International Test Conference
Charlotte, NC, Sep 30-Oct 2, 2003

Defect Tolerance at the End of the Roadmap

Mahim Mishra and Seth C. Goldstein

Carnegie Mellon University

Purpose

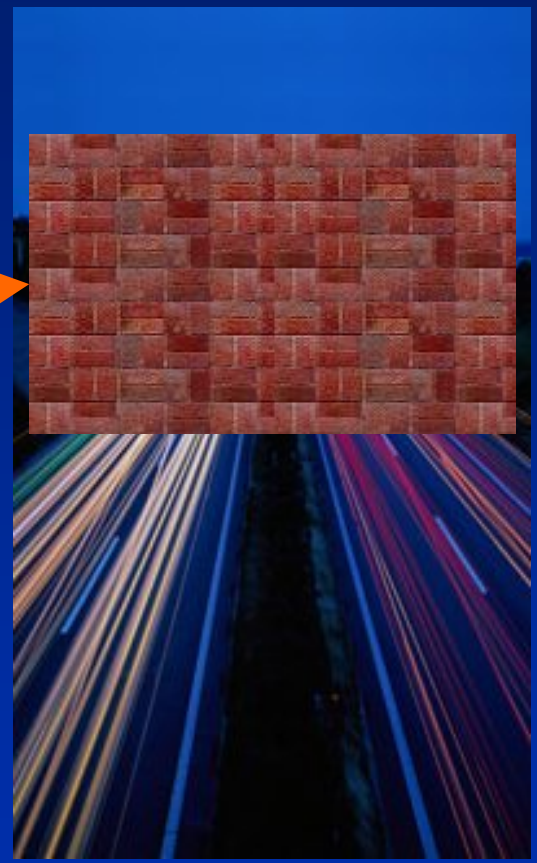
- Future technologies: EUV Lithography and Chemically Assembled Electronic Nanotechnology
  - Single-digit nanometer feature sizes
  - Extreme device densities
- Problem: much higher defect rates
  - Defect tolerance becomes key issue
- We outline a defect tolerance strategy
  - novel testing method

Talk Outline

- Introduction – need for defect tolerance
- Outline of defect tolerance strategy
  - Testing requirements
- Description of proposed test strategy
- Evaluation using simulations

Towards the end of the (ITRS) Roadmap

- Feature sizes approach single-digit nanometers
- Physical and economic limits to scaling



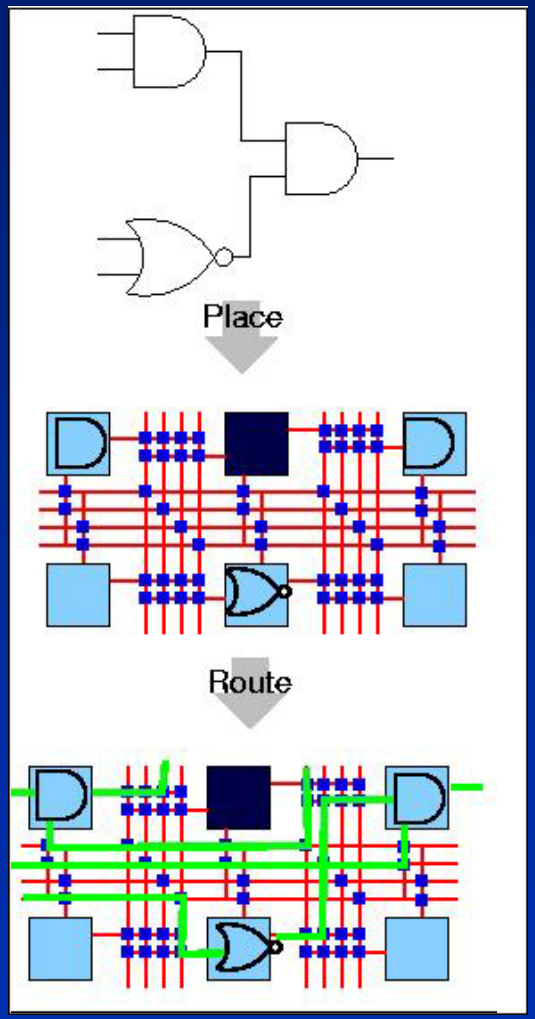
- New Technologies
  - Chemically Assembled electronic Nanotech. (CAEN)
  - Extreme Ultraviolet (EUV) Lithography

New technologies: caveats

- Extremely high defect densities
  - As high as 10% of fabric logic and routing resources
- Cannot throw away defective fabrics
  - Defect-free yield: close to 0%

- Must find a way to use defective fabrics

Part of the solution: reconfigurability



- **Regular, periodic**

  computing fabric
  - e.g., Field-Programmable Gate Arrays (FPGAs)

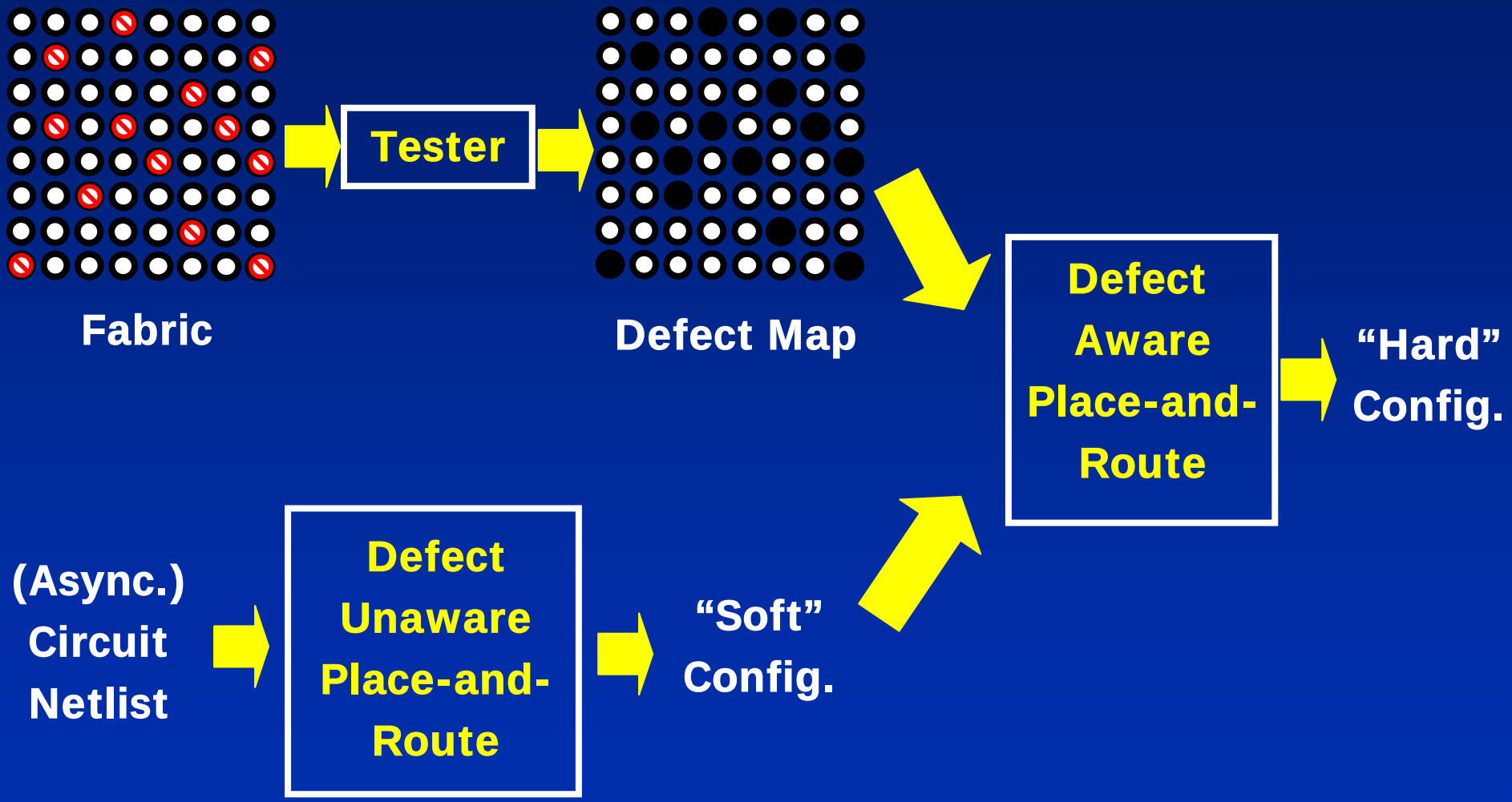
- User programs as **circuits**

- **Helps achieve defect tolerance**

New challenges for defect tolerance

- New **testing** techniques to locate all the defects
  - generate a ***defect map***
- New, quick, **place-and-route** algorithms
  - utilize the defect map
- Must **scale** with fabric size and number of defects

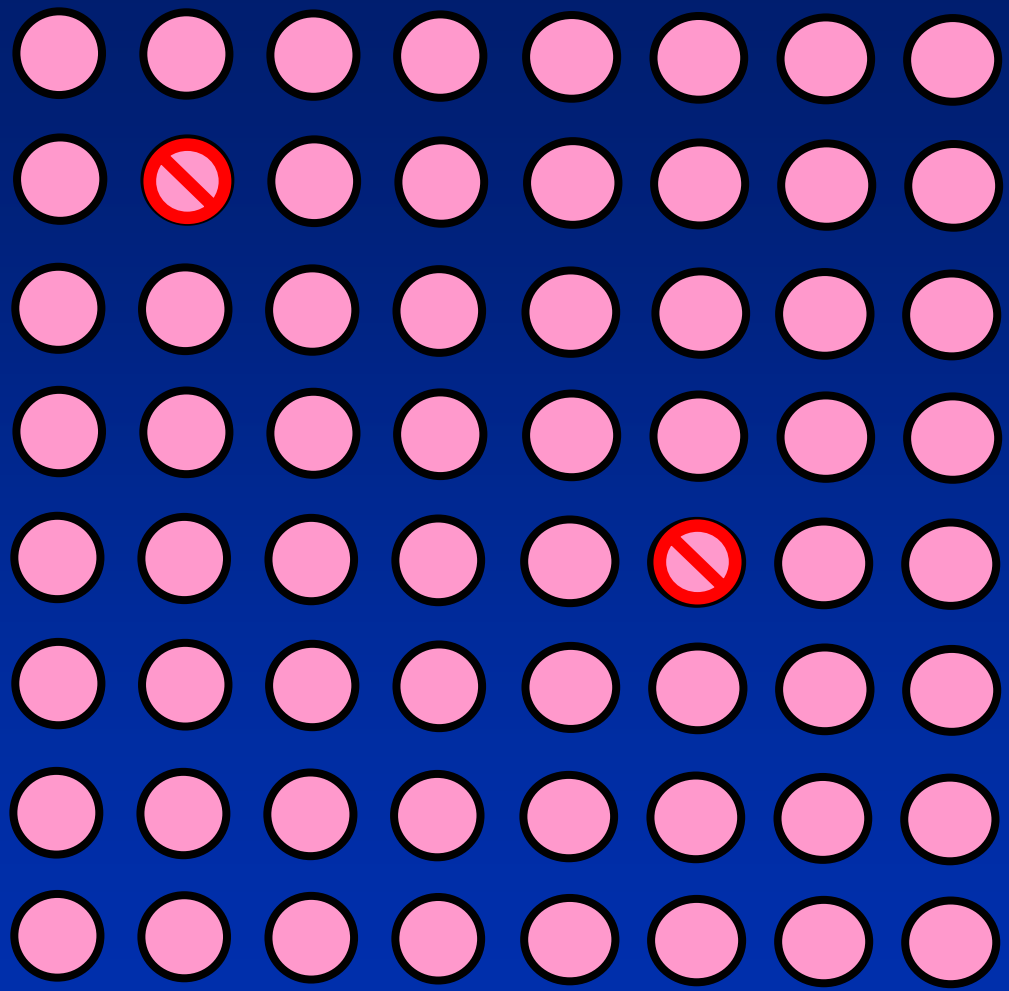
Proposed tool flow



Testing to locate defects

- **Required**: scalable testing method to locate defects in large reconf fabrics
  - Capable of dealing with large defect densities
  - Quick
- Very different from previous FPGA testing methods
  - Using defective chips: not a goal so far
  - Similar approach: ***Teramac*** custom computer at HP
- **Goal of this work**
  - Show that such a testing method is possible
  - Develop some of the new, smart techniques required

Testing: previous methods



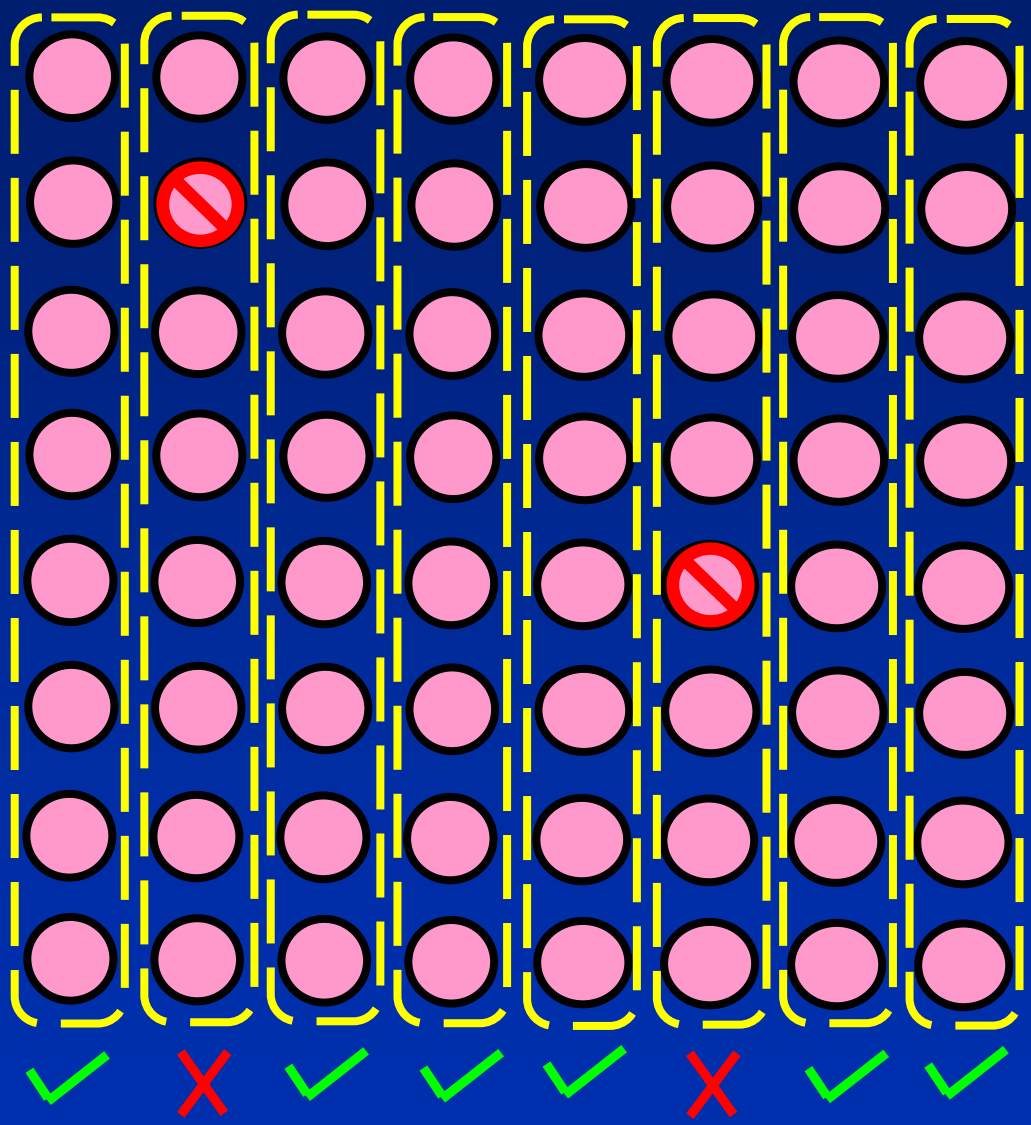
Defect-free, unknown

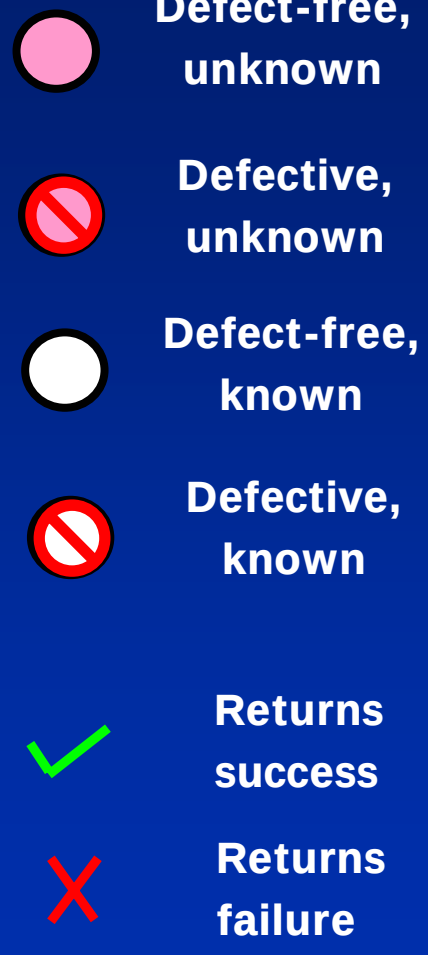
Defective, unknown

Defect-free, known

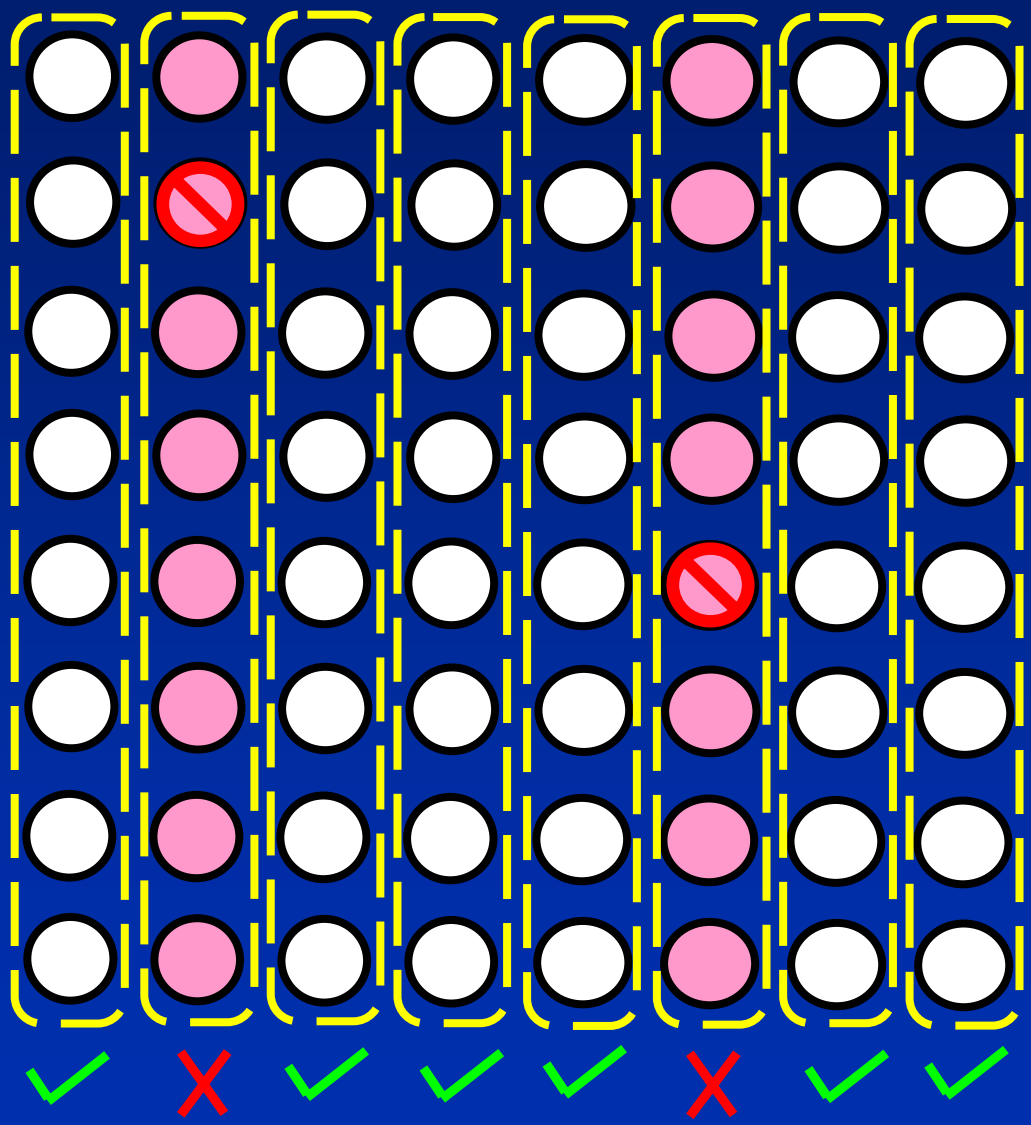
Defective, known

Testing: previous methods





Testing: previous methods



Defect-free, unknown

Defective, unknown

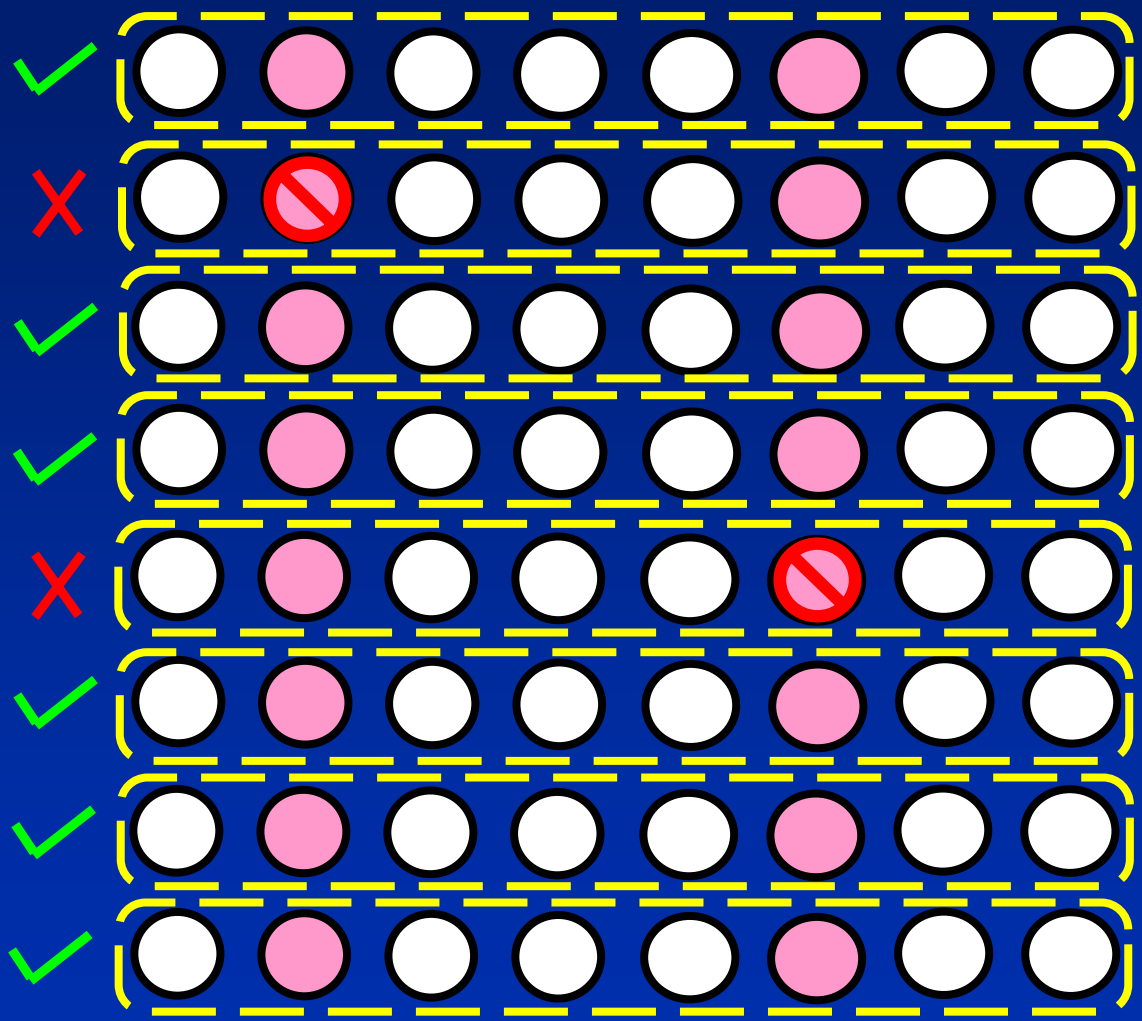
Defect-free, known

Defective, known

Returns success

Returns failure

Testing: previous methods



Defect-free, unknown

Defective, unknown

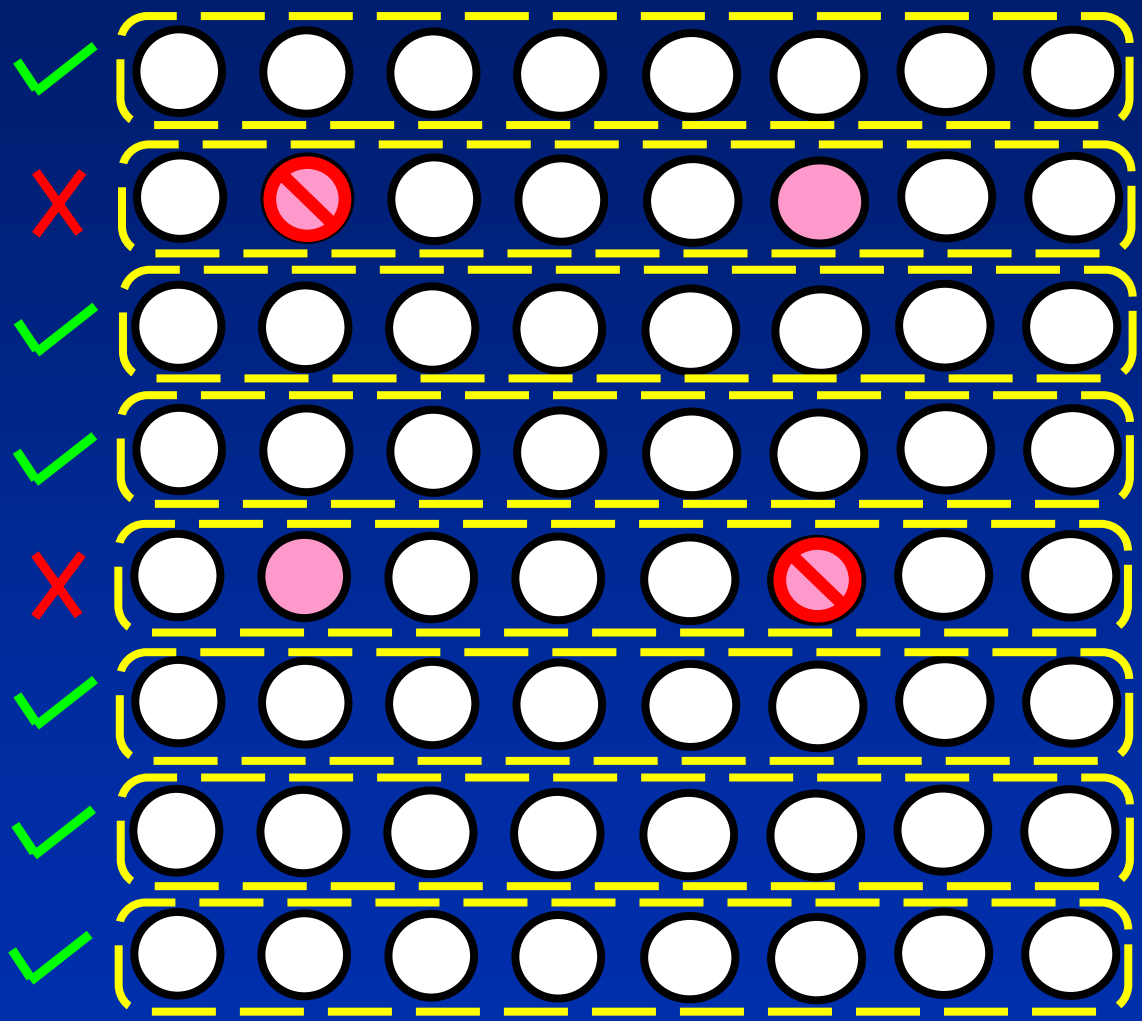
Defect-free, known

Defective, known

Returns success

Returns failure

Testing: previous methods



Defect-free, unknown

Defective, unknown

Defect-free, known

Defective, known

Returns success

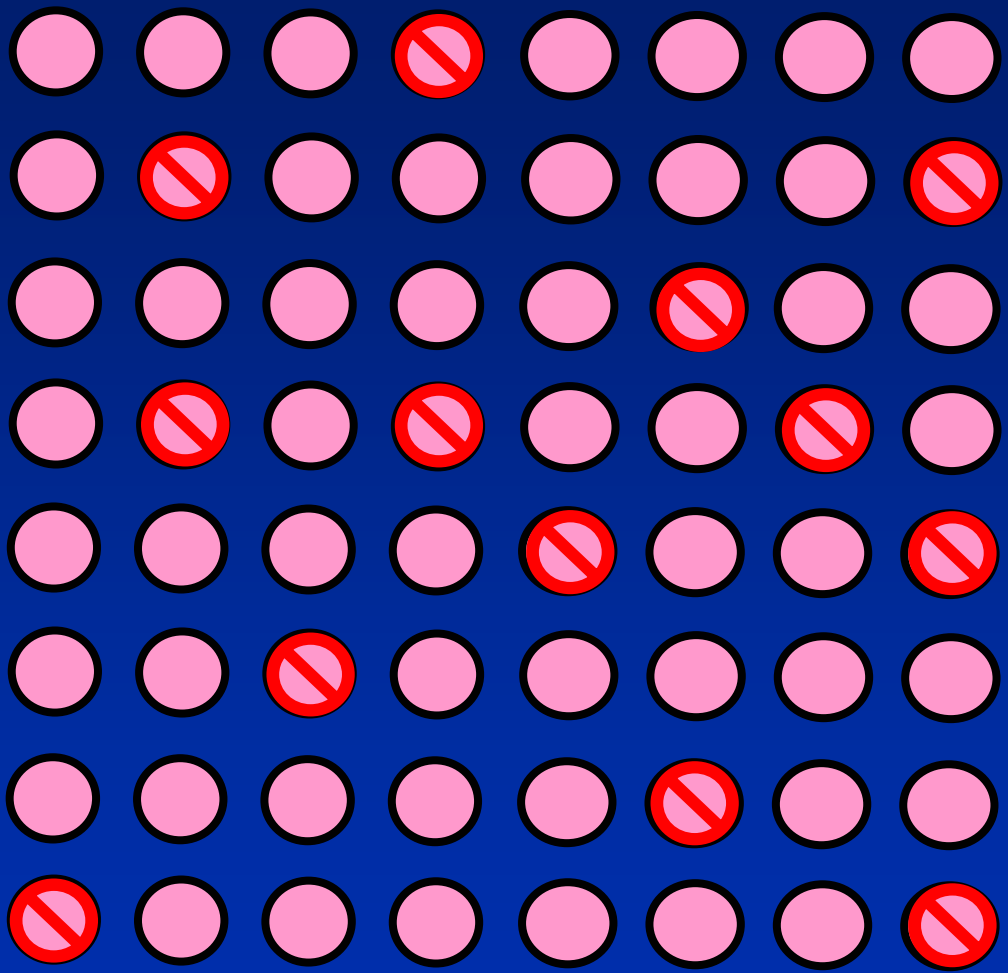
Returns failure

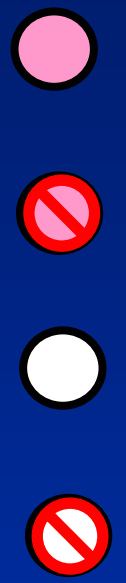
Testing with high defect rates

- Previous method: works for low defect rates
  - Uses **“binary”** circuits
  - Requires significant number of defect-free test-circuits

- Will not work for high defect rates
  - Each test circuit has multiple defects
  - Very, very few circuits with 0 defects

Testing: high defect rates





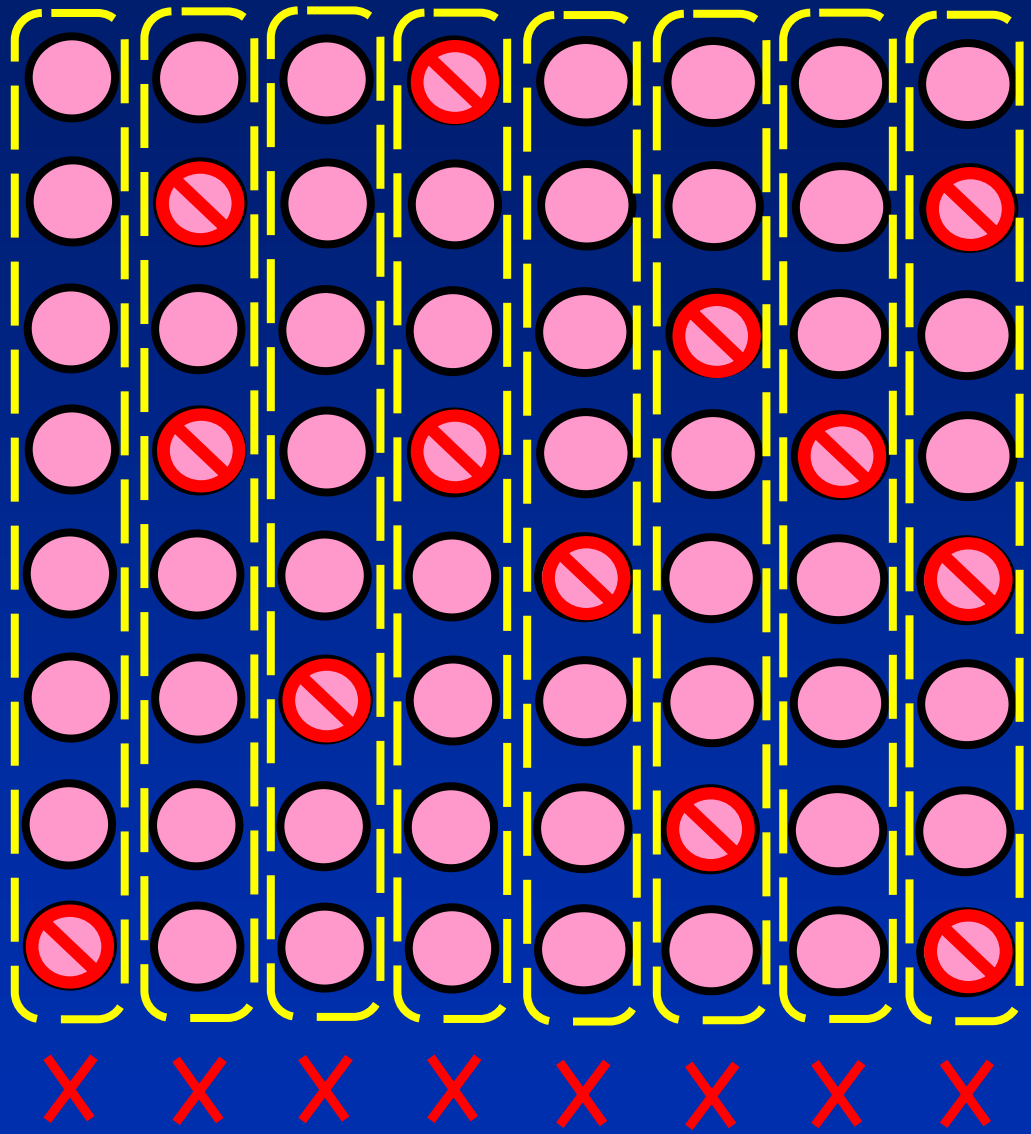
Defect-free, unknown

Defective, unknown

Defect-free, known

Defective, known

Testing: high defect rates



Defect-free, unknown

Defective, unknown

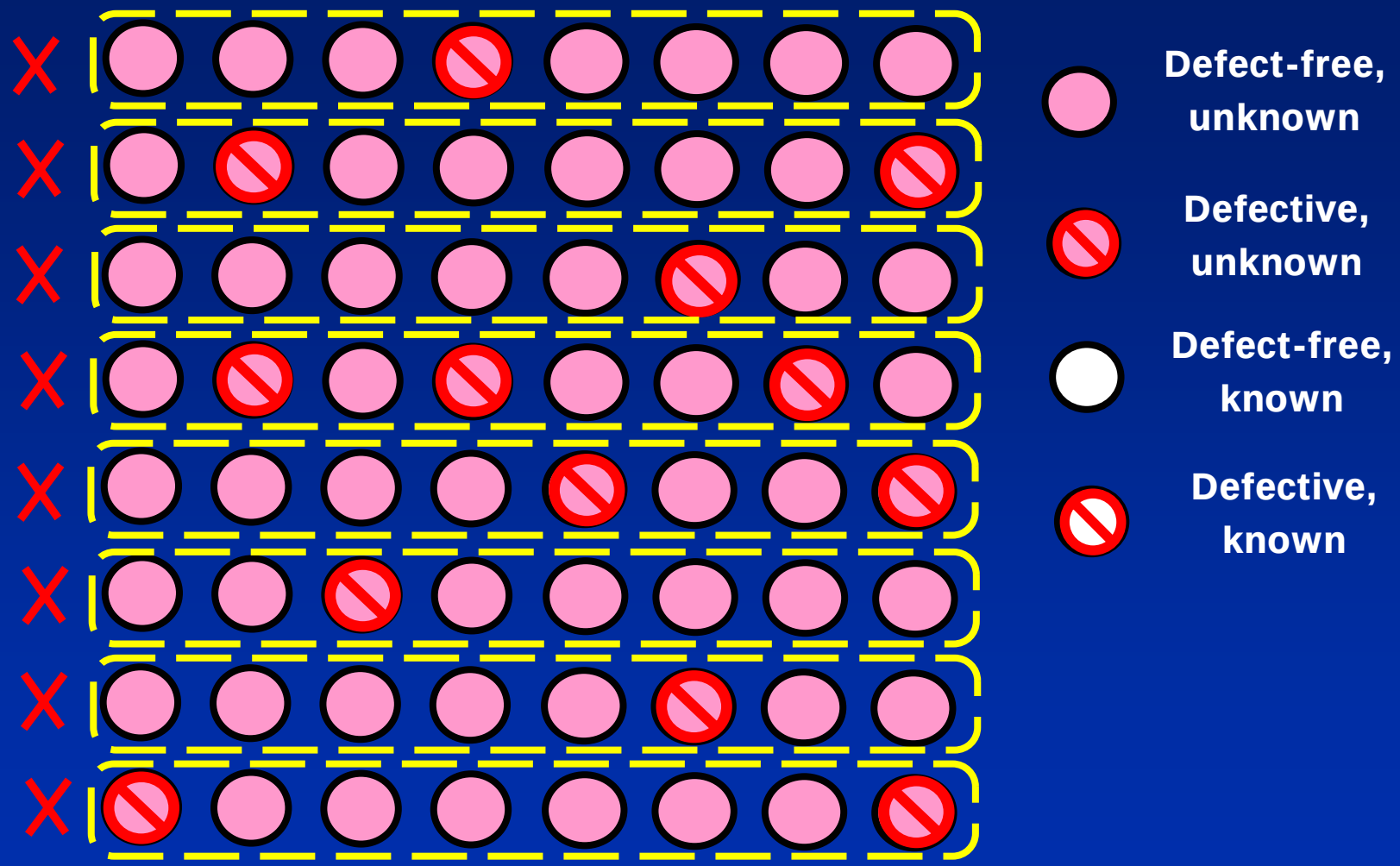
Defect-free, known

Defective, known

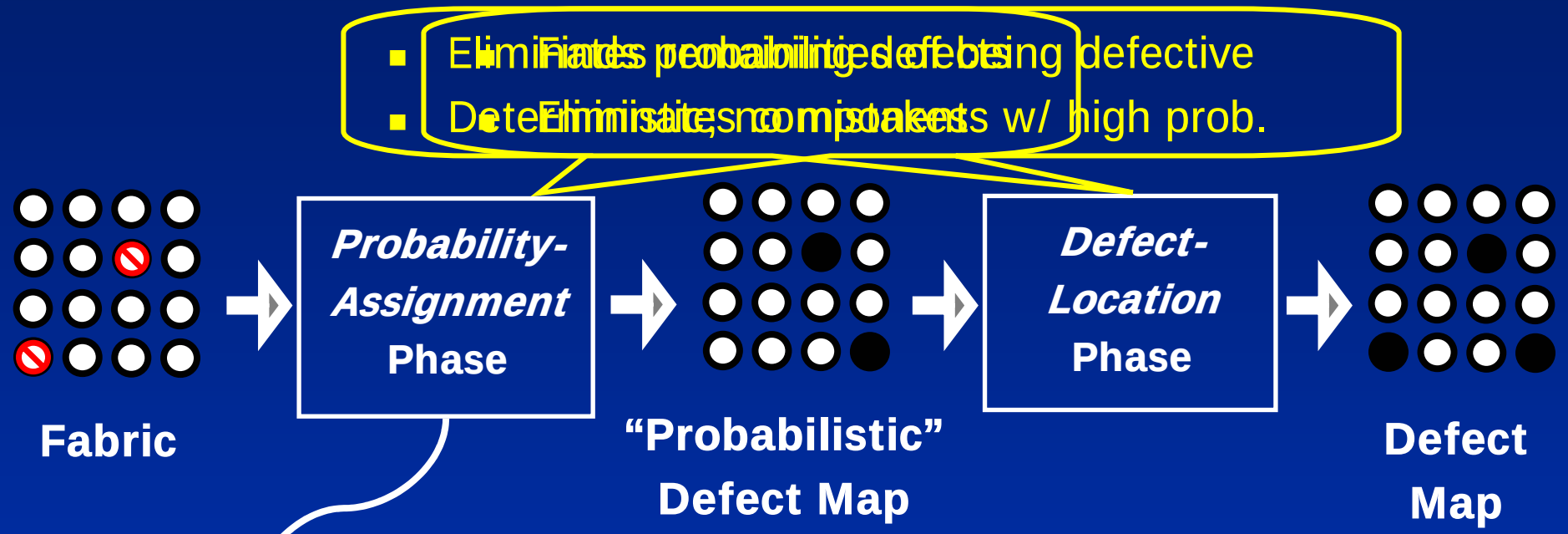
Returns success

Returns failure

Testing: high defect rates



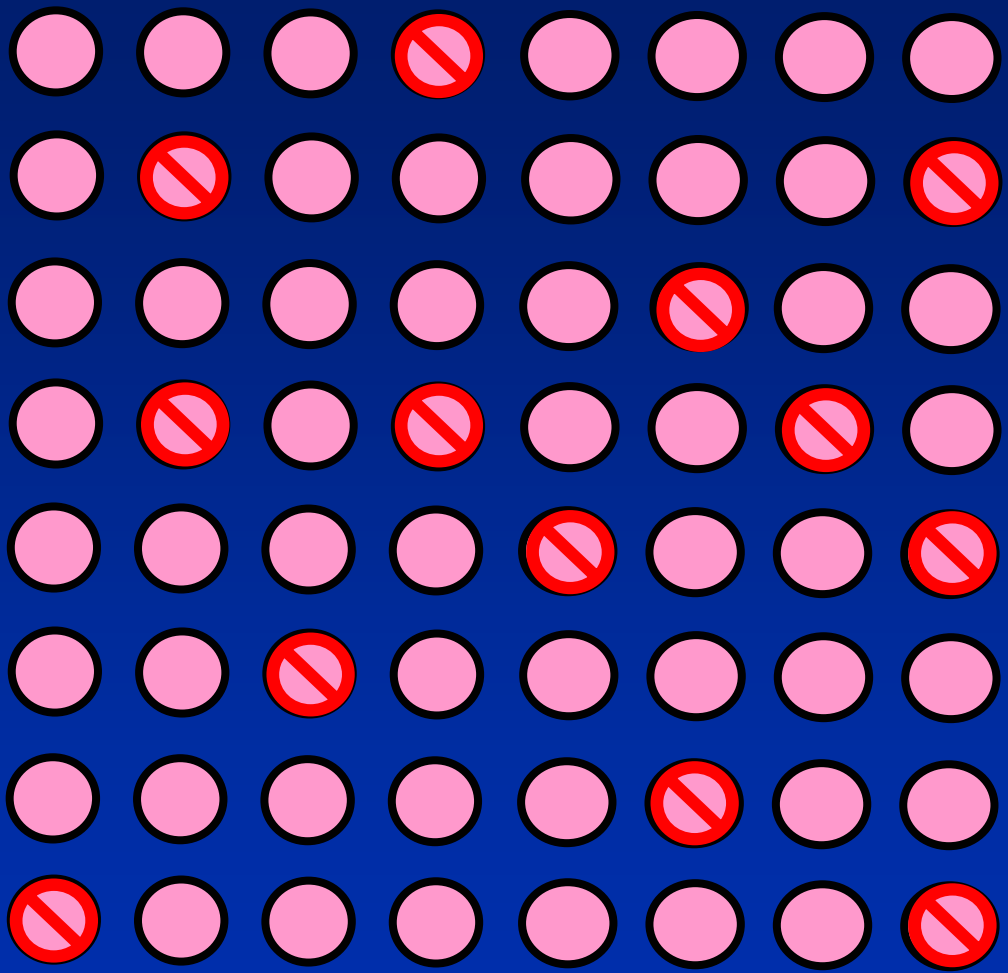
Dealing with high defect rates: our algorithm



2 key ideas:

- More **powerful test-circuits**
  - More than binary info; e.g. approximate defect counts
- More **powerful analysis techniques**

Probability assignment: example



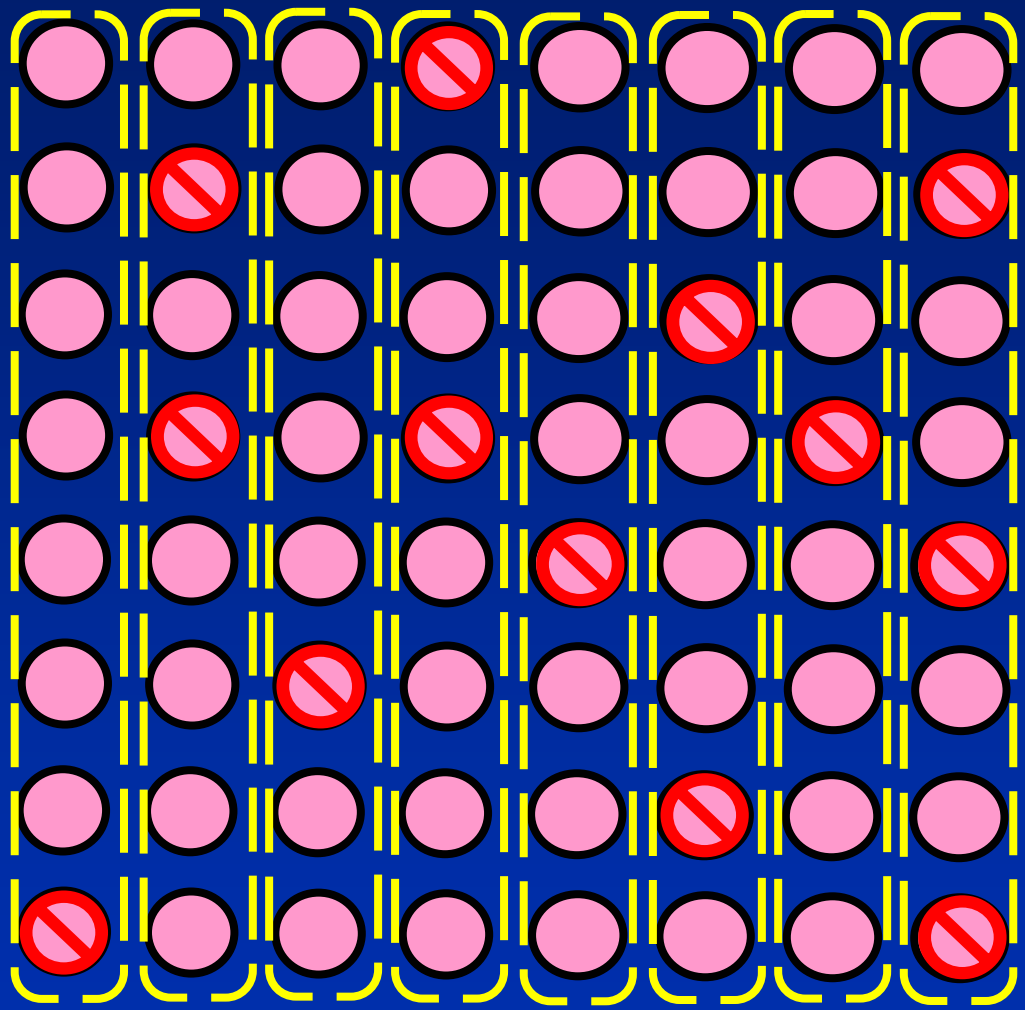
Defect-free, unknown

Defective, unknown

Defect-free, known

Defective, known

Probability assignment: example



Defect-free, unknown

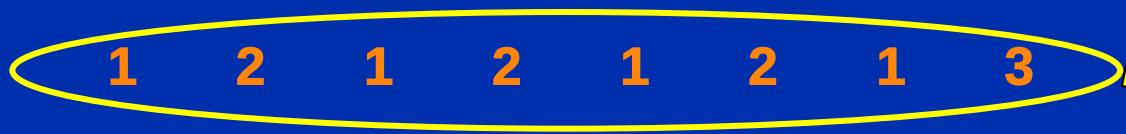
Defective, unknown

Defect-free, known

Defective, known

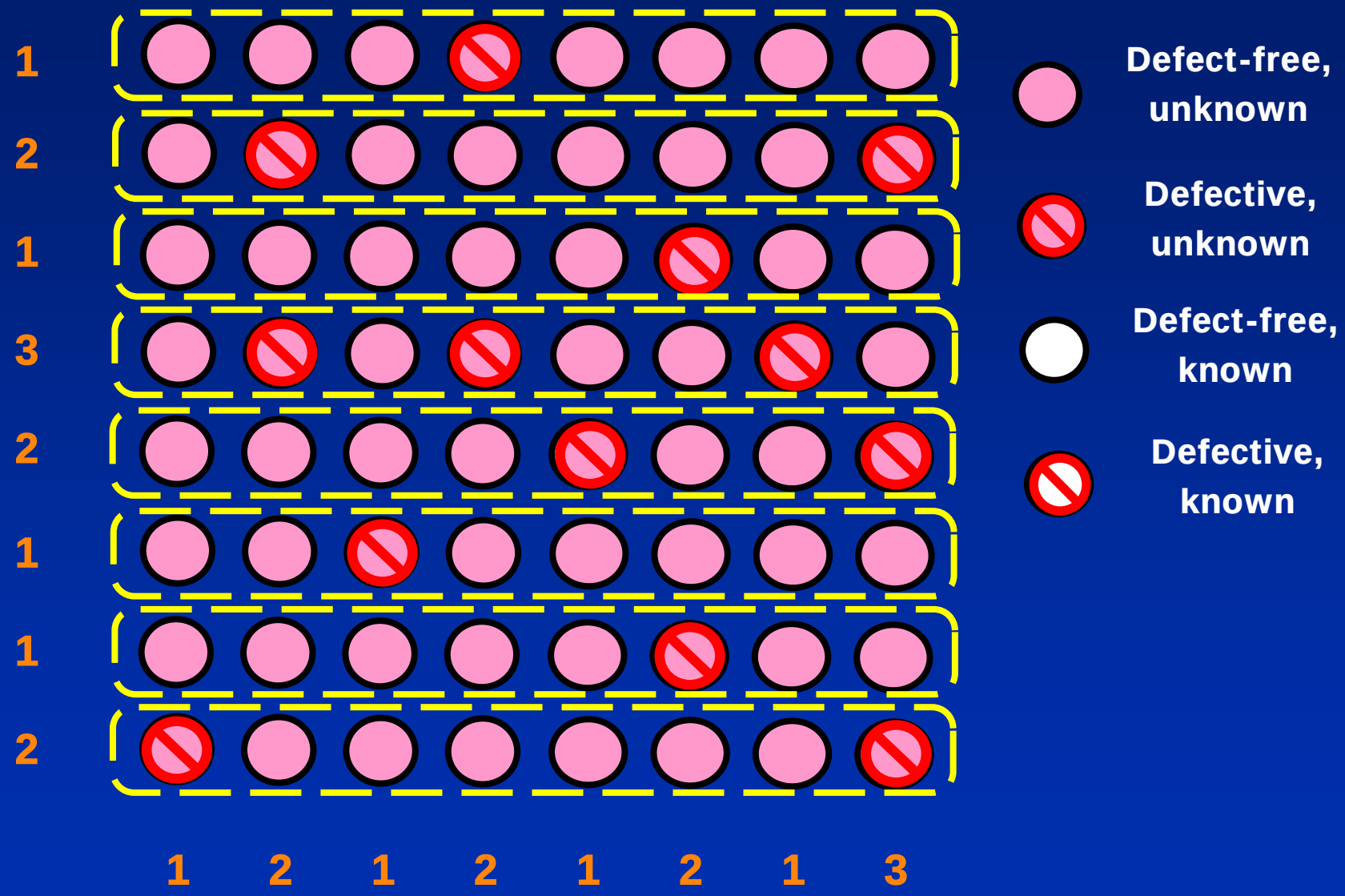
Test circuits:

- ***Counter***
- ***None-some-many***

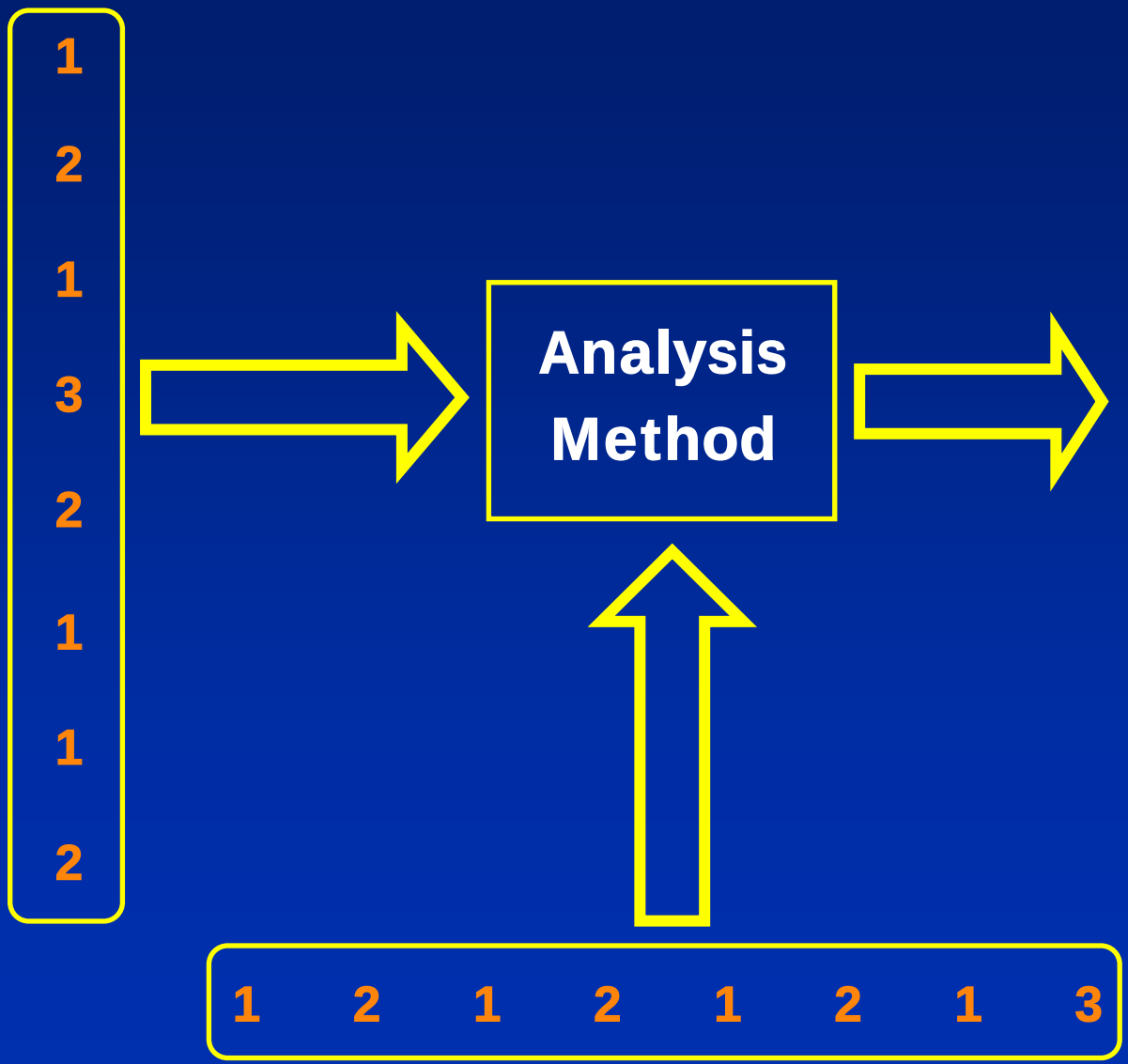


More than binary information!

Probability assignment: example



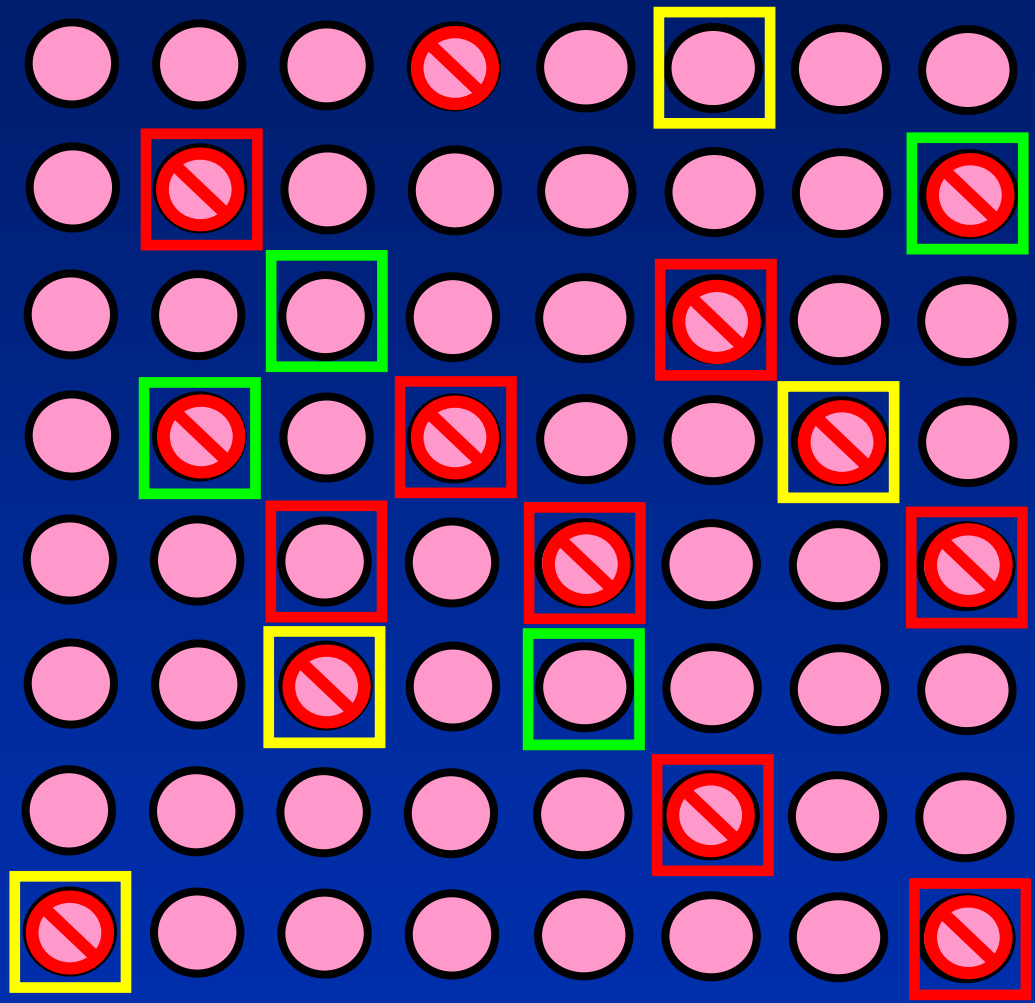
Probability assignment: example



Analysis methods:

- ***Sorting***
- ***Bayesian***

Probability assignment: example

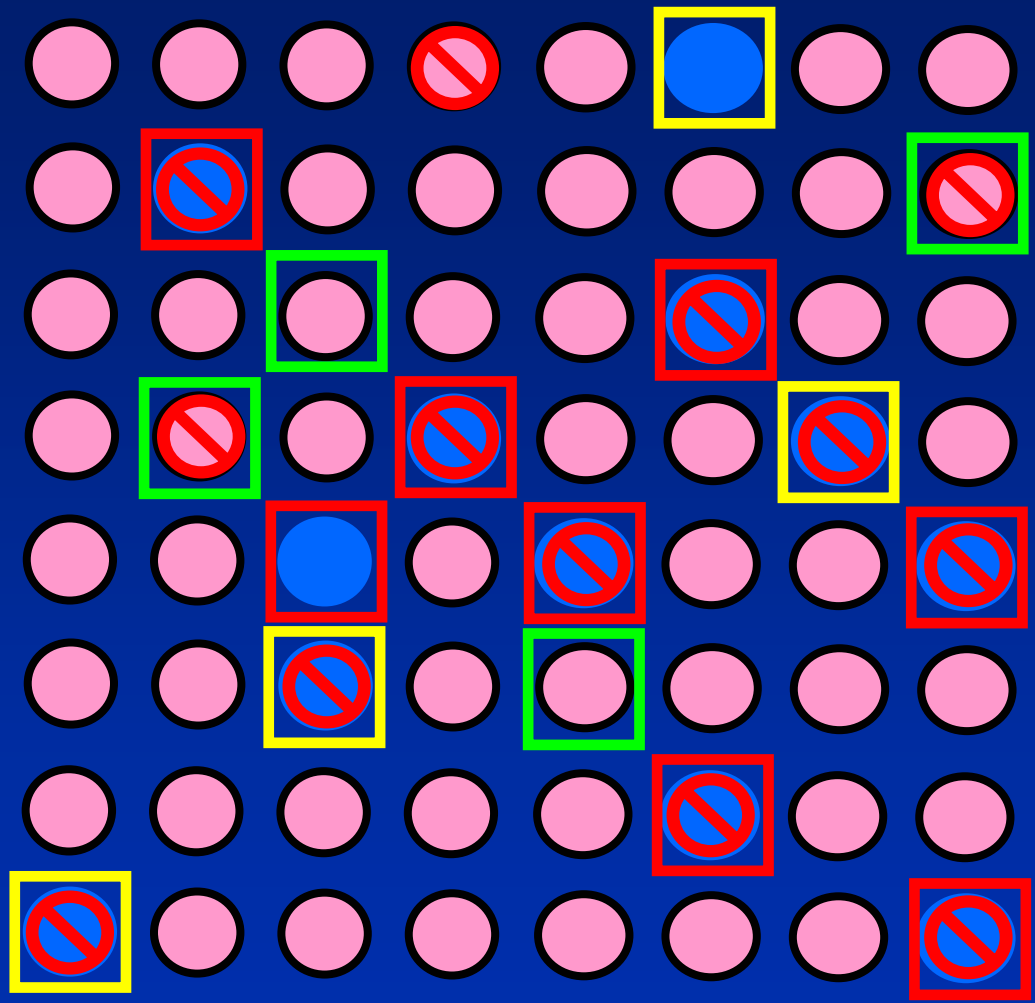


Defect Probabilities

higher

lower

Probability assignment: example

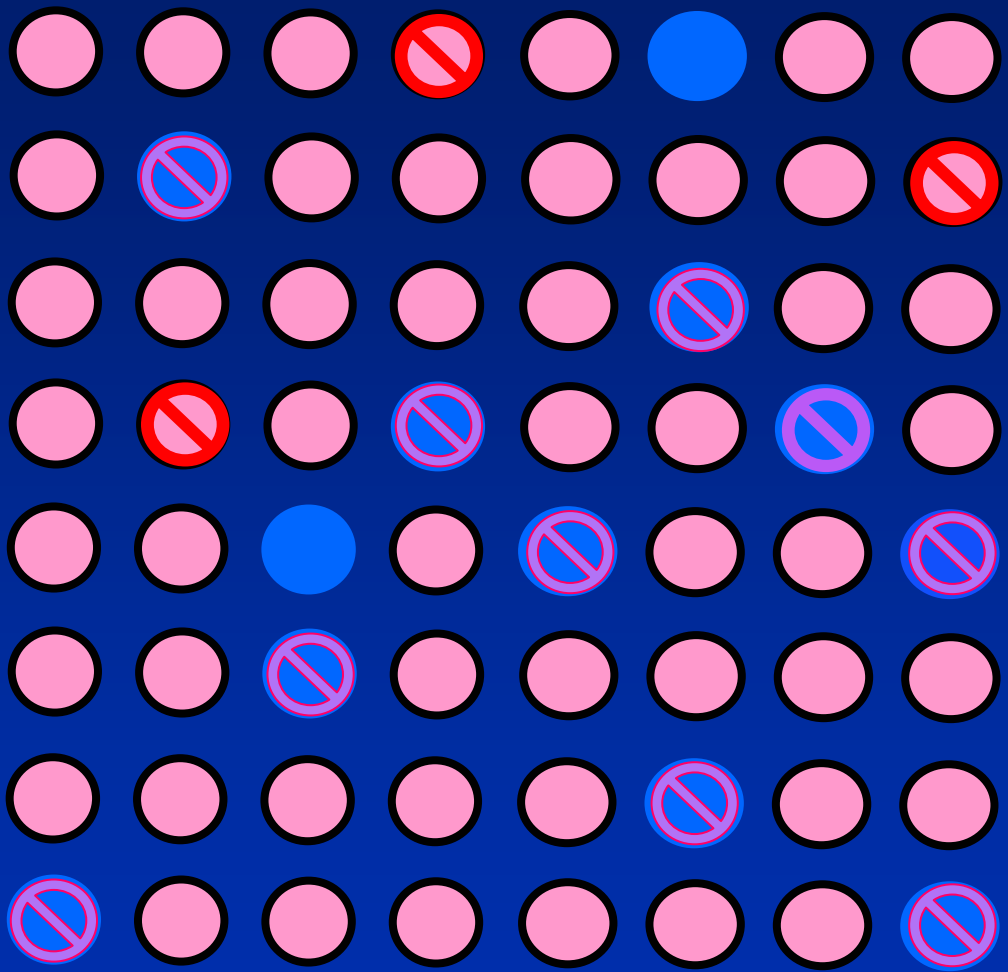


Removed:

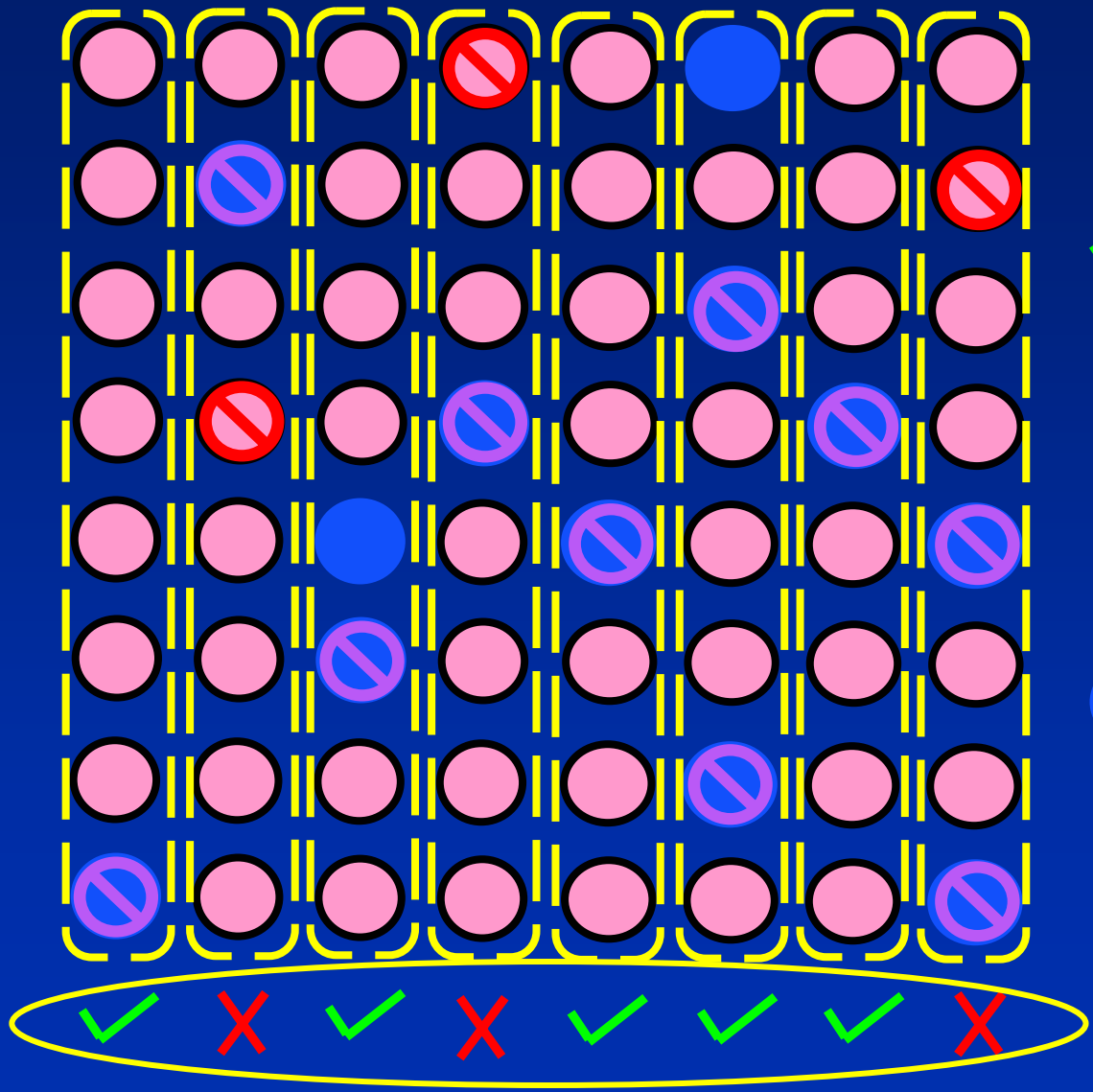
&

Assumed Defective

Probability assignment: example



Defect location: example



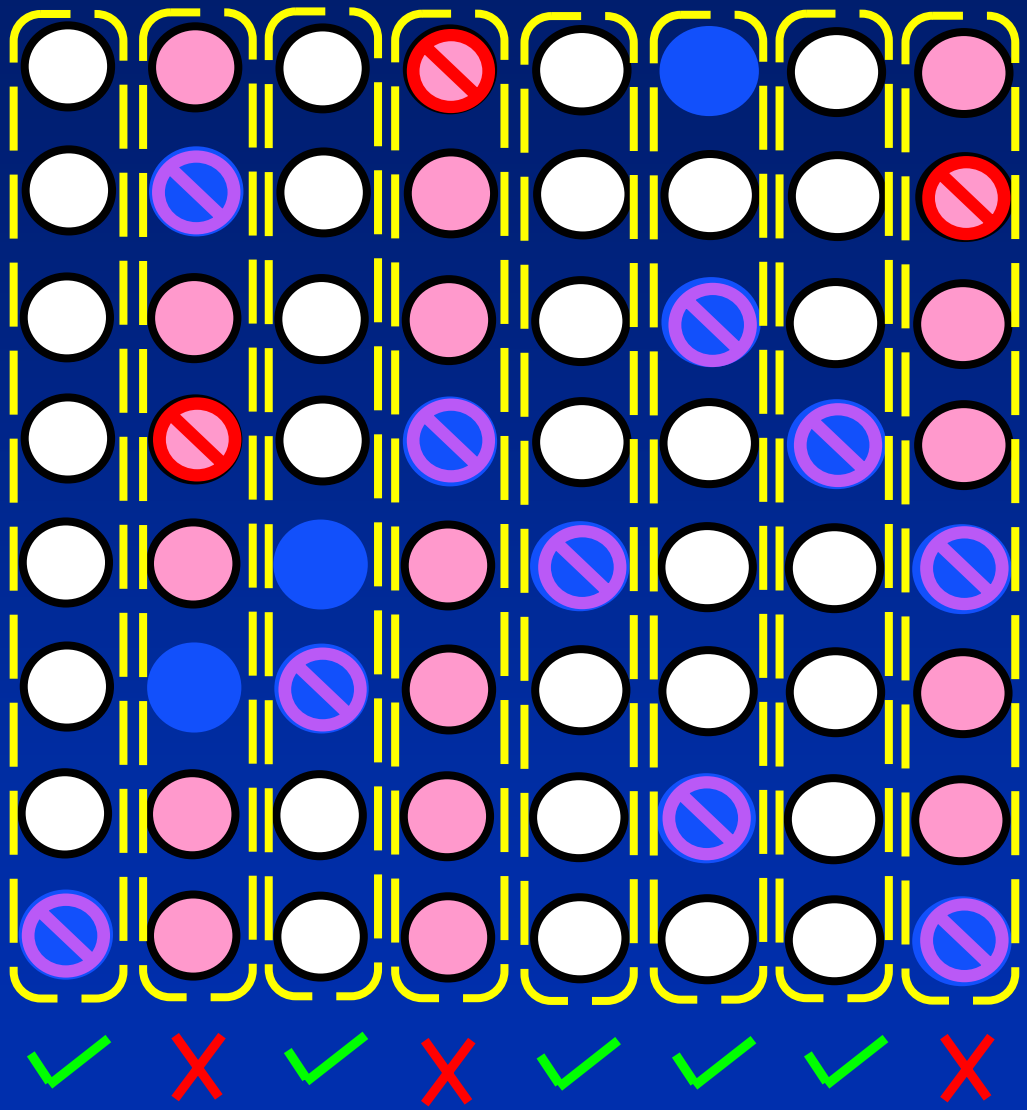
Returns success

Returns failure

Assumed Defective

Binary Information

Defect location: example

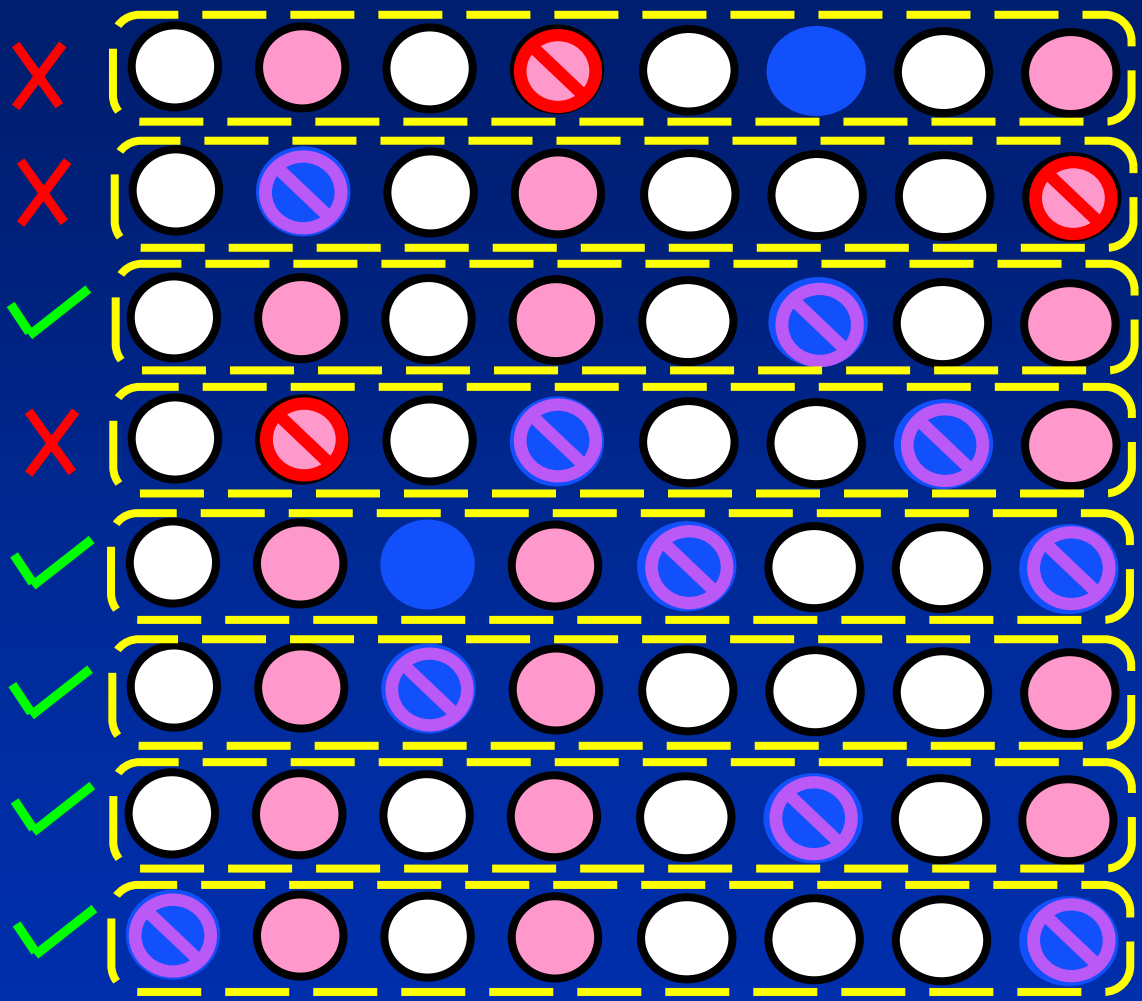


Returns success

Returns failure

Assumed Defective

Defect location: example

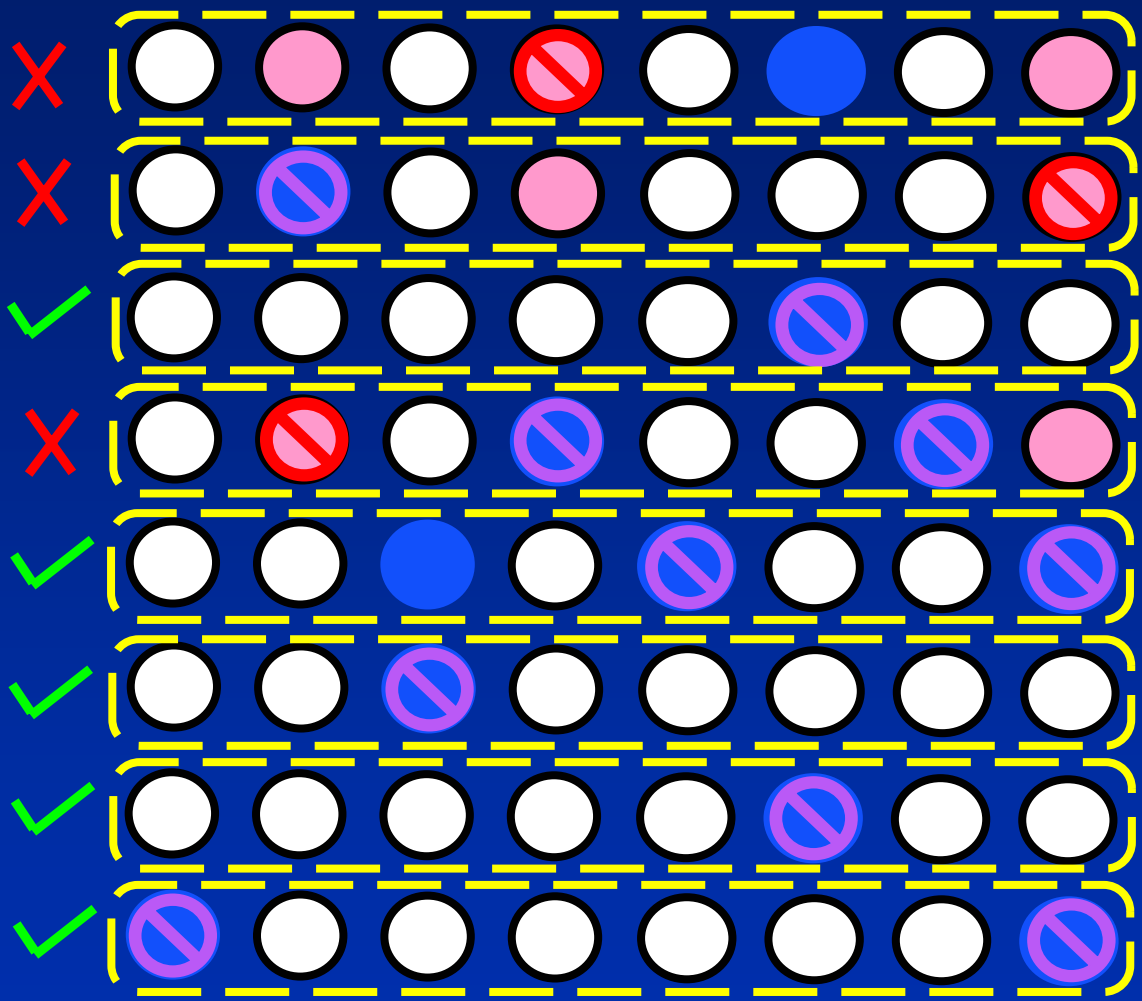


Returns success

Returns failure

Assumed Defective

Defect location: example

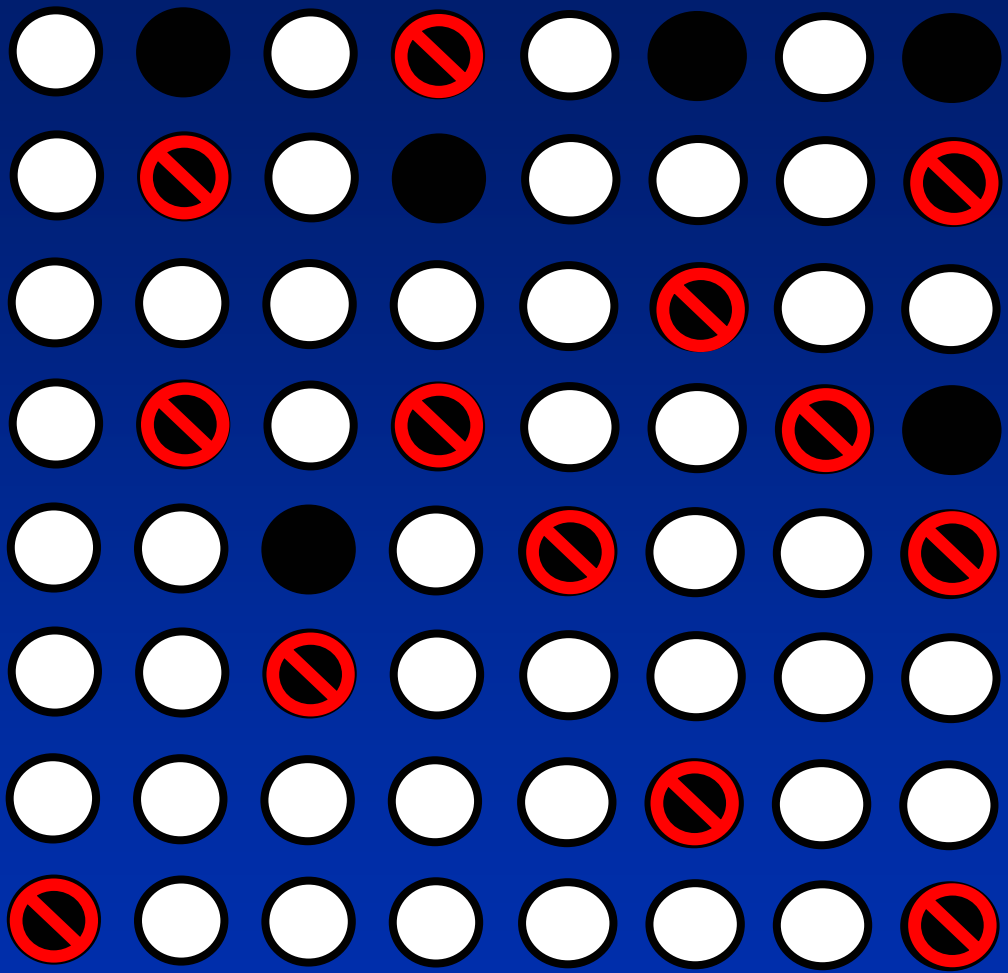


Returns success

Returns failure

Assumed Defective

Final defect map

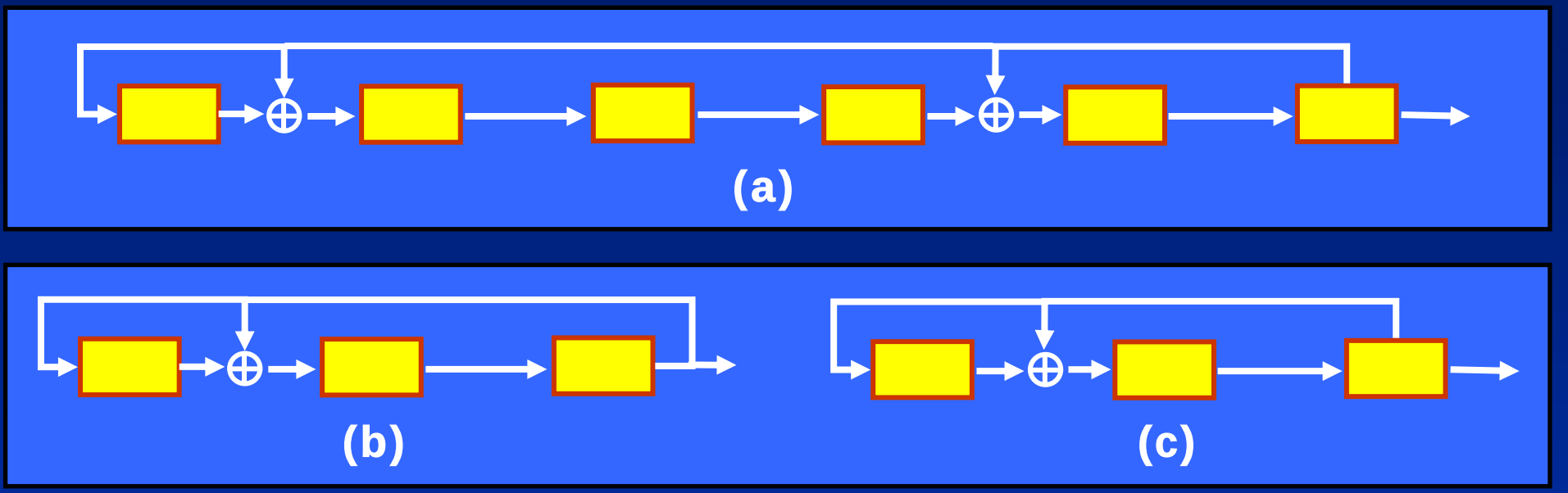


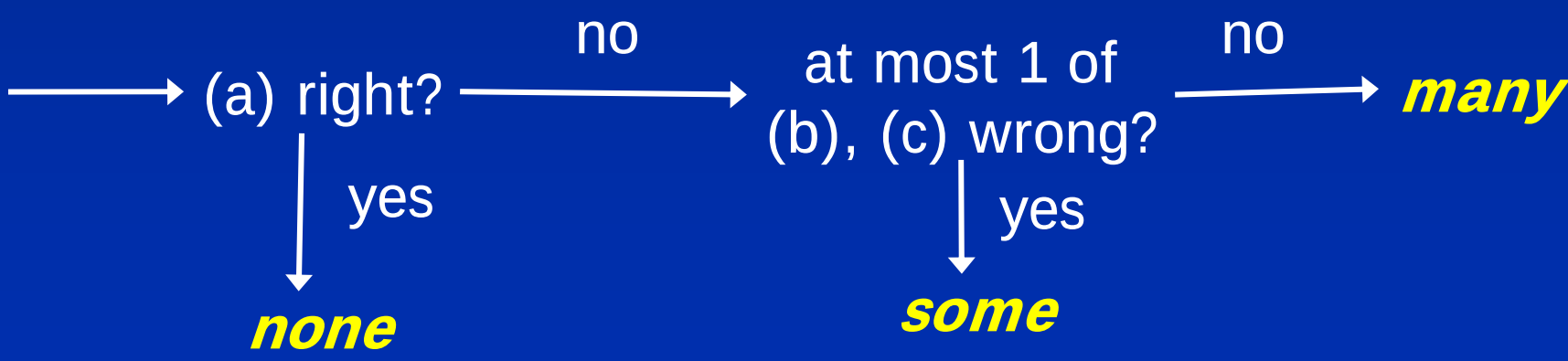
Test circuits for *prob. assignment*

- Idealized ***counter*** circuits
  - Conceptual circuits
  - Return defect counts, upto threshold *t*
  - Higher threshold $\Rightarrow$ more powerful circuit

- ***None-some-many*** circuits
  - Tell if ***none***, ***some*** or ***many*** defects
  - Less powerful than counters, easier to realize
  - e.g., our LFSR-based design

None-some-many circuits



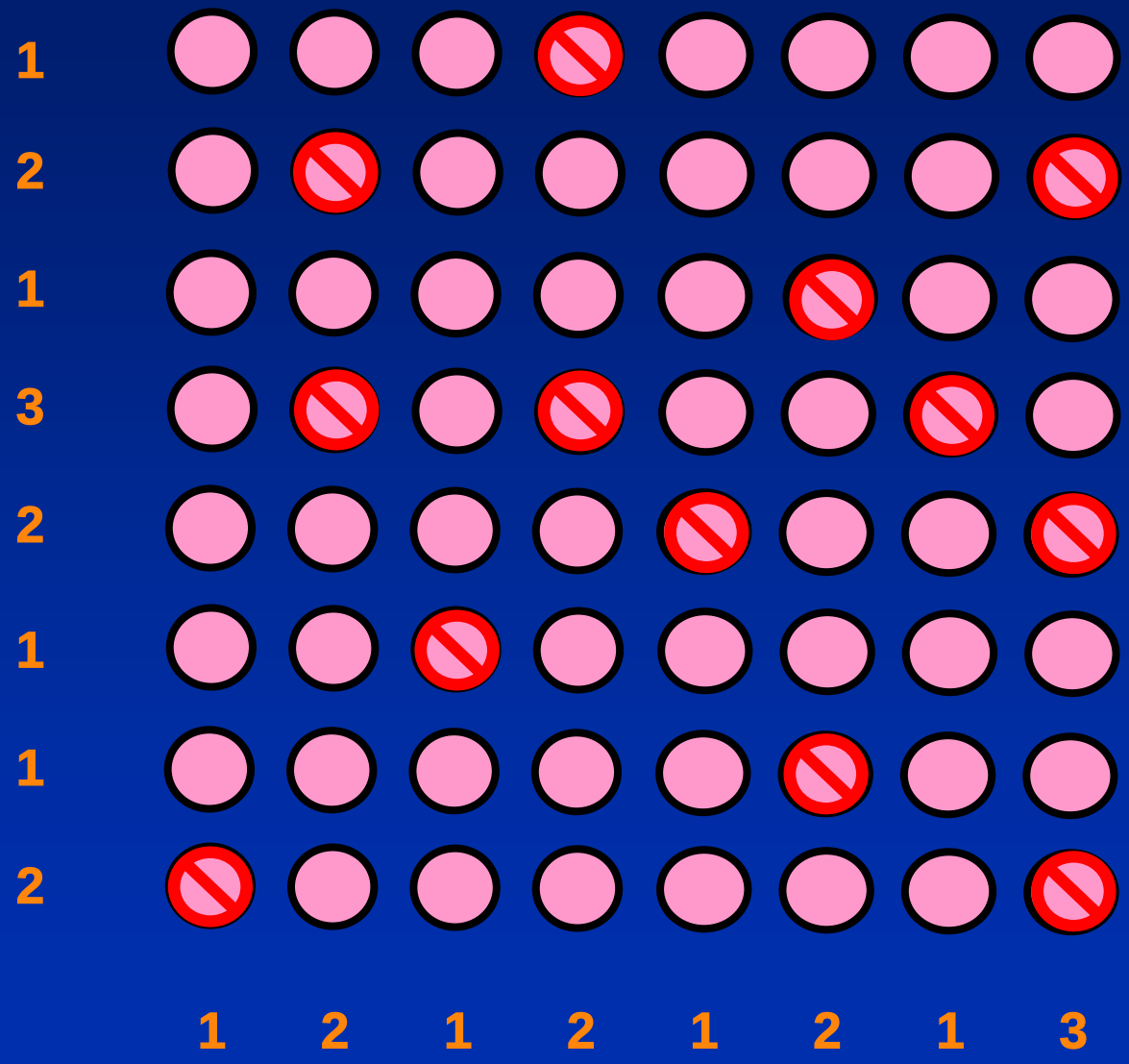


Analysis methods

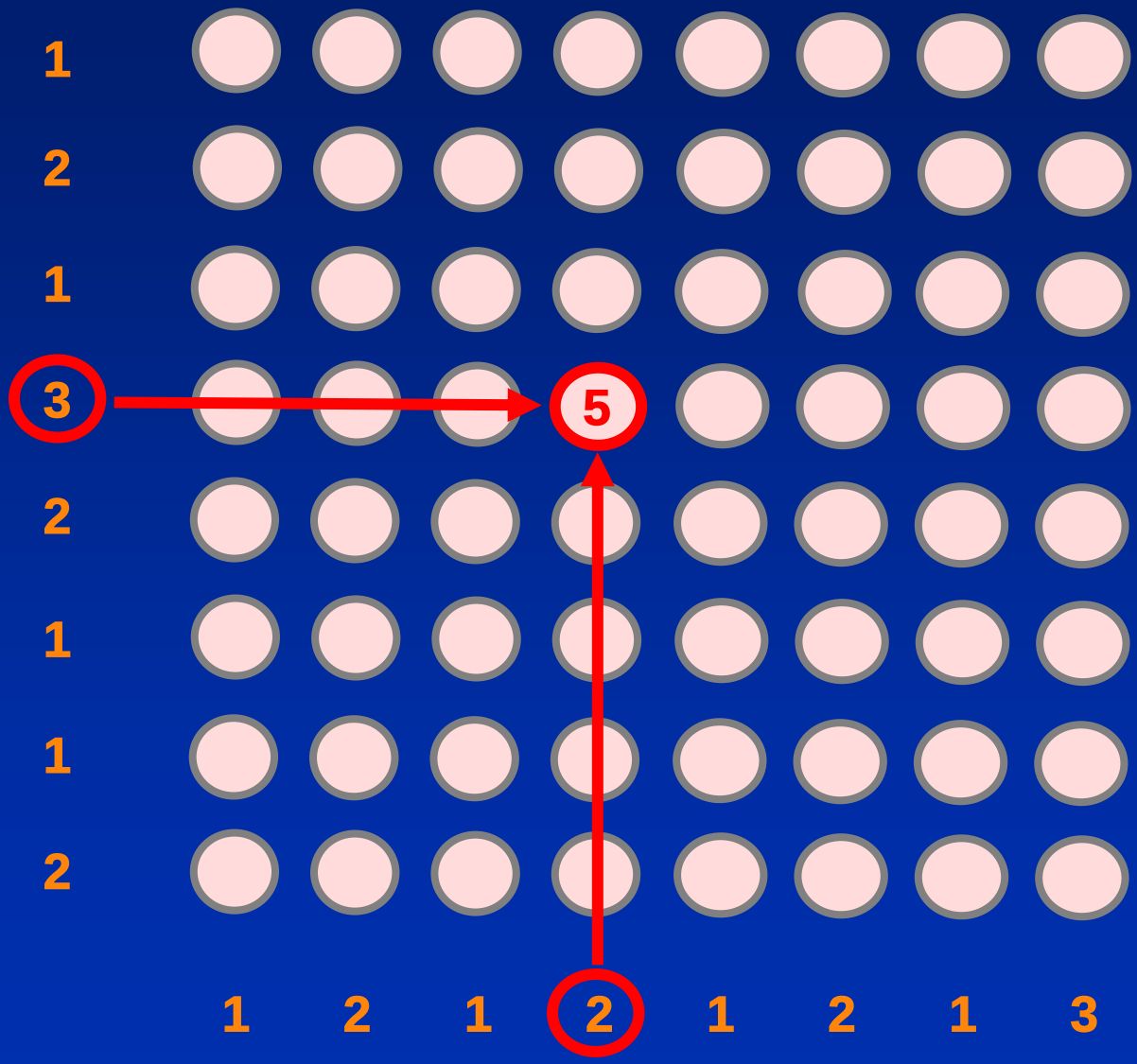
- ***Sorting*** analysis
  - Example

- ***Bayesian*** analysis

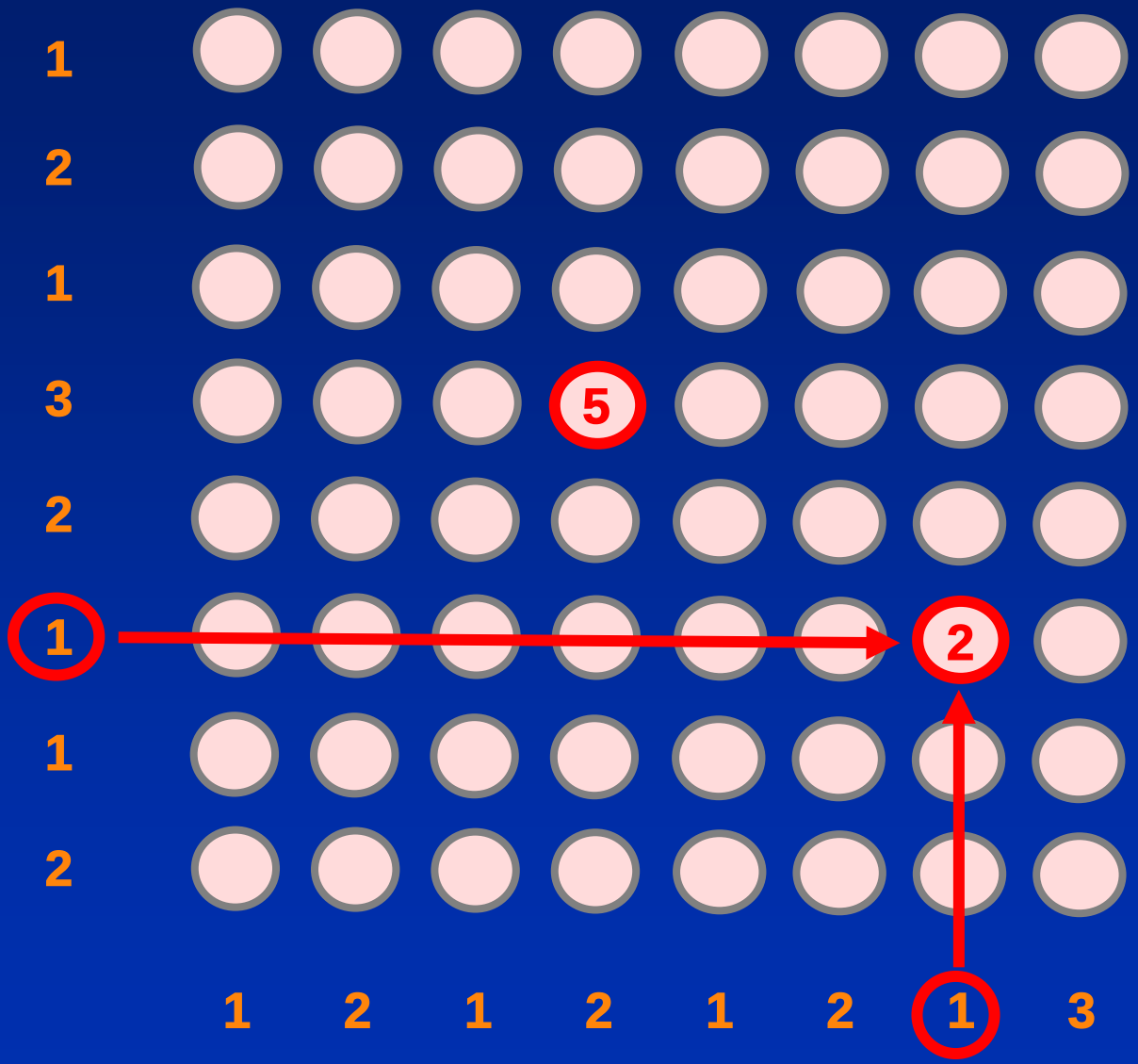
Analysis methods: *Sorting* analysis



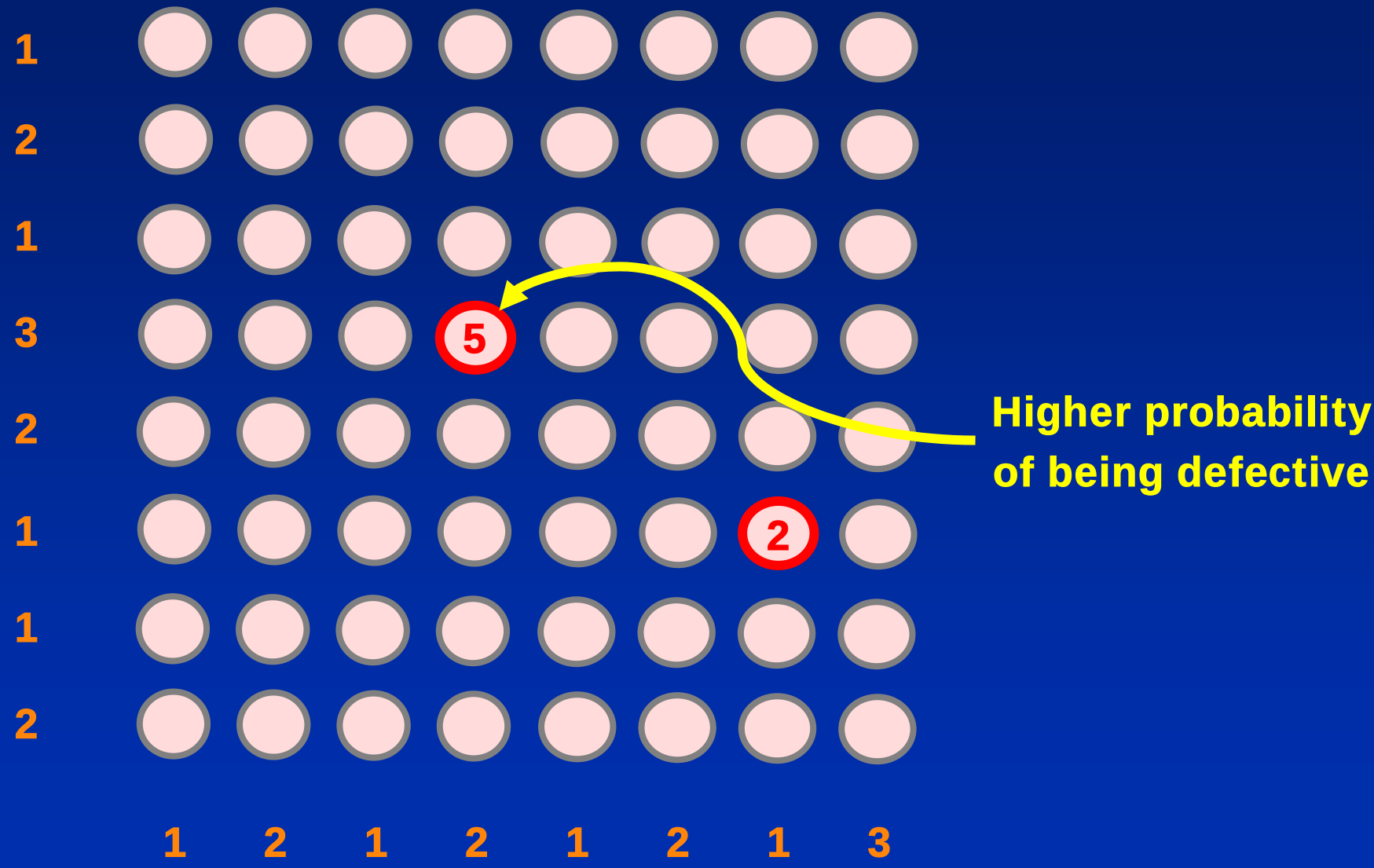
Analysis methods: *Sorting* analysis



Analysis methods: *Sorting* analysis



Analysis methods: *Sorting* analysis



Analysis methods: comparison

- Ease of implementation
  - Bayesian: harder to implement (restricted circuits)
  - Sorting: no restrictions
- Complexity
  - Bayesian: $O(n^2)$ best case
  - Sorting: $O(n \log n)$
- Quality of results: ~10% better for Bayesian

Algorithm: discussion

- ***Minimally-adaptive*** algorithm
  - Minimal rerouting required at test time
- No **false negatives**
  - After defect-location phase, all defects identified
- **Algorithm complexity**:
  - circuit size k
  - defect rate p
  - $k \times k$ fabric
  - requires $O(kp)$ test-circuit orientations

Evaluation

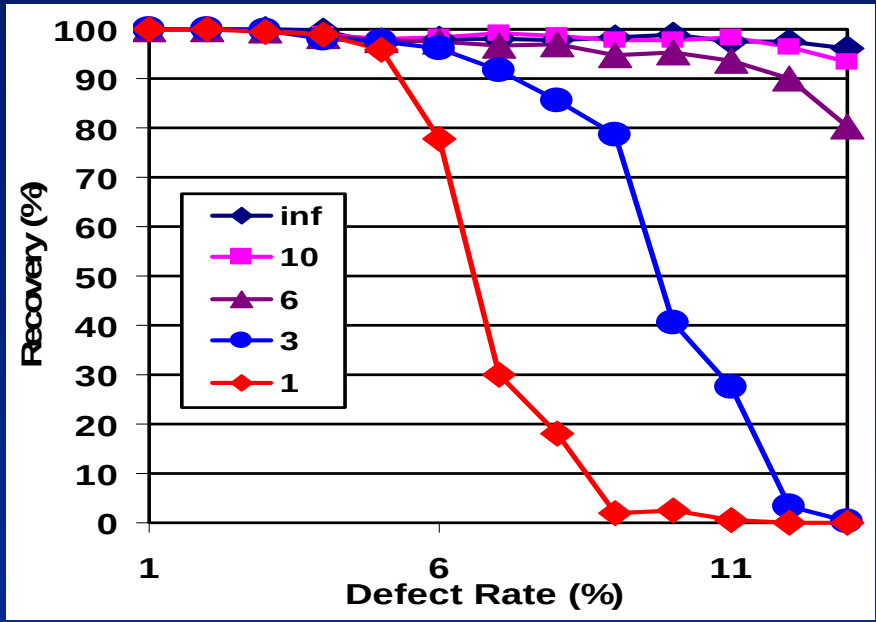
- Quality metric: ***recovery***
  - percentage of defect-free components marked *not defective*
- Each simulated test circuit: 100 components
- Simulated defect rates: 1 to 13%
  - 1 to 13 defects on average per test circuit
  - Results valid for circuits with this many defects

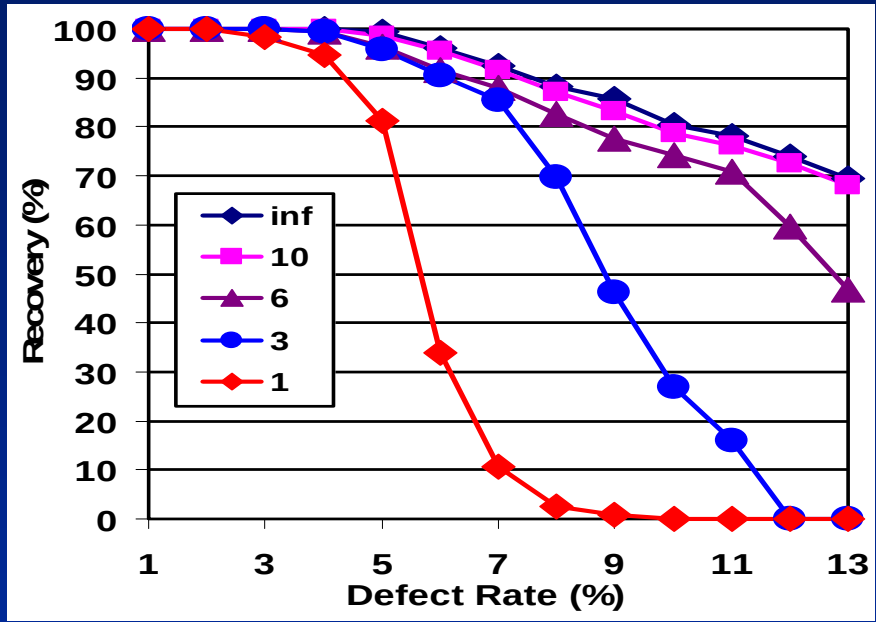
Evaluation results: comparison

Bayesian

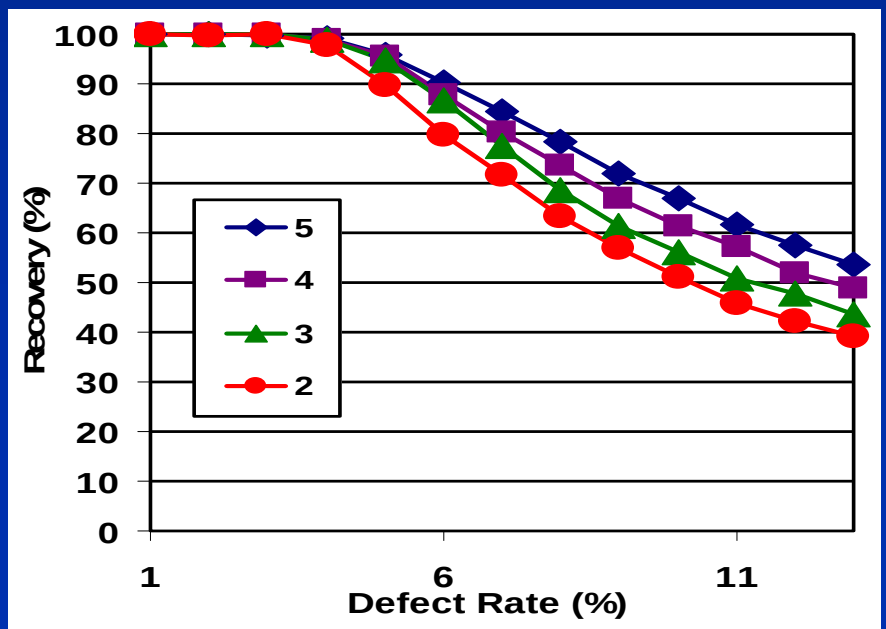
sorting

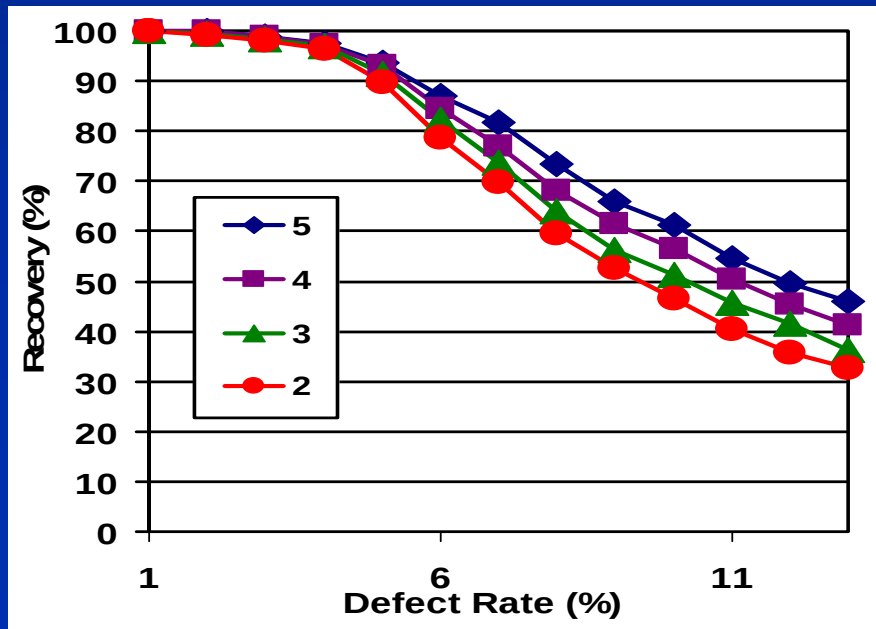
Ctr



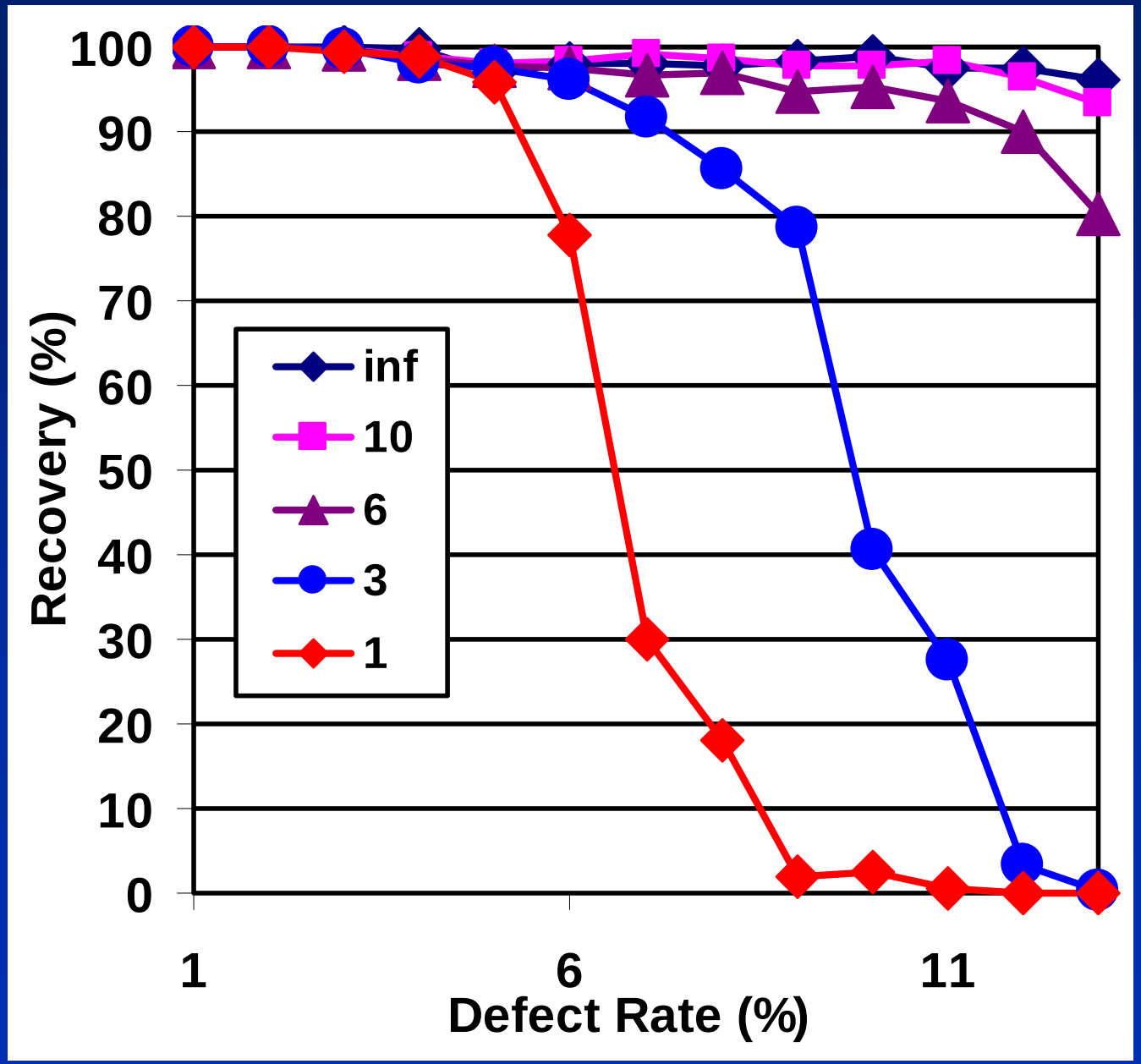


Nsm





Eval.: *counter* circuits, *Bayesian* anal.

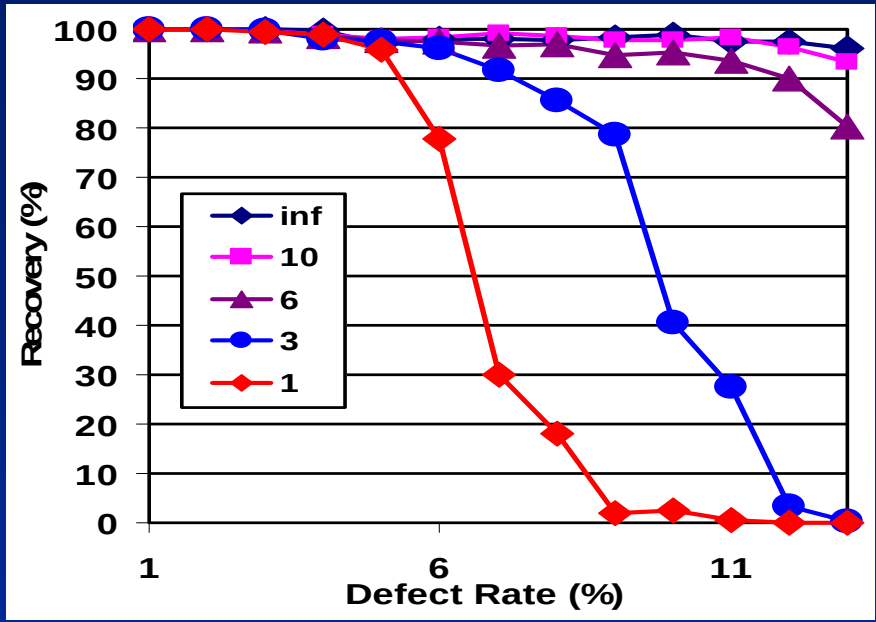


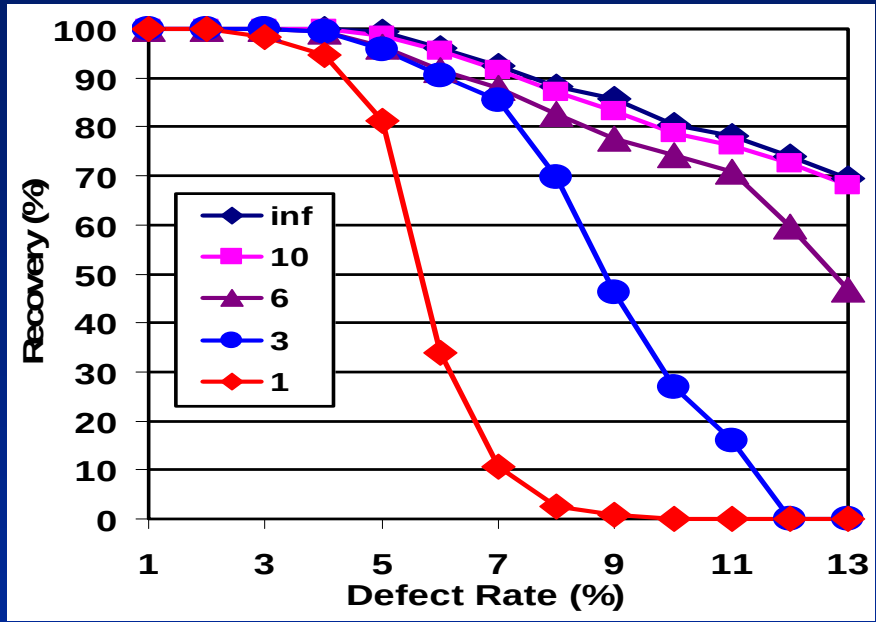
Evaluation results: comparison

Bayesian

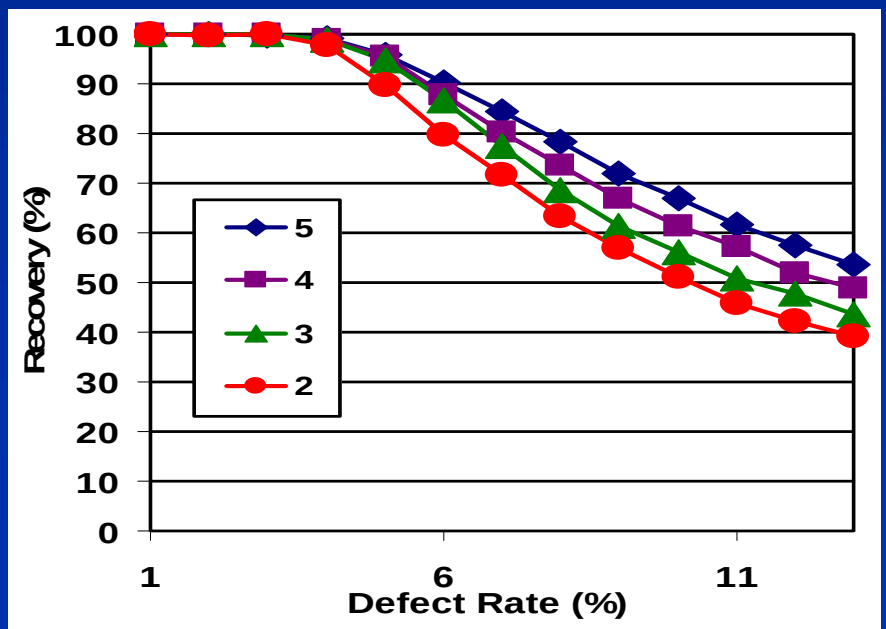
sorting

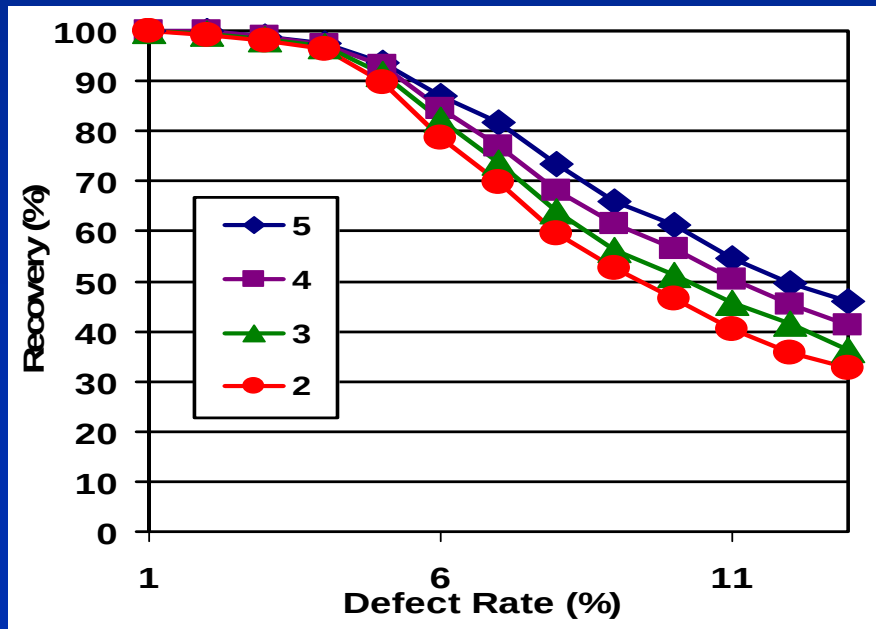
Ctr



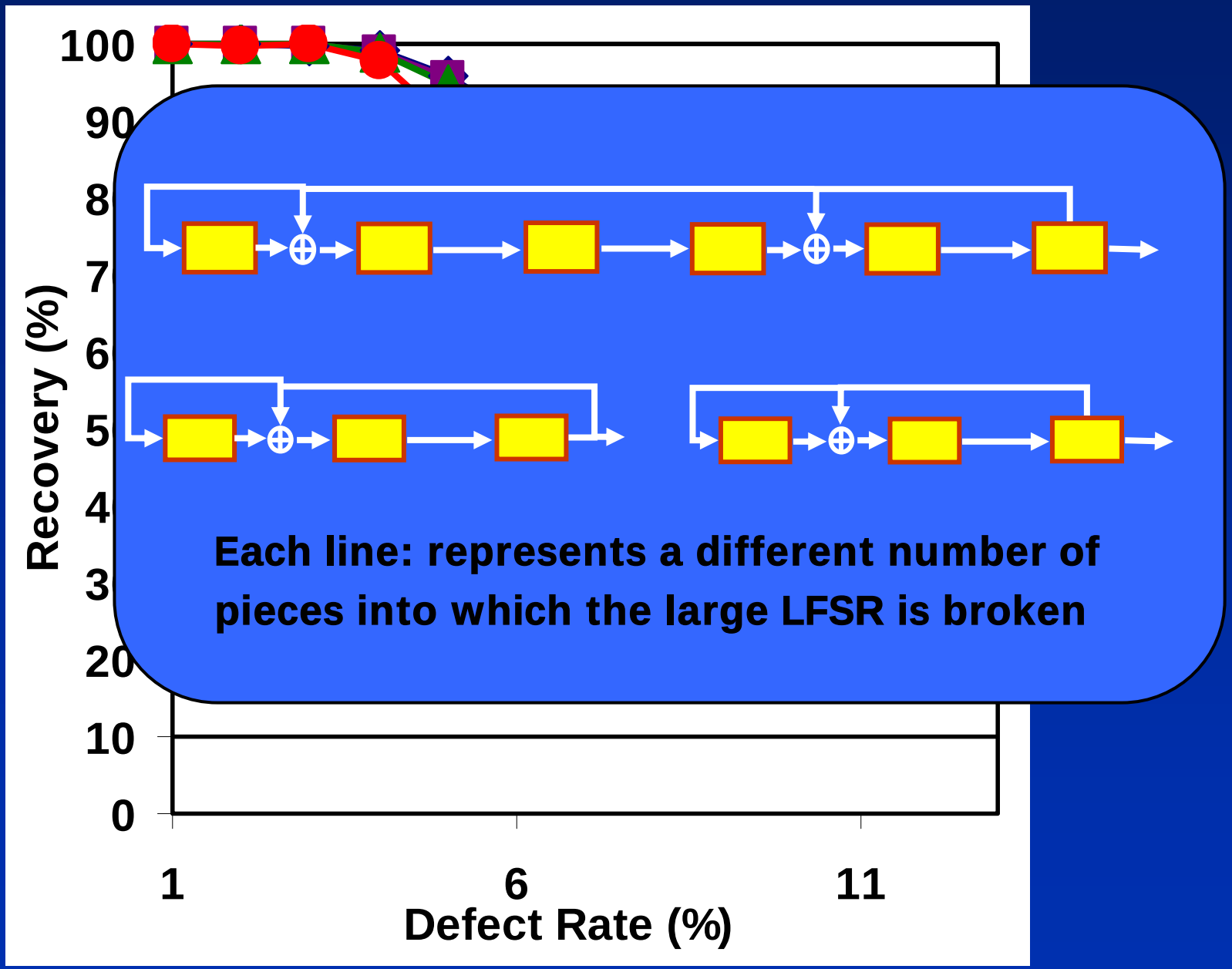


Nsm





Eval.: *n-s-m* circuits, *Bayesian* anal.

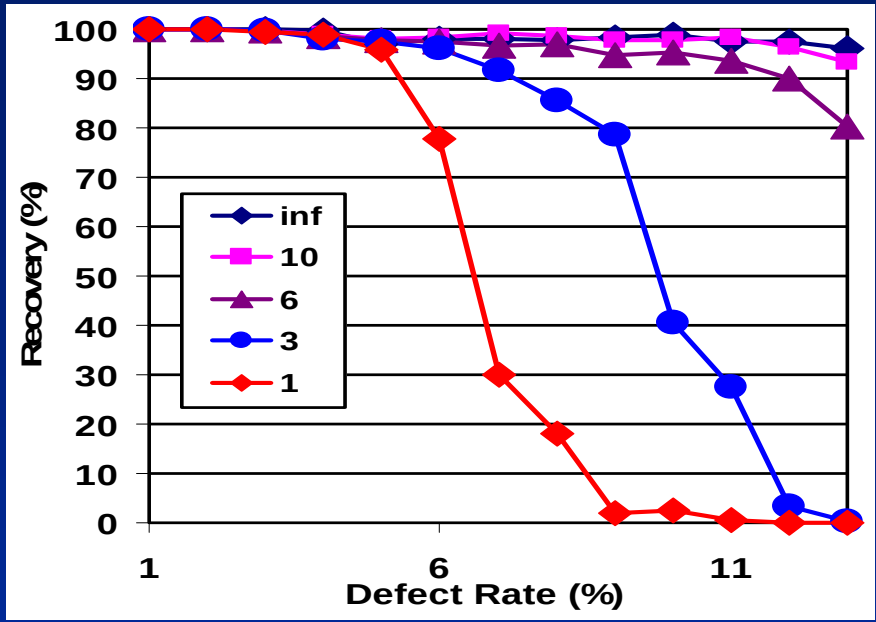


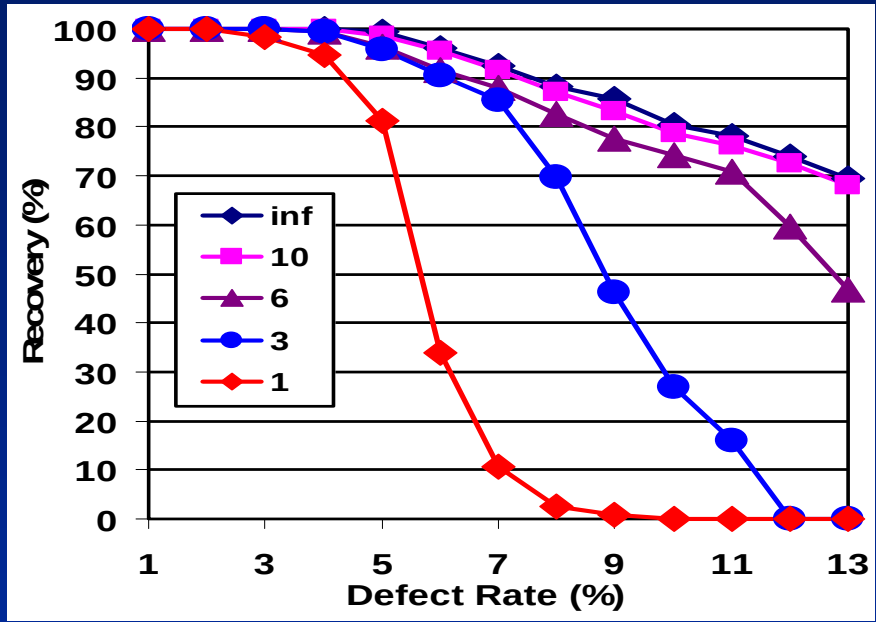
Evaluation results: comparison

Bayesian

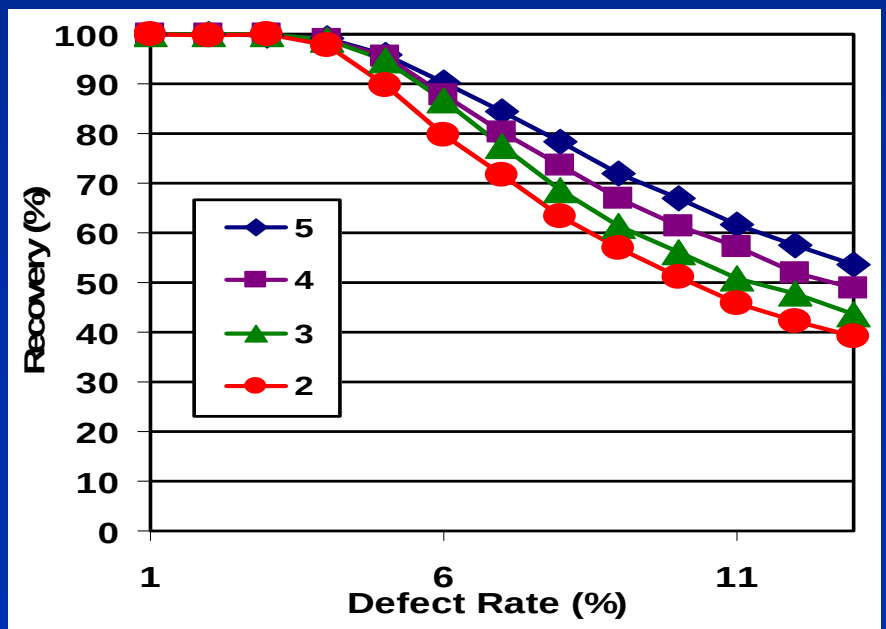
sorting

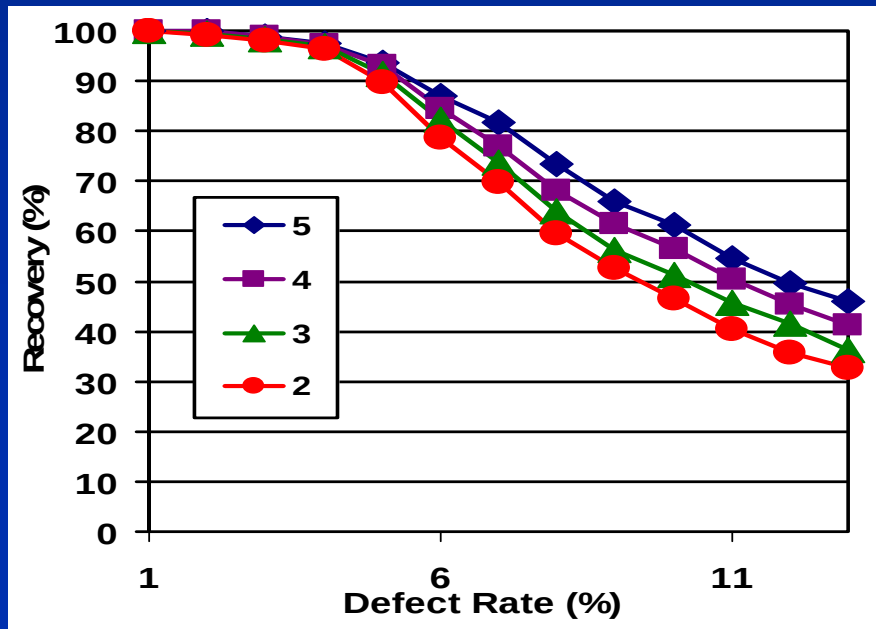
Ctr





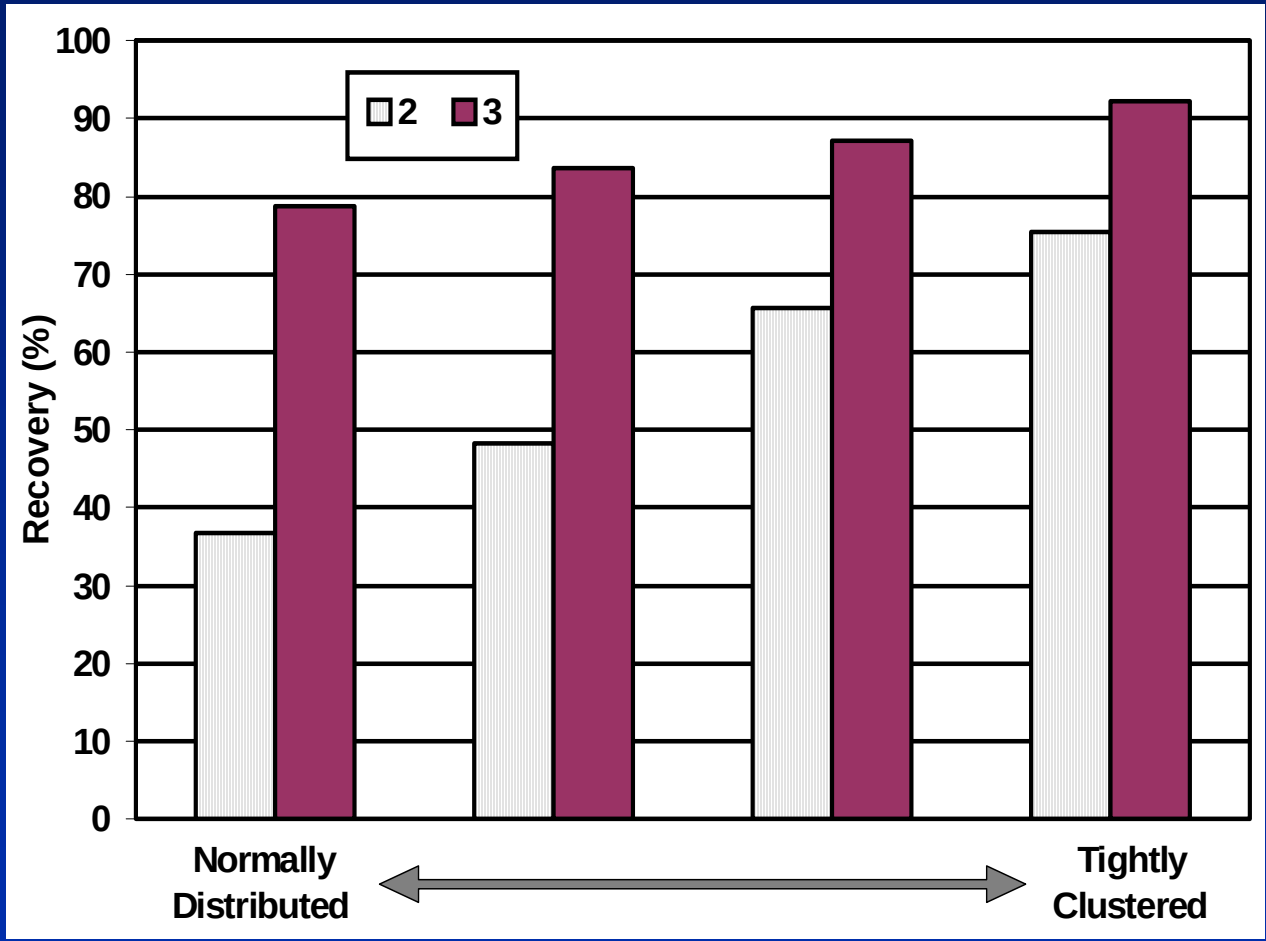
Nsm





Evaluation results: clustered defects

- So far: uniformly distributed defects
- In VLSI: defects often clustered



- Clustered defects ⇒ higher recovery

Discussion

- Low threshold *counter* circuits give good results
  - Implementation may be possible for particular defects
- Trade-off between *Bayesian* and *sorting* analysis

Conclusions

- **New manufacturing paradigm**
  - Reduced manufacturing complexity and cost
  - Increased post-fabrication testing and defect-tolerant place-and-route effort
- Defect tolerance is a **major challenge**
- **Locate defects** and **configure around them**
- **Scalable** testing with **high recovery** is possible

Backup slides

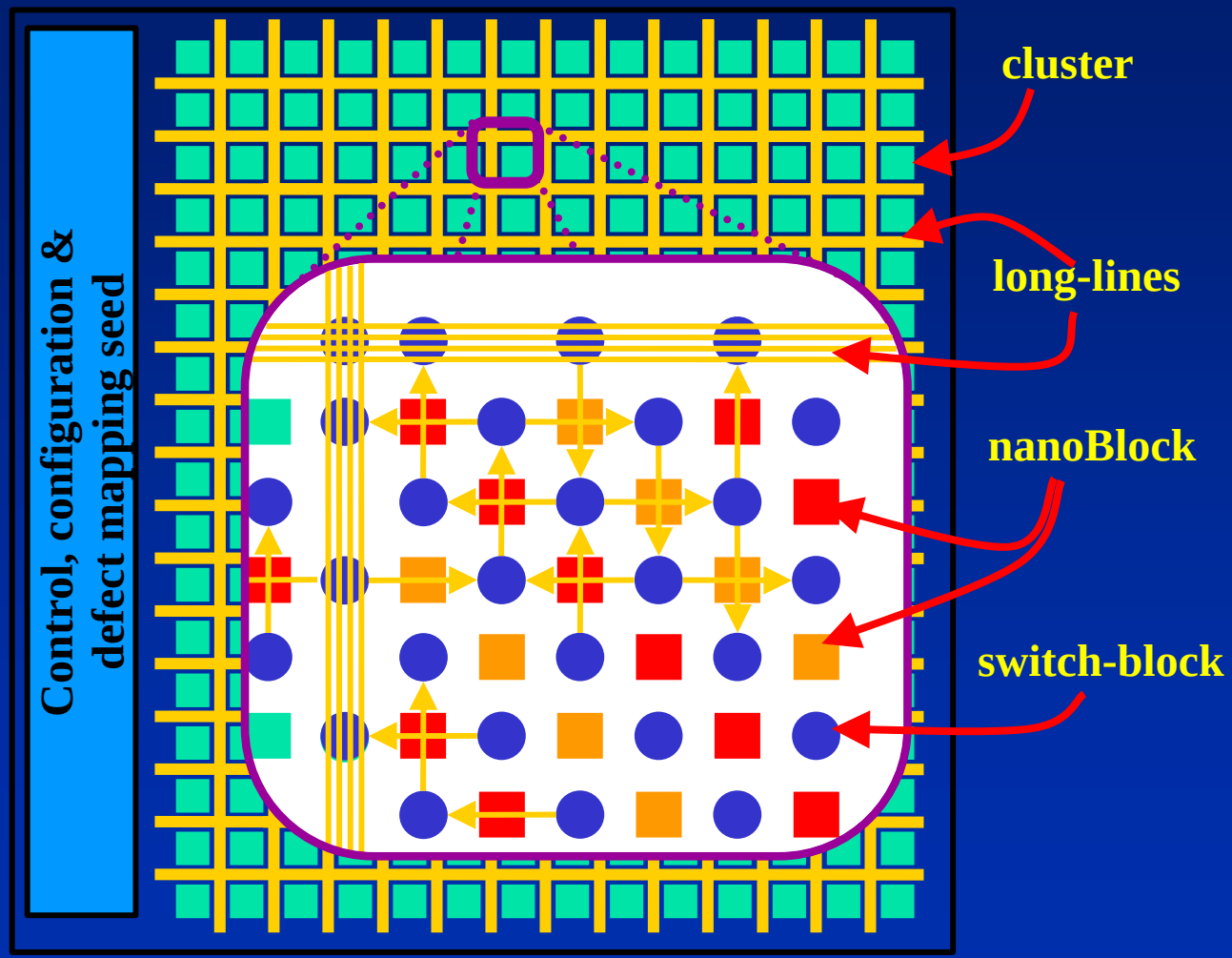
- Fabric architecture
- Algorithm
- Probability calculation
- Wave testing
- Individual results graphs

Candidate fabric architecture

- Proposed algorithms adaptable to any fabric arch
  - Should have fine-grained reconfigurability and plenty of routing resources
  - Rich interconnect resources: greatly eases testing and reconfiguration

- For purpose of this talk: consider architecture similar to island-style FPGAs
  - e.g., the *nanoFabric* architecture for CAEN-based fabrics

nanoFabric architecture (ISCA'01)



Algorithm (Part 1: prob. assignment)

1 mark all fabric components `not suspect`

2 for *iteration* from 1 to N_1 do

3 while *termination condition* not met do

4 for all fabric components marked `not suspect` do

5 configure components into *type 1 test circuits* using a particular *tiling*

6 compute defect probability for each component using circuit results from current *iteration*

7 done

8 done

9 mark components with high defect probability as `suspect`

10 done

Algorithm (Part 2: defect location)

```
11 for iteration from 1 to N2 do
12    while termination condition not met do
13       for all fabric components marked not suspect or not
         defective do
14          configure components into type 2 test-circuits using
            a particular tiling
15          for all circuits with correct output do
16             mark all circuit components not defective
17          done
18       done
19    done
20    mark some suspect components not suspect
21 done
```

Analysis methods: *Sorting* analysis

- Let component c_1 have defect counts c_{11}, c_{12}, …, c_{1n}, and c_2 have counts c_{21}, c_{22}, …, c_{2n}, for n circuits each
- Prob_defect(c_1) > Prob_defect(c_2) if $\Sigma c_{1i} > \Sigma c_{2i}$
- Complexity: O(n log$n$) for each probability calculation step

Probability calculation

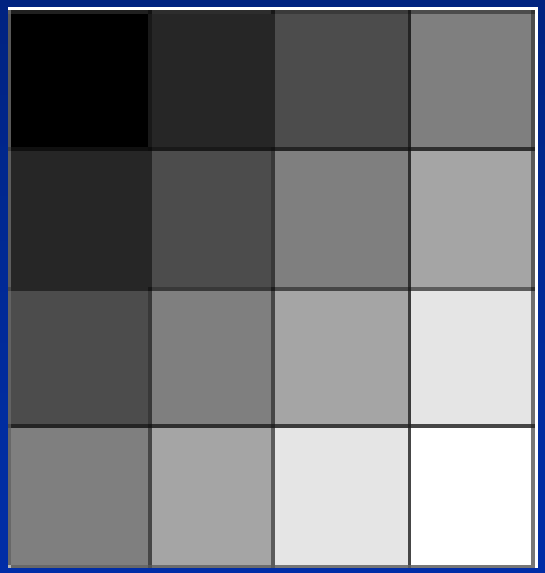
A is the event of the component being good, and *B* is the event of obtaining the defect counts c_1, c_2, ….for it,

$$\begin{aligned}\text{Prob}(A|B) &= \frac{\text{Prob}(A \cap B)}{\text{Prob}(B)} \\ &= \frac{\text{Prob}(A \cap B)}{\text{Prob}(A \cap B) + \text{Prob}(\overline{A} \cap B)}\end{aligned}$$

If *k* is the circuit size and *p* is the fabric defect rate,

$$\text{Prob}(A|B) = \frac{1}{1 + \dfrac{(1-p)^{k-1}k^k}{p^{k-1}(k-c_1)(k-c_2)\ldots(k-c_k)}}$$

Scaling with fabric size



- Testing proceeds in a wave through fabric
  - darker areas test and configure their adjacent lighter ones.
- Total time required: time for this wave to traverse the fabric
  - square root of the fabric size.