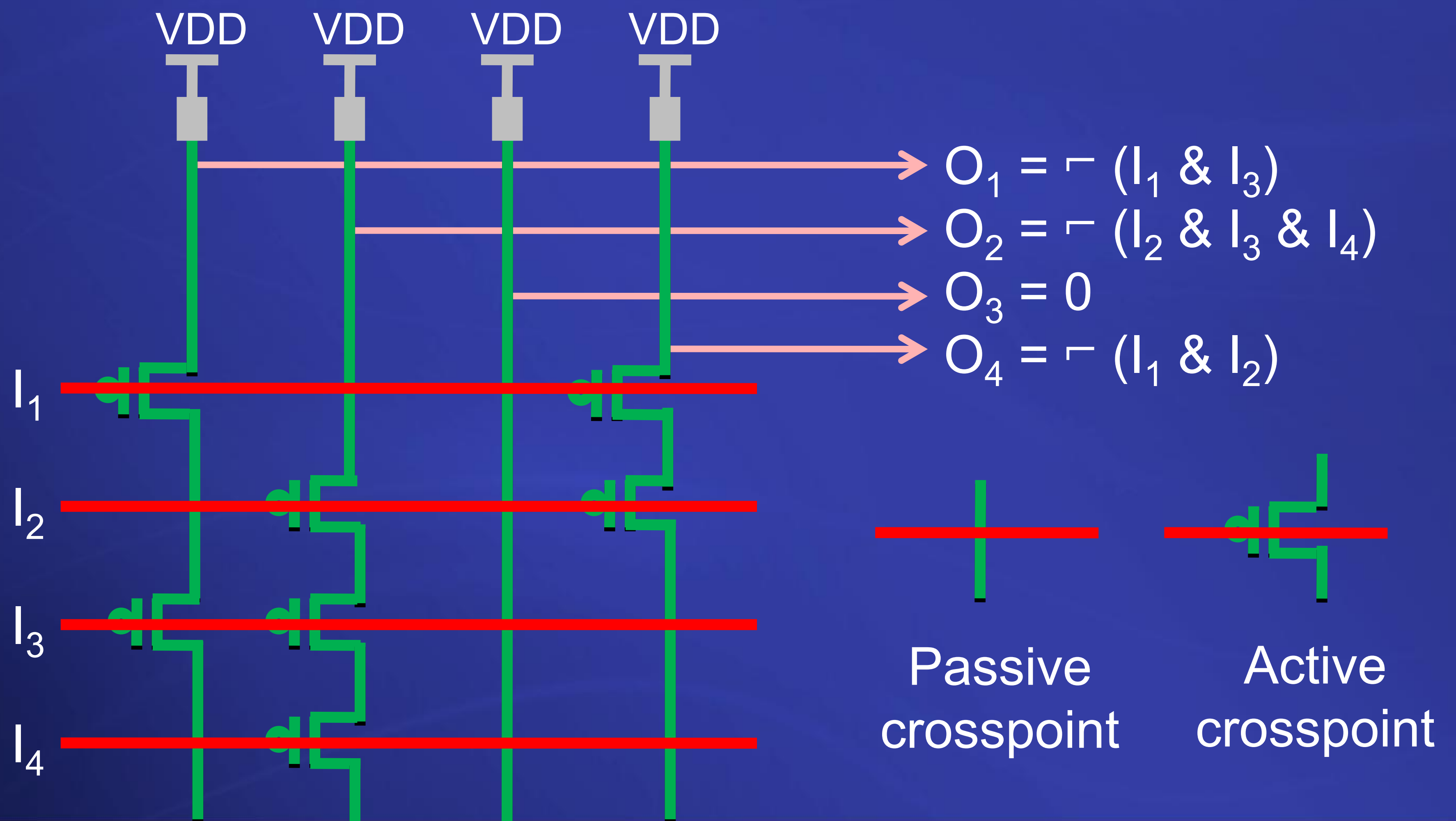
- Diode based crossbars





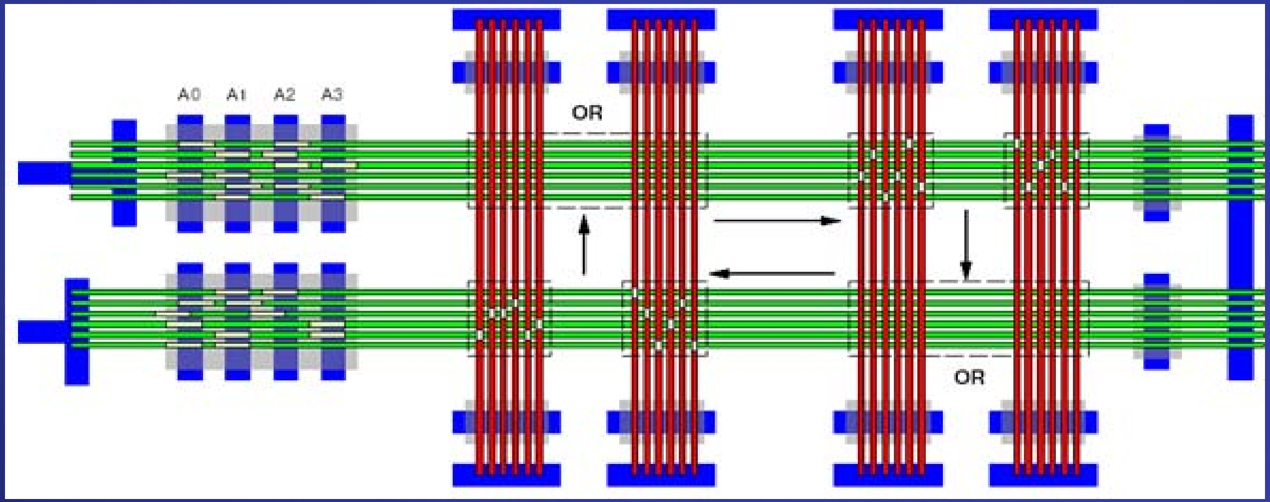
Emerging Nanotechnologies

- FET based crossbars (focus of this work)



Architectures

- Using diode based crossbars
  - CMOL [Likharev_05]
  - NanoFabric [Goldstein_01]
  - nanoPLA [DeHon_04]

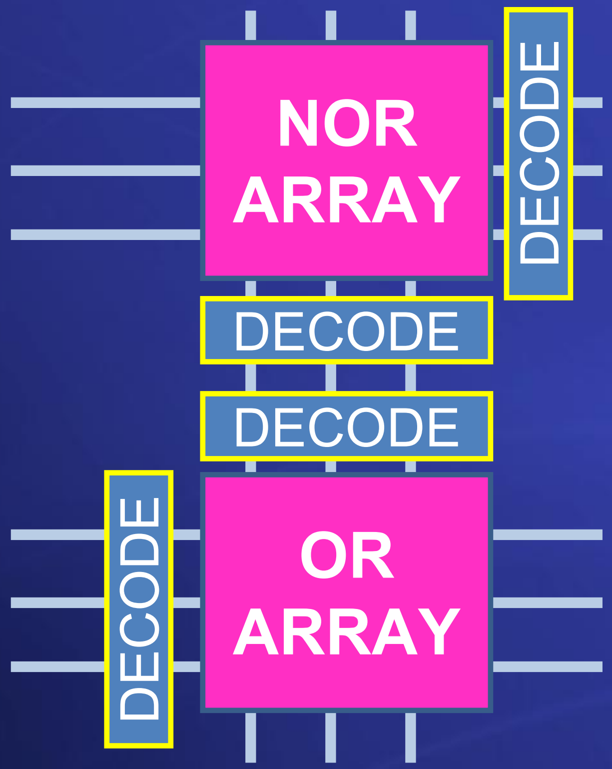


NanoPLA, [DeHon_04]

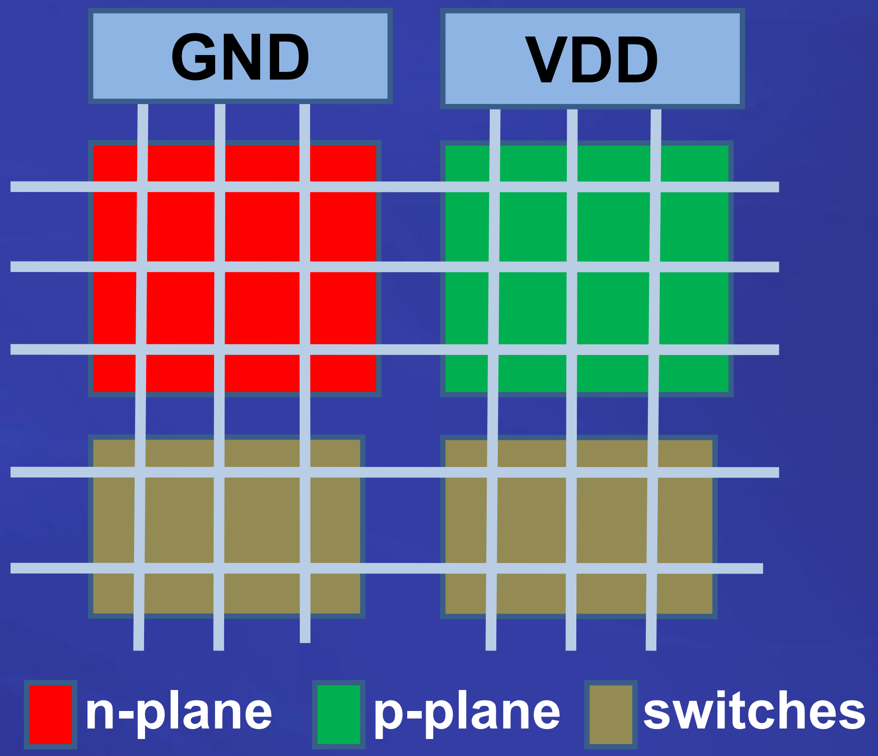
Architectures

- Using FET based crossbars:

NOR blocks for nanoPLA



FET arrays with switch blocks



Motivation

- Basic, regular, and stochastic manufacturing
- High defect rate → low yield rate
  - Open or shorted nanowires
  - Defective crosspoints
- High variation in manufacturing process
  - Resistance, capacitance, etc.
  - Results in delay variation
- Variation is extremely larger than CMOS technology
  - 100 – 200% compared to 10-15%

Objective

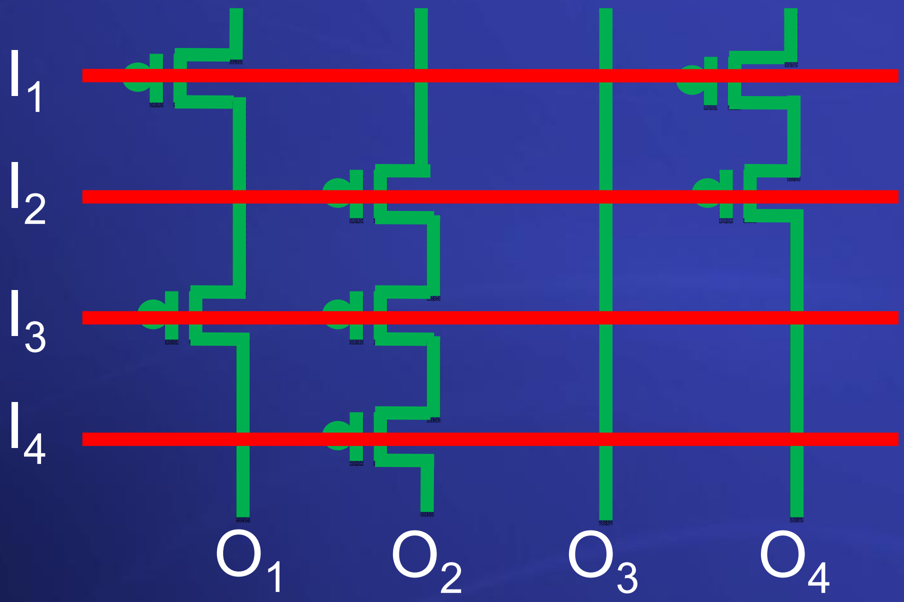
- Variation aware mapping for FET-based crossbars
- Variations are delay differences of individual FETs
- Different optimization goals and different constraints
- Defect tolerant mapping

Outline

- Introduction and Motivation
- Definitions
- Algorithms
- Defect Tolerance
- Experimental Results
- Conclusion

Function Matrix (FM)

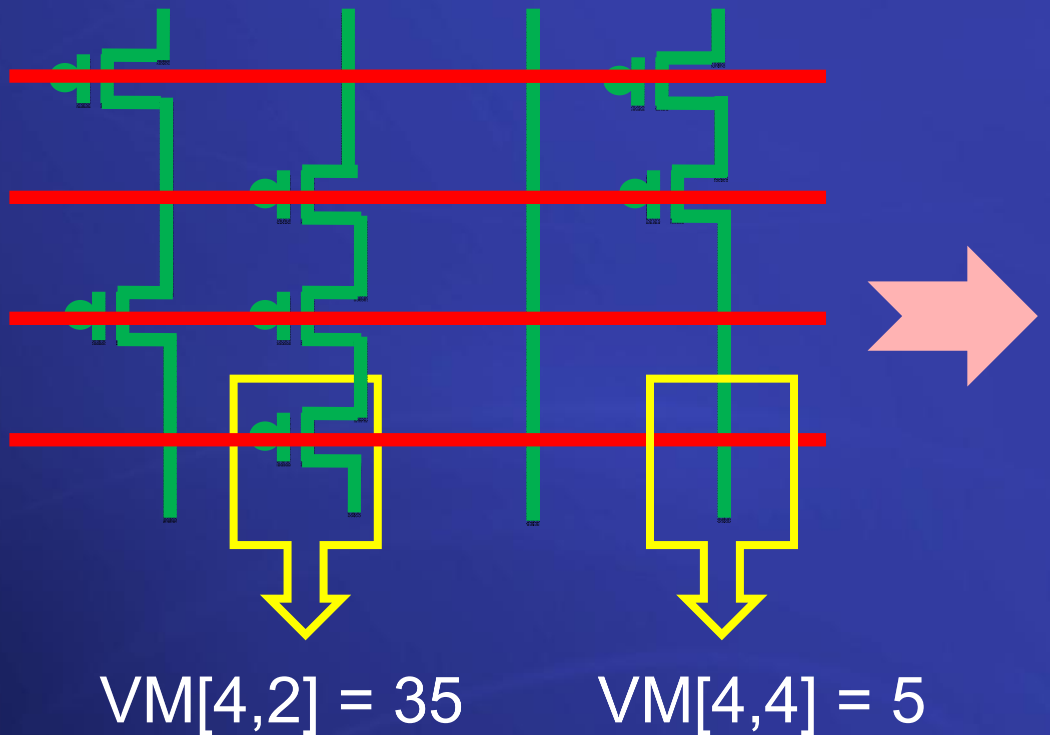
- The logic function to be mapped to a crossbar.
- $FM_{i,j} = \begin{cases} 1, & \text{if output } j \text{ depends on input } i \\ 0, & \text{otherwise} \end{cases}$



Function Matrix			
1	0	0	1
0	1	0	1
1	1	0	0
0	1	0	0

Variation Matrix (VM)

- Delay of individual crosspoints using lumped delay modeling



Variation Matrix			
55	10	45	45
20	5	40	10
95	75	15	15
45	35	50	5

Variation Matrix

- VM entries extracted by a characterization testing procedure
  - Delay testing
    - Taking advantage of programmability of crossbars
    - All crossbar outputs read simultaneously
    - Both falling and rising transitions applied through each crosspoint
  - One test configuration
    - All crosspoints are activated (all FM elements are ‘1’)
  - For each input, two controlling transitions are applied
    - All other inputs are stable at the non-controlling values

Variation Matrix & Function Matrix

- Variation Matrix: Property of individual crossbars and different for any crossbar
- Function Matrix: Property of the logic function and fixed for all crossbars

VM 1			
55	10	45	45
20	5	40	10
95	75	15	15
45	35	50	5

VM 2			
10	50	20	35
40	55	5	90
90	65	35	45
70	60	65	50

FM			
1	0	0	1
0	1	0	1
1	1	0	0
0	1	0	0

Input/Output Mapping Vectors

- For the mapping of a function to a $n \times m$ crossbar
  - Input Mapping Vector (IMV)
    IMV[i] = j, if input x_i is assigned to horizontal nanowire j
  - Output Mapping Vector (OMV)
    OMV[i] = j, if output f_i is assigned to vertical nanowire j

	Mapping 1			
I_1	1	0	0	1
I_2	0	1	0	1
I_3	1	1	0	0
I_4	0	1	0	0
	O_1	O_2	O_3	O_4

	Mapping 2			
I_2	0	1	0	1
I_3	0	1	1	0
I_1	0	0	1	1
I_4	0	1	0	0
	O_3	O_2	O_1	O_4

For Mapping 1:
IMV = {1, 2, 3, 4}
OMV = {1, 2, 3, 4}

For Mapping 2:
IMV = {3, 1, 2, 4}
OMV = {3, 2, 1, 4}

Cost Function

- Based on VM and actual mapping (FM, IMV/OMV)
- For every output f_i

$$C(f_i) = \sum_{k=1}^{n} FM[k][i] \times VM[IMV[k]]\,[OMV[i]]$$

VM			
55	10	45	45
20	5	40	10
95	75	15	15
45	35	50	5

FM			
1	0	0	1
0	1	0	1
1	1	0	0
0	1	0	0
f_1	f_2	f_3	f_4

For IMV = {1, 2, 3, 4}
OMV = {1, 2, 3, 4}

Cost (f_1) = 55 + 95 = 150
Cost (f_2) = 5 + 75 + 35 = 115
Cost (f_3) = 0
Cost (f_4) = 45 + 10 = 55

Optimization Goals

- Objective 1: Minimize the maximum delay
- Objective 2: Balance all output delays

VM			
55	10	45	45
20	5	40	10
95	75	15	15
45	35	50	5

FM			
1	0	0	1
0	1	0	1
1	1	0	0
0	1	0	0
f_1	f_2	f_3	f_4

For IMV = {1, 2, 3, 4}
OMV = {1, 2, 3, 4}

Cost (f_1) = 55 + 95 = 150
Cost (f_2) = 5 + 75 + 35 = 115
Cost (f_3) = 0
Cost (f_4) = 45 + 10 = 55

→ Objective 1: 150 (Minimize maximum cost)
→ Objective 2: 150 – 55 = 95 (Minimize maximum difference)

Outline

- Introduction and Motivation
- Definitions
- Algorithms
- Defect Tolerance
- Experimental Results
- Conclusion

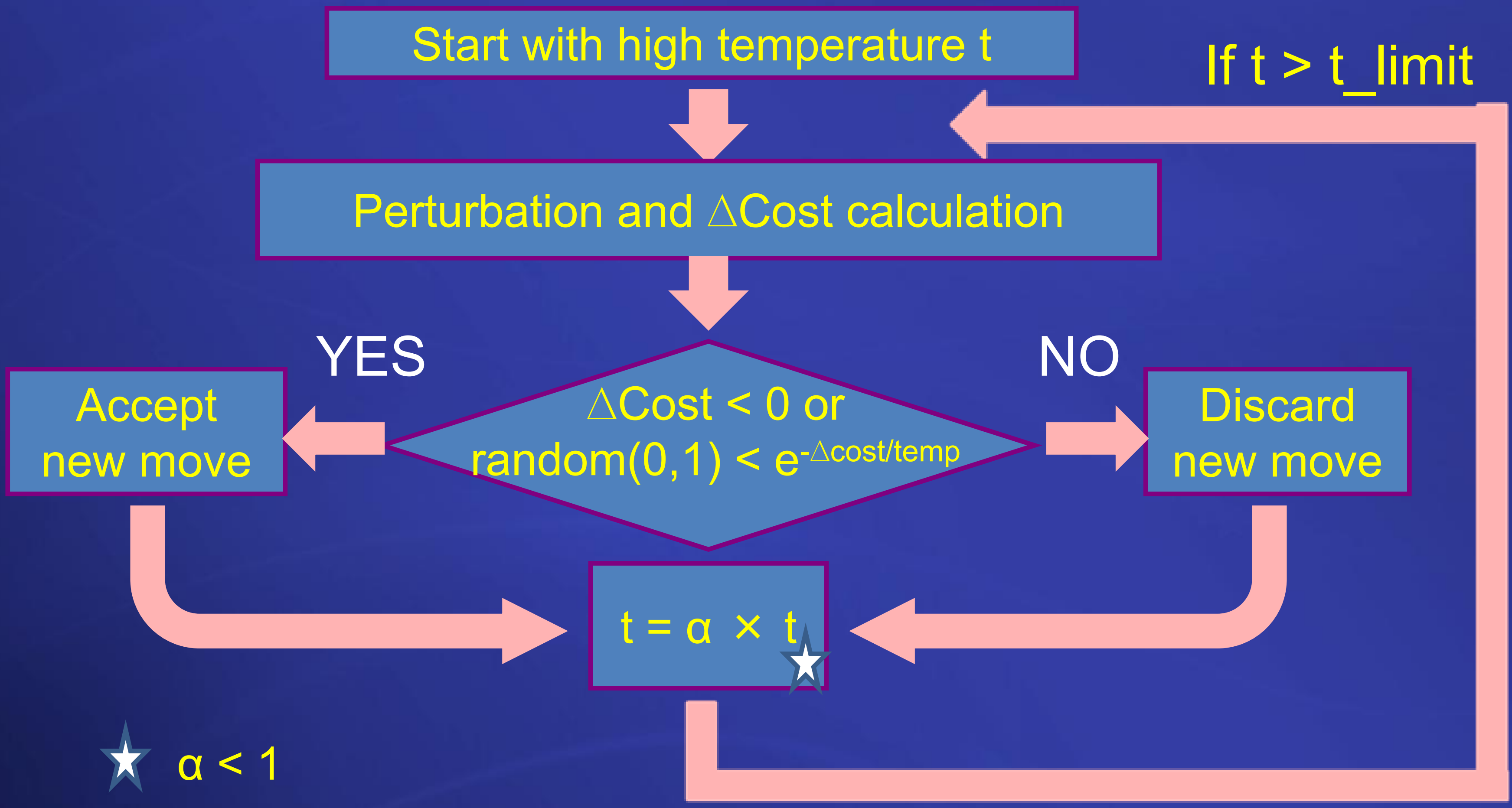
Exhaustive Search

- Try all possible solutions and find the best solution
- There are n! x m! possible solutions for n x m crossbar
- Intractable for large crossbars
- We require combinatorial optimization methods: Simulated Annealing

Simulated Annealing

- A general-purpose optimization method
  - Widely used in VLSI design automation
- Based on energy distribution minimization of metals
  - First heats and then cools gradually
- Perturbations are random
- Acceptance of new perturbations is
  - $\Delta cost < 0$ or
  - $\Delta cost > 0$ with probability $e^{-\Delta cost/temp}$
  - To avoid local optimums

Simulated Annealing Algorithm



Moves

- Swapping in only Input Vector, OMV remains same:

IMV = {1, 2, 3, 4} → new IMV = {4, 2, 3, 1}
OMV = {1, 2, 3, 4} → new OMV = {1, 2, 3, 4}

	FM			
I_1	1	0	0	1
I_2	0	1	0	1
I_3	1	1	0	0
I_4	0	1	0	0
I/O	O_1	O_2	O_3	O_4

→

	Mapping			
I_4	0	1	0	0
I_2	0	1	0	1
I_3	1	1	0	0
I_1	1	0	0	1
I/O	O_1	O_2	O_3	O_4

Swapped (I_4 and I_1)

Moves

- Swapping in only Output Vector, IMV remains same:

IMV = {1, 2, 3, 4} → new IMV= {1, 2, 3, 4}
OMV = {1, 2, 3, 4} → new OMV = {1, 2, 4, 3}

	FM			
I_1	1	0	0	1
I_2	0	1	0	1
I_3	1	1	0	0
I_4	0	1	0	0
I/O	O_1	O_2	O_3	O_4

→

	Mapping			
I_1	1	0	1	0
I_2	0	1	1	0
I_3	1	1	0	0
I_4	0	1	0	0
I/O	O_1	O_2	O_4	O_3

Swapped

Moves

- Swapping in both Input and Output Vectors:

IMV = {1, 2, 3, 4} → new IMV = {4, 2, 3, 1}

OMV = {1, 2, 3, 4} → new OMV = {1, 2, 4, 3}

	FM			
I_1	1	0	0	1
I_2	0	1	0	1
I_3	1	1	0	0
I_4	0	1	0	0
I/O	O_1	O_2	O_3	O_4

→

	Mapping			
I_4	0	1	0	0
I_2	0	1	1	0
I_3	1	1	0	0
I_1	1	0	1	0
I/O	O_1	O_2	O_4	O_3

Swapped (rows I_4 … I_1)

Swapped (columns O_4, O_3)

Outline

- Introduction and Motivation
- Definitions
- Algorithms
- Defect Tolerance
- Experimental Results
- Conclusion

Defect Tolerance

- Sources of defects:
  - Nanowires may be broken or misaligned
  - Crosspoints may be defective
- All defects are modeled by crosspoint defects
  - Stuck-open crosspoint: cannot be ***activated***
    - Crosspoint is unusable.
  - Stuck-closed crosspoint: cannot be ***deactivated***
    - All nanowires and crosspoints intersecting at this crosspoint are unusable.

Defect Tolerance

- Infinite VM entries for defective crosspoints
  - (1, 1) and (4, 2) stuck-open
  - (3, 3) stuck-close
- Infinite cost when defective
- Simulated Annealing will discard defective crosspoints
  - $\triangle$Cost $= \infty$
  - $e^{(-\infty/\text{temperature})} = 0$

Defects in VM			
∞	10	∞	45
20	5	∞	10
∞	∞	∞	∞
45	∞	∞	5

🟨 Stuck open

🟩 Stuck closed

Outline

- Introduction and Motivation
- Definitions
- Algorithms
- Defect Tolerance
- Experimental Results
- Conclusion

Experiment Setup for Optimization

- Two Objectives
  - Objective 1: Minimizing maximum cost
  - Objective 2: Balancing output delays
- Comparison with exhaustive search
  - 6x6 crossbars
  - Crosspoint Usage Ratio (CR): 30% and 40%
  - Output Usage Ratio (OR): 60%
  - 500 random crossbars
- Cost reduction for 16x16 crossbars
  - Crosspoint Usage ratio (CR): 20%, 30%, and 40%
  - Output Usage Ratio (OR): 80%
  - 10,000 random crossbars

Experiment Setup for Defect Tolerance

- Success for defect free mapping
  - 8x8, 16x16, and 32x32 crossbars
- Different Constraints
  - Crosspoint Usage Ratio (CR): 20%, 30%, 40%
  - Output Usage Ratio (OR) : 80%
- Different Defect Rates: 5% and 10%
- 1,000 random crossbars generated

Experimental Results

- Exhaustive search vs. Simulated Annealing for 6x6 crossbars

	Objective 1			Objective 2		
	RAND	EXH	SA	RAND	EXH	SA
CR = 30%	189.30%	–	3.57%	2.325%	–	50%
CR = 40%	144.20%	–	2.60%	1,917%	–	50%
Runtime Overhead	–	2320x	1	–	1940x	1

RAND : Variation Unaware Mapping
EXH : Exhaustive Search
SA : Simulated Annealing

Experimental Results

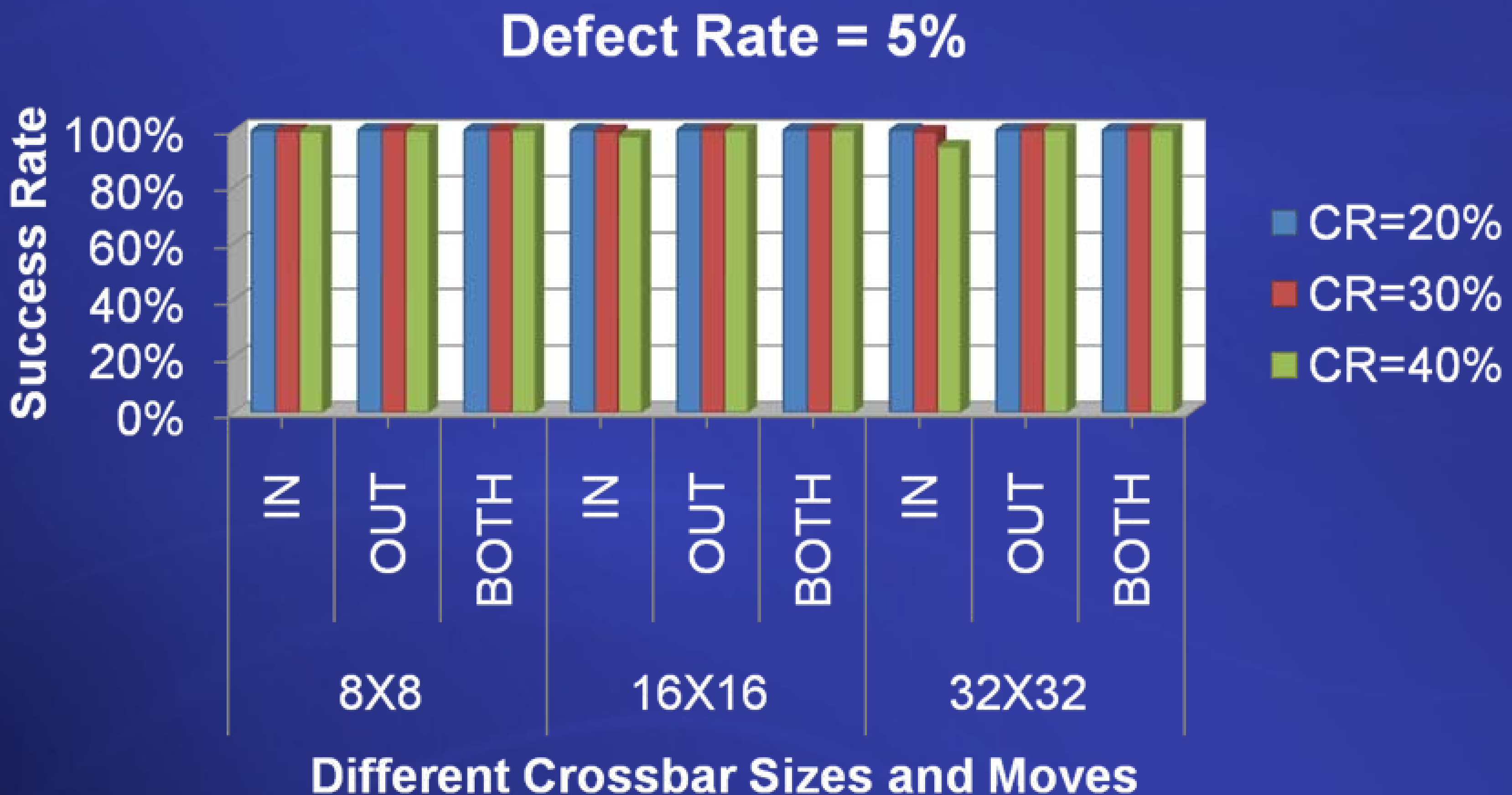
- Cost reduction for 16x16 crossbars

	Objective 1			Objective 2		
CR	IN	OUT	BOTH	IN	OUT	BOTH
20%	1.61	1.85	1	2.82	3.25	1
30%	1.52	1.56	1	2.40	2.41	1
40%	1.47	1.38	1	2.49	2.17	1

IN: Move in only IMV
OUT: Move in only OMV
BOTH: Move in both IMV and OMV
CR = Crosspoint Usage Ratio

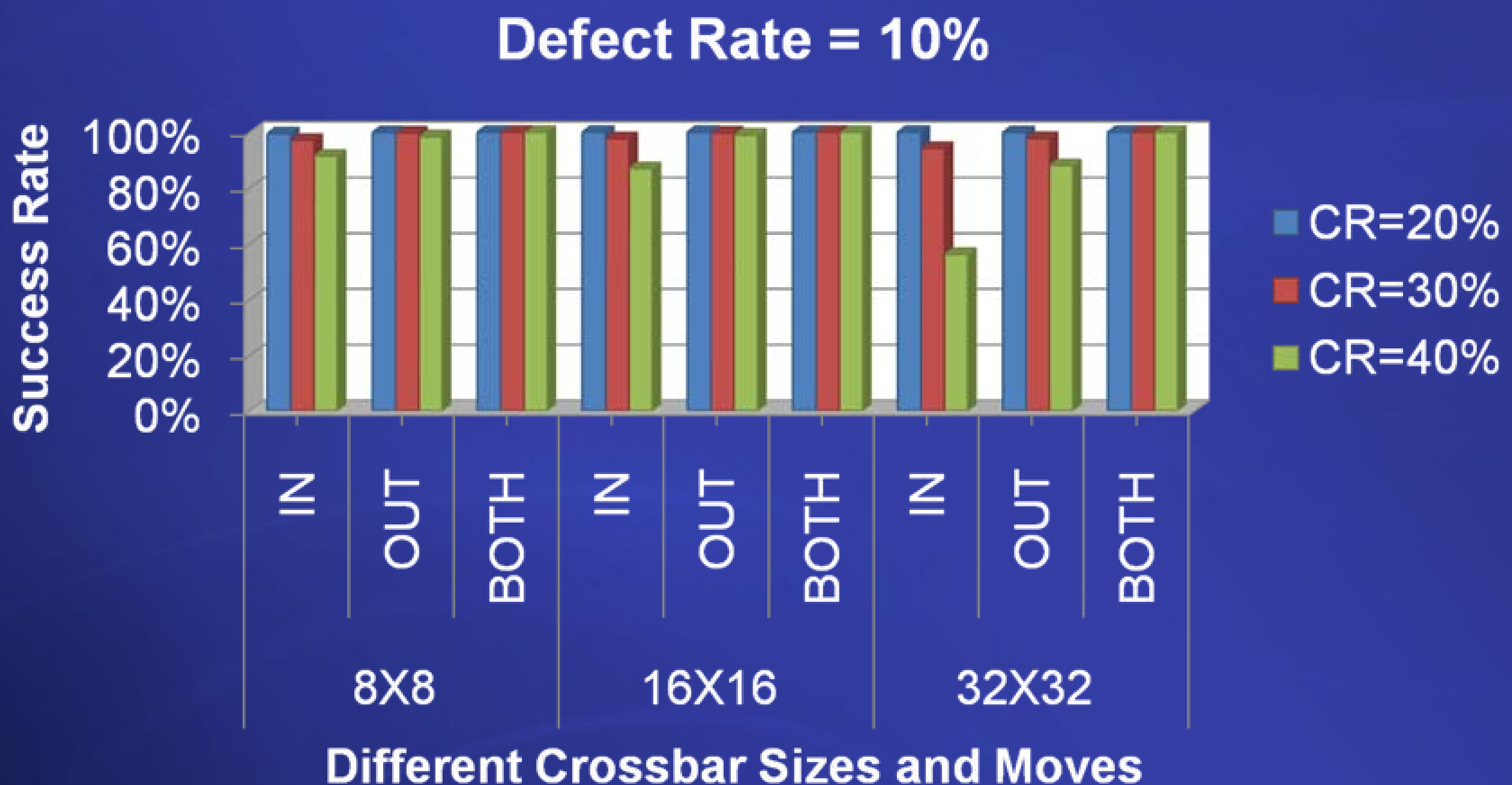
Experimental Results

- Defect free mapping success



Experimental Results

- Defect free mapping success



Outline

- Introduction and Motivation
- Definitions
- Algorithm
- Defect Tolerance
- Experimental Results
- Conclusion

Conclusion

- Low controllability in nanomanufacturing
  - High defect rate
  - Extreme parametric variation
- Variation Tolerant Logic Mapping technique
  - For FET-based crossbars
  - Programmability and interchangeability of crossbars
  - Formulated using Simulated Annealing
  - Efficient also for defect tolerance

Variation Tolerant Logic Mapping for Crossbar Array Nano Architectures

Cihan Tunc and Mehdi B. Tahoori

Karlsruhe Institute of Technology, Germany
Northeastern University, Boston, USA

Karlsruher Institut für Technologie

Northeastern University