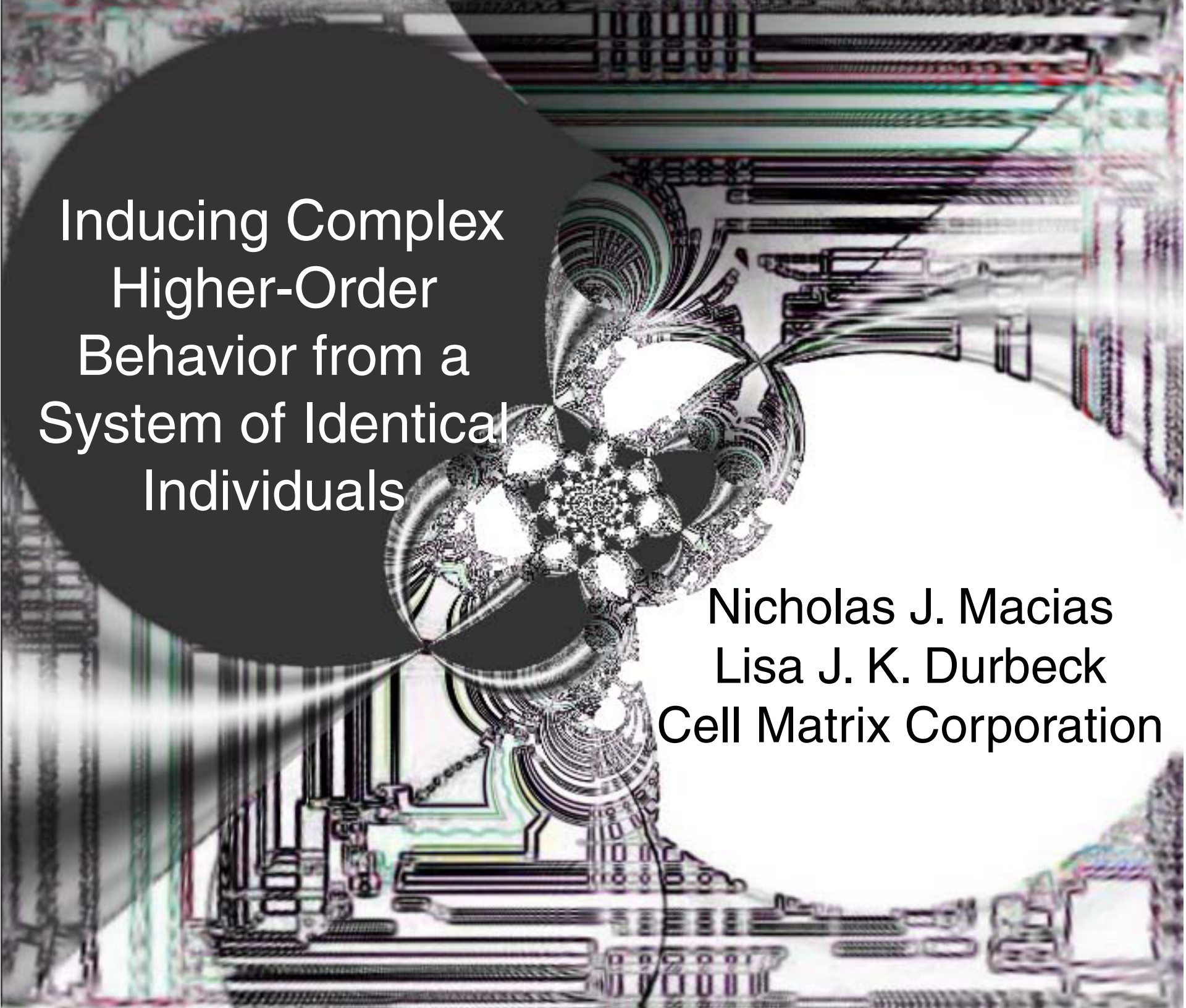
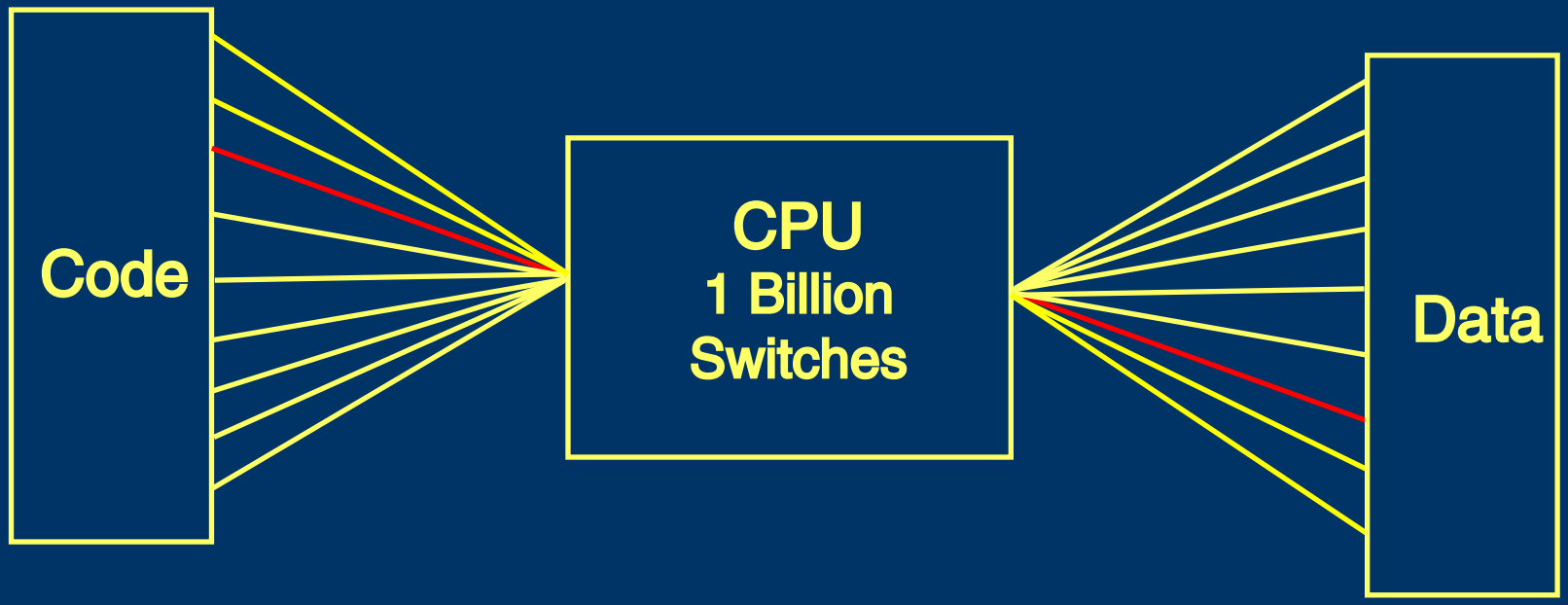


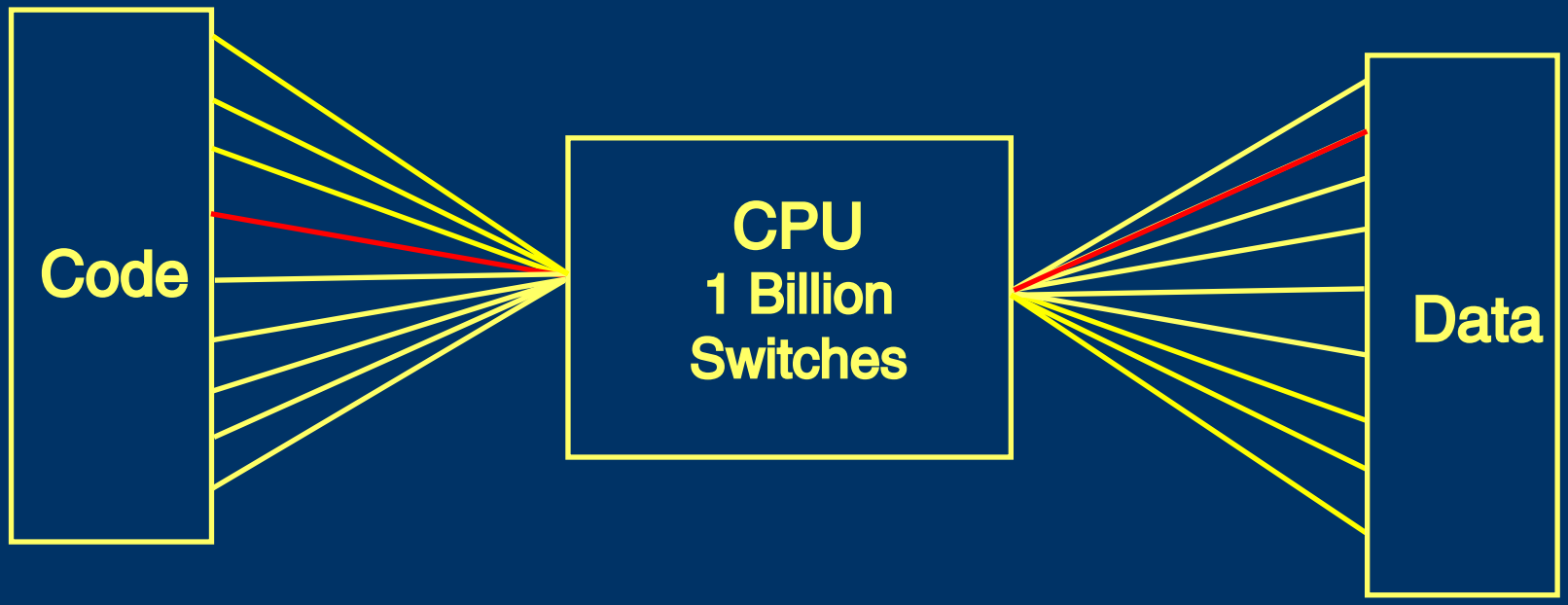
Inducing Complex Higher-Order Behavior from a System of Identical Individuals

Nicholas J. Macias
Lisa J. K. Durbeck
Cell Matrix Corporation

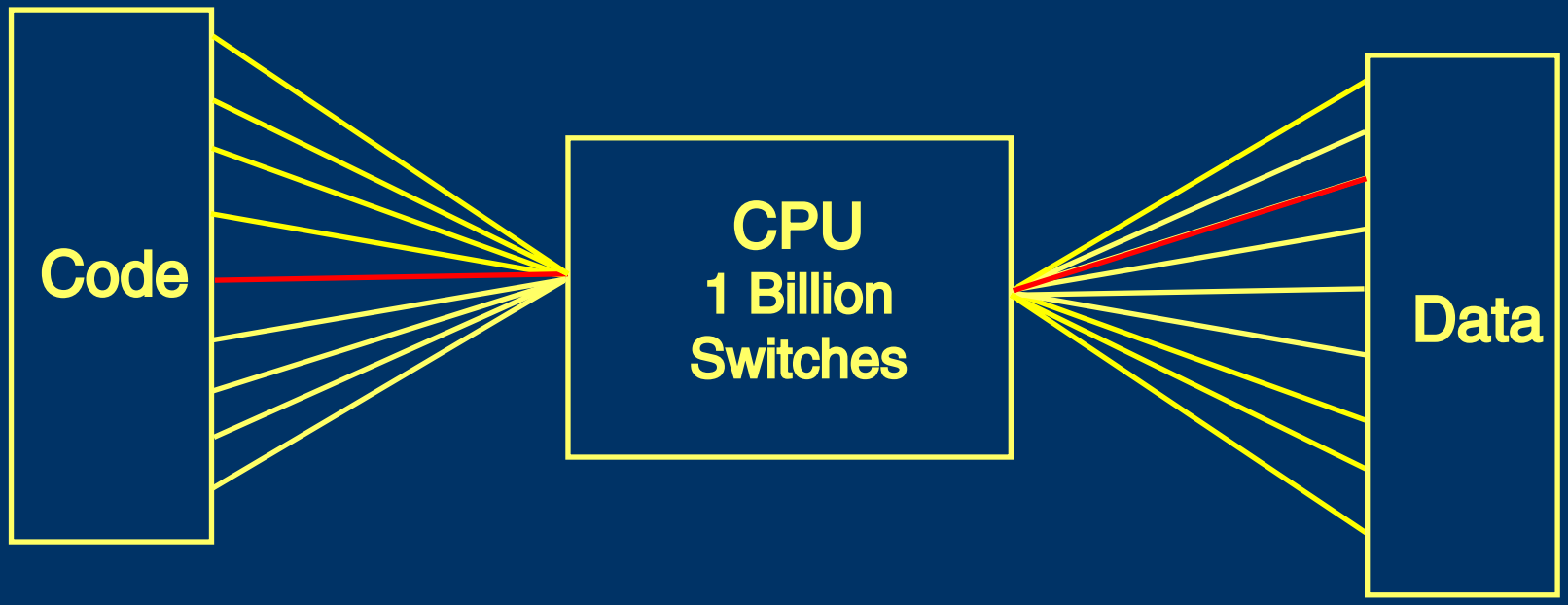
Traditional System



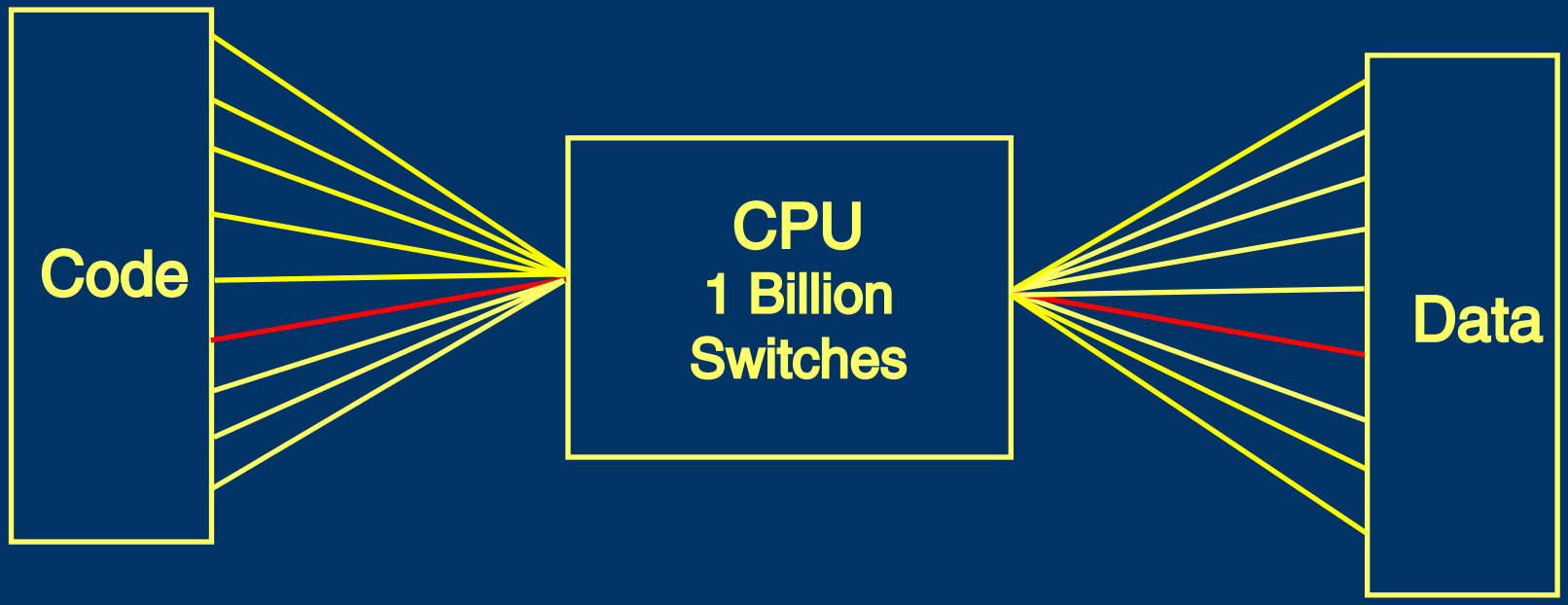
Traditional System



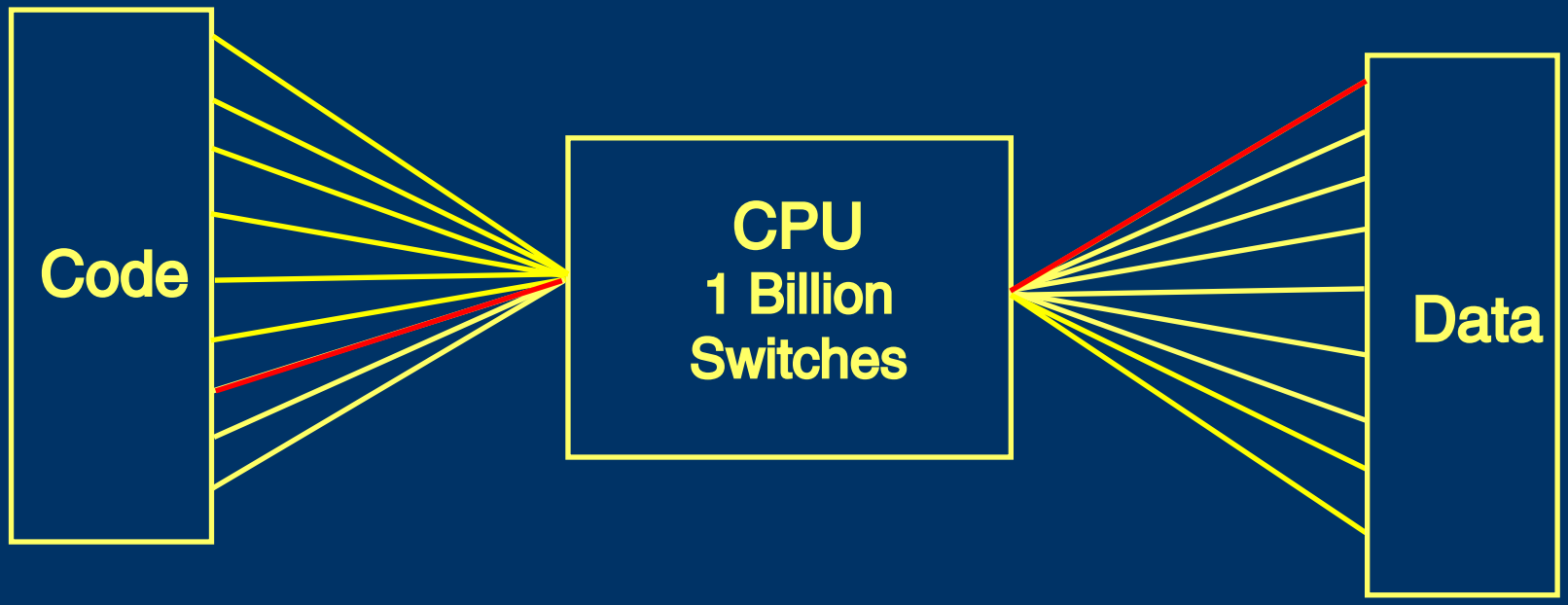
Traditional System



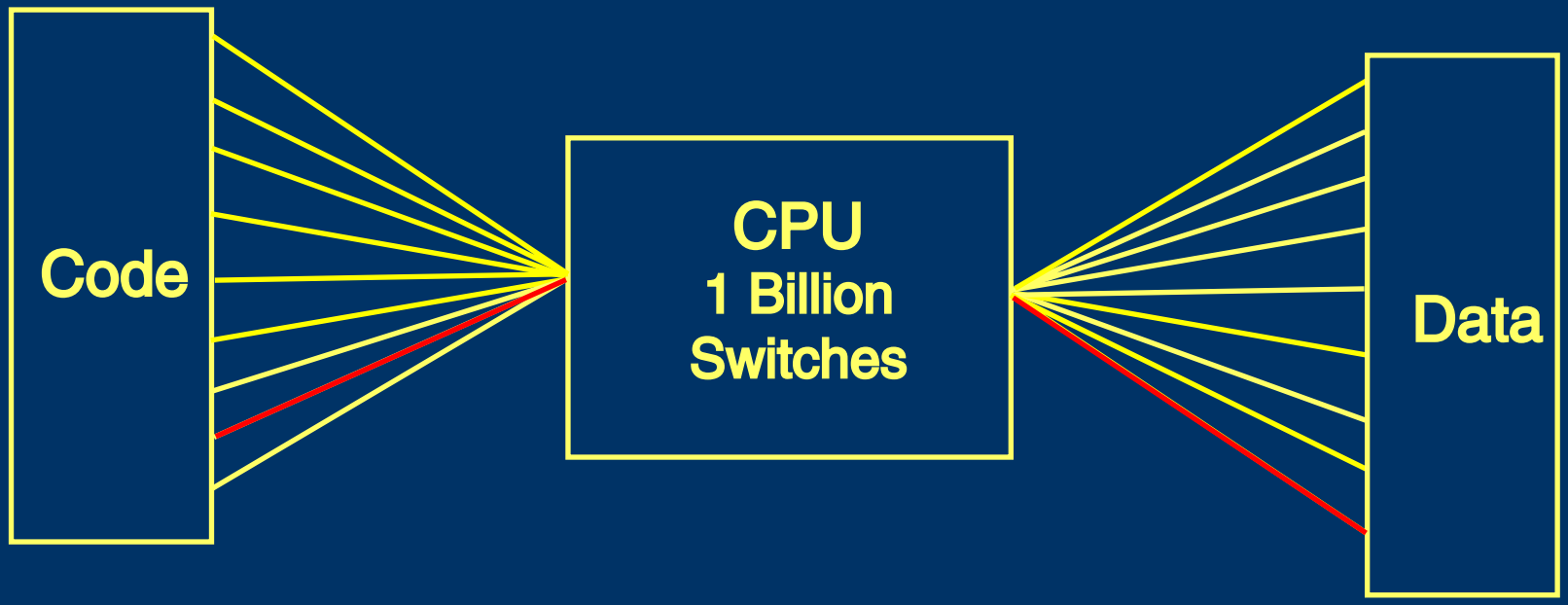
Traditional System



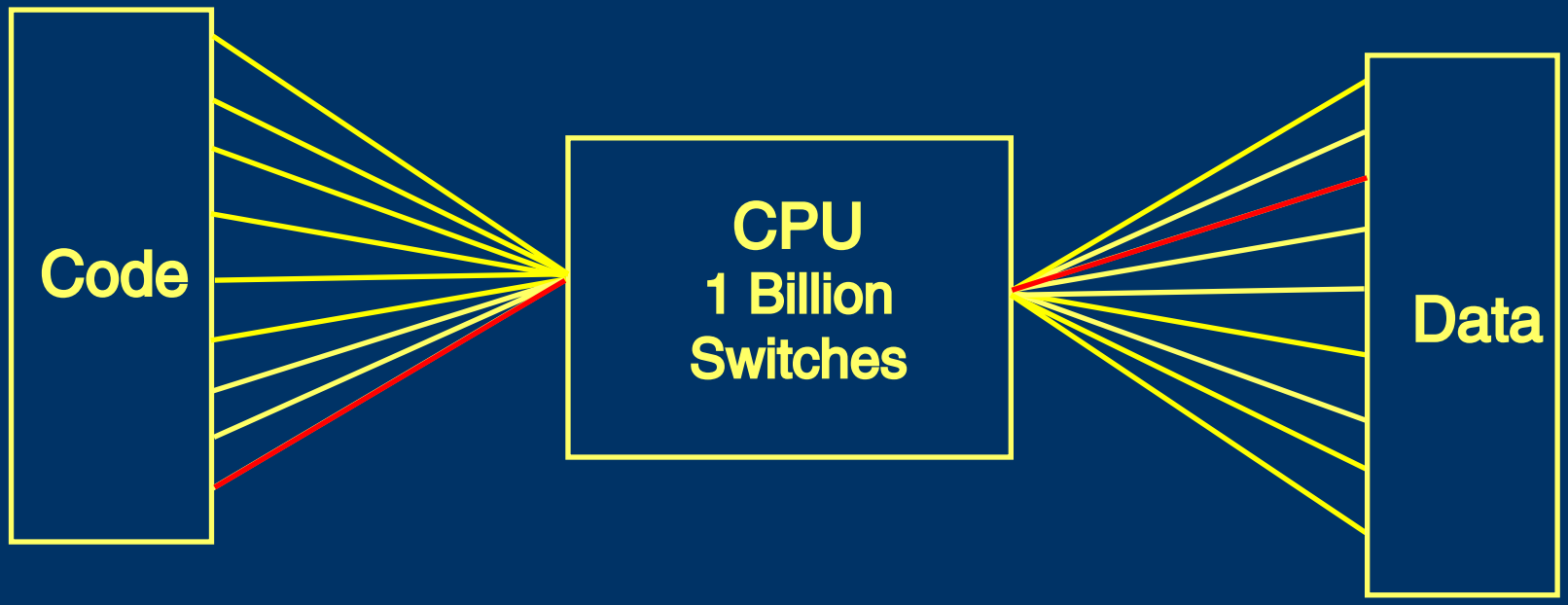
Traditional System



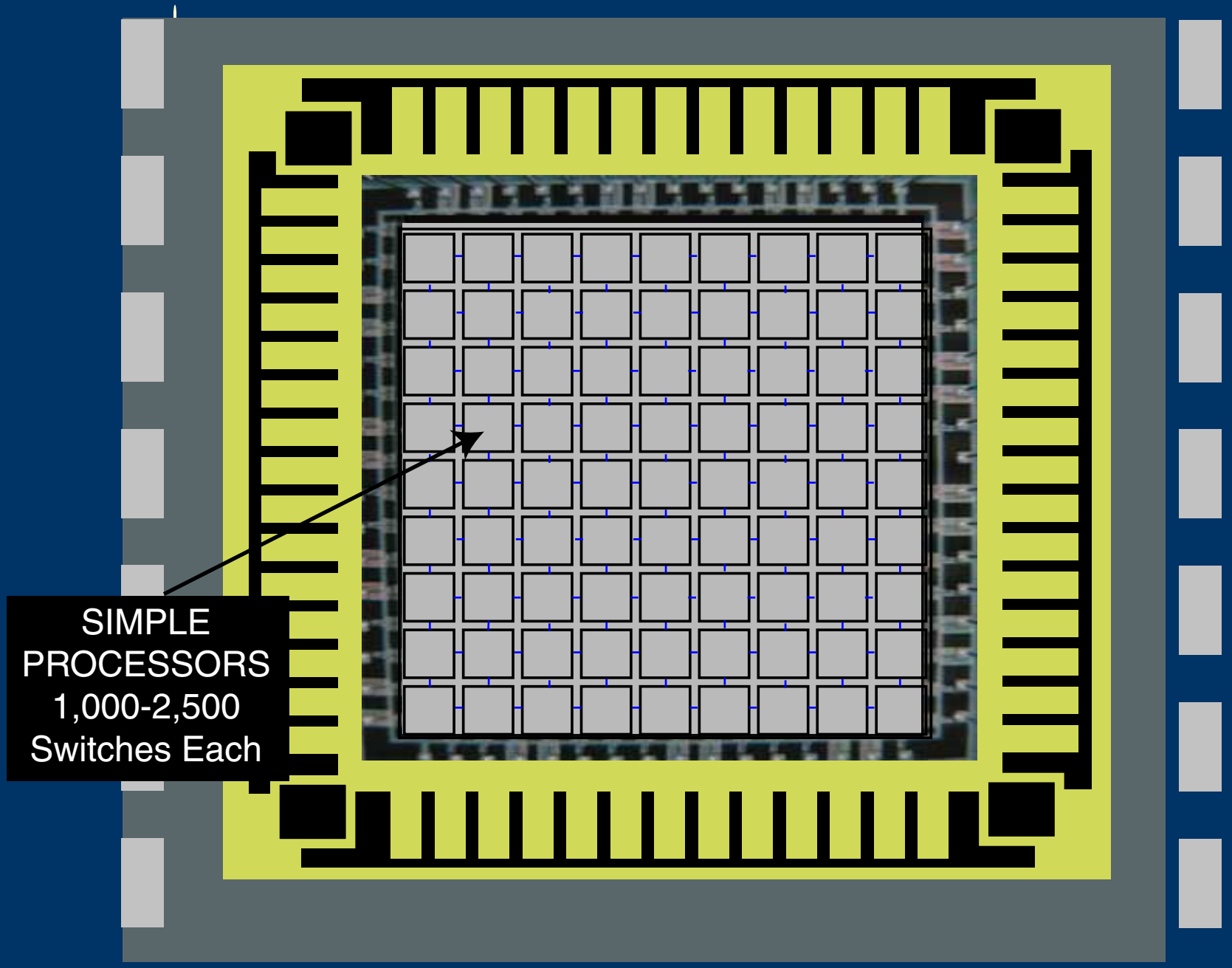
Traditional System



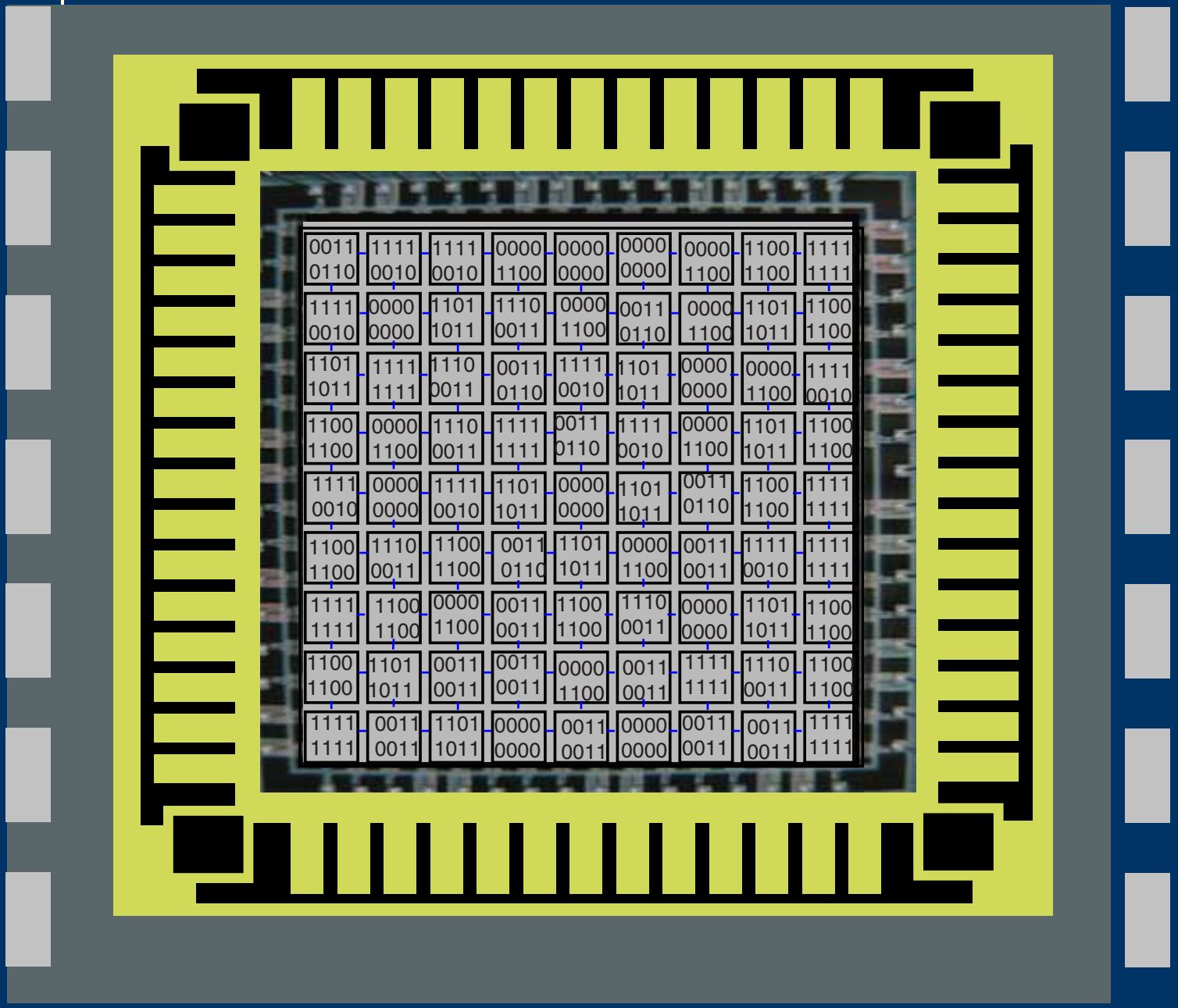
Traditional System



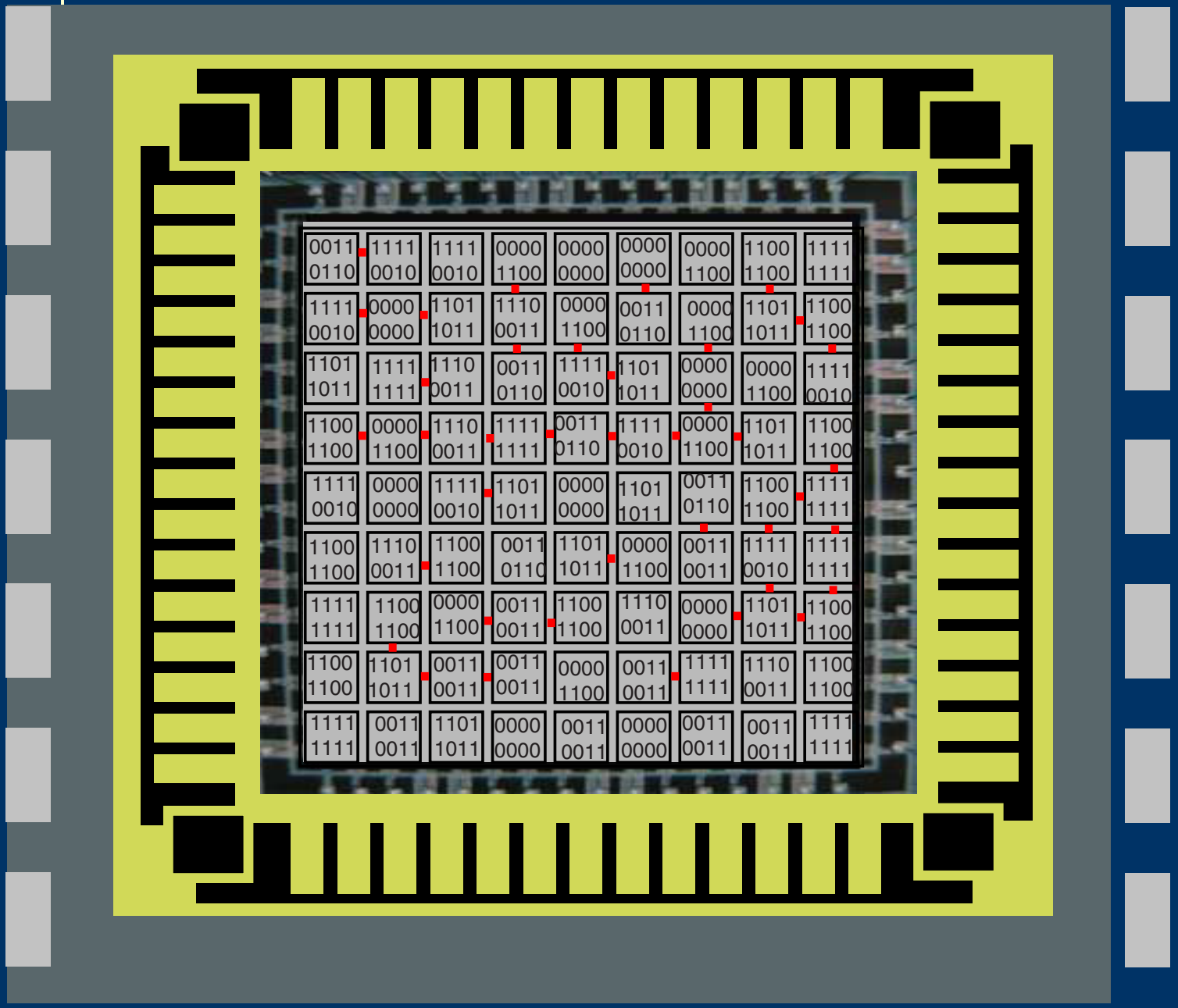
Cell Matrix Approach



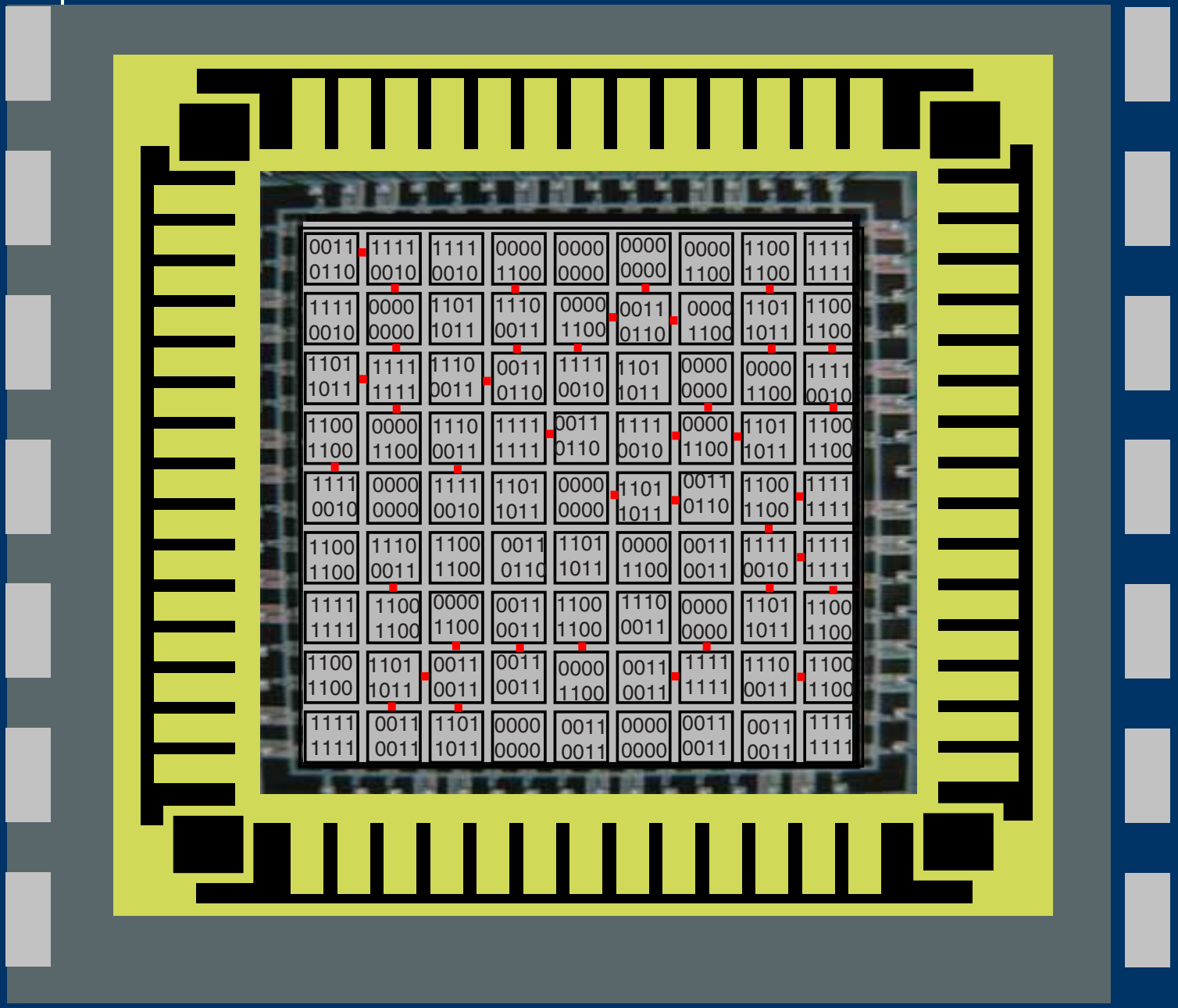
Cell Matrix Approach



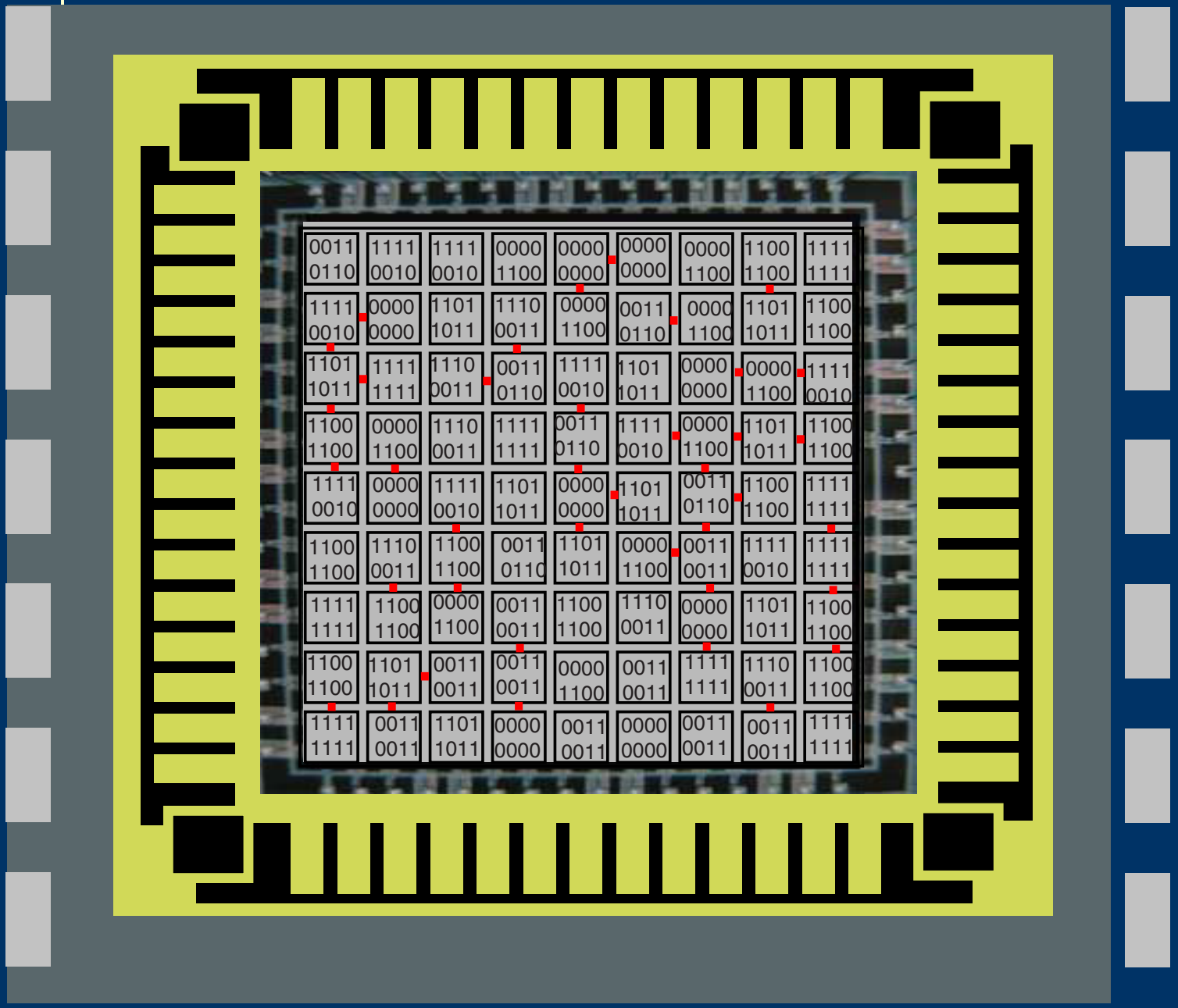
Cell Matrix Approach



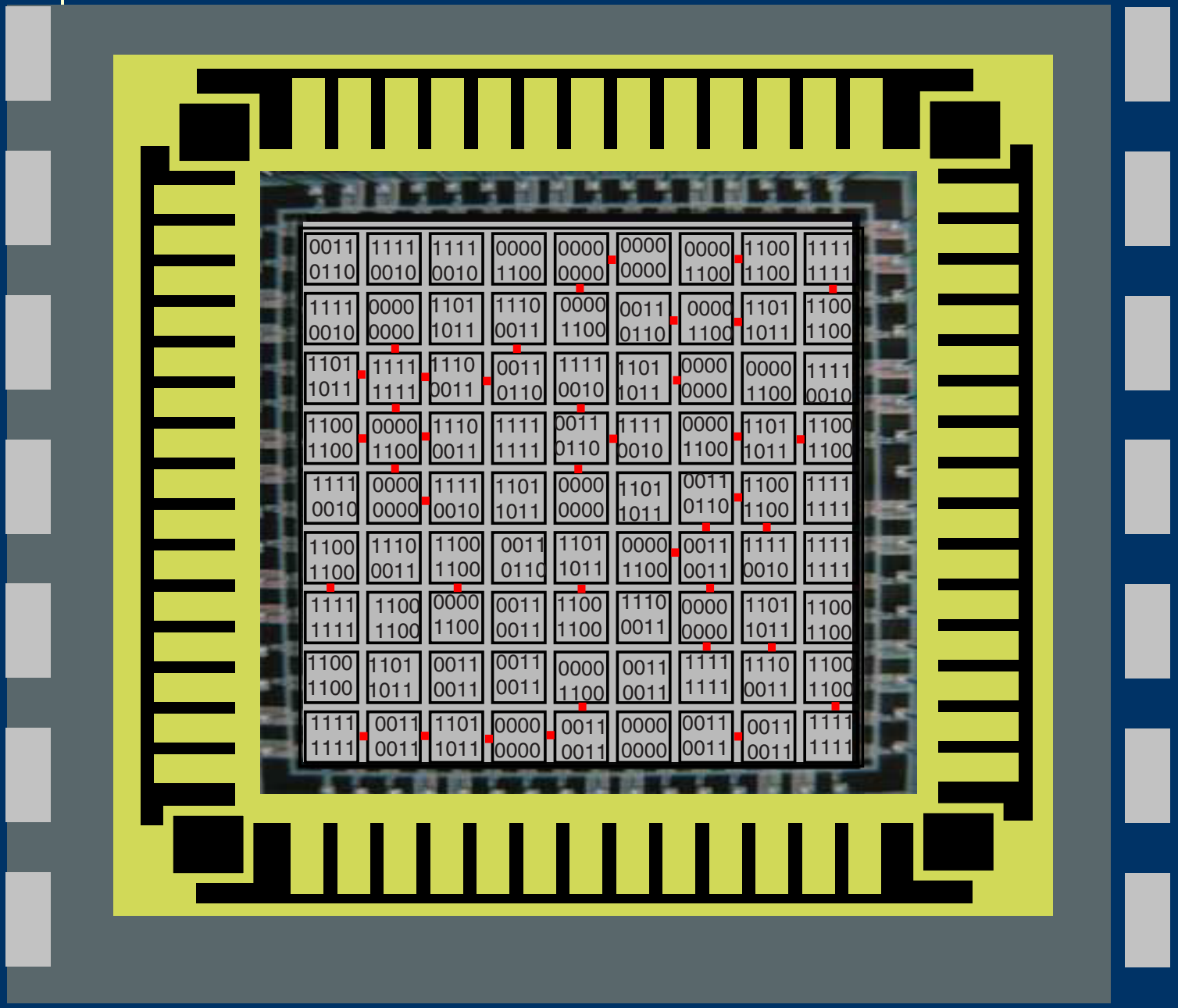
Cell Matrix Approach



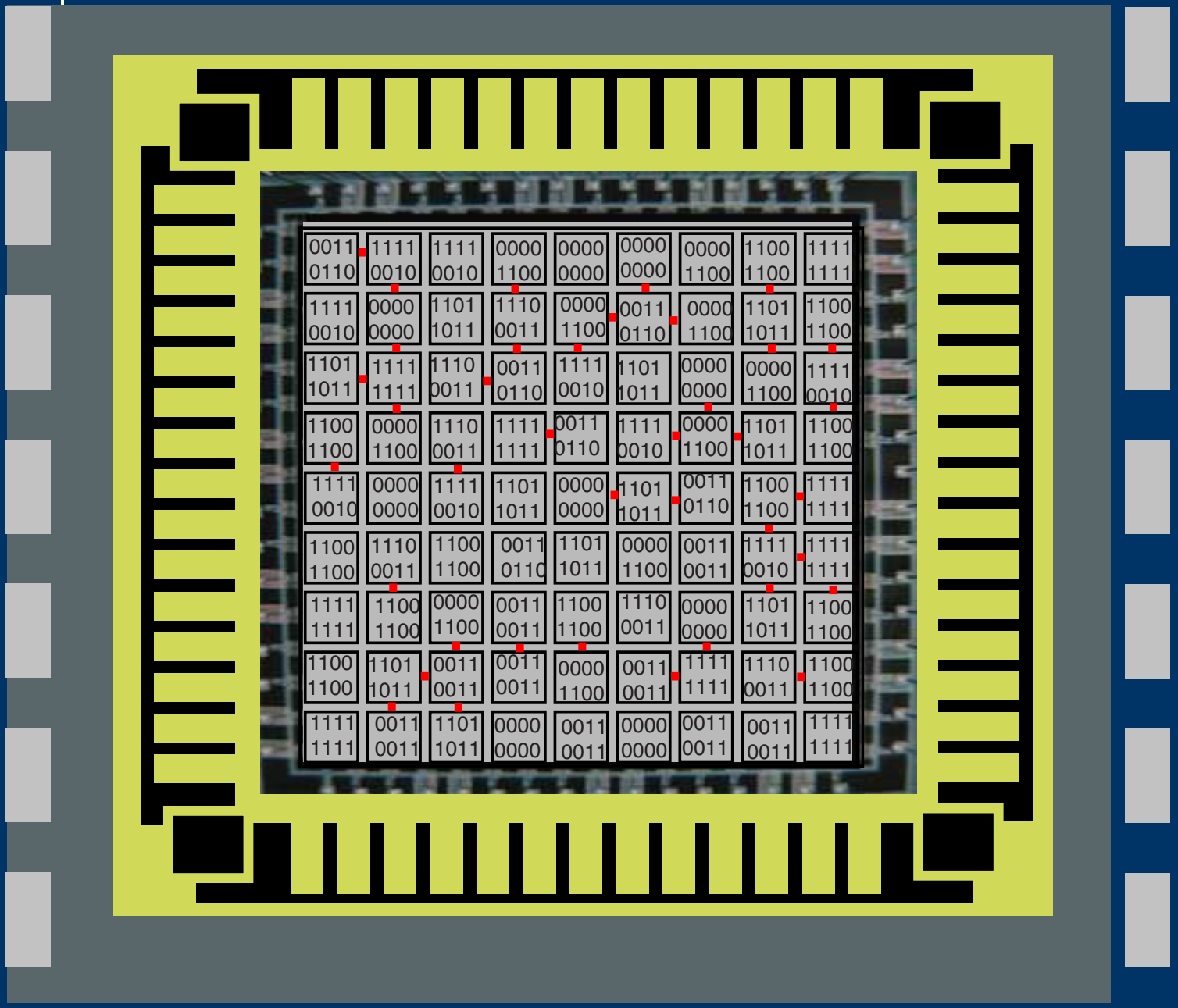
Cell Matrix Approach



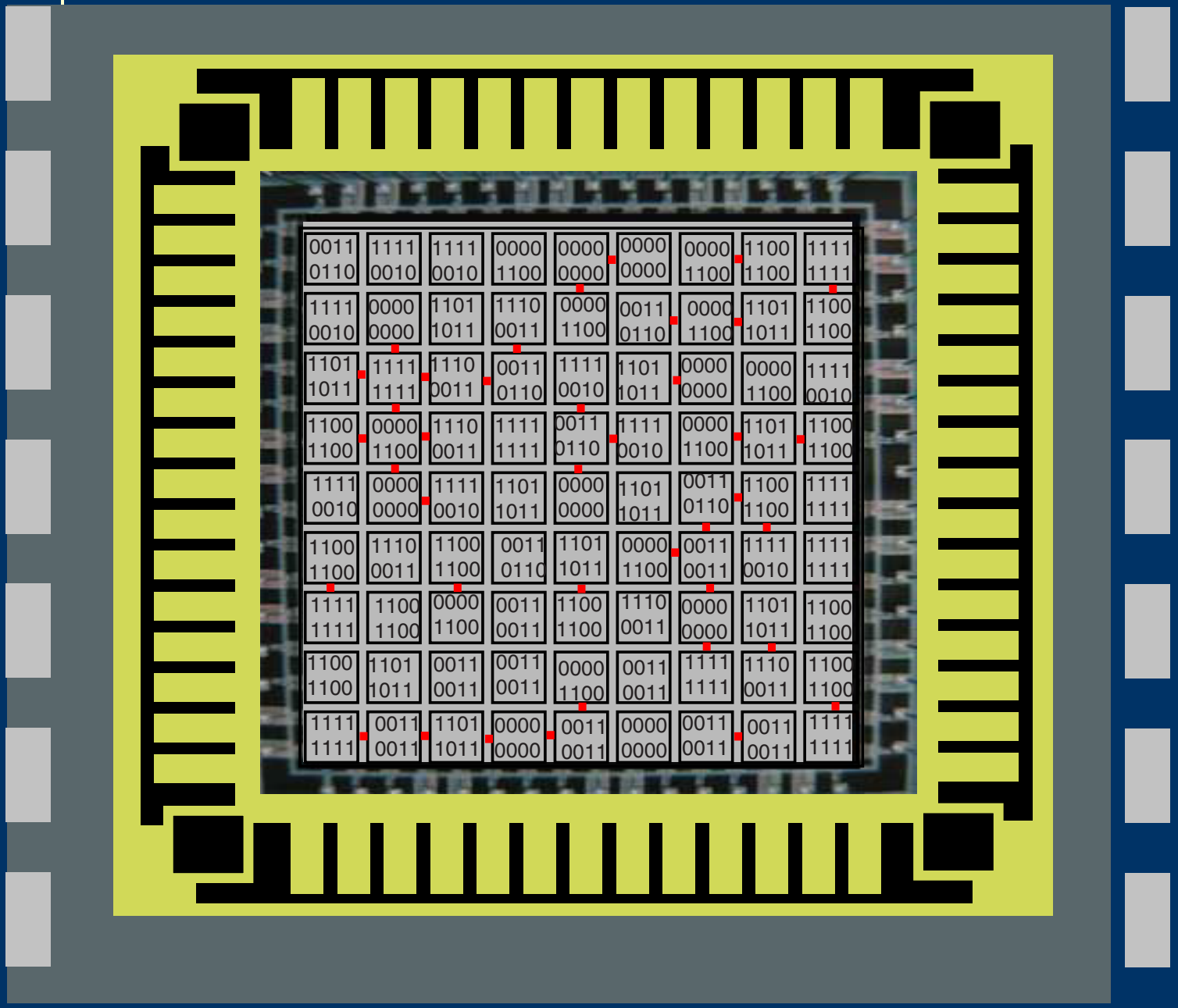
Cell Matrix Approach



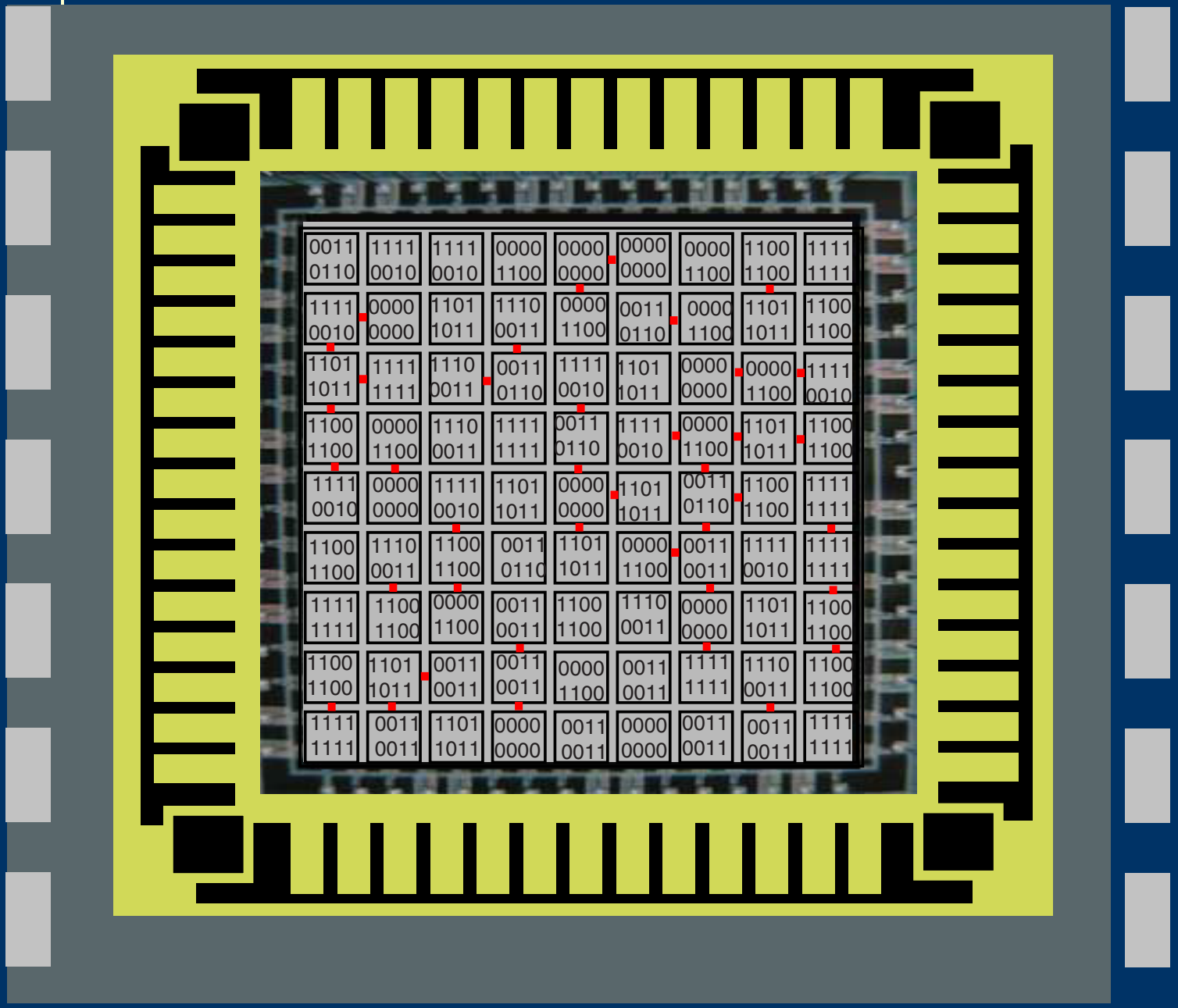
Cell Matrix Approach



Cell Matrix Approach



Cell Matrix Approach



Some Problems are Well Suited to Parallel Solution

- Simulation of Gas Dynamics
  - Use one processor per gas molecule
  - Simple interaction among processors
  - **Extremely** fast time step advance

Some Problems are Well Suited to Parallel Solution

- Simulation of Gas Dynamics
  - Use one processor per gas molecule
  - Simple interaction among processors
  - **Extremely** fast time step advance
  - Simulating one mole requires 6.02×10^{23} processors!

Other Well-Suited Problems

- Search
- Image Processing
- Finite Element Analysis
- Solution of Diophantene Equations
- Gene Sequence Alignment
- Neural Networks/Genetic Algorithms

Parallelism is the Key

- The problems scale nicely
- May require significant number of processors

Other Complexities Abound...

- How to program 10^{23} processors
- How to setup the interconnect among processors
- How to coordinate the operation of these processors
- How to handle the (inevitable) failures of individual processors

AUTONOMY

- Best way to handle a massively parallel system is with a massively parallel system
- Don't make a separate control system.
- Set the system up to control itself

SIMPLICITY

- Need to be able to actually manufacture the system
- Looking towards Nanotechnology
- Grow as a crystal?

Three Basic Goals

- Massive Parallelism (order 10^{23})
- Autonomous Operation
- Simplicity

THE CELL MATRIX SATISFIES ALL THREE

- ✓ Massive Parallelism (order 10^{23})
- ✓ Autonomous Operation
- ✓ Simplicity

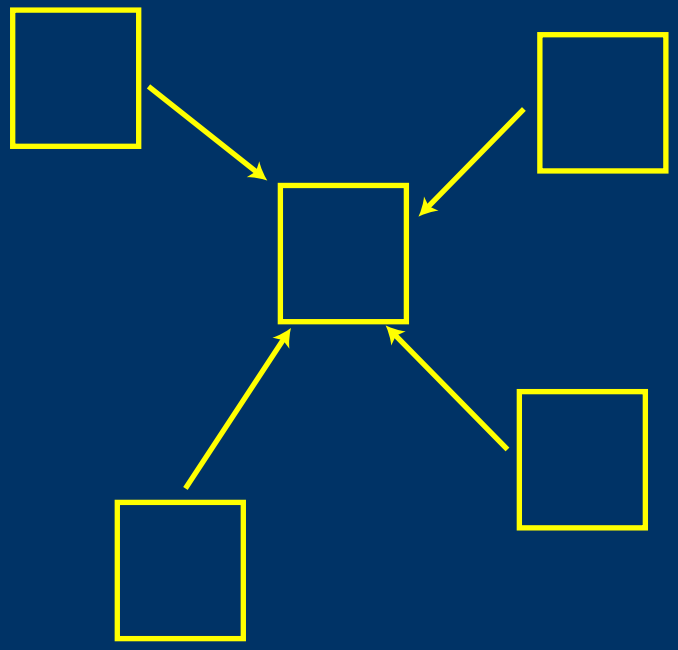
Remainder of Talk

- Architectural Details
- Specific Problem Description
- Cell Matrix Solution
- Summary, Future Work, Questions

Atomic Unit of System

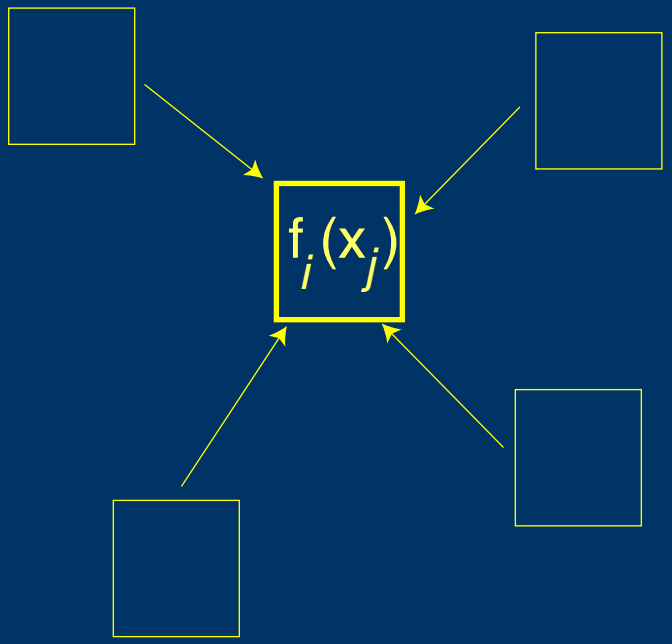
Cell Characteristics

1. Receive Information from Other Cells



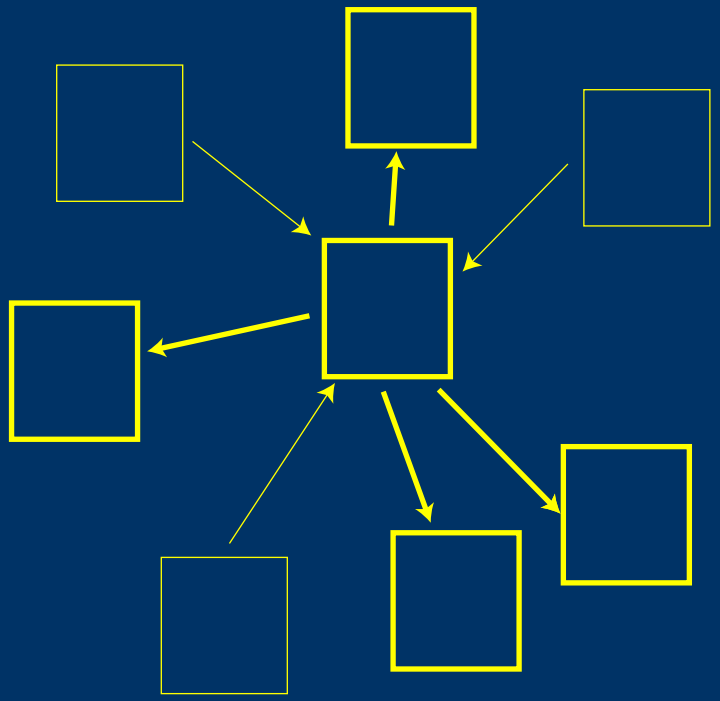
Cell Characteristics

2. Process Incoming Information

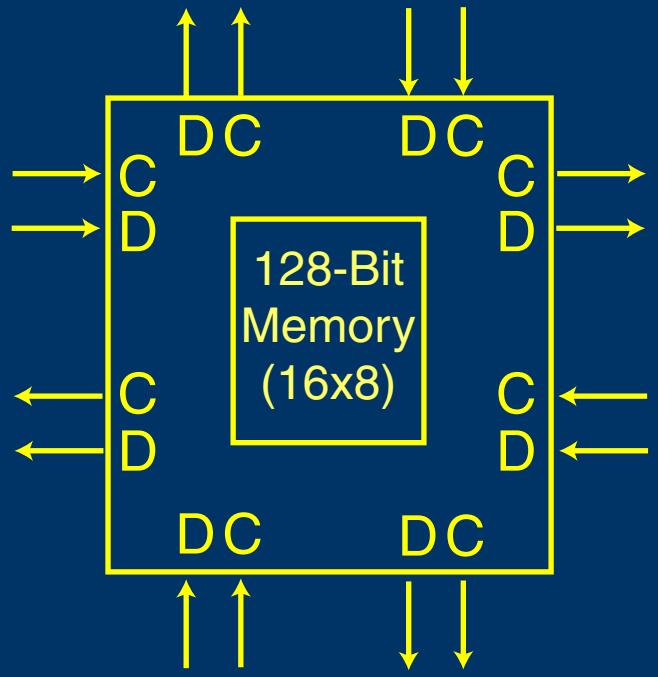


Cell Characteristics

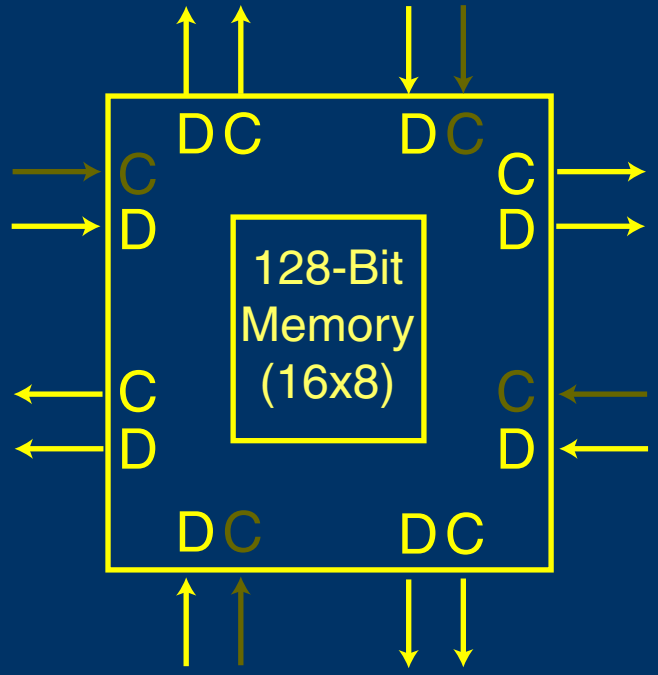
3. Transmit Data to Other Cells



Atomic Unit of Cell Matrix



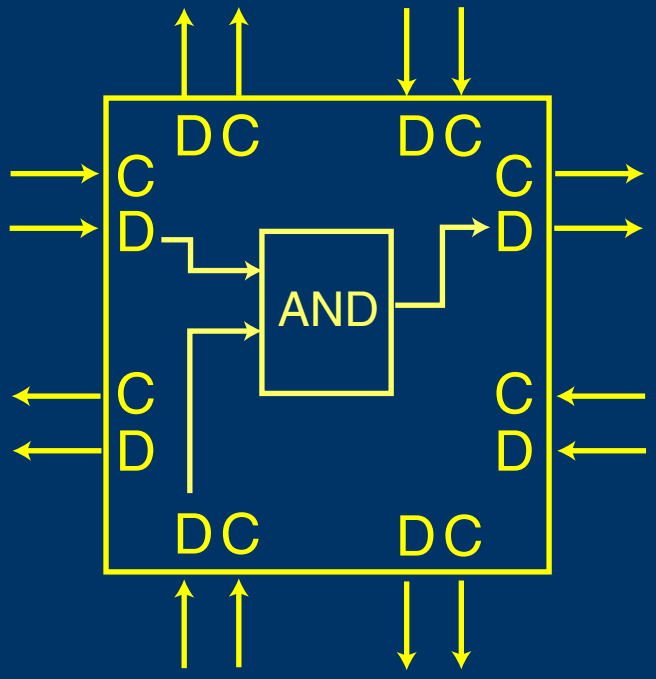
Atomic Unit of Cell Matrix



Cell Matrix Truth Table

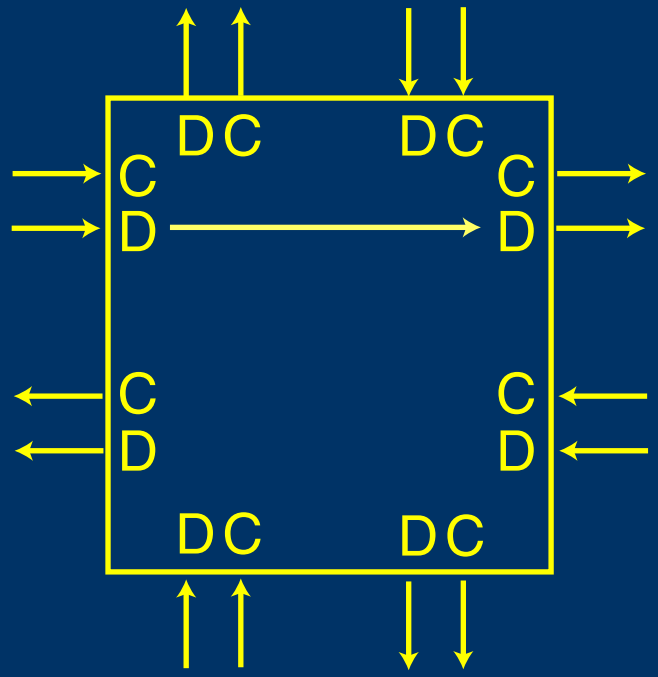
DN	DS	DW	DE	CN	CS	CW	CE	DN	DS	DW	DE
0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	1
0	0	1	1	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	1	0	1
1	0	1	1	0	0	0	1	1	1	0	1
1	1	0	0	0	0	0	1	1	0	1	0
1	1	0	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1	1
1	1	1	1	0	0	0	1	0	0	1	1

Cells Implement Simple Logic Functions



$DE_{out} = DS_{in}.AND.DW_{in}$

Sometimes Very Simple



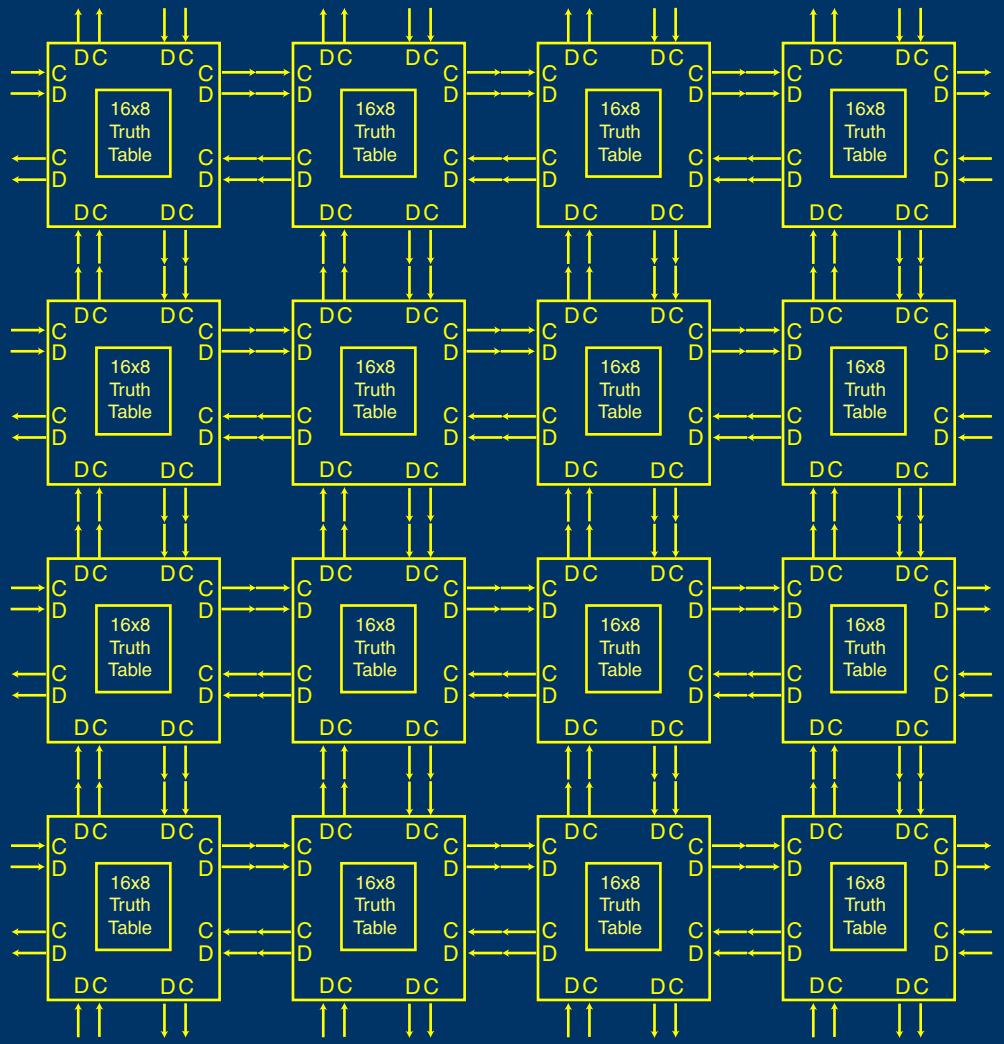
$DE_{out} = DW_{in}$

A Cell's Truth Table
Completely Specifies
the Behavior of that cell

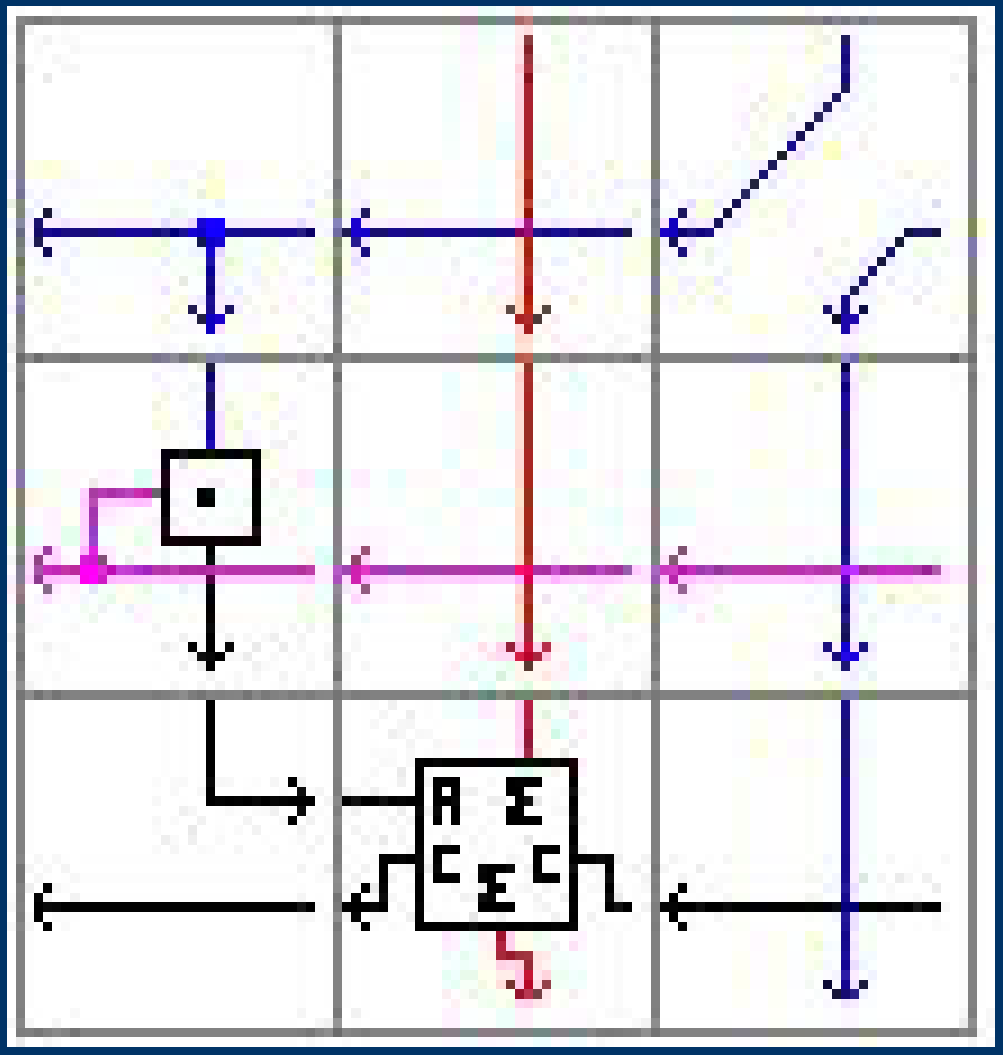
Truth Table for DW->DE

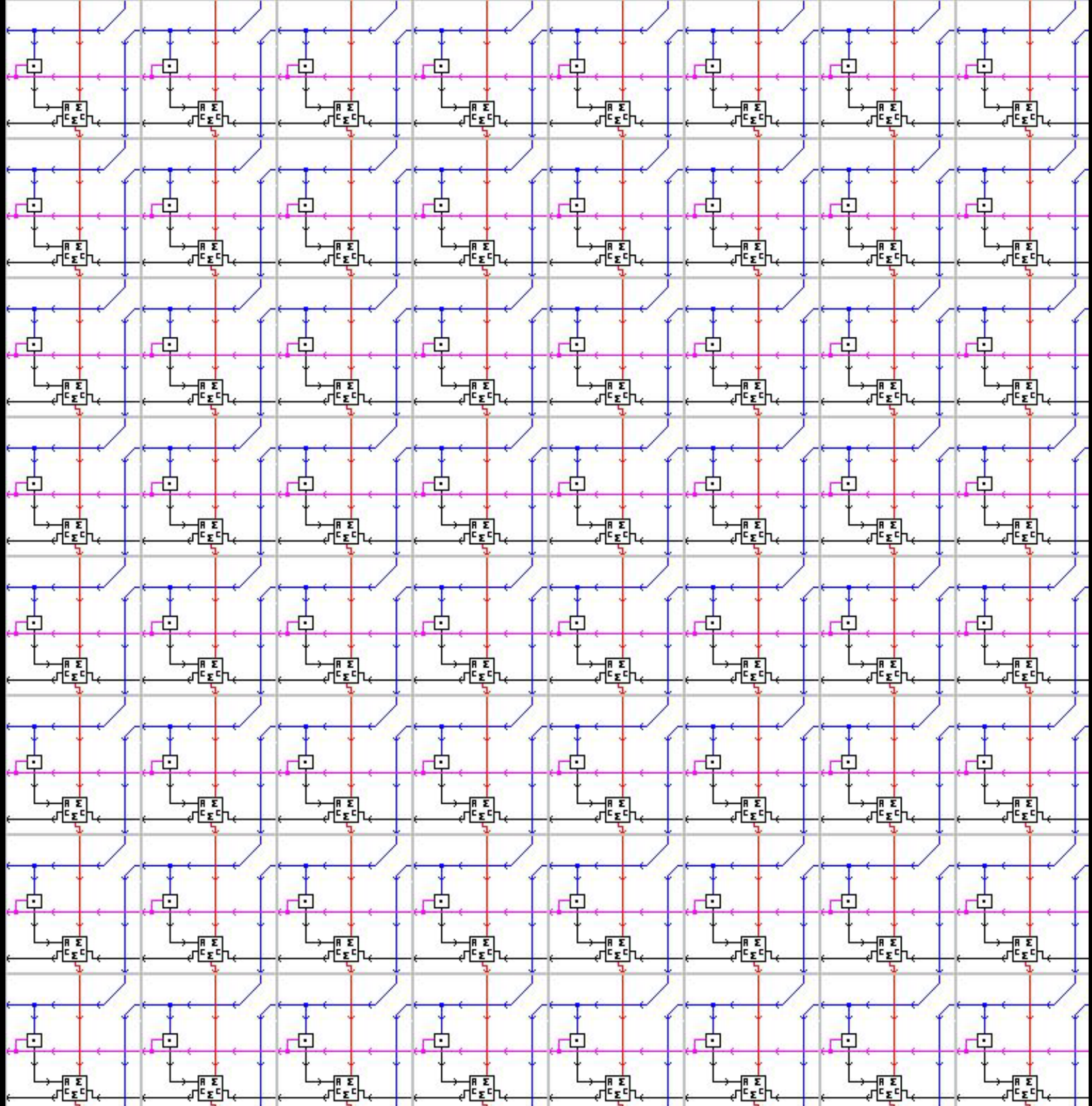
DN	DS	DW	DE	CN	CS	CW	CE	DN	DS	DW	DE
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1
0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
1	0	1	1	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	1

Tiling of Cells

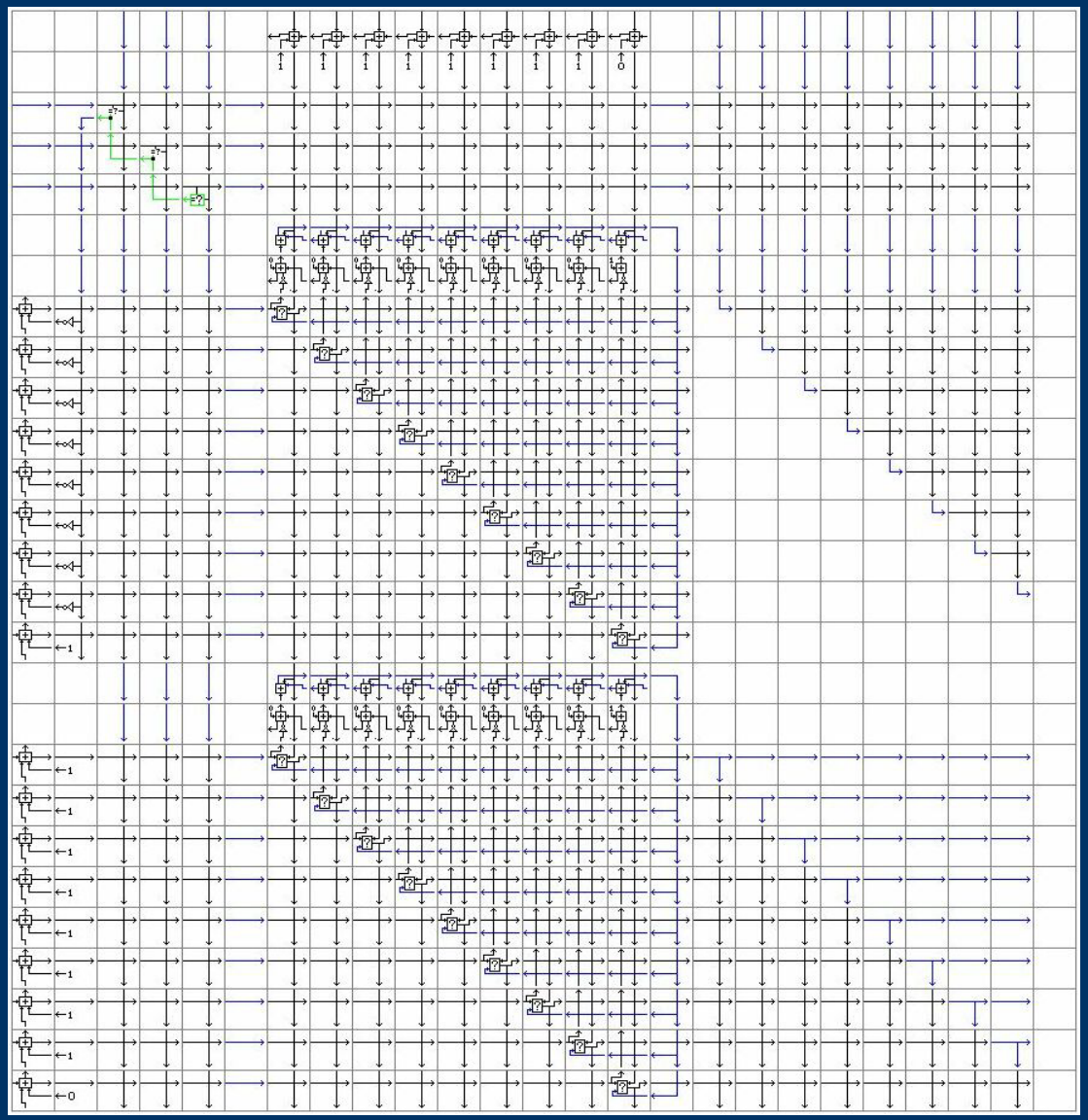


One-Bit Multiplier





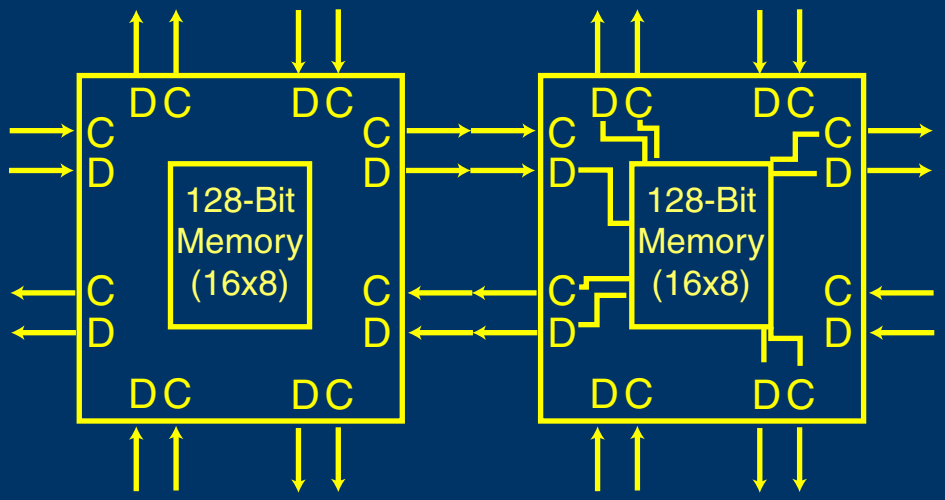
Sequence Alignment Circuit



Source: Bin Wang's
Master's Thesis
Utah State University
2001

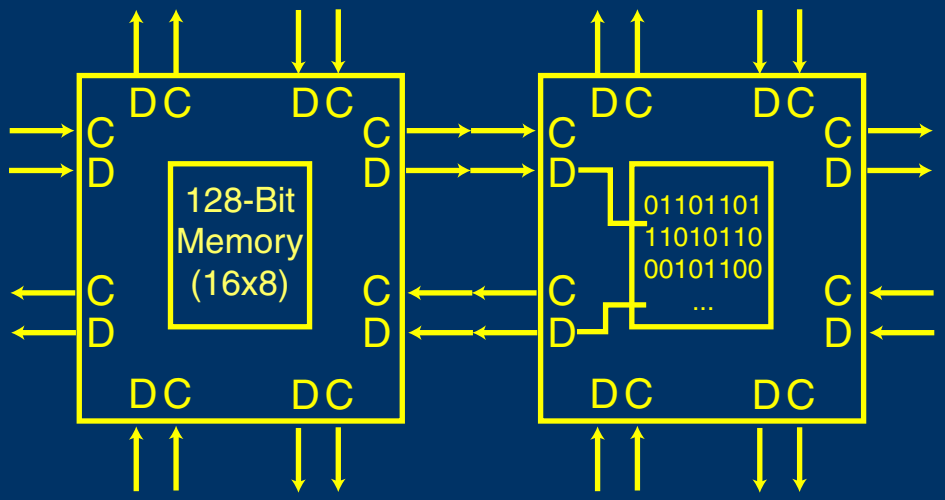
How are Cells Configured?

- Cells typically exchange data with neighbors
  - That data is processed via cell's Truth Table



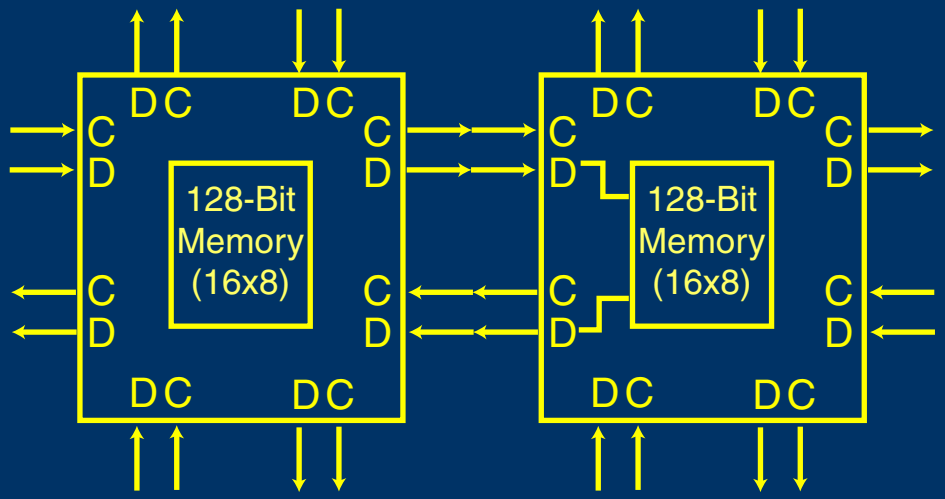
There is a Dual Mode...

- Under certain circumstances, cells may pass *truth tables* to neighbors

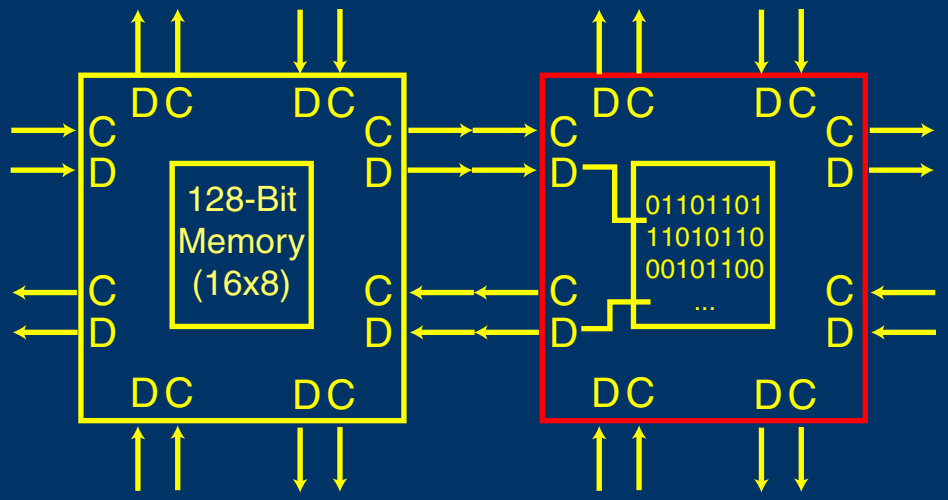


Each Cell Operates in One of Two Modes

D-Mode:
Cells Exchange DATA

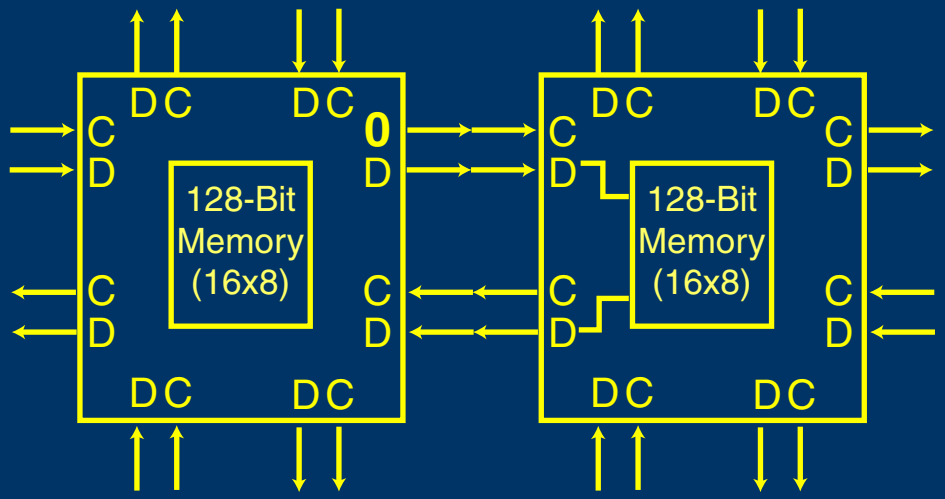


C-Mode:
Cells Exchange CODE

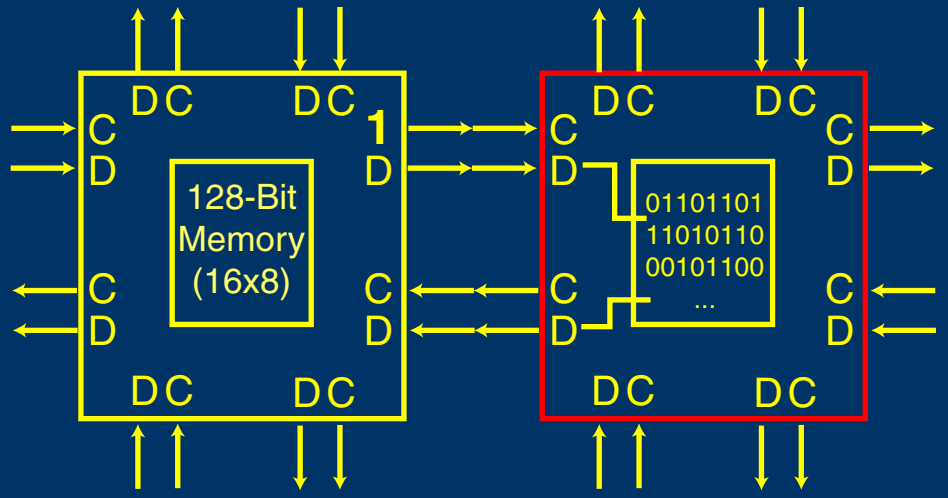


C-Mode is Indicated by Setting a C Input to1

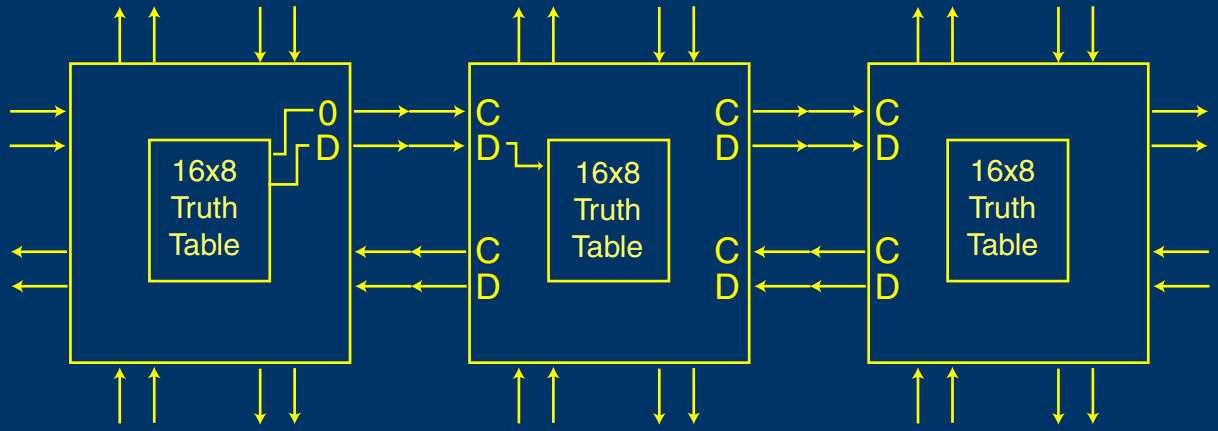
D-Mode:
Cells Exchange DATA



C-Mode:
Cells Exchange CODE



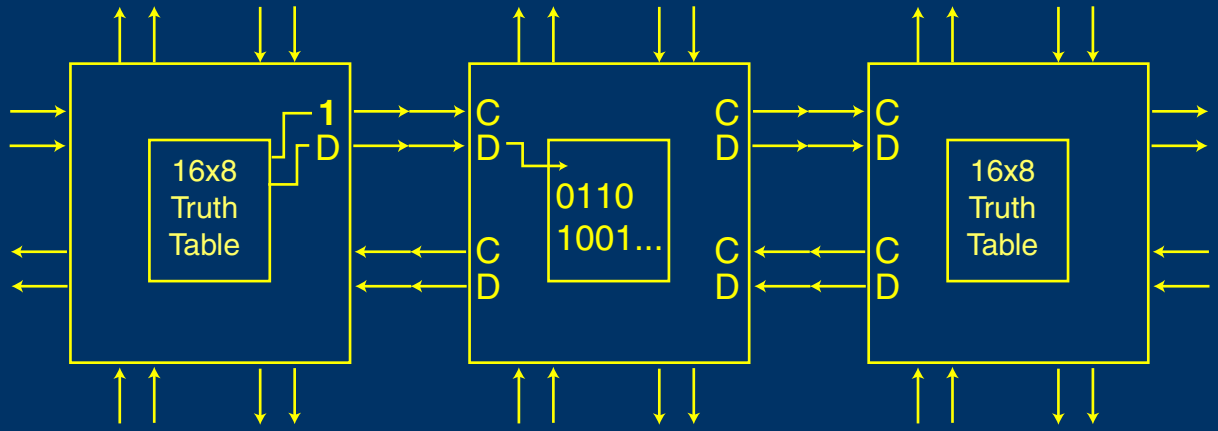
X May Pass Data to Y



X Y Z

Y is in D Mode

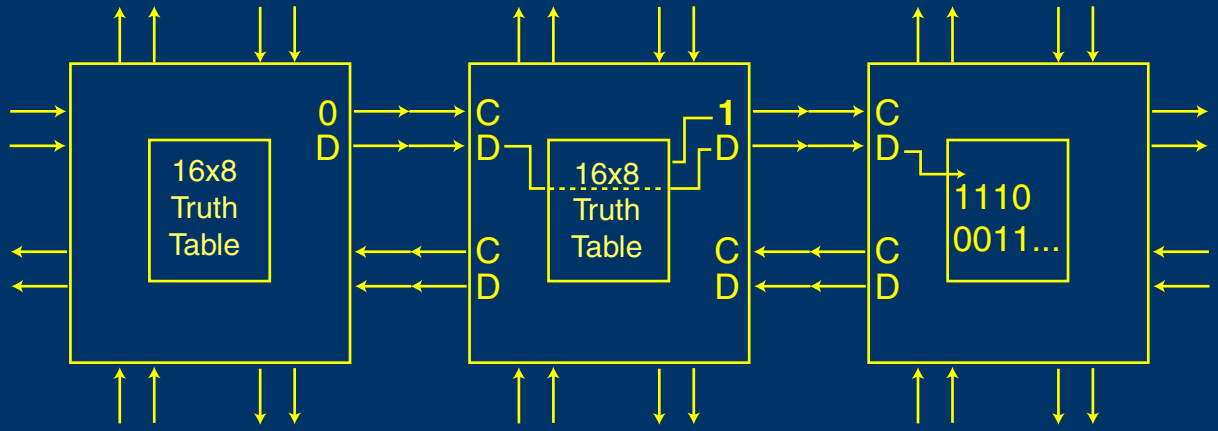
X May Configure Y



X Y Z

Y is in C Mode

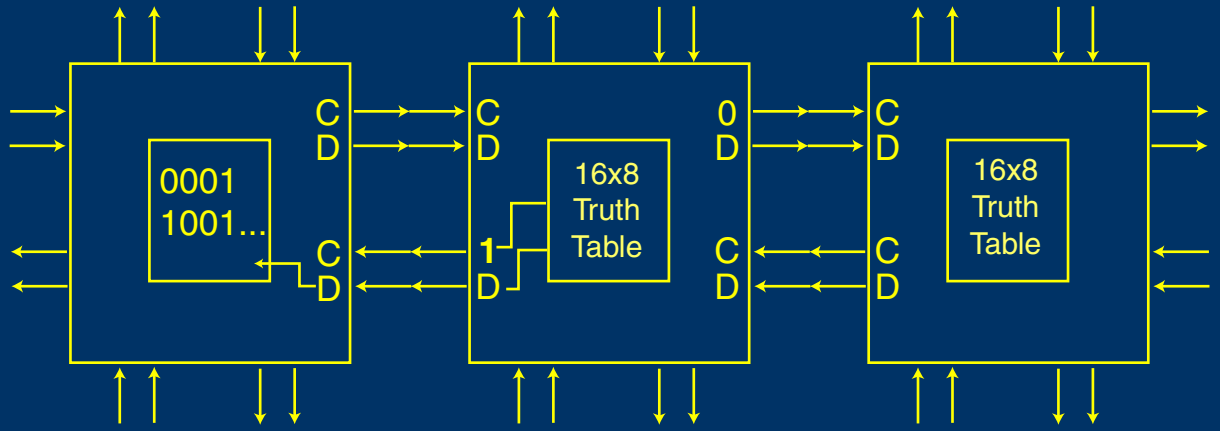
Y May Then Configure Z



X Y Z

Z is in C Mode

Or Y May Reconfigure X



X Y Z

X is in C Mode

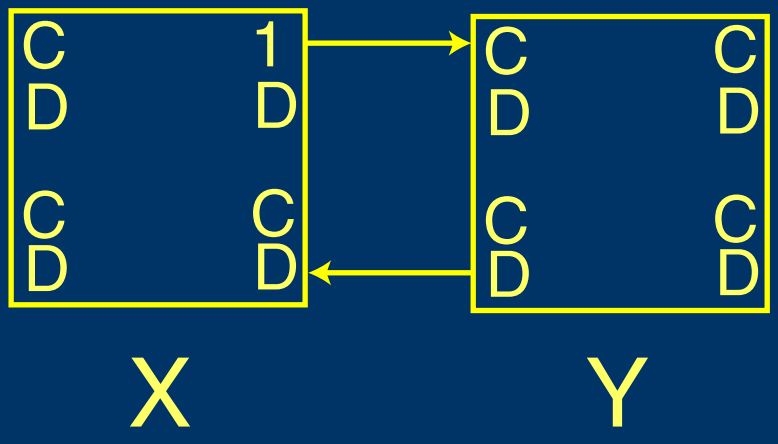
SELF DUALITY

There is a perfect interchangability between *configurers* and *configurees*

What Can Cells Do?

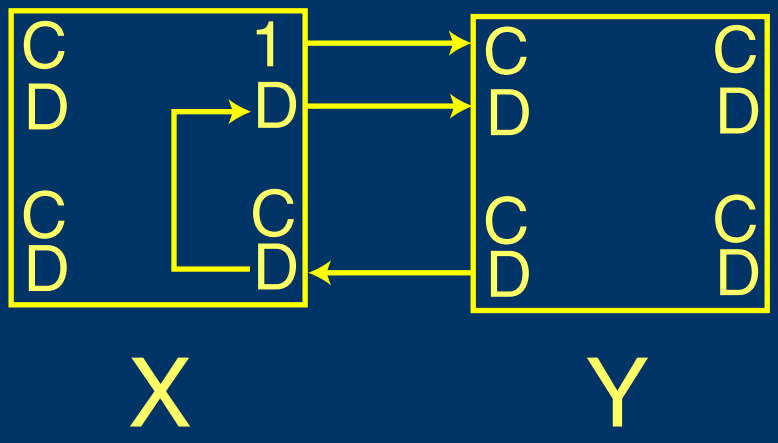
- Logic circuits
- Wires
- Cell reader
- Non-destructive cell reader
- Cell replicator
- Wire builder
- Guard cell/wall/ring

Cell Reader



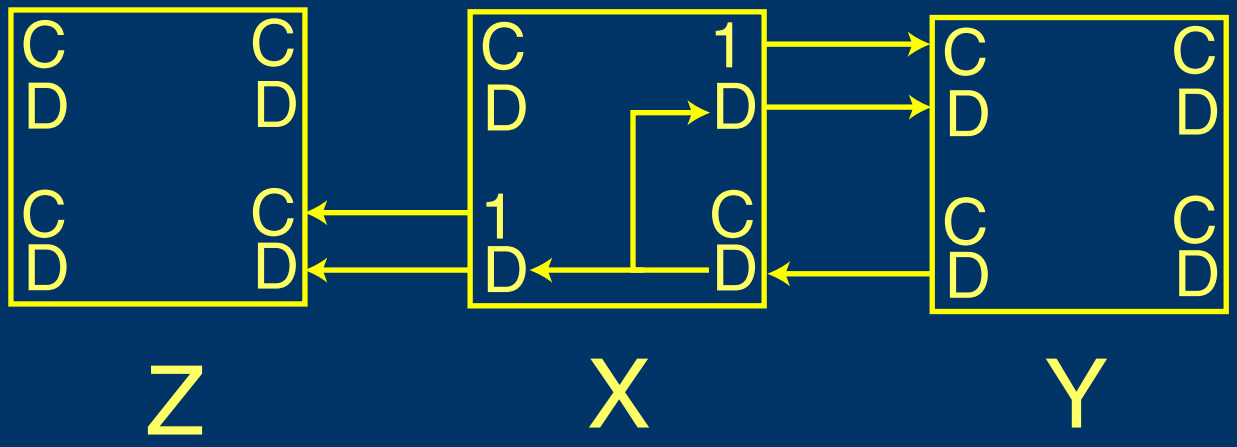
X is reading Y's Truth Table

Non-Destructive Cell Reader



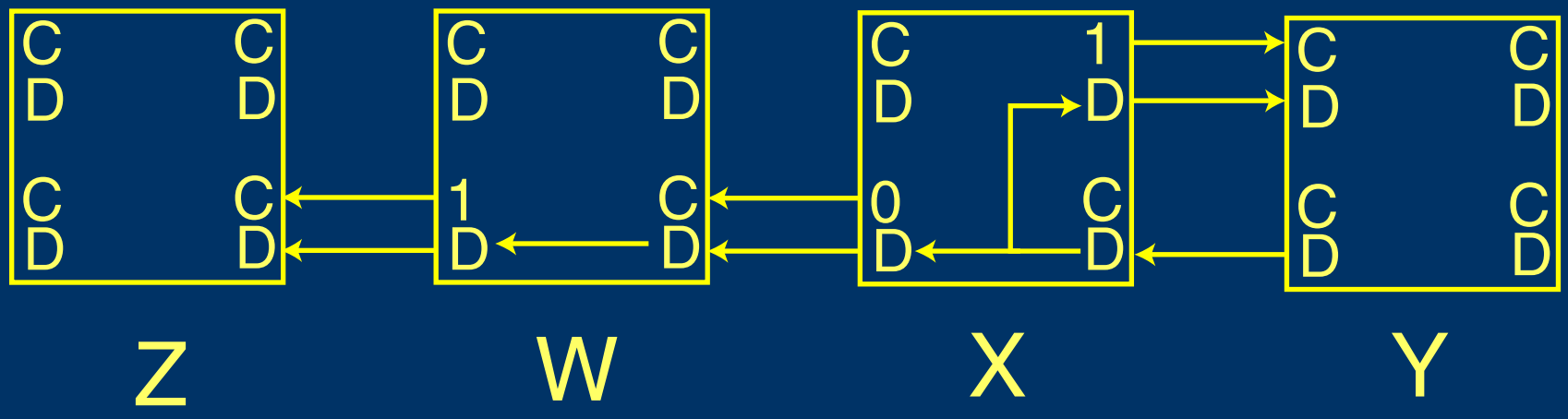
X is reading Y's Truth Table
and copying it back to Y

Cell Replicator



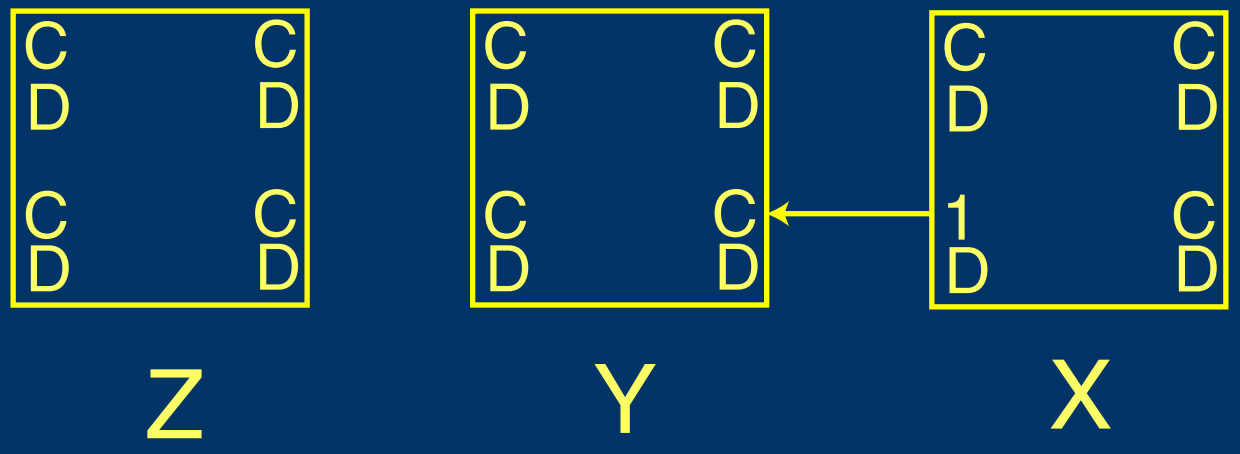
X is reading Y's Truth Table,
copying it back to Y,
and also copying it to Z

Remote Cell Replicator



X is reading Y's Truth Table,
copying it back to Y,
and also copying it to Z

Guard Cell

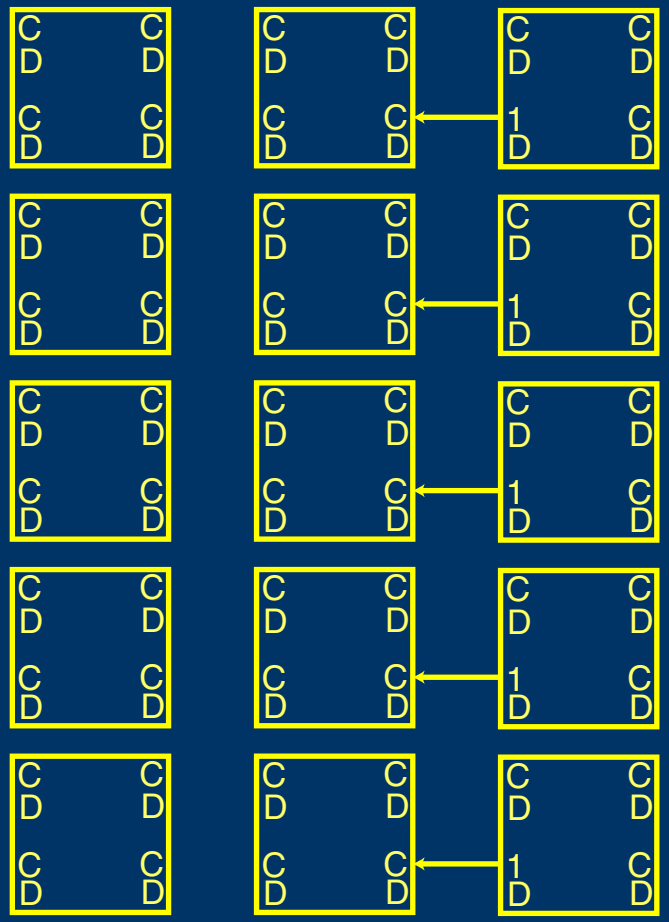


X is "guarding" cell Y

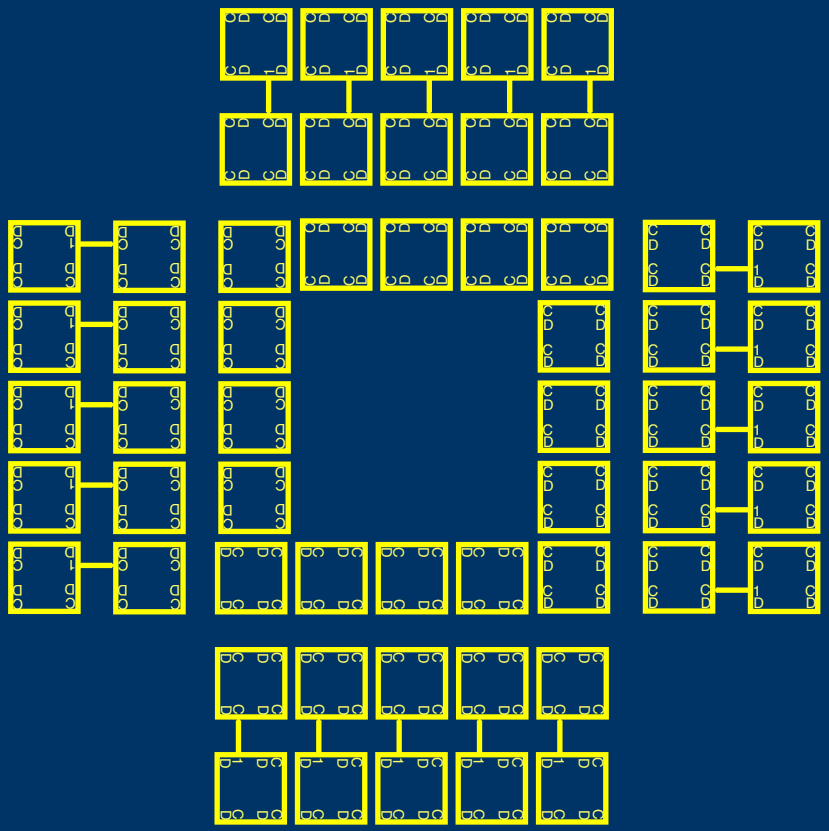
Y can not assert its C outputs

Thus, Z can not use Y to configure X

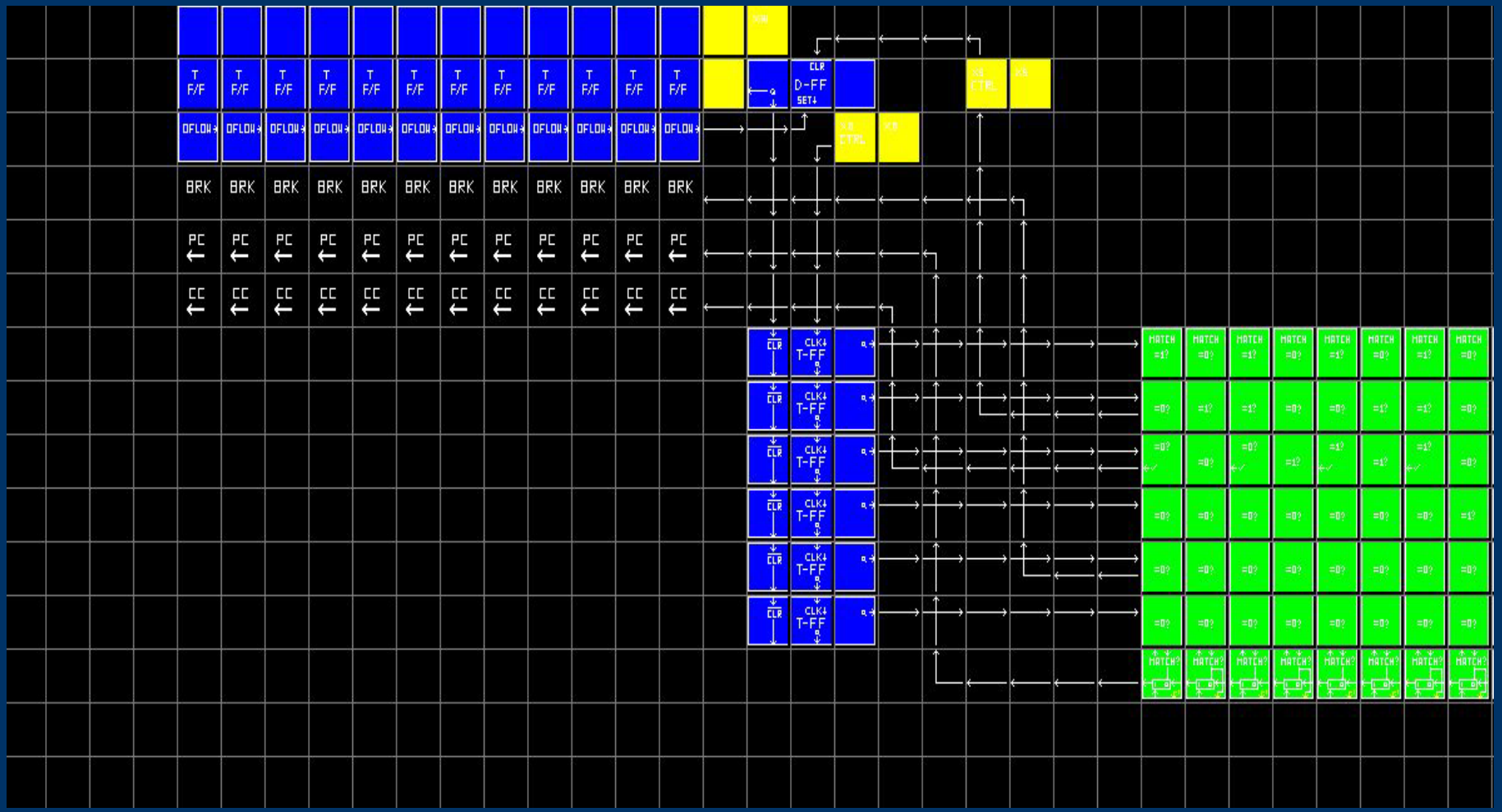
Guard Wall

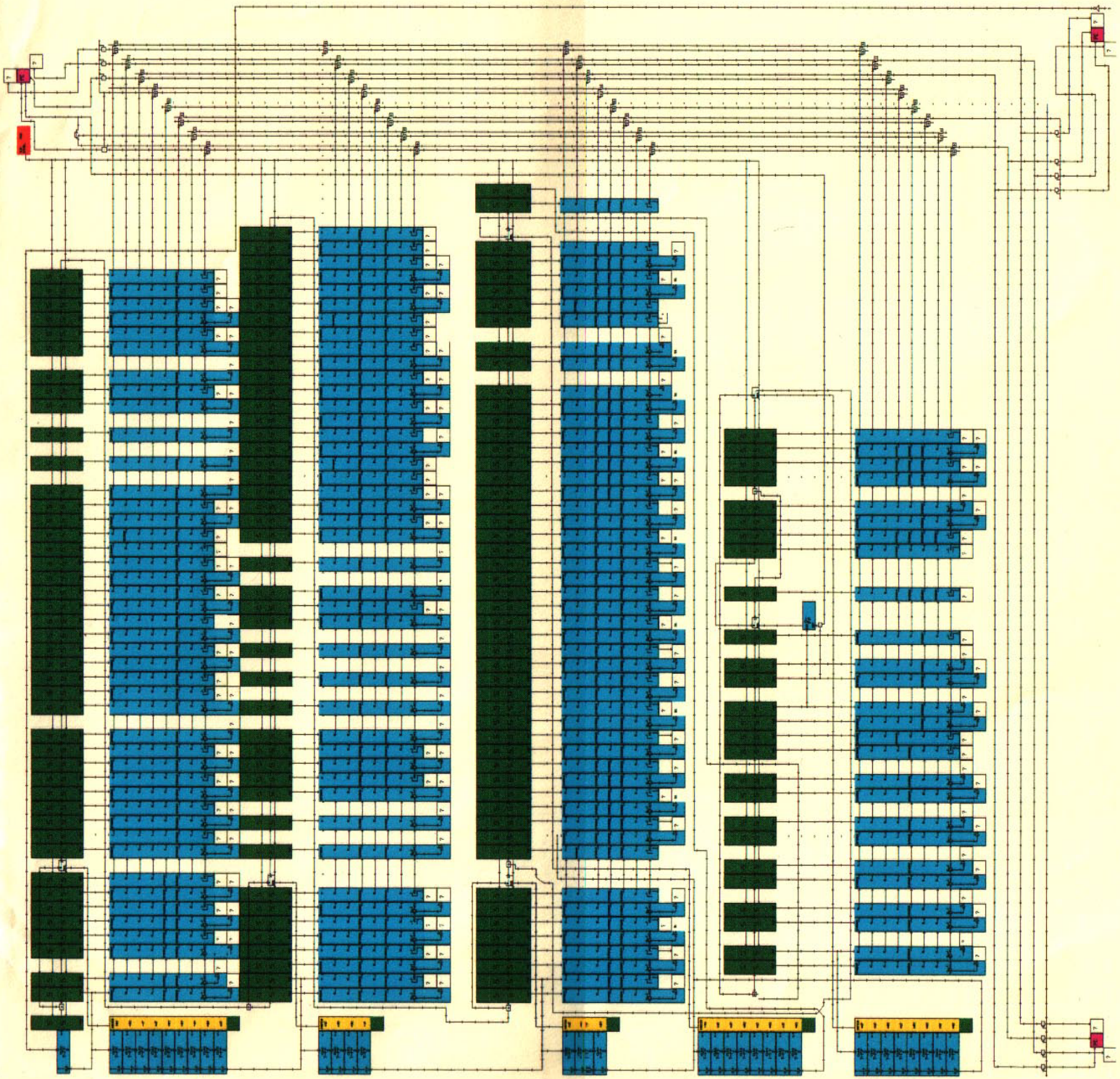


Guard Ring



Overflow-Resistant Counter



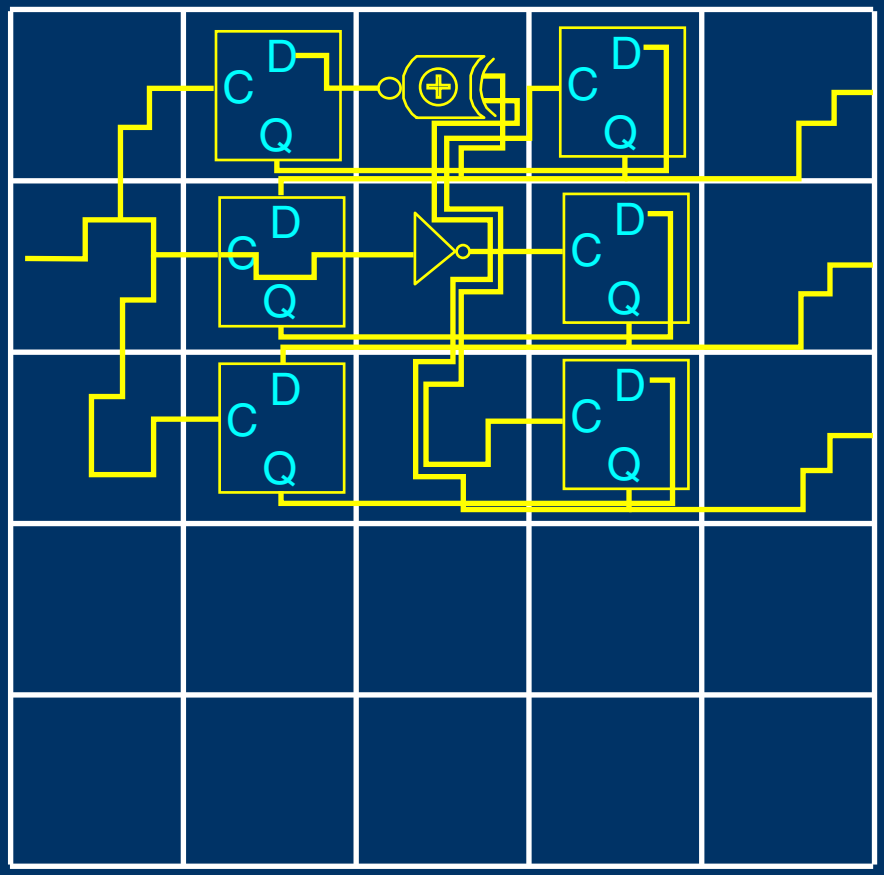


CASE STUDY: Autonomous Self-Repairing Circuits

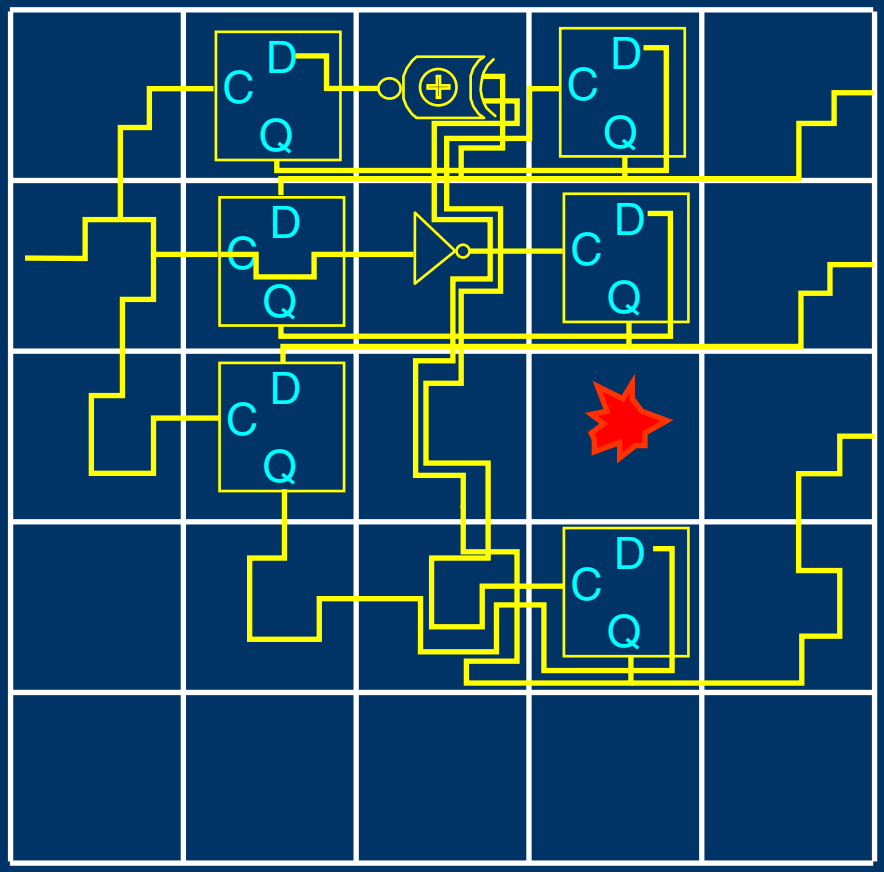
Funded in part under
NASA SBIR
NAS2-01049

February-August 2001

Primary Goals



Primary Goals



Remote Environments

Pluto $\longleftrightarrow$ Earth=5.8 Billion Kilometers
Round-trip signal time=11 Hours

Remote Environments

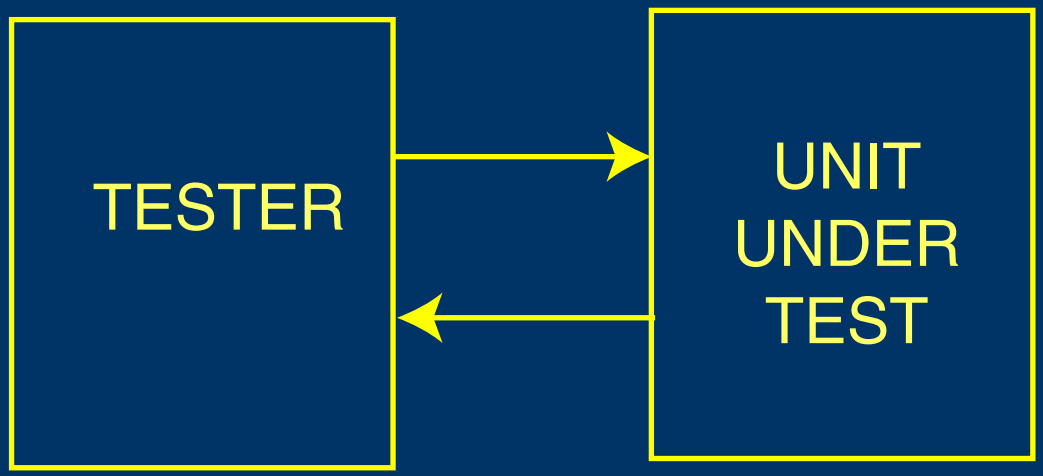
Pluto $\longleftrightarrow$ Earth=5.8 Billion Kilometers
Round-trip signal time=11 Hours

Return of autonomy

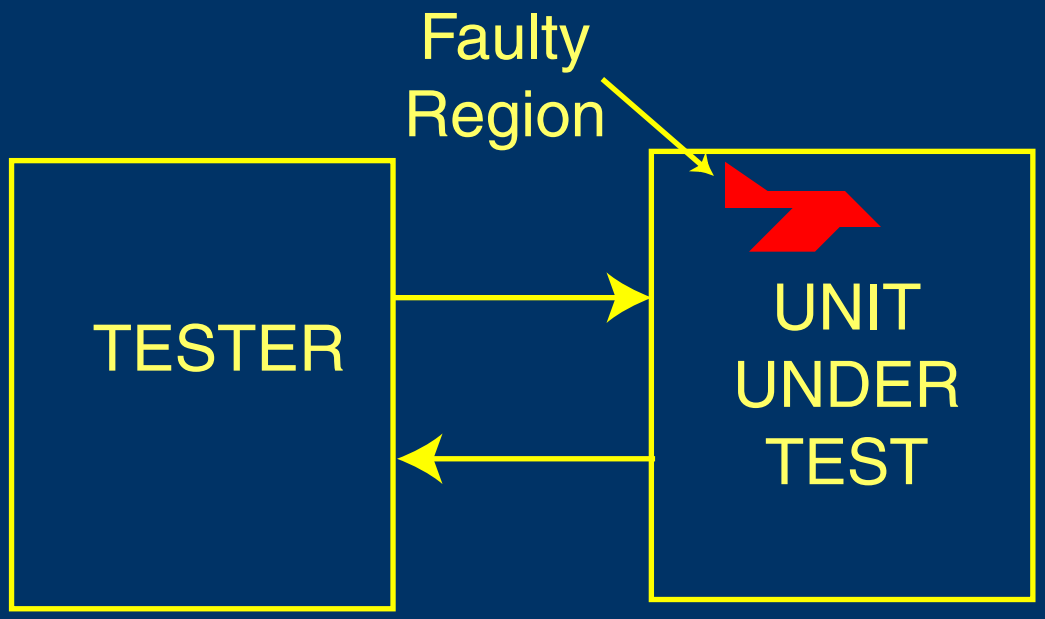
Basic Idea

Something like SCANDISK
but for hardware

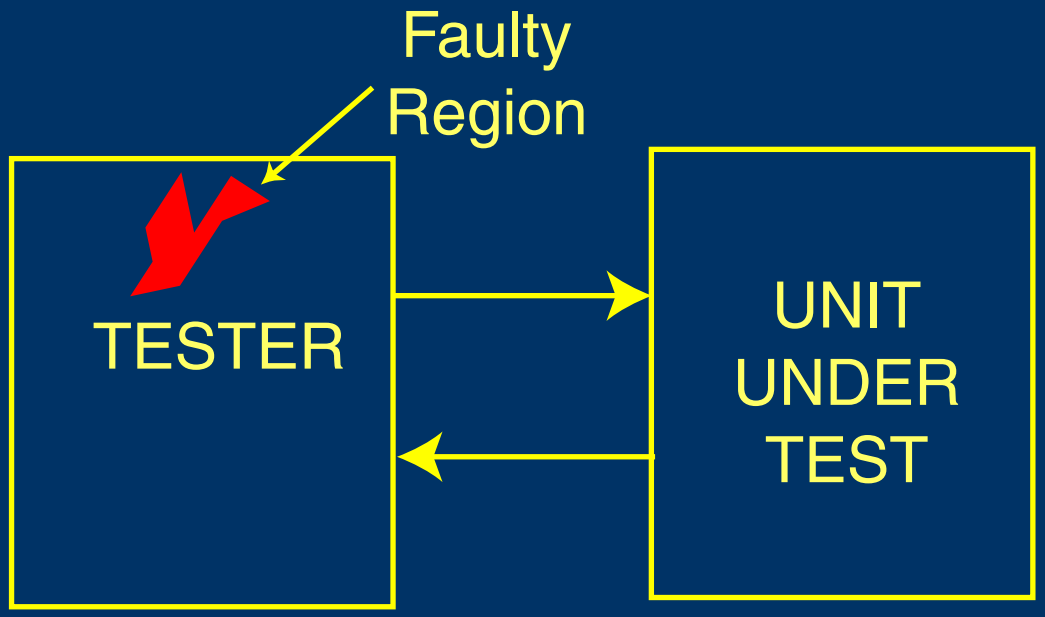
Basic Idea



Basic Idea



Basic Idea



We want a duality
between objects which are
testing for faults and
objects which are **being tested**
for faults

Supercell Approach

Create larger, more powerful atomic unit from a collection of cells

Two Phases

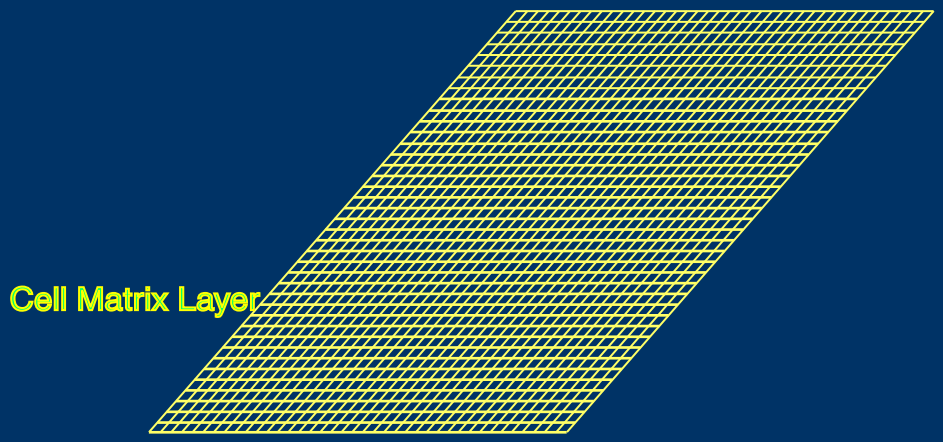
Tiling Phase:

- Test HW
- Isolate bad regions
- Tile good regions

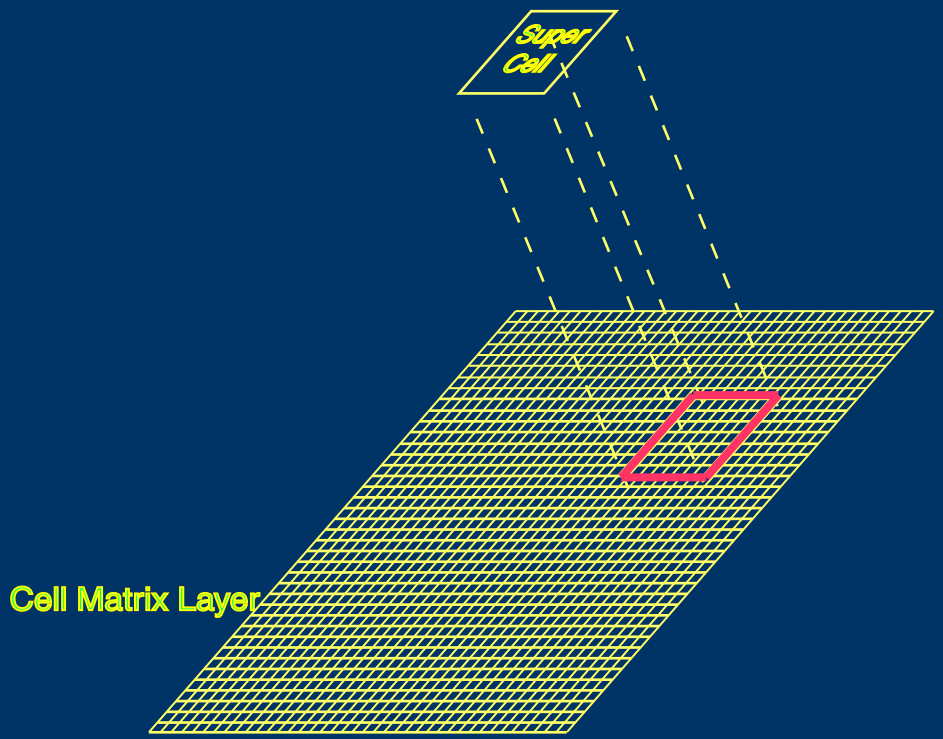
Self-Organizing Phase:

- Implement target circuit

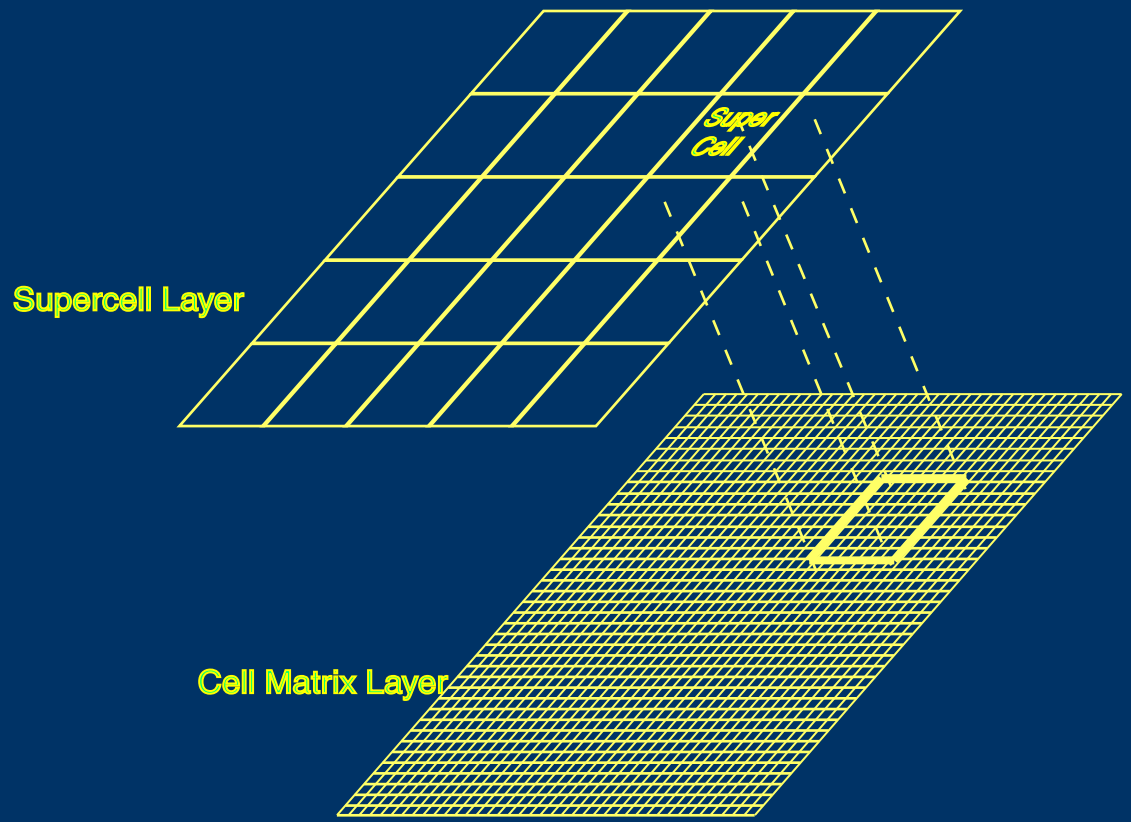
Supercell Hierarchy



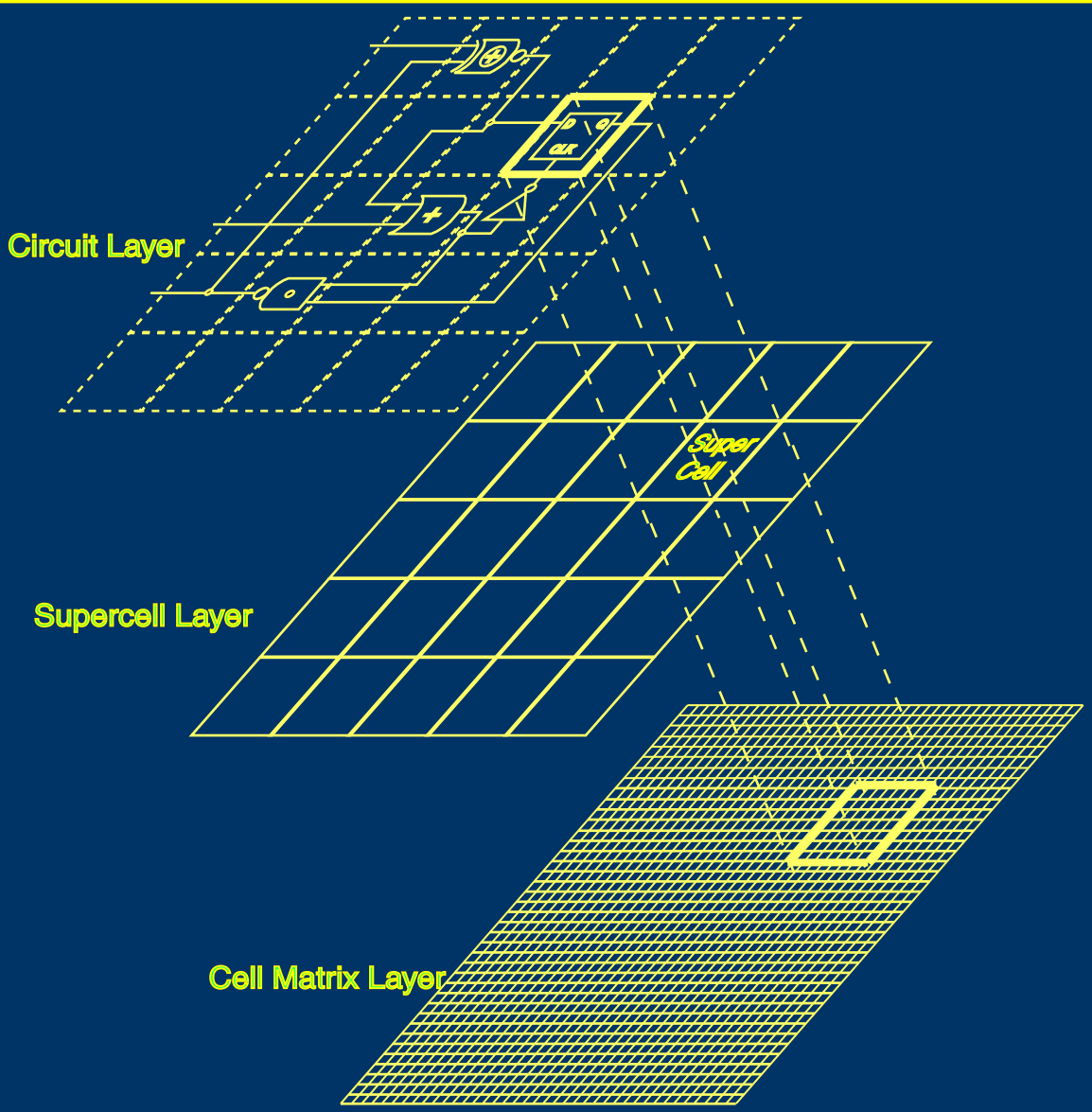
Supercell Hierarchy



Supercell Hierarchy



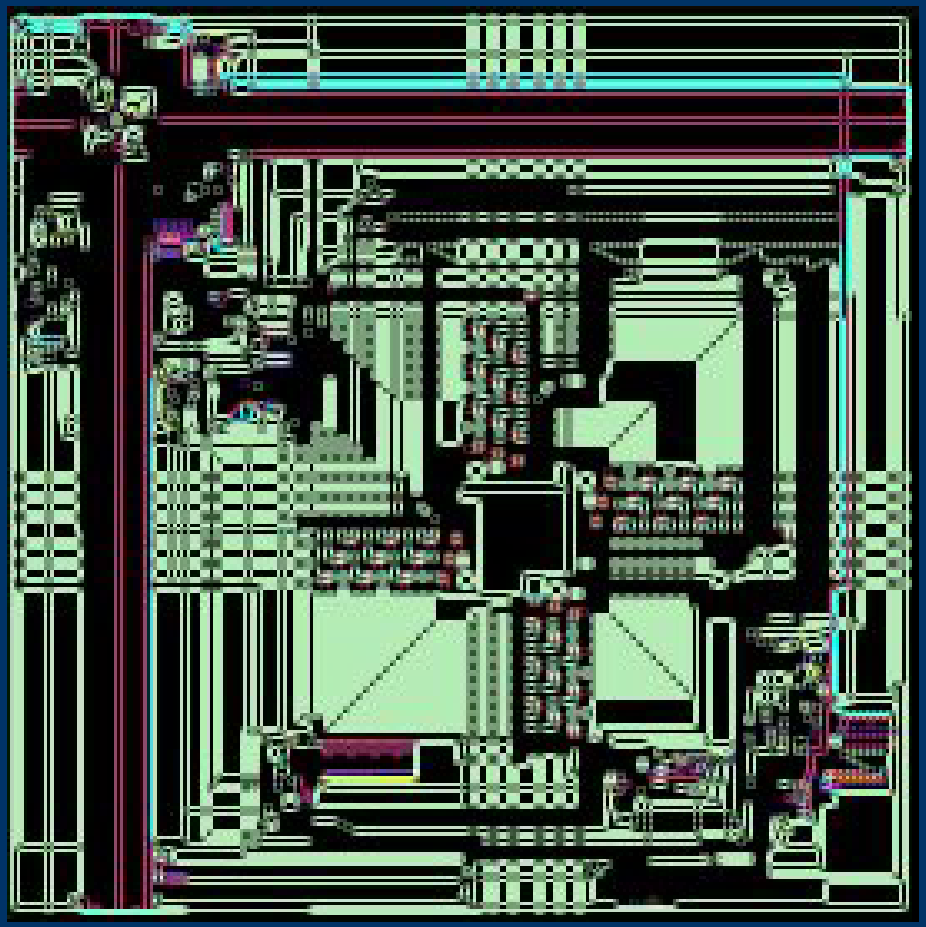
Supercell Hierarchy



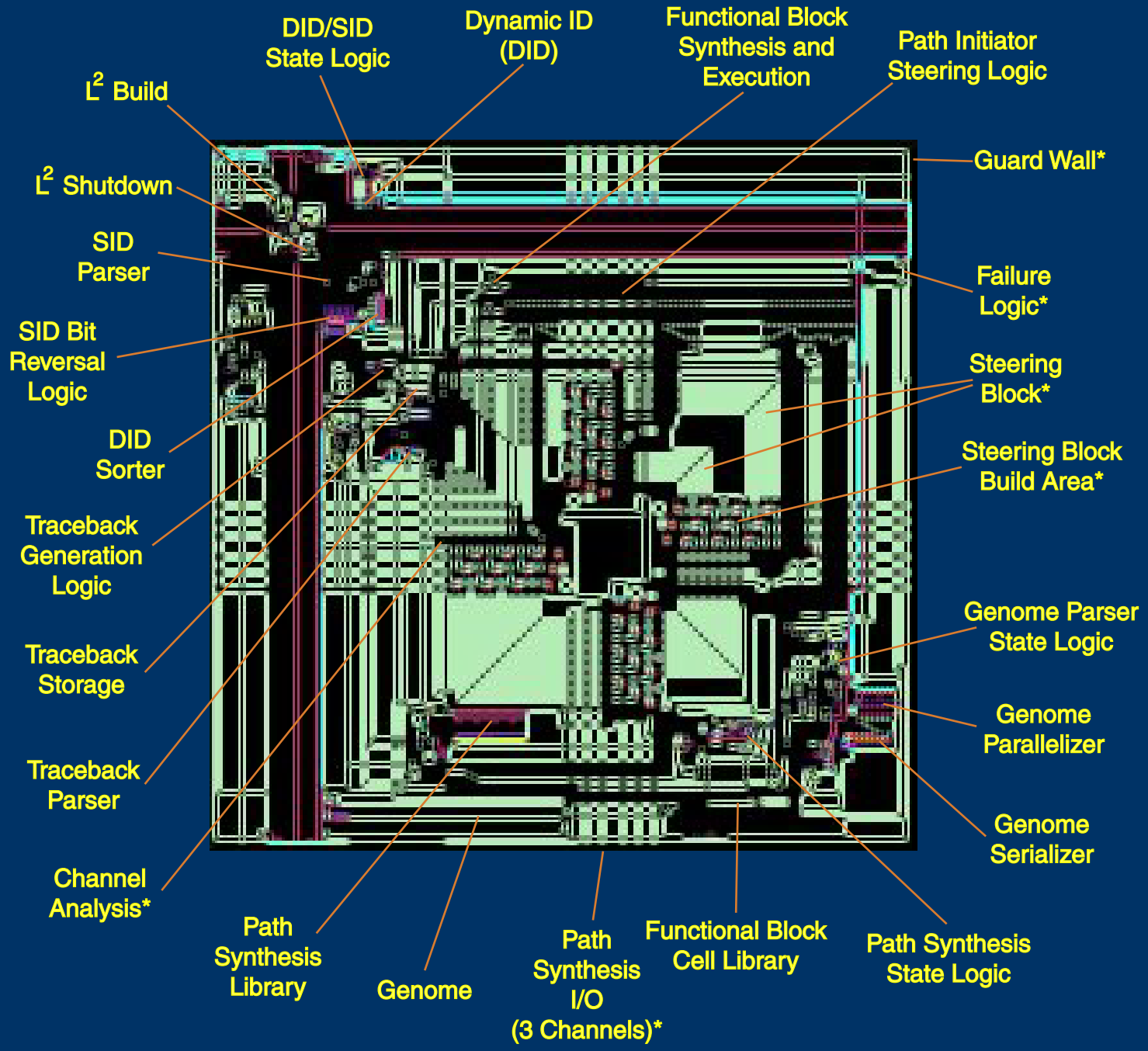
Supercell Tasks

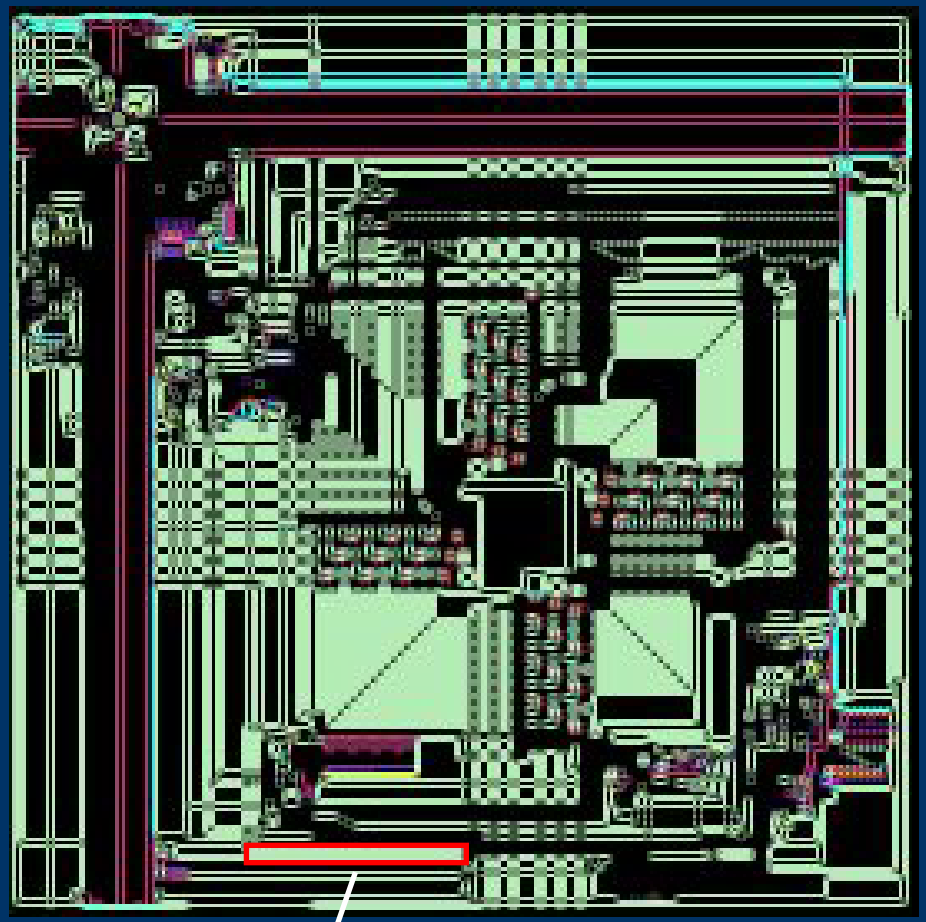
- Fault testing
- Fault isolation
- Creation of new Supercells
- Specialization into elements of final circuit
- Creation of connections to other Supercells
- Assistance of other Supercells' routing requirements

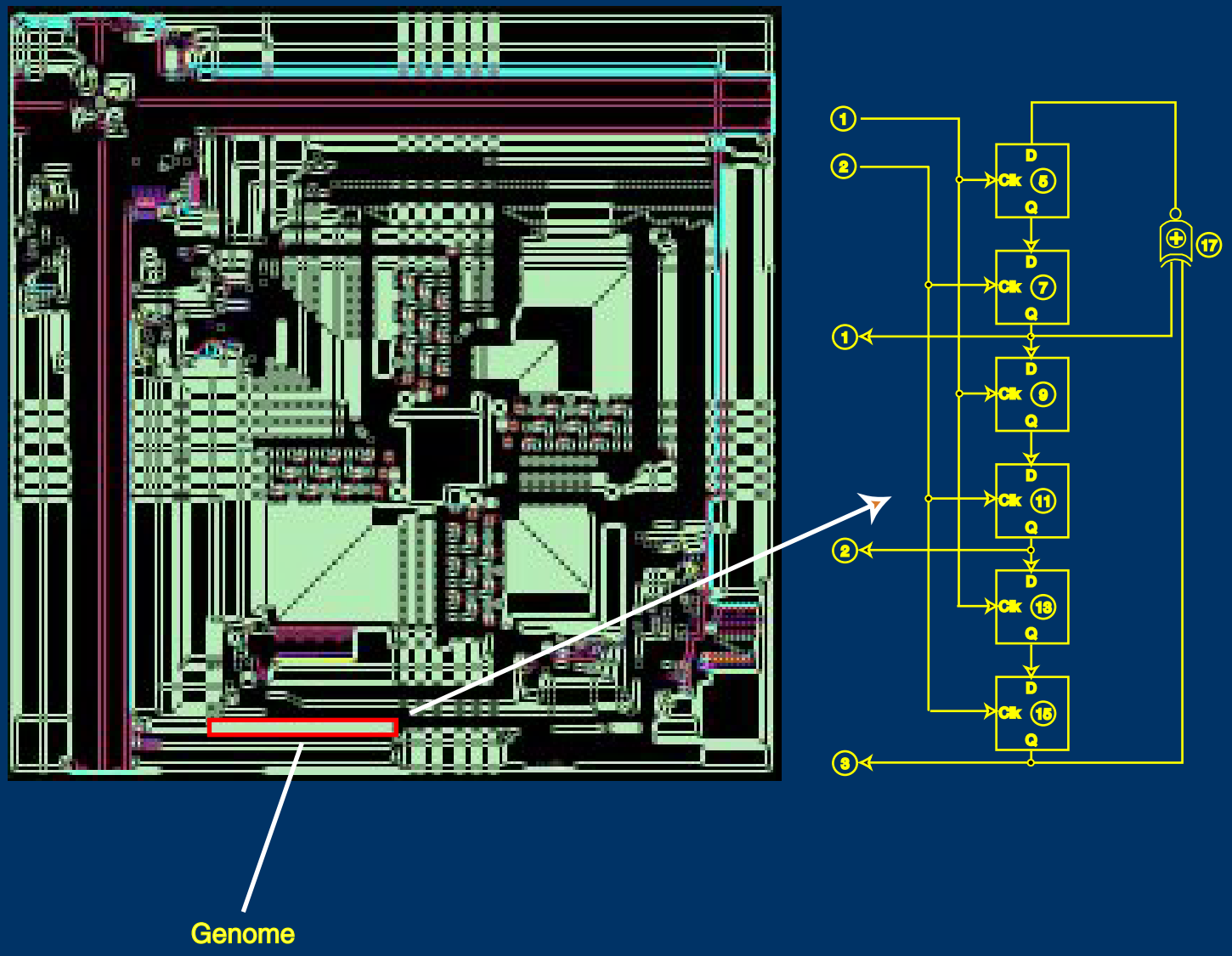
Supercell Image



72,900 Cell Matrix Cells
270x270


DID/SID State Logic
Dynamic ID (DID)
Functional Block Synthesis and Execution
Path Initiator Steering Logic
L2 Build
L2 Shutdown
SID Parser
SID Bit Reversal Logic
DID Sorter
Traceback Generation Logic
Traceback Storage
Traceback Parser
Channel Analysis*
Guard Wall*
Failure Logic*
Steering Block*
Steering Block Build Area*
Genome Parser State Logic
Genome Parallelizer
Genome Serializer
Path Synthesis Library
Genome
Path Synthesis I/O (3 Channels)*
Functional Block Cell Library
Path Synthesis State Logic

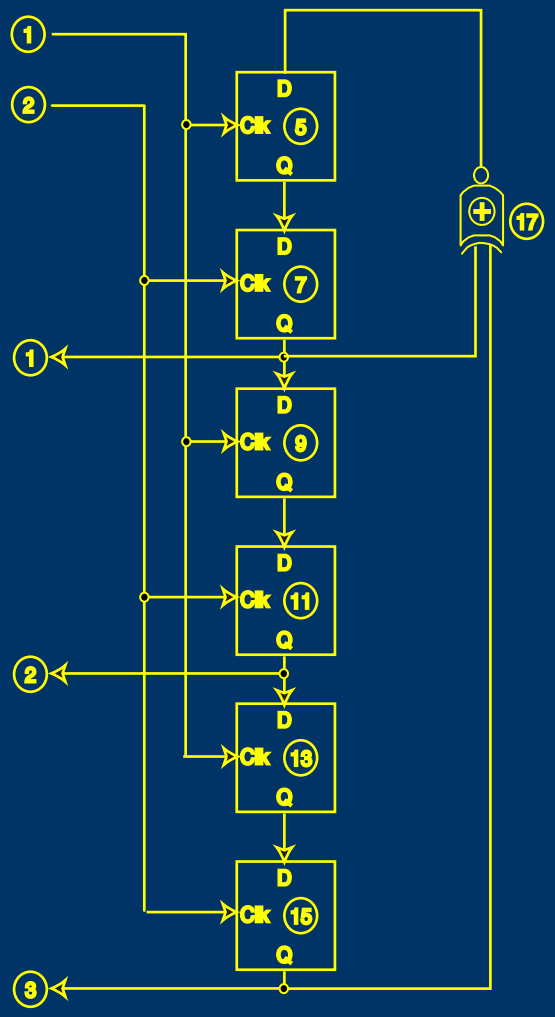

Genome


1
2
D
Clk
5
Q
D
Clk
7
Q
17
1
D
Clk
9
Q
D
Clk
11
Q
2
D
Clk
13
Q
D
Clk
15
Q
3
Genome

Genome Mapping

1	7	0	18	1	1
2	11	0	18	1	1
3	15	0	3	0	0
5	17	1	2	19	9
7	5	2	2	19	9
11	9	2	2	19	9
13	11	1	2	19	9
15	13	2	2	19	9
17	7	15	2	20	1

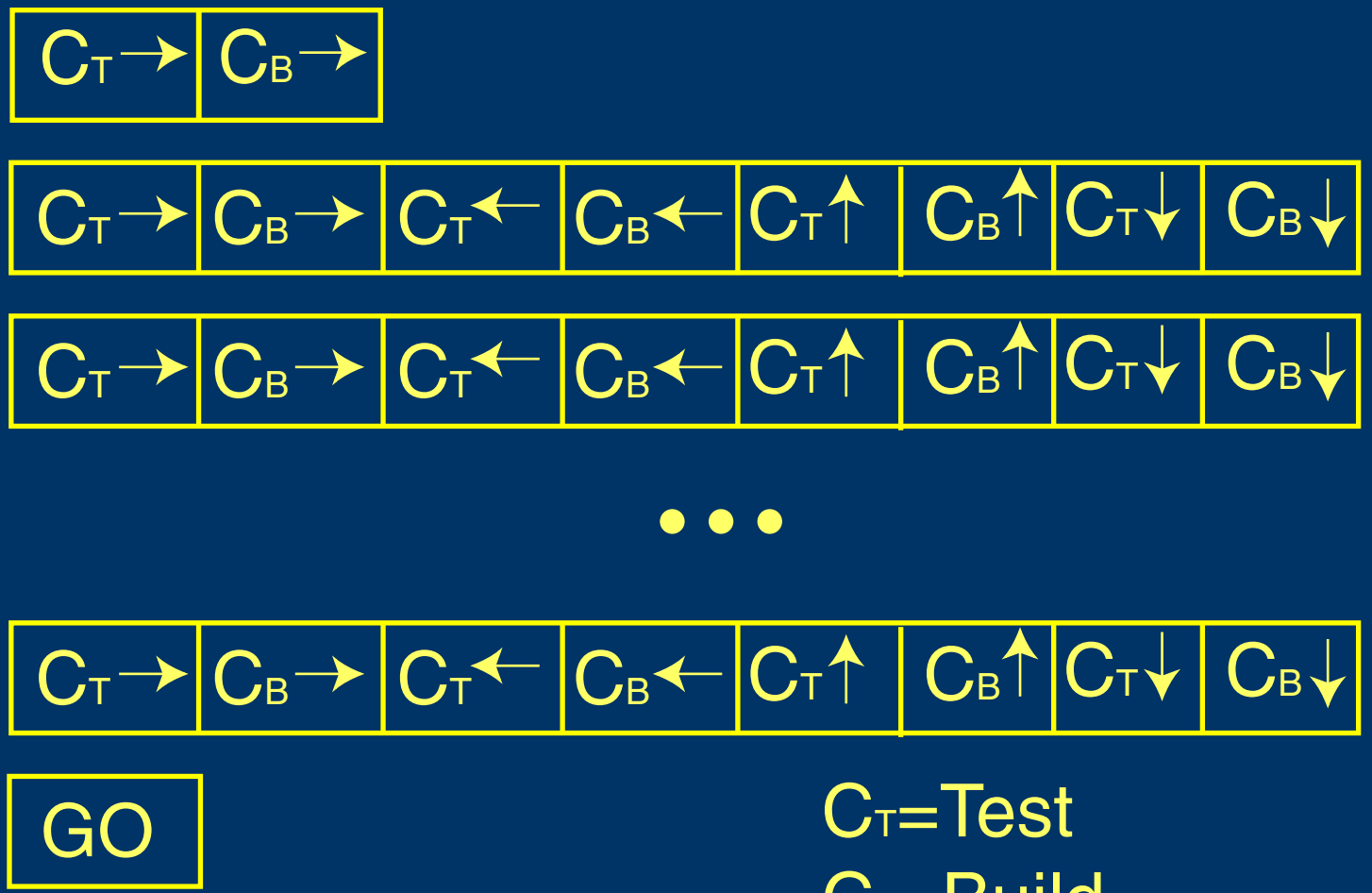
Node IDs Function



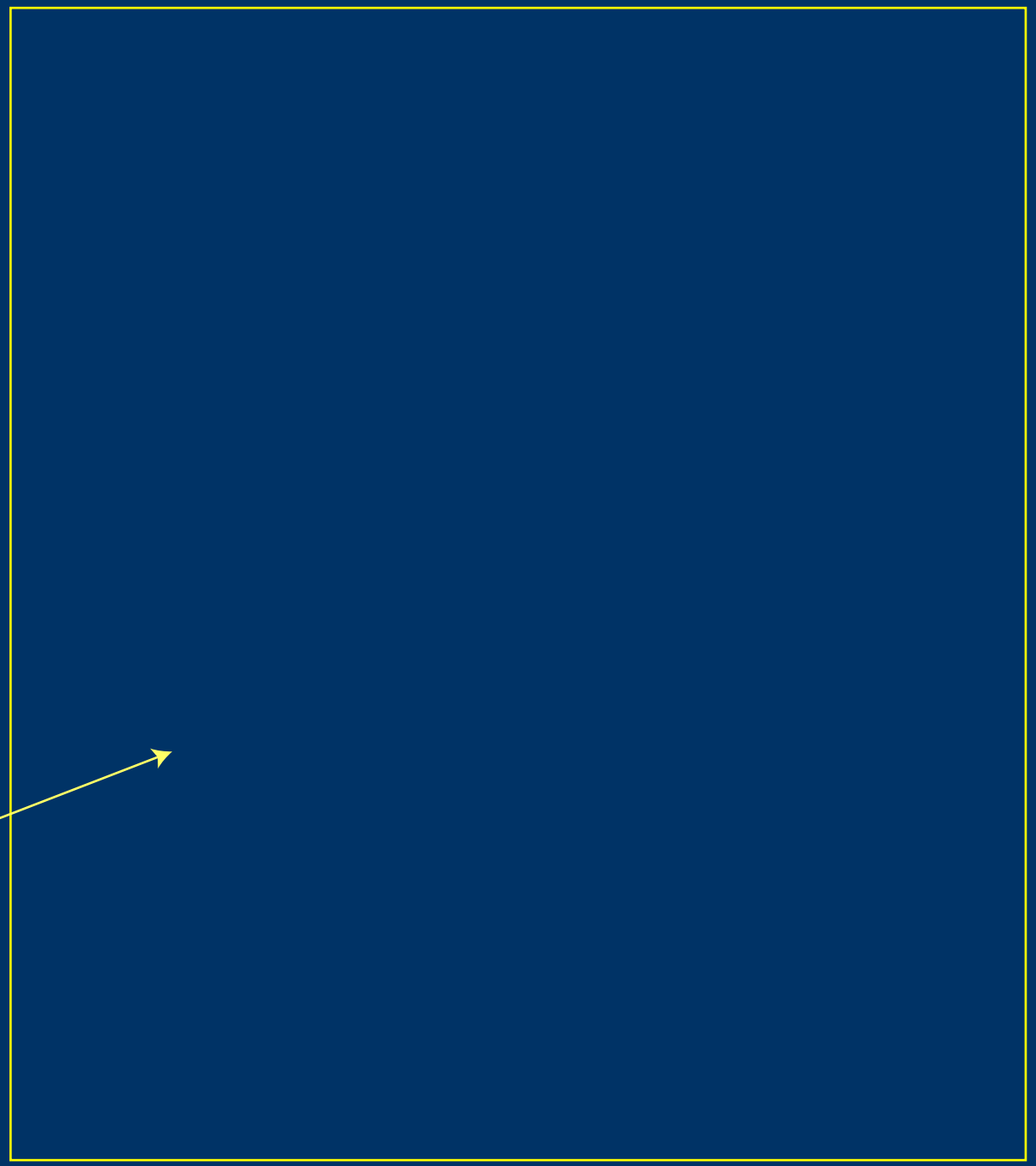
Methodology

- Code target circuit into genome
- Embed genome into Supercell layout
- Create binary configuration string "C" corrresponding to Supercell
- Send "C" into an empty Cell Matrix

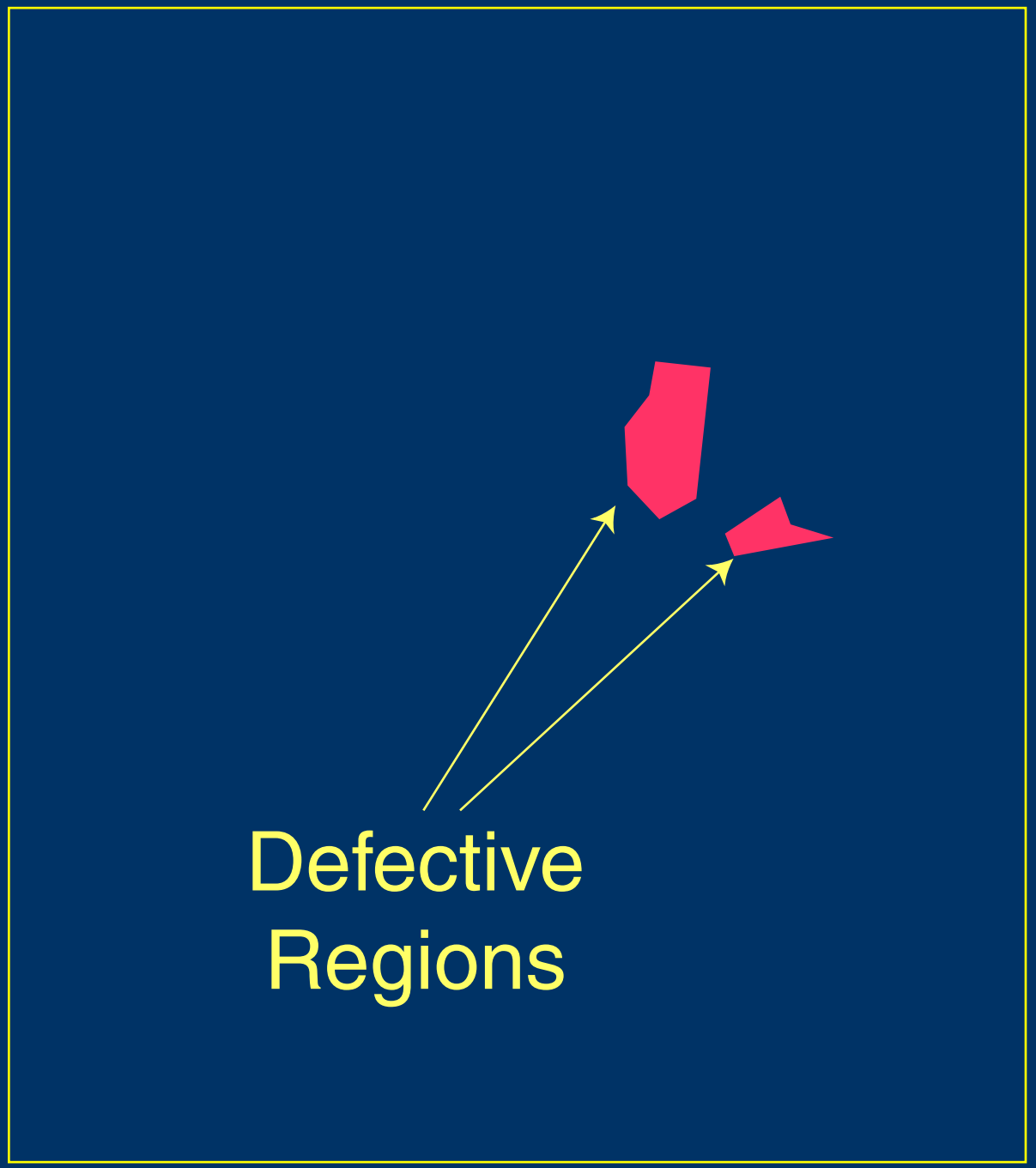
Configuration String



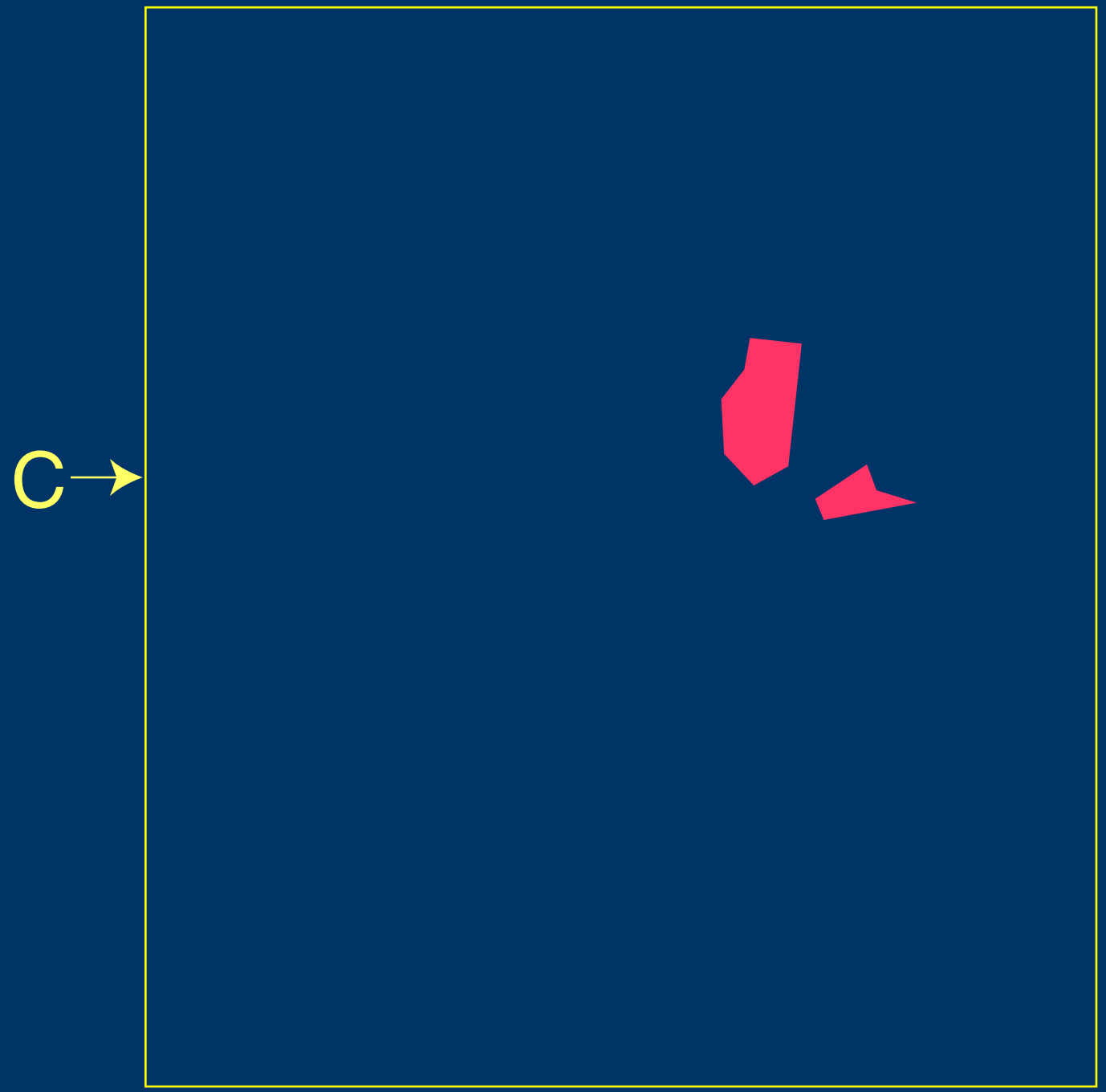
Tiling Phase



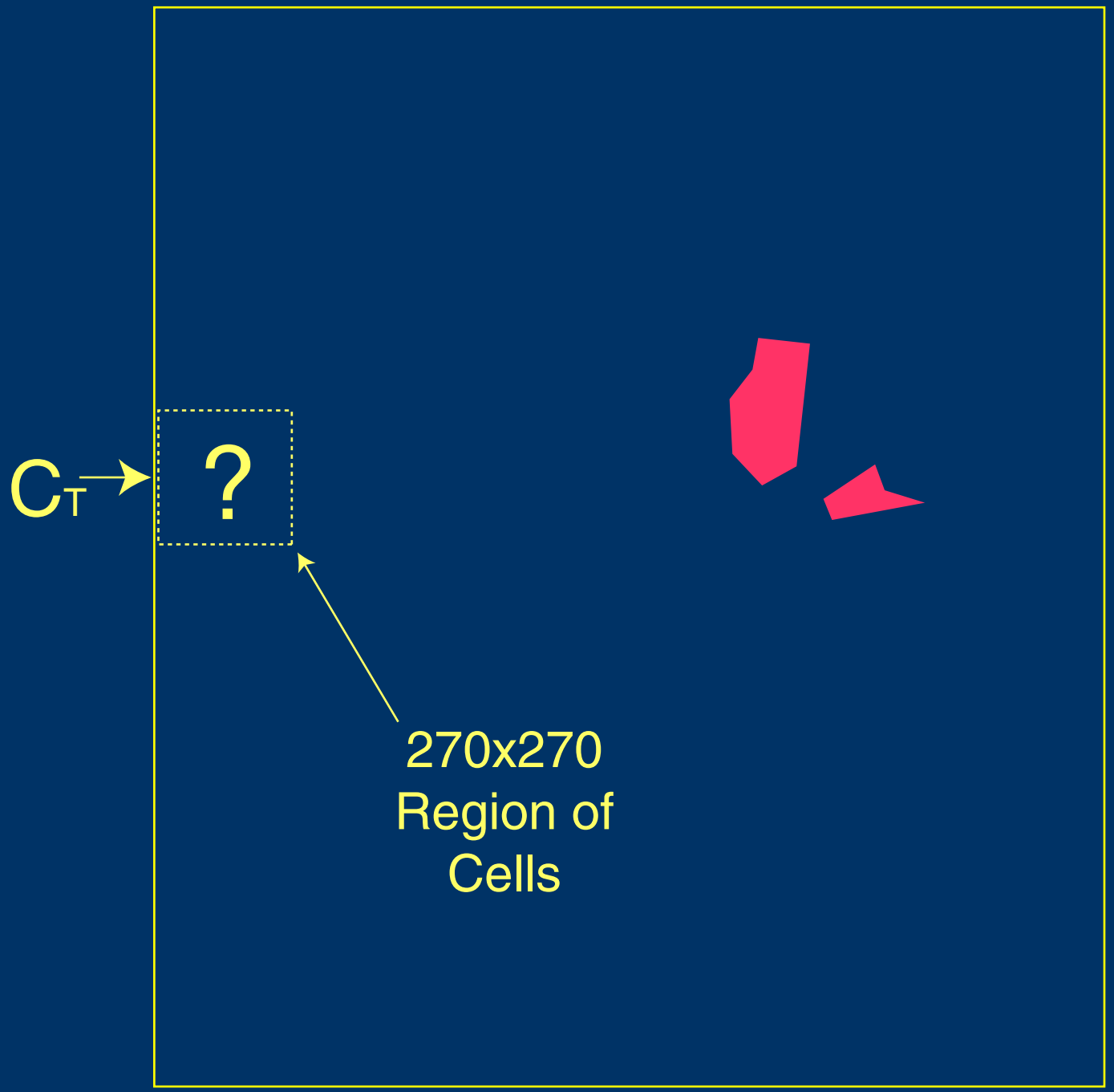
Tiling Phase



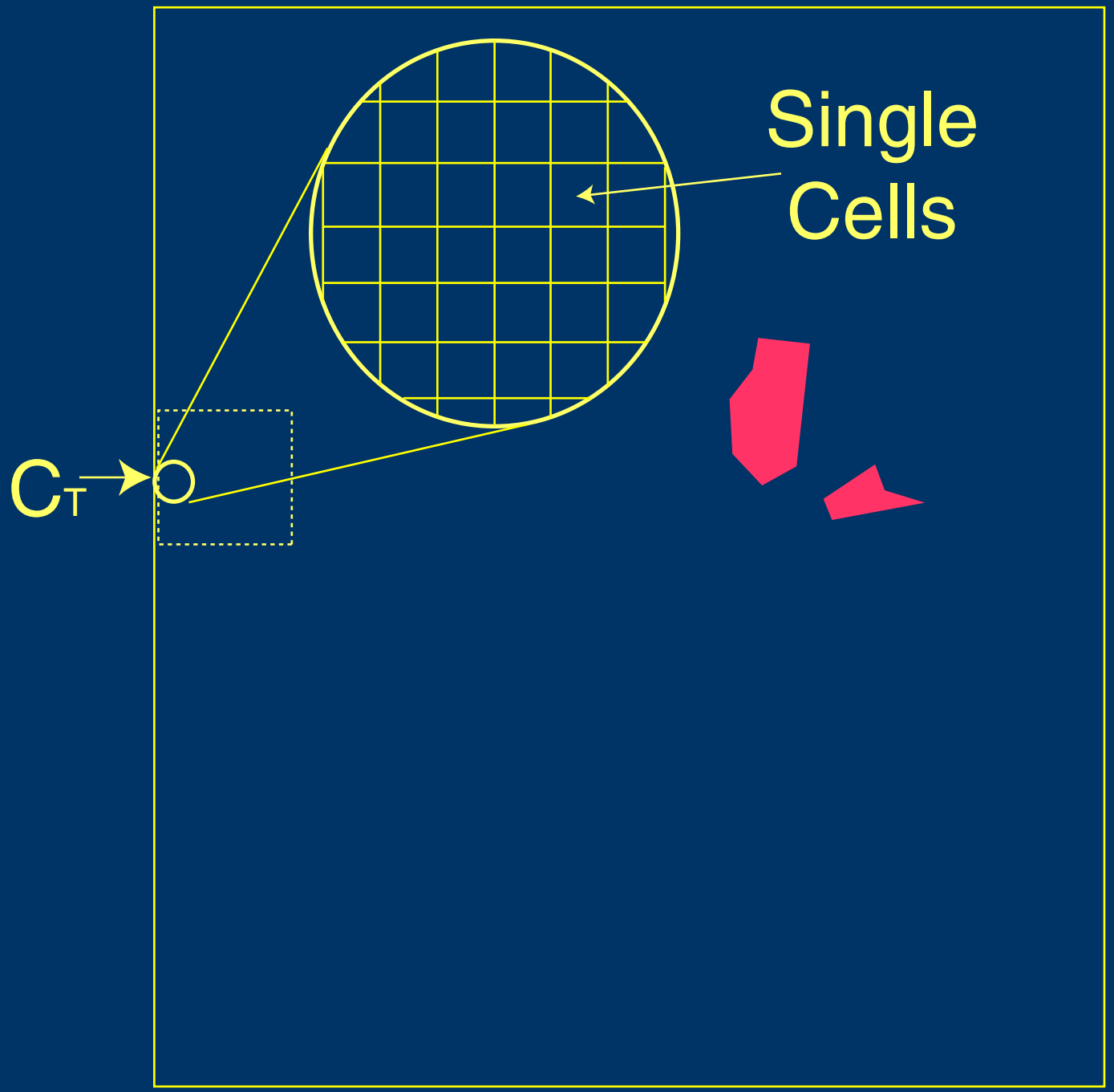
Tiling Phase



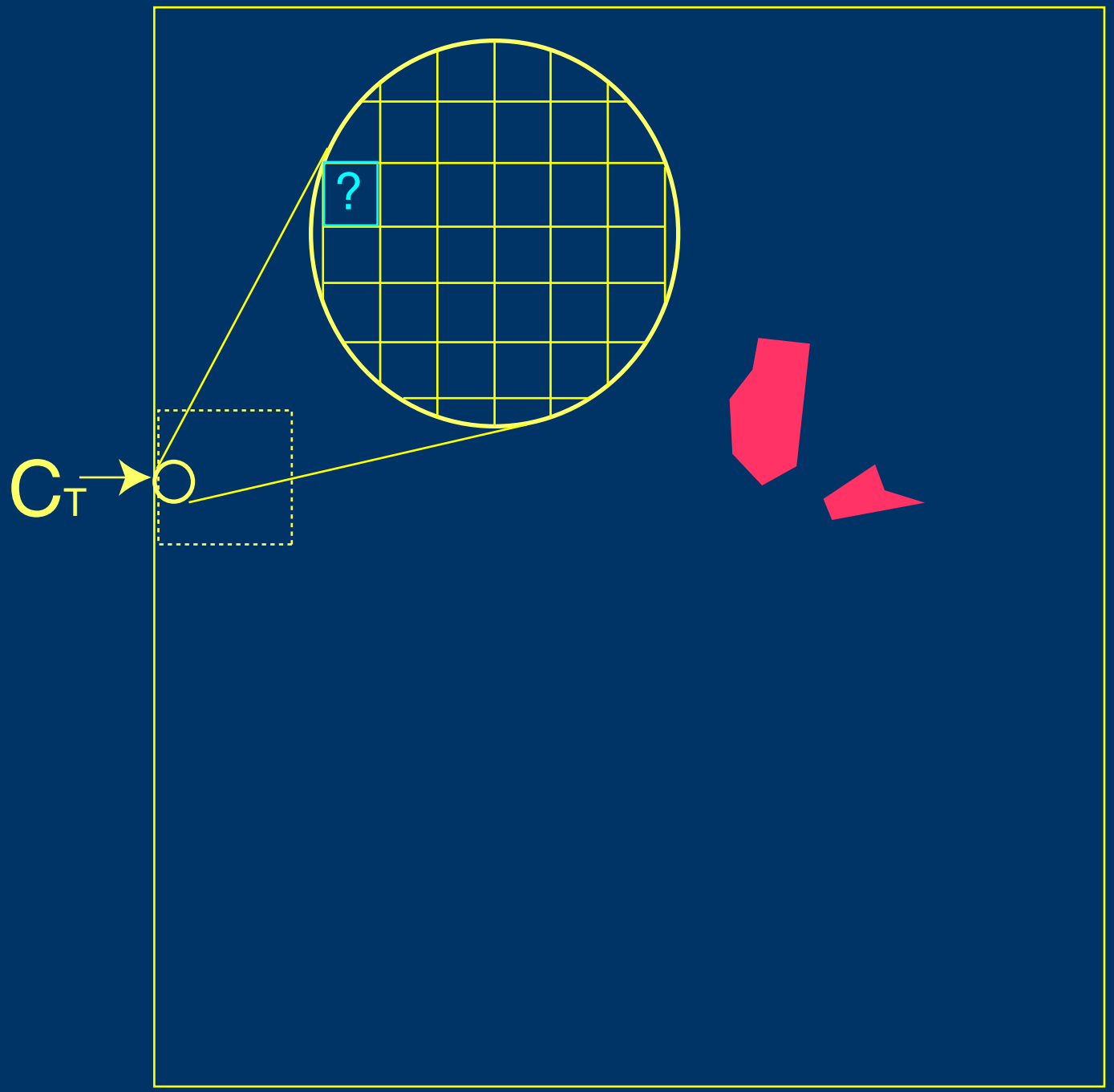
Test Phase



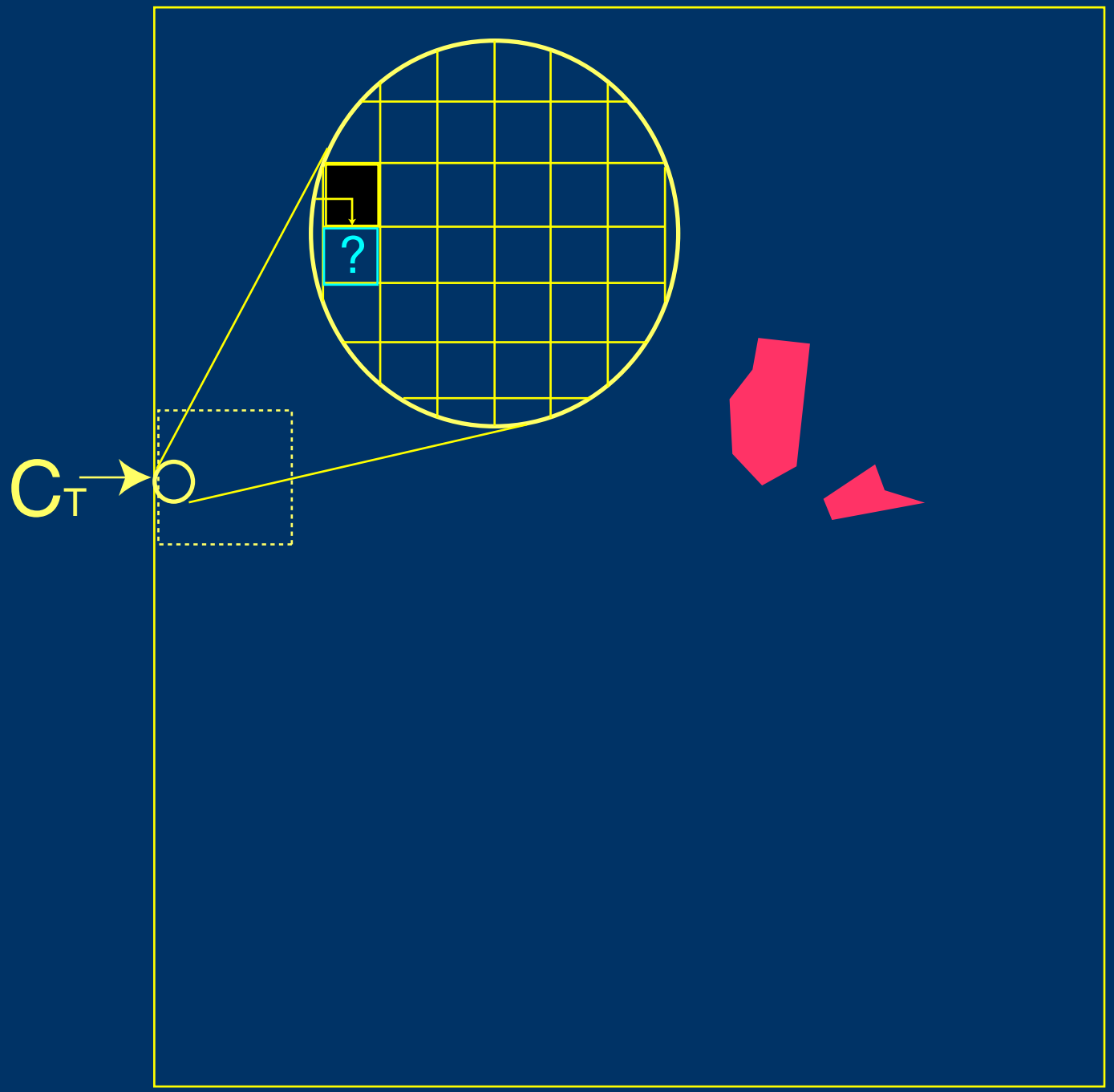
Test Phase



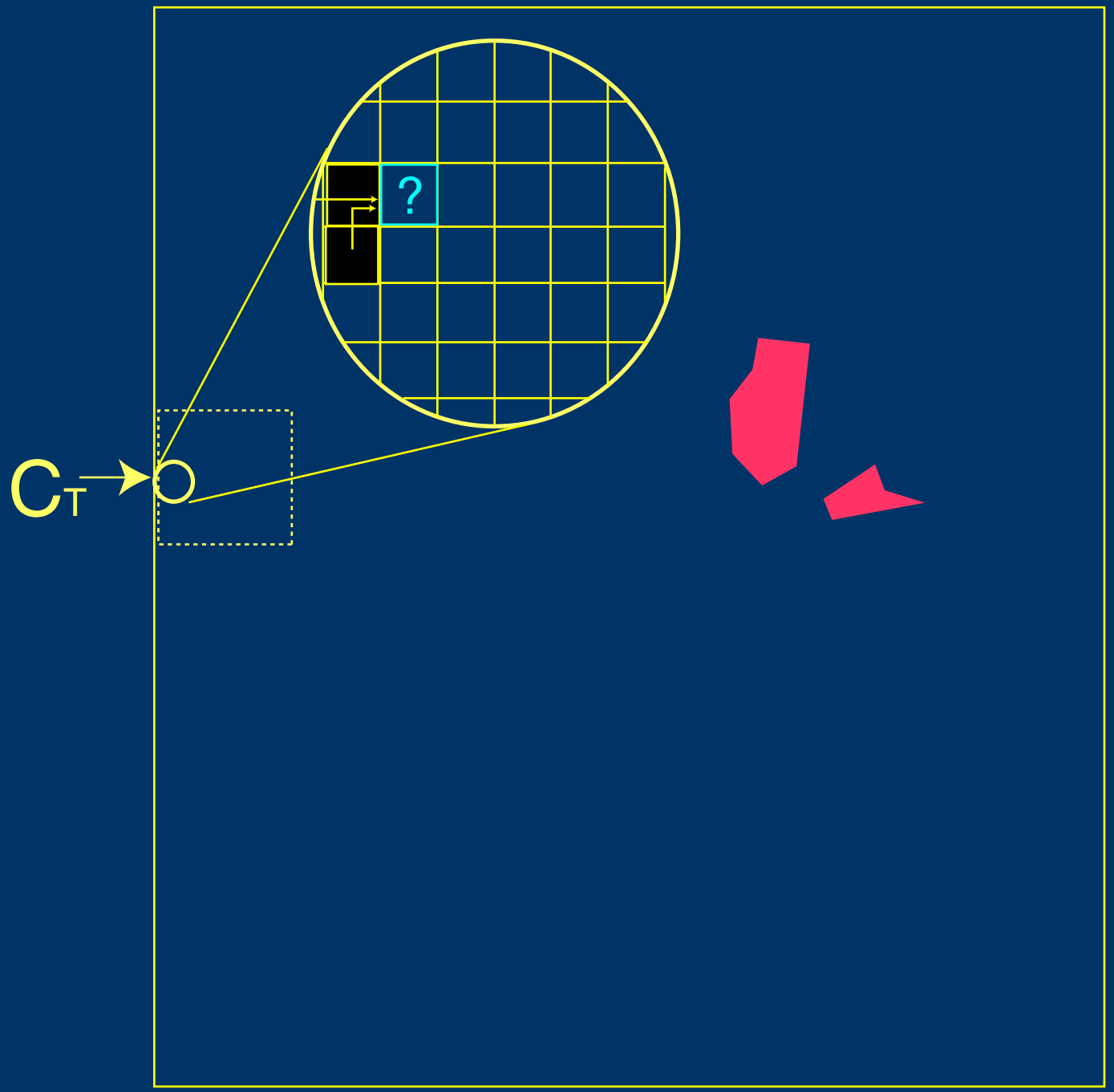
Test Phase



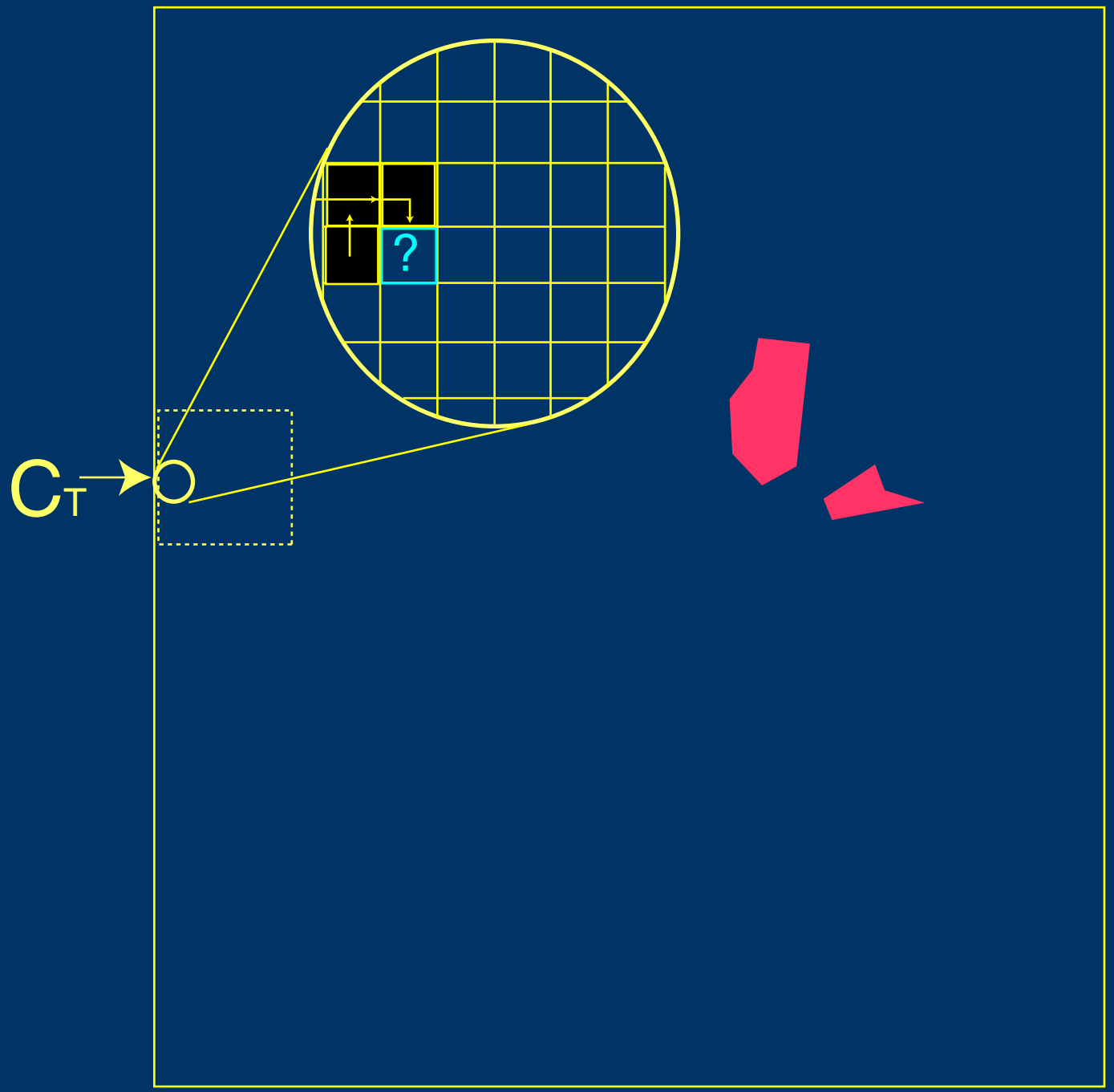
Test Phase



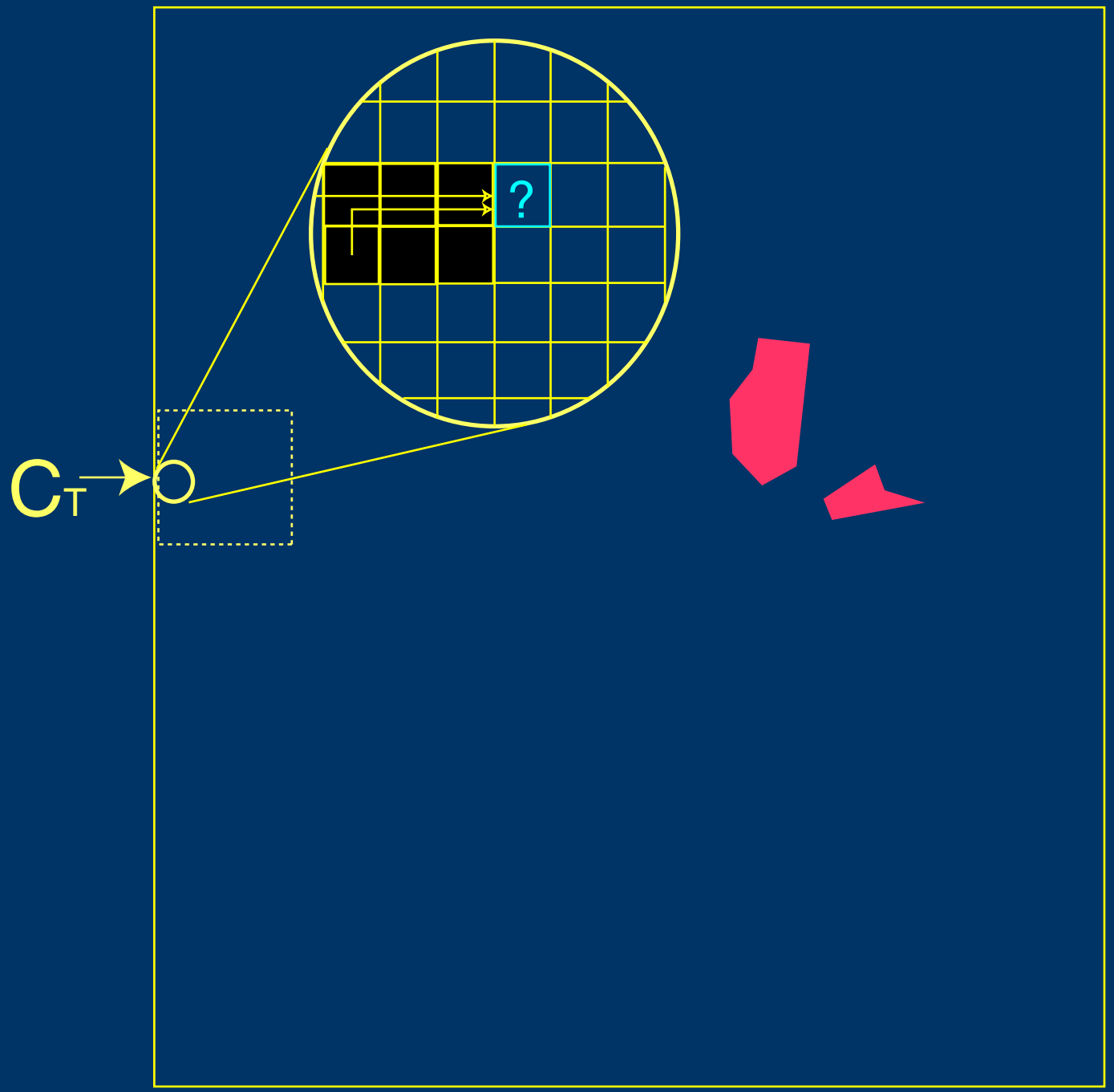
Test Phase



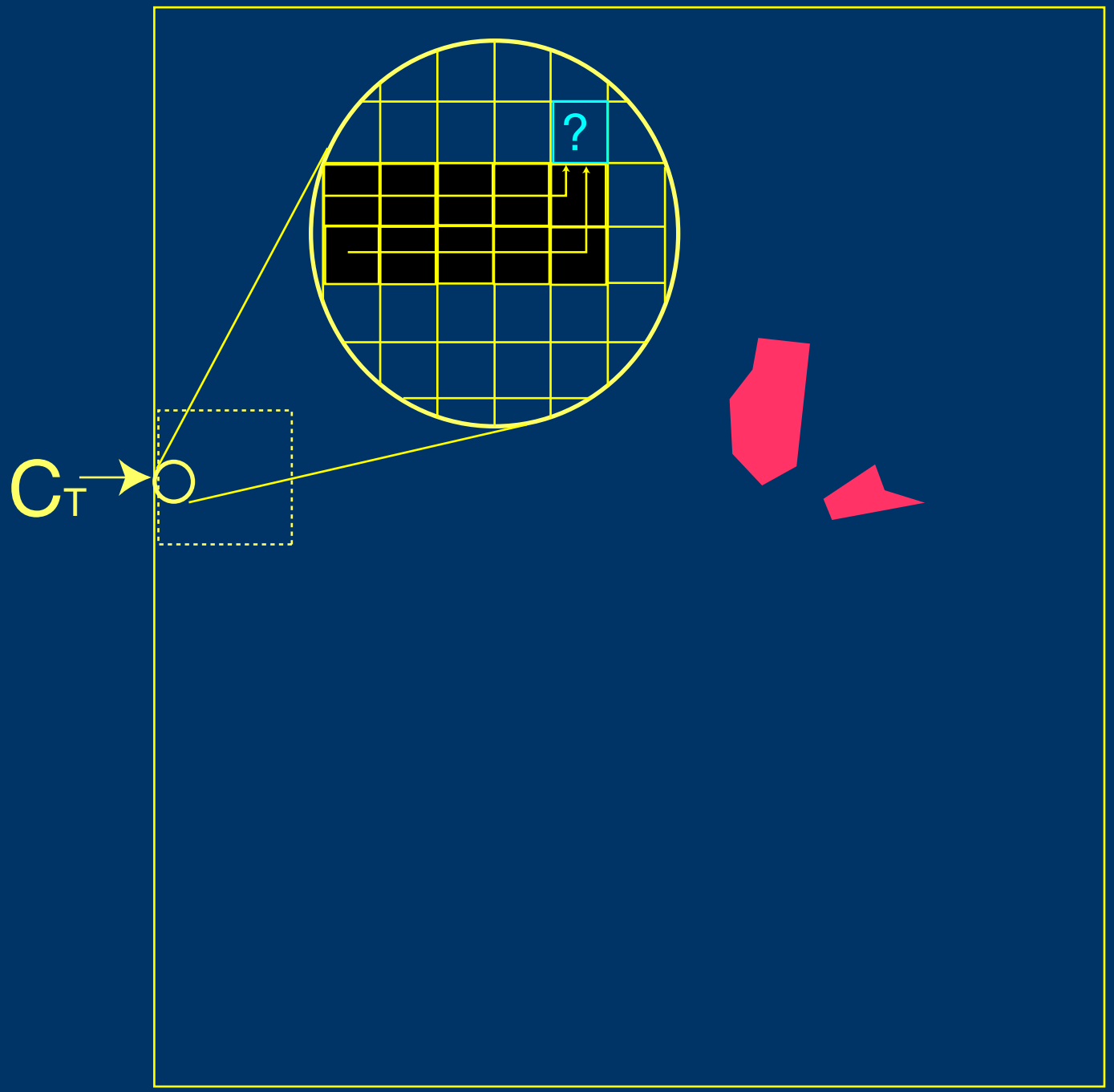
Test Phase



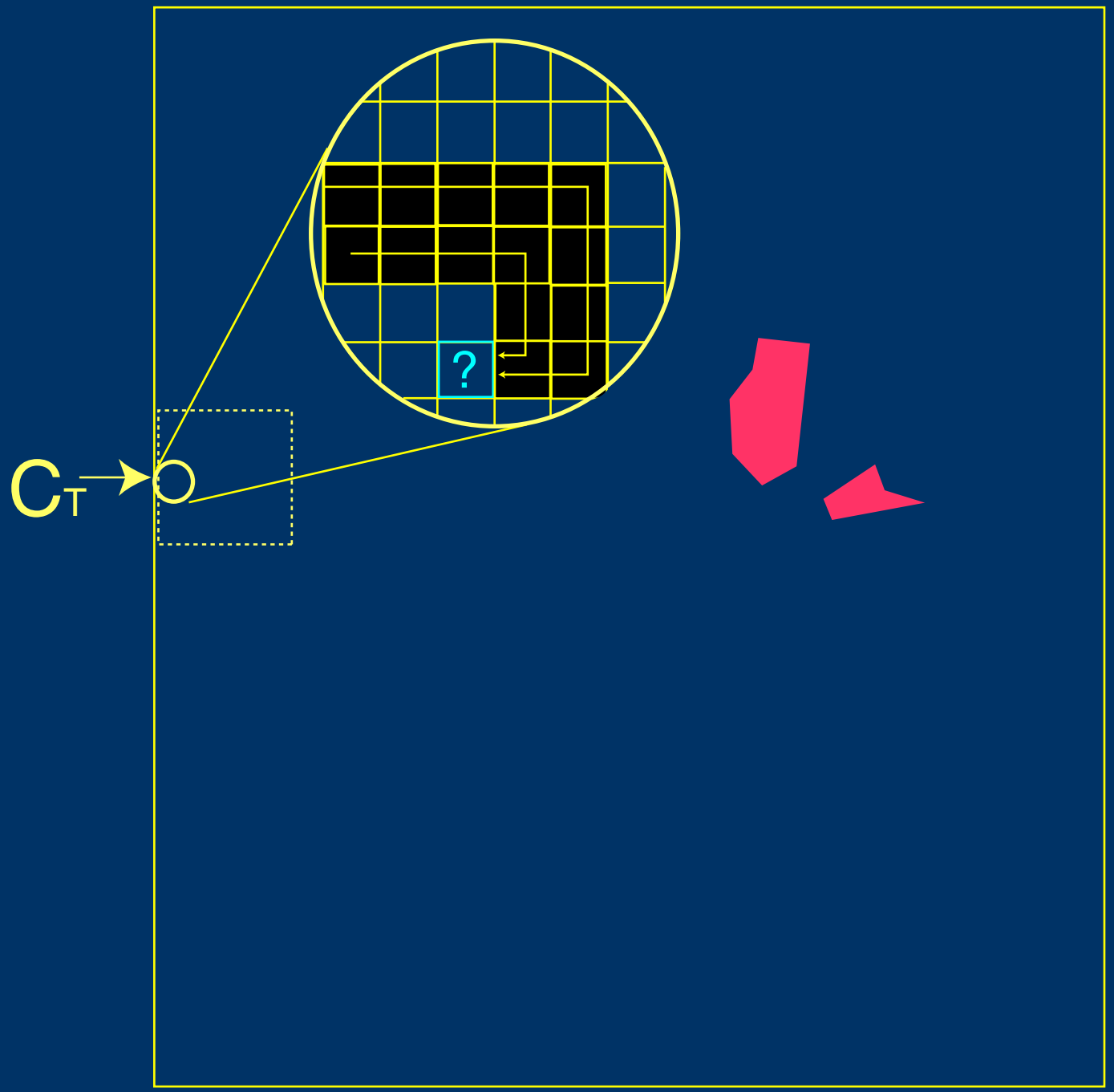
Test Phase



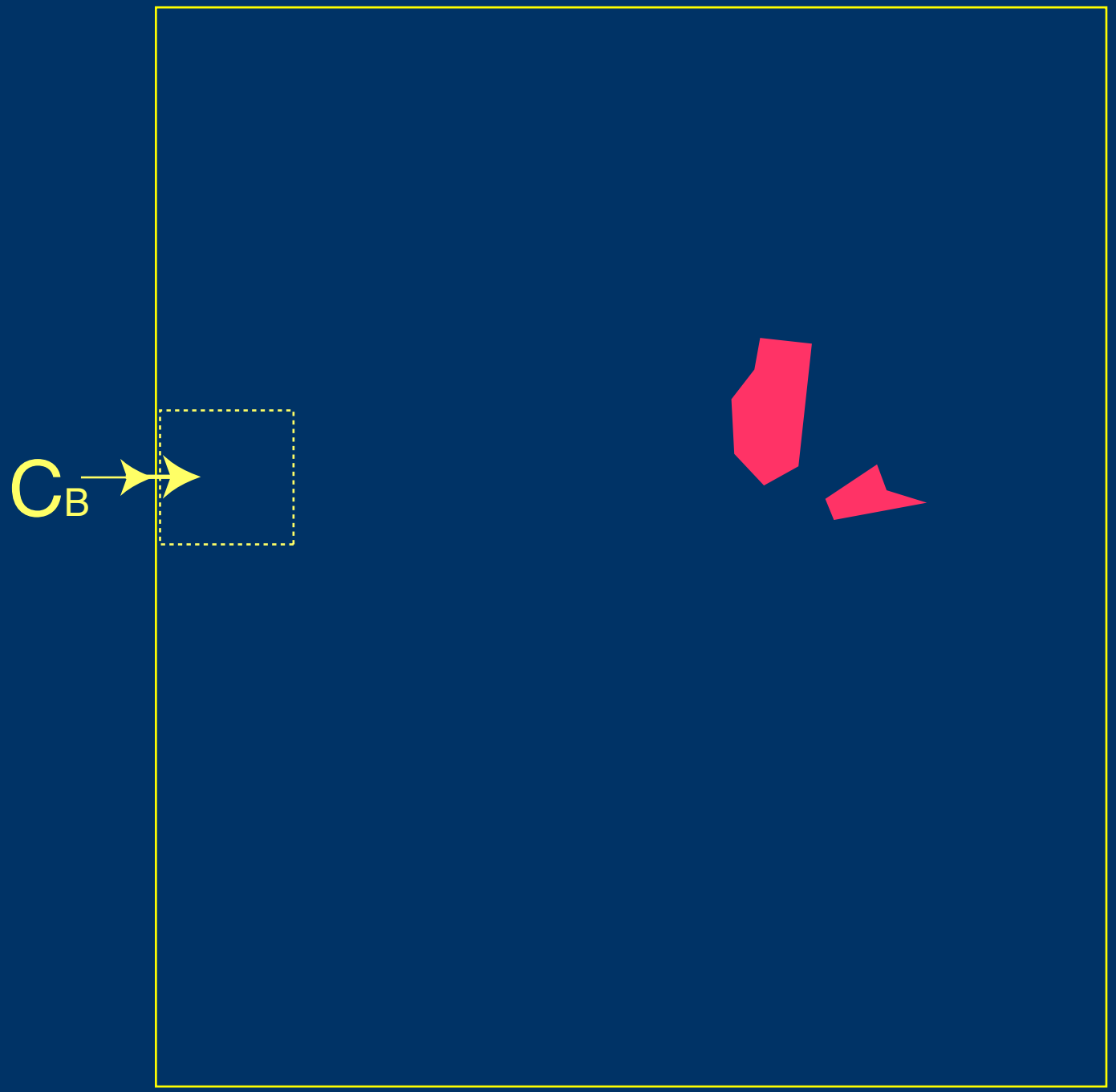
Test Phase



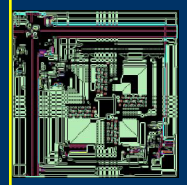
Test Phase



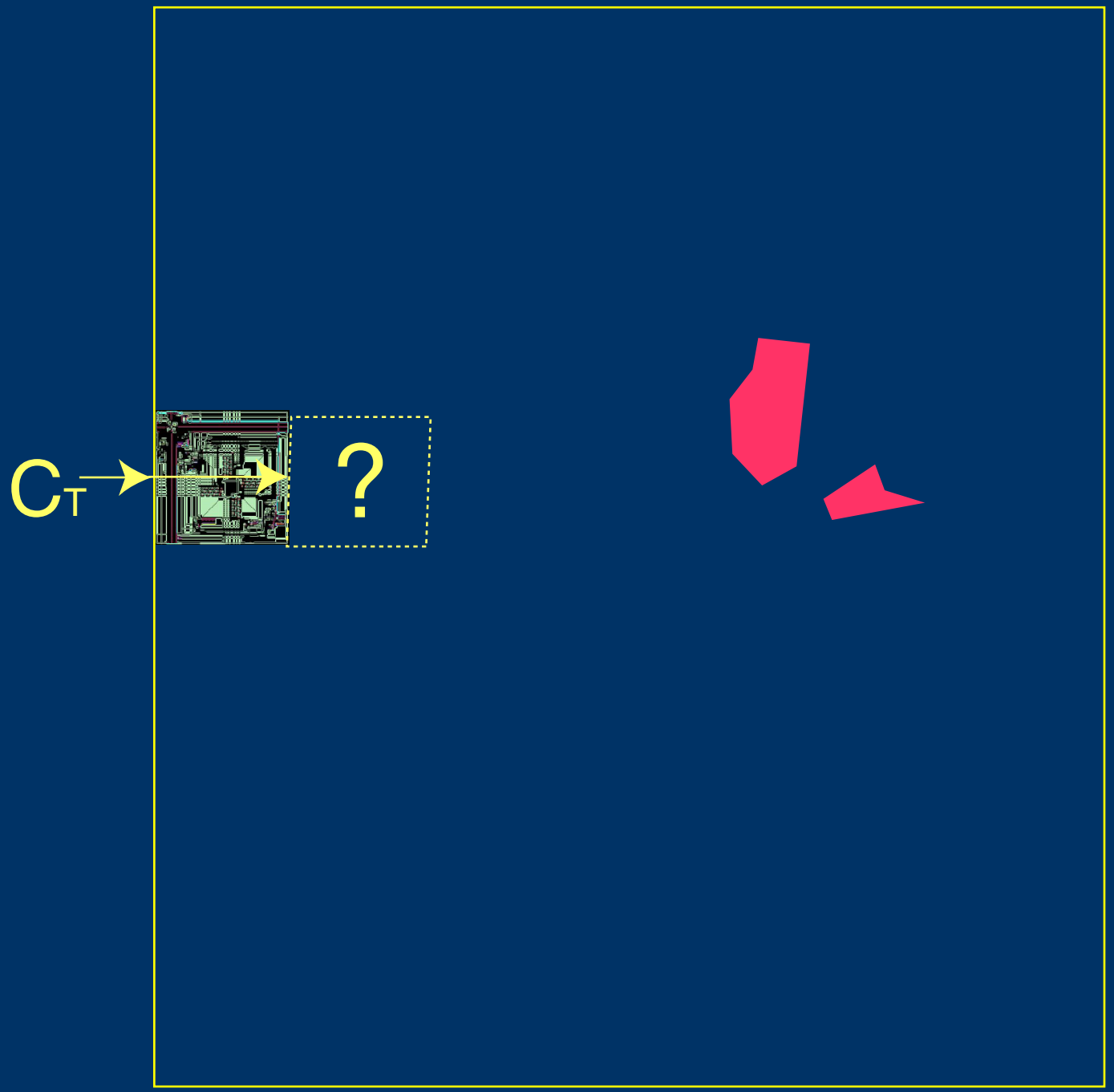
Build Phase



First Supercell Configured

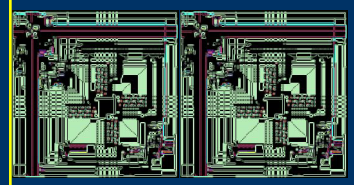


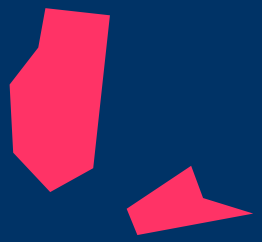
Test Phase



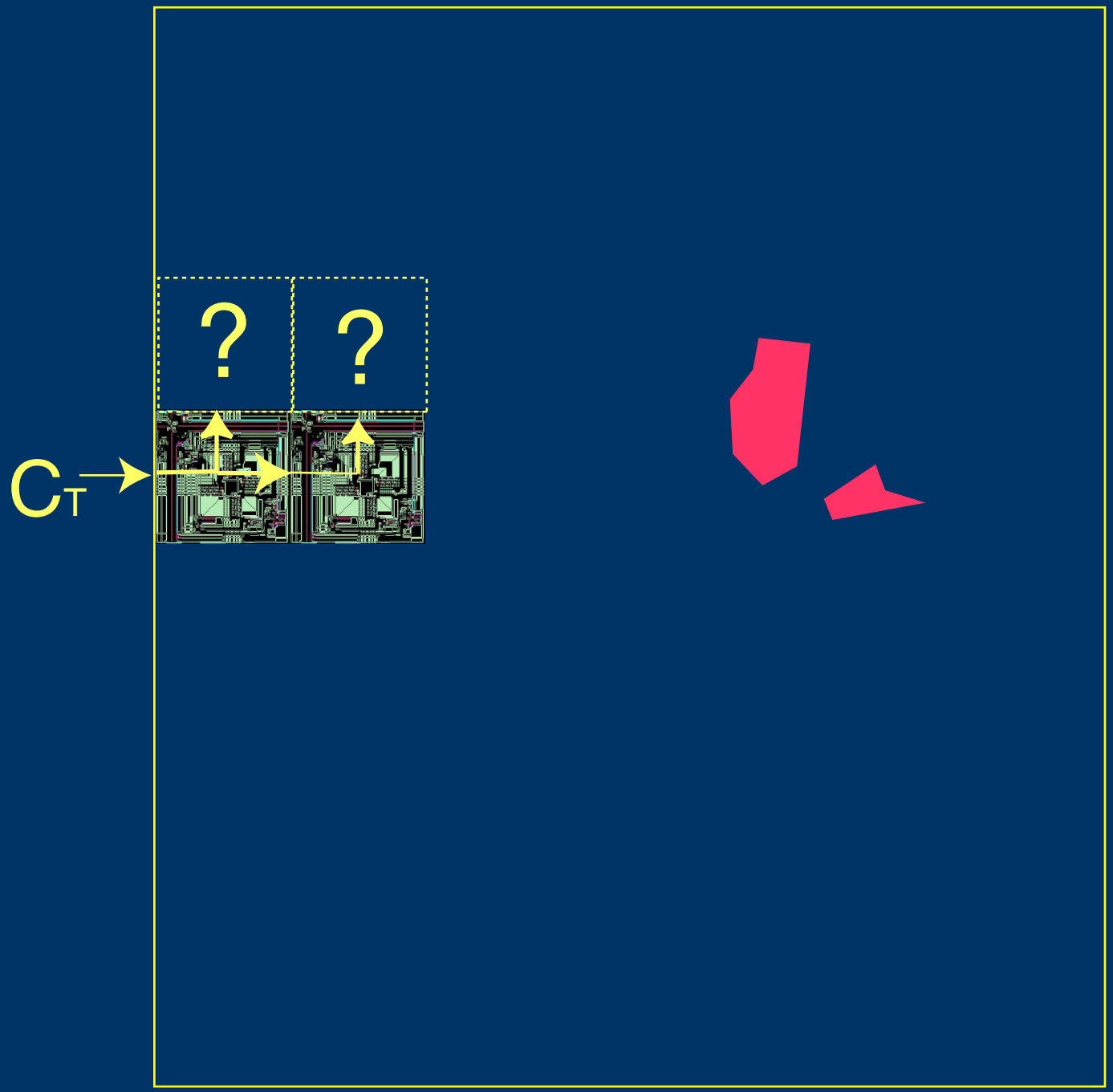
Build Phase

Two Supercells Configured

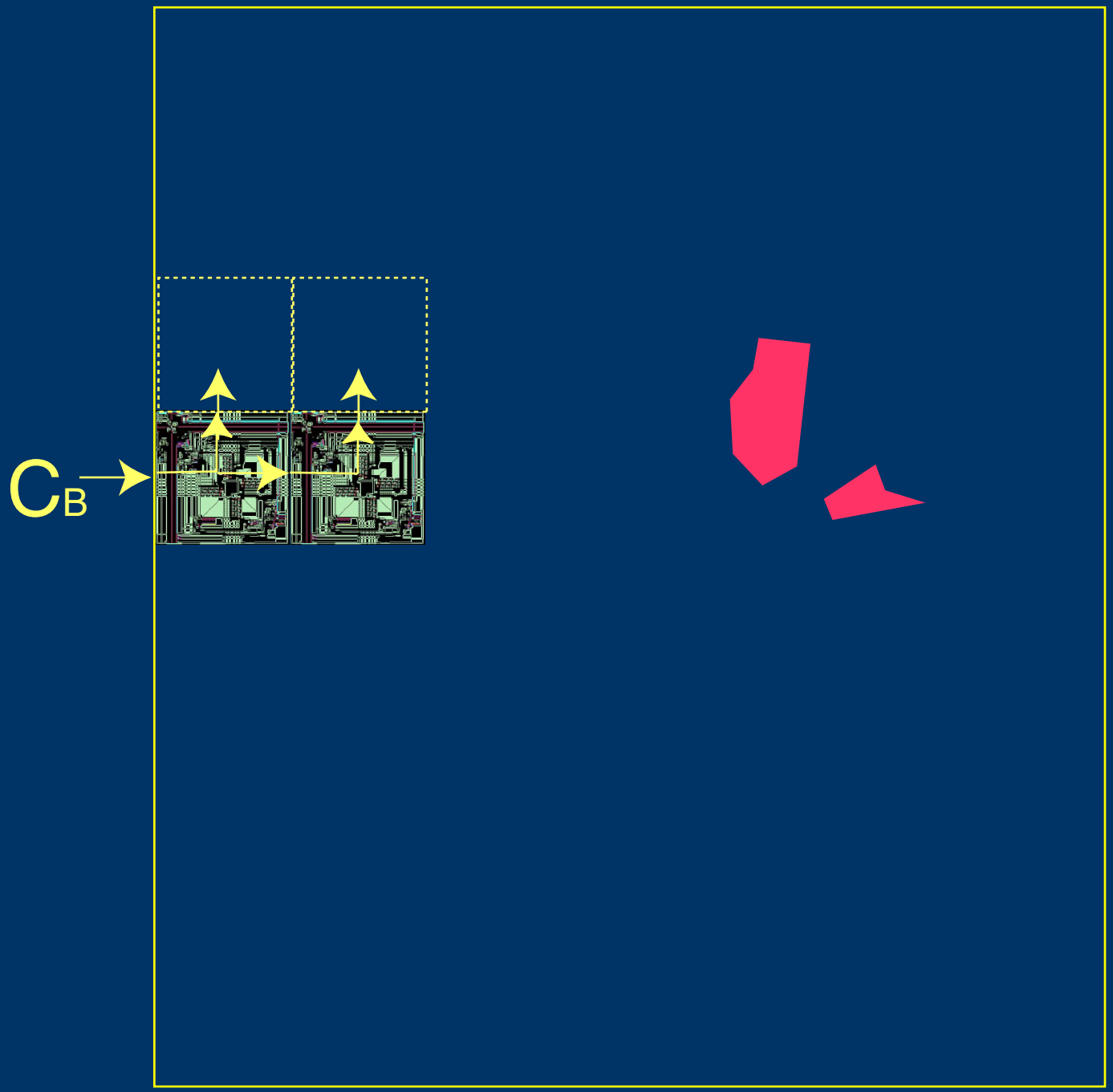


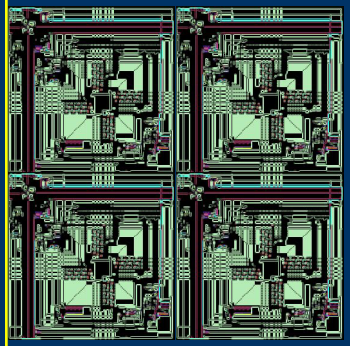


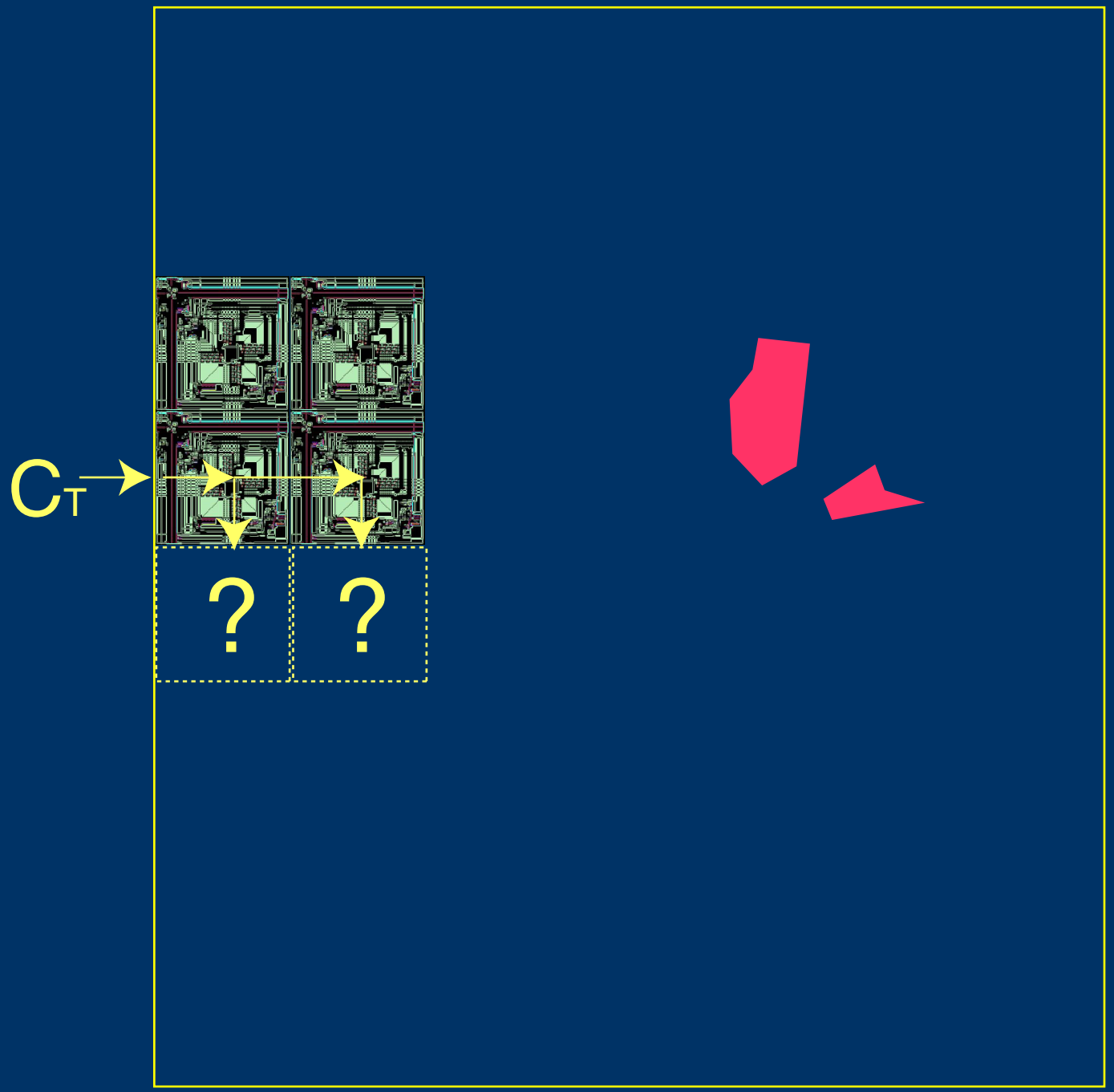
Two Regions Tested in Parallel

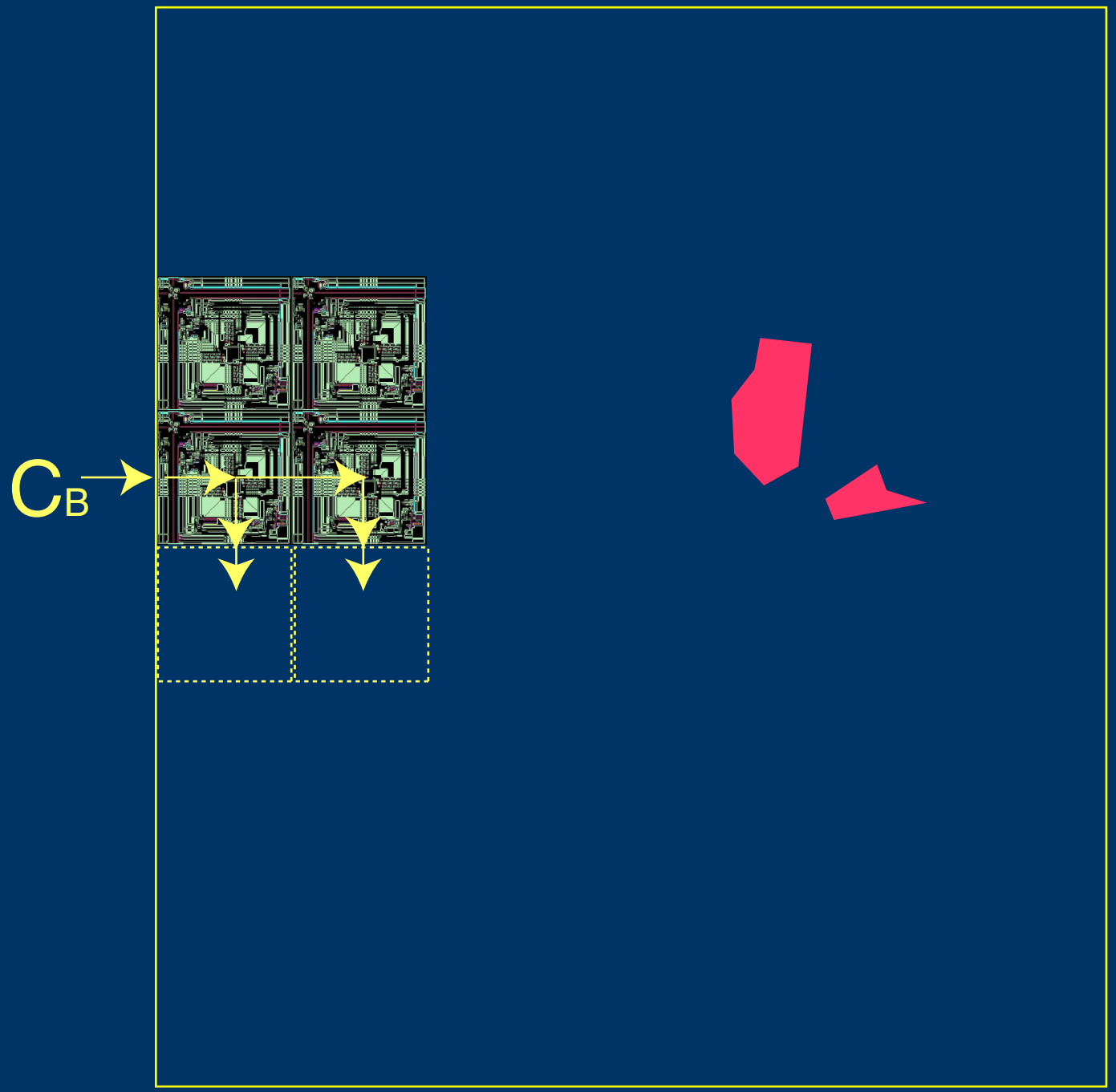


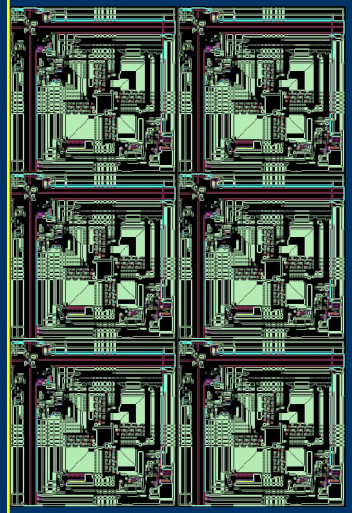
Two Regions Configured in Parallel



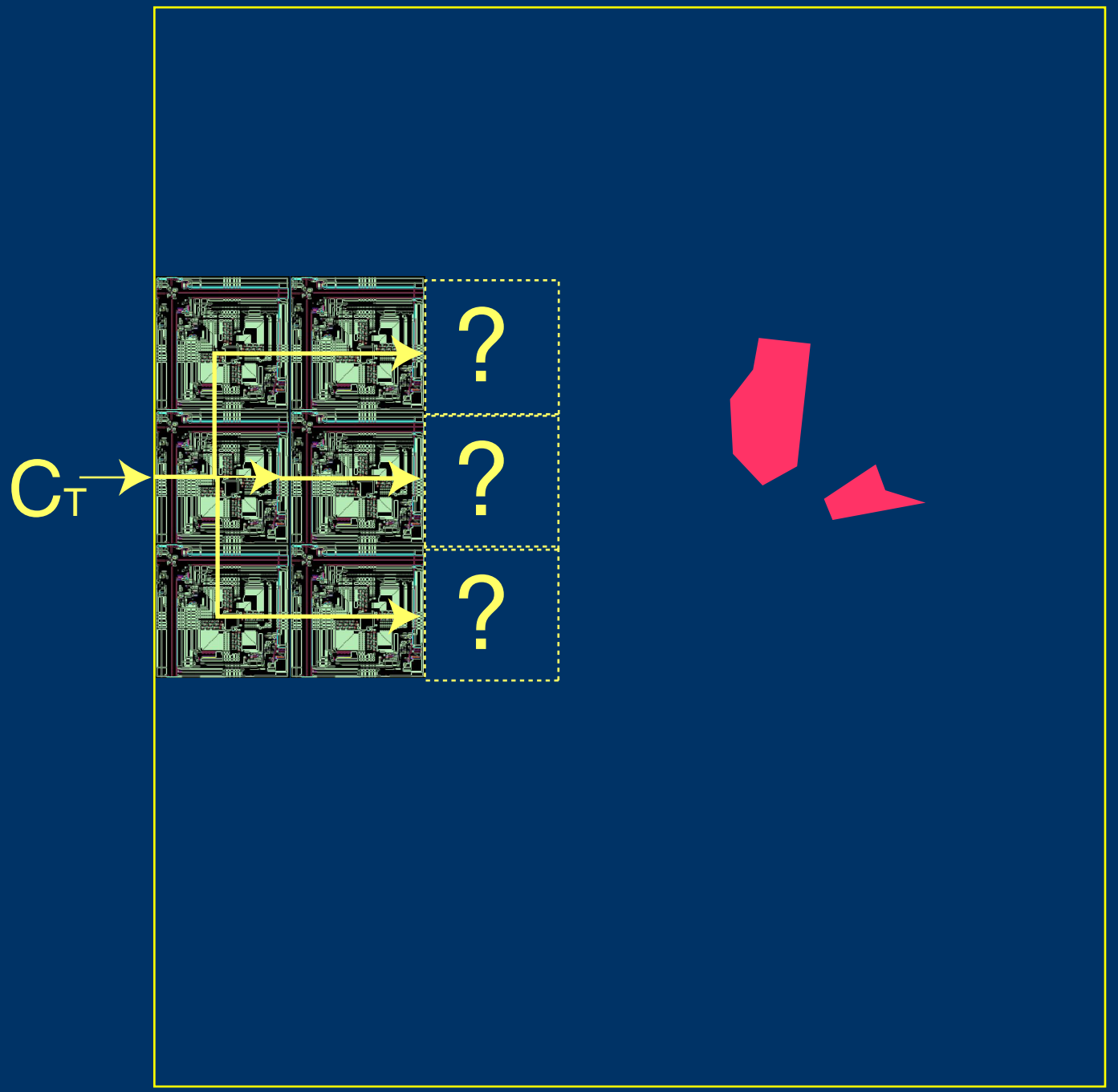



C_T
?
?

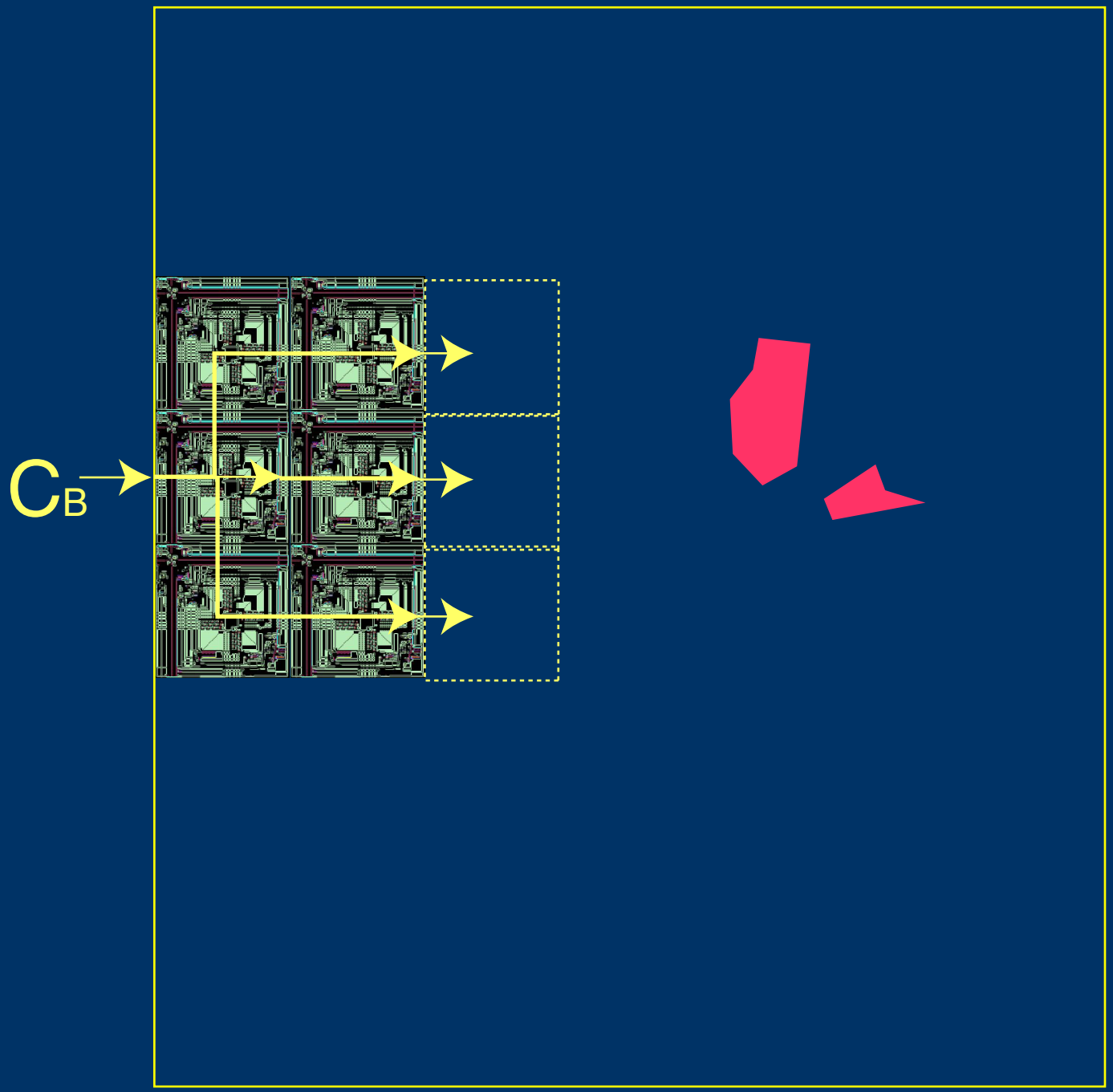

C_B

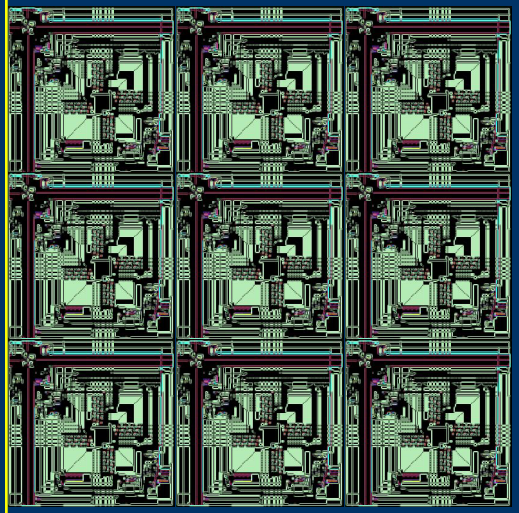


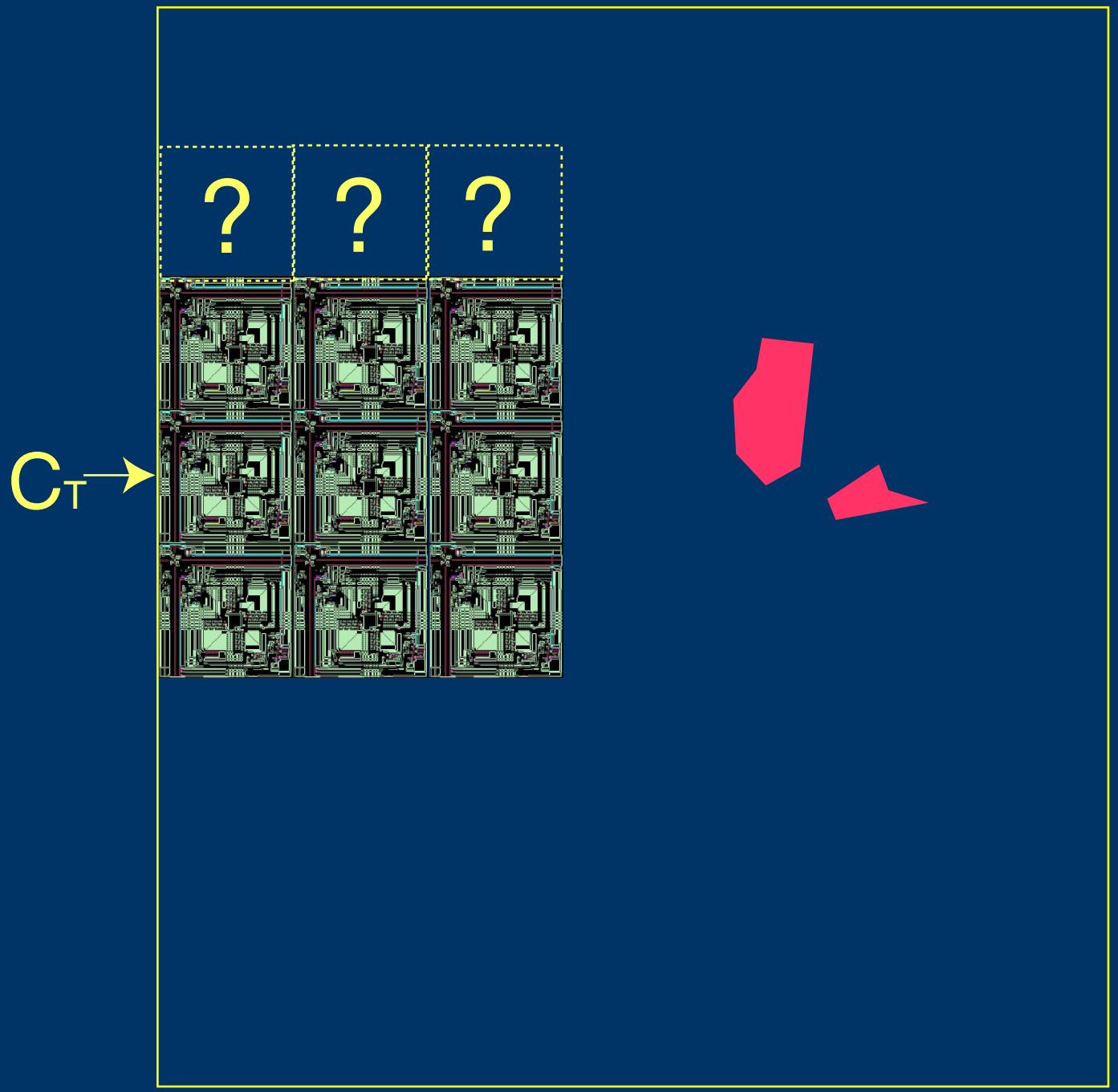
Three Regions Tested in Parallel

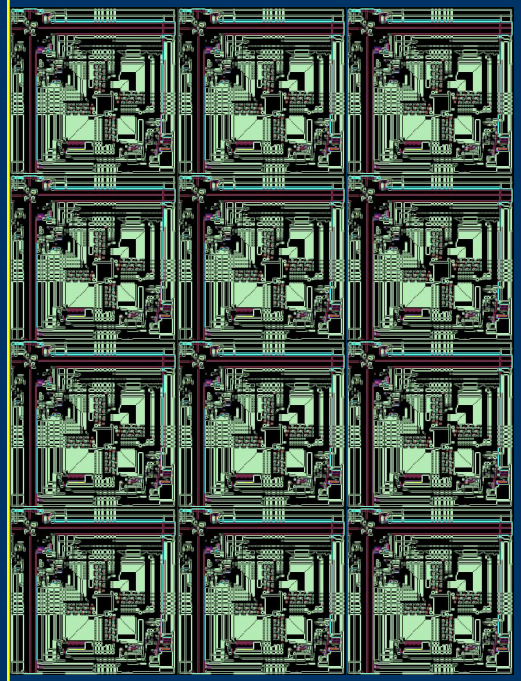


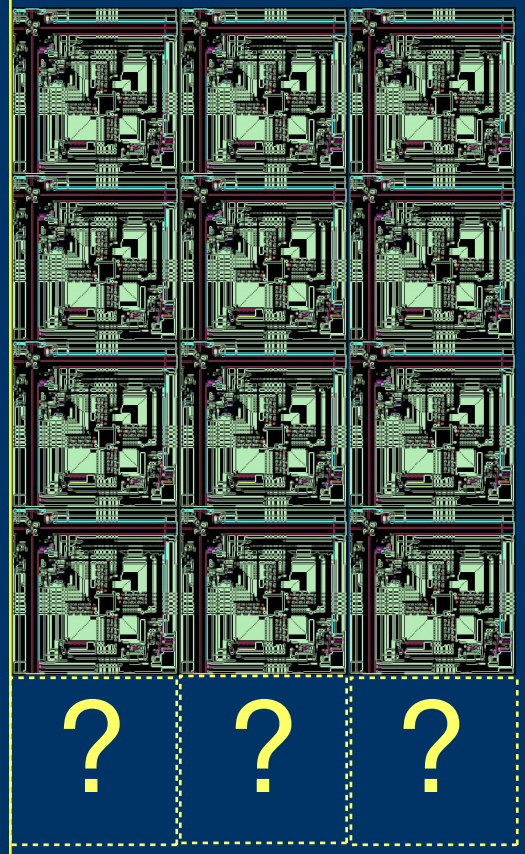
Three Regions Built in Parallel

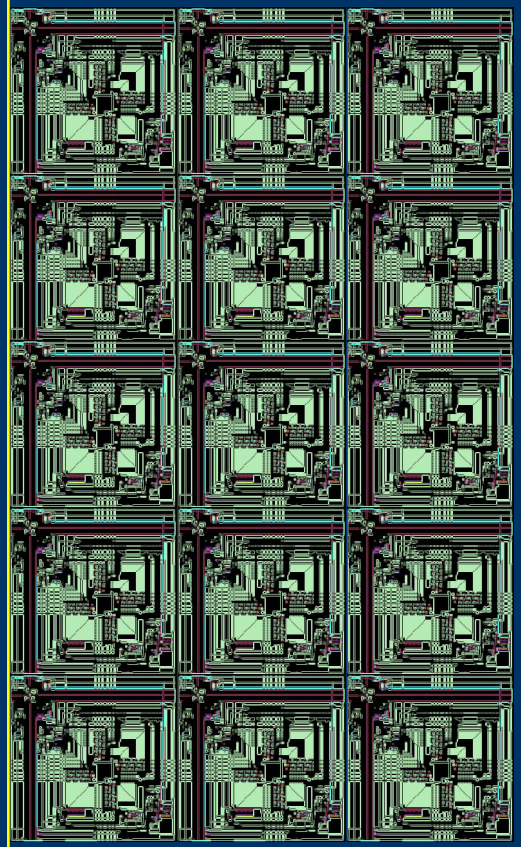




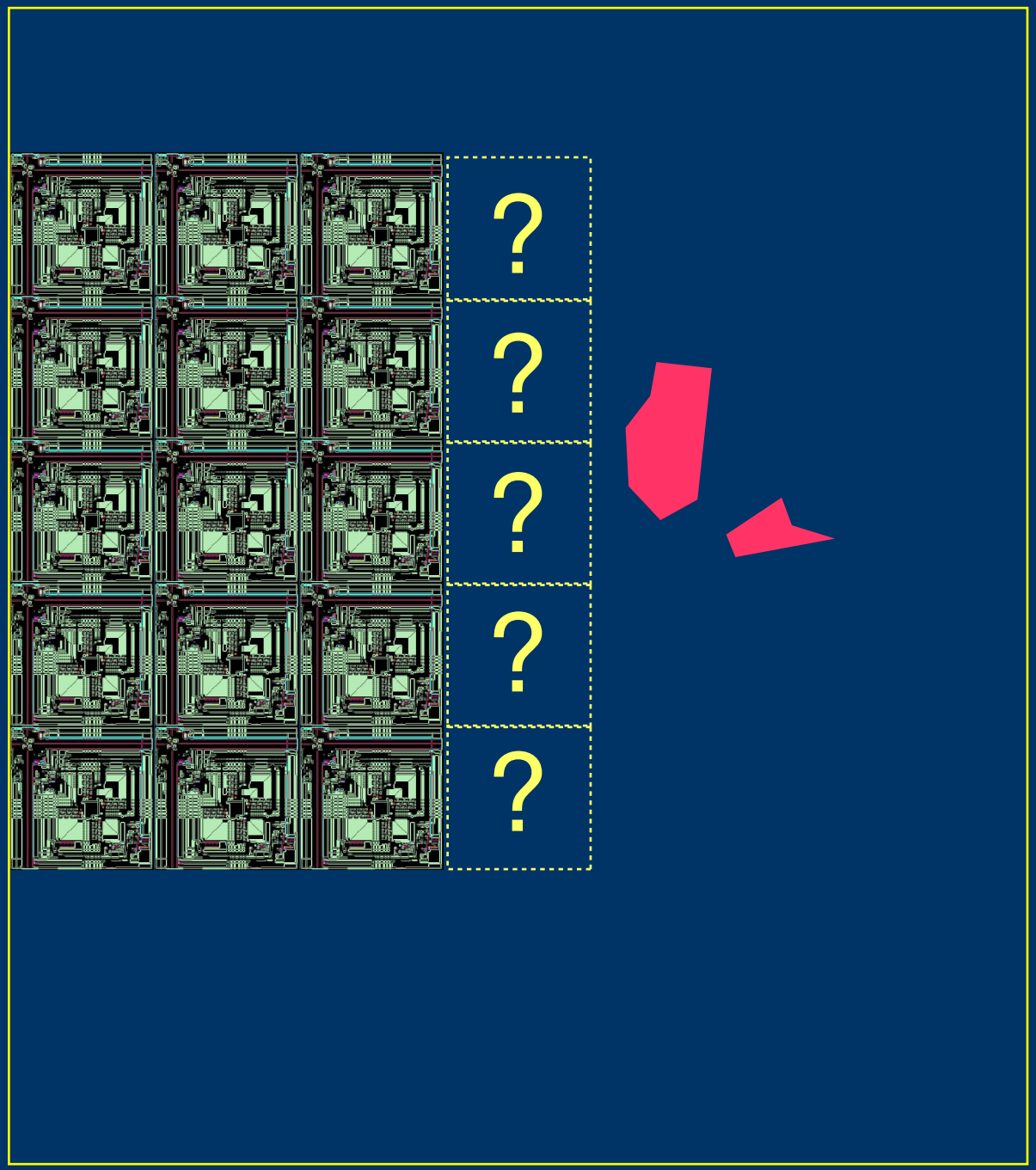

? ? ?
C_T





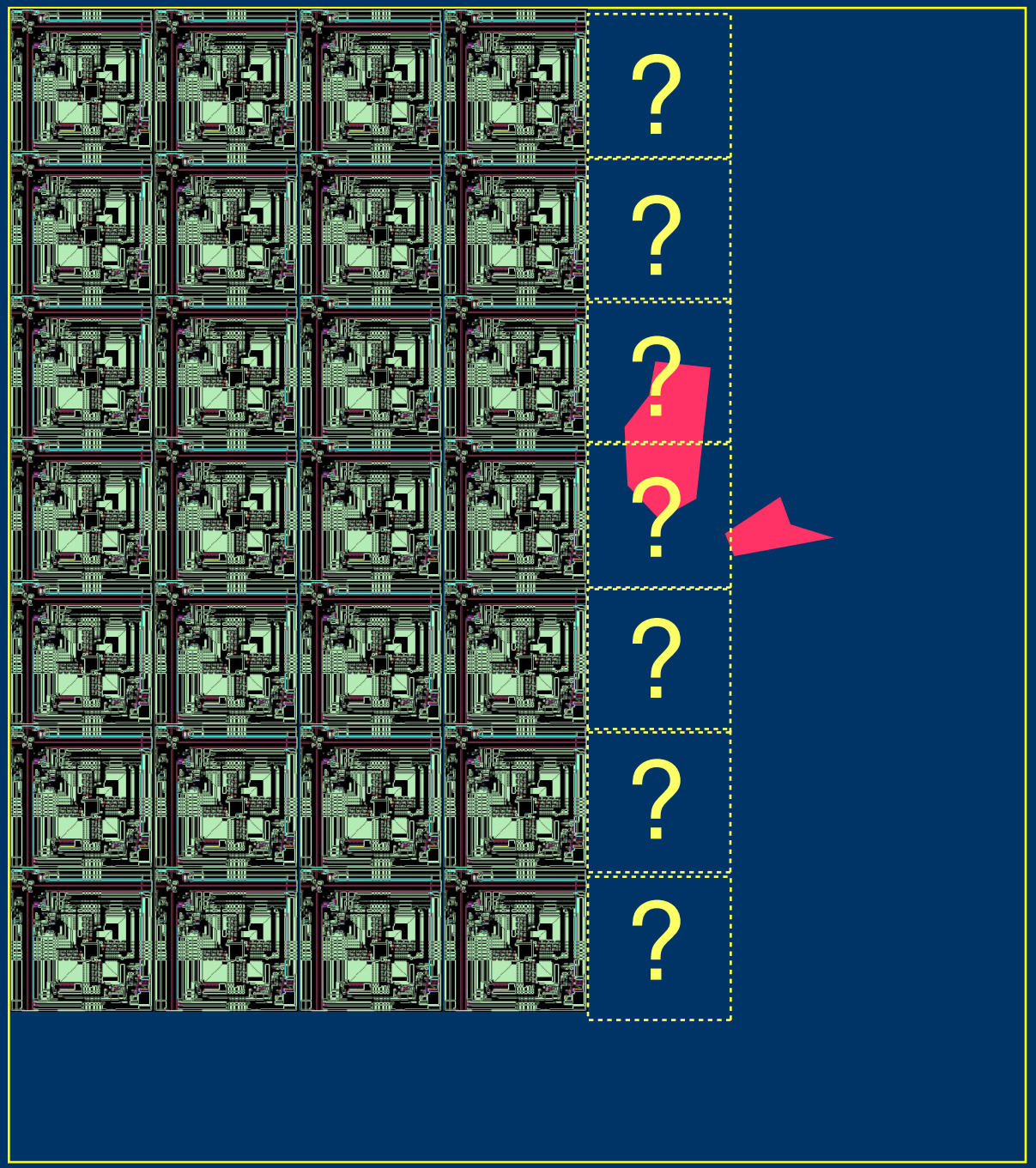


Five Regions Tested in Parallel



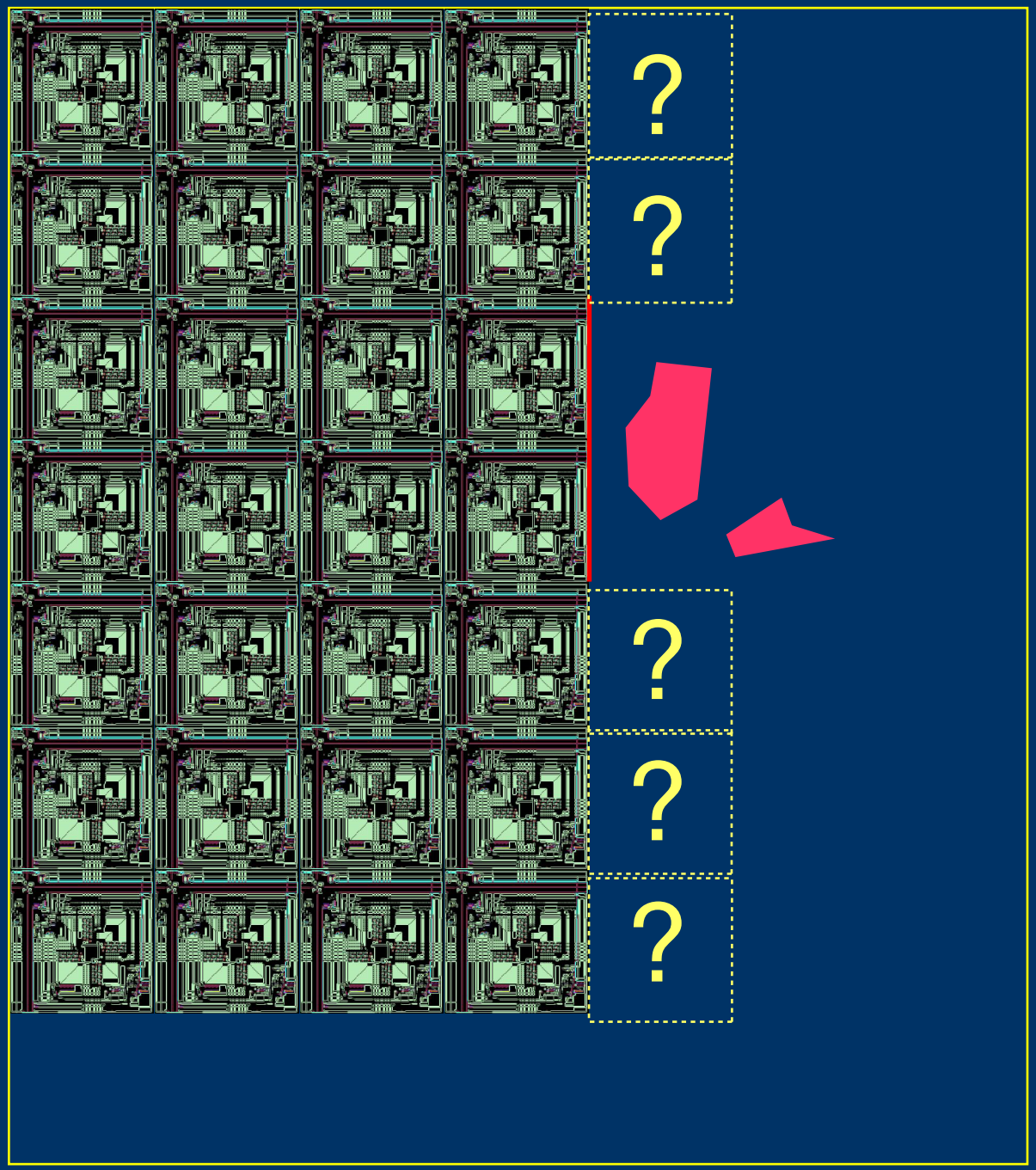
Five Regions Configured in Parallel

Test Phase: →



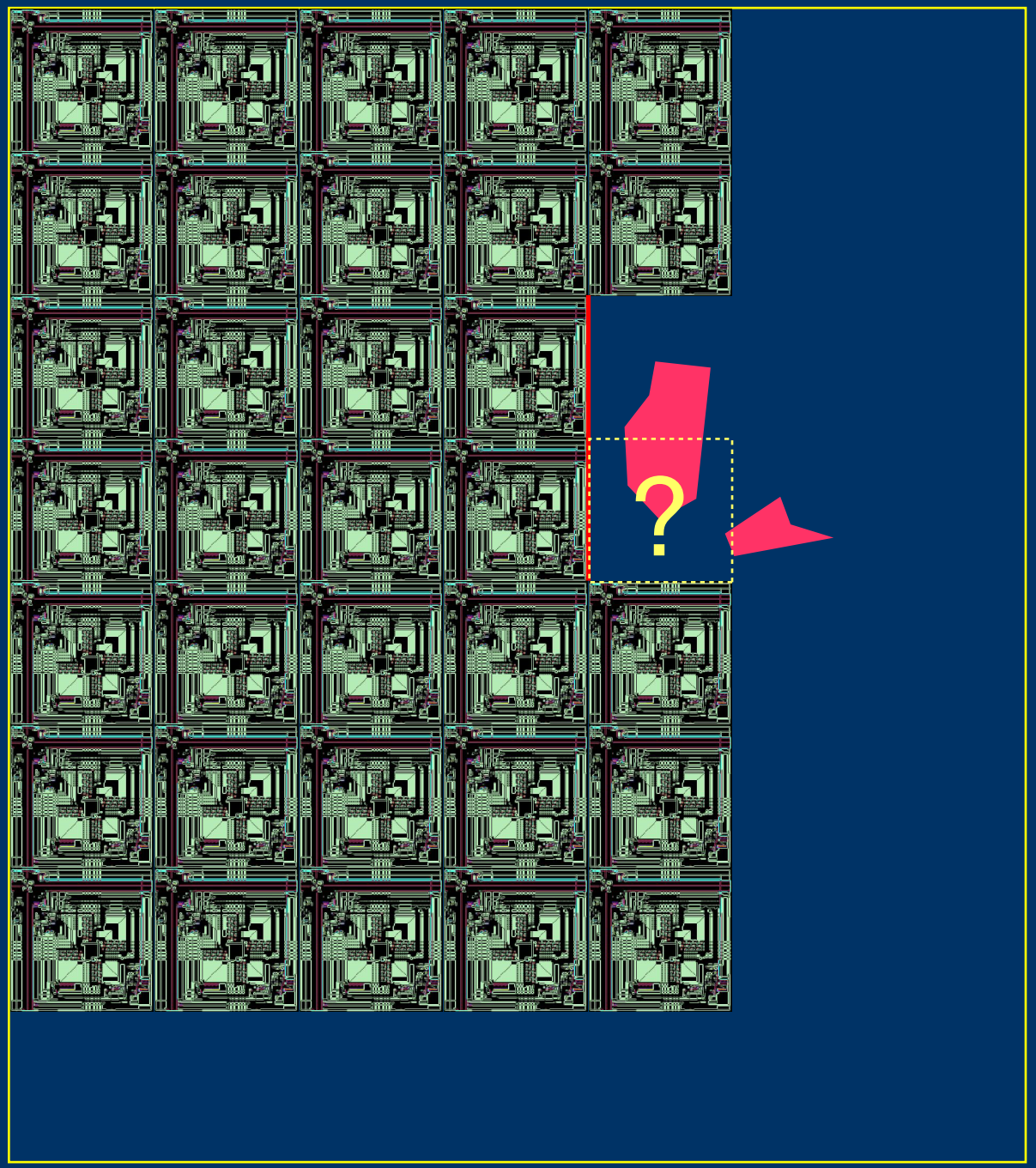
Test Phase: →

Test Phase: →



Build Phase: →

Test Phase: ↑

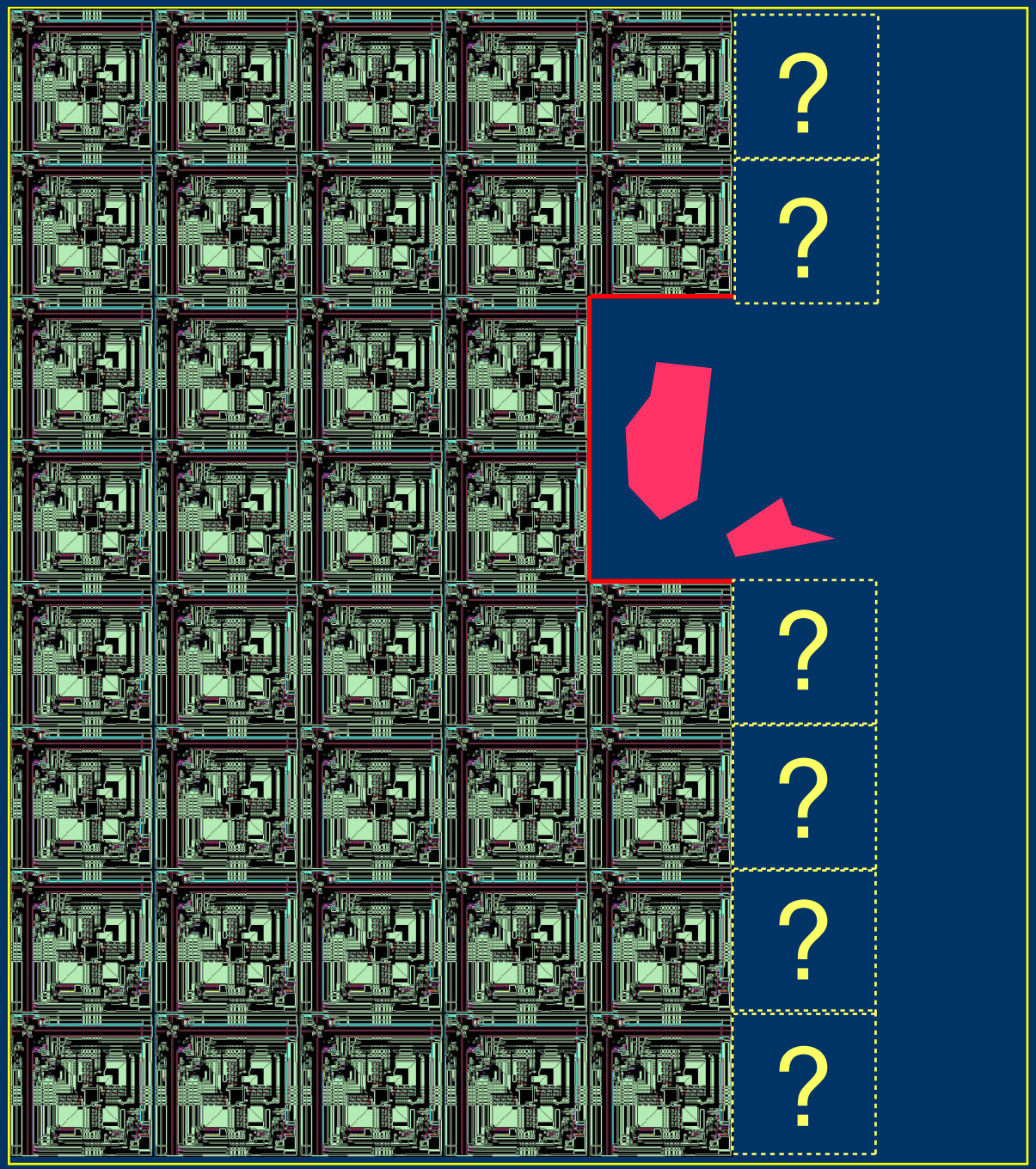


Build Phase: ↑

Test Phase: ↓

Build Phase:↓

Test Phase: →

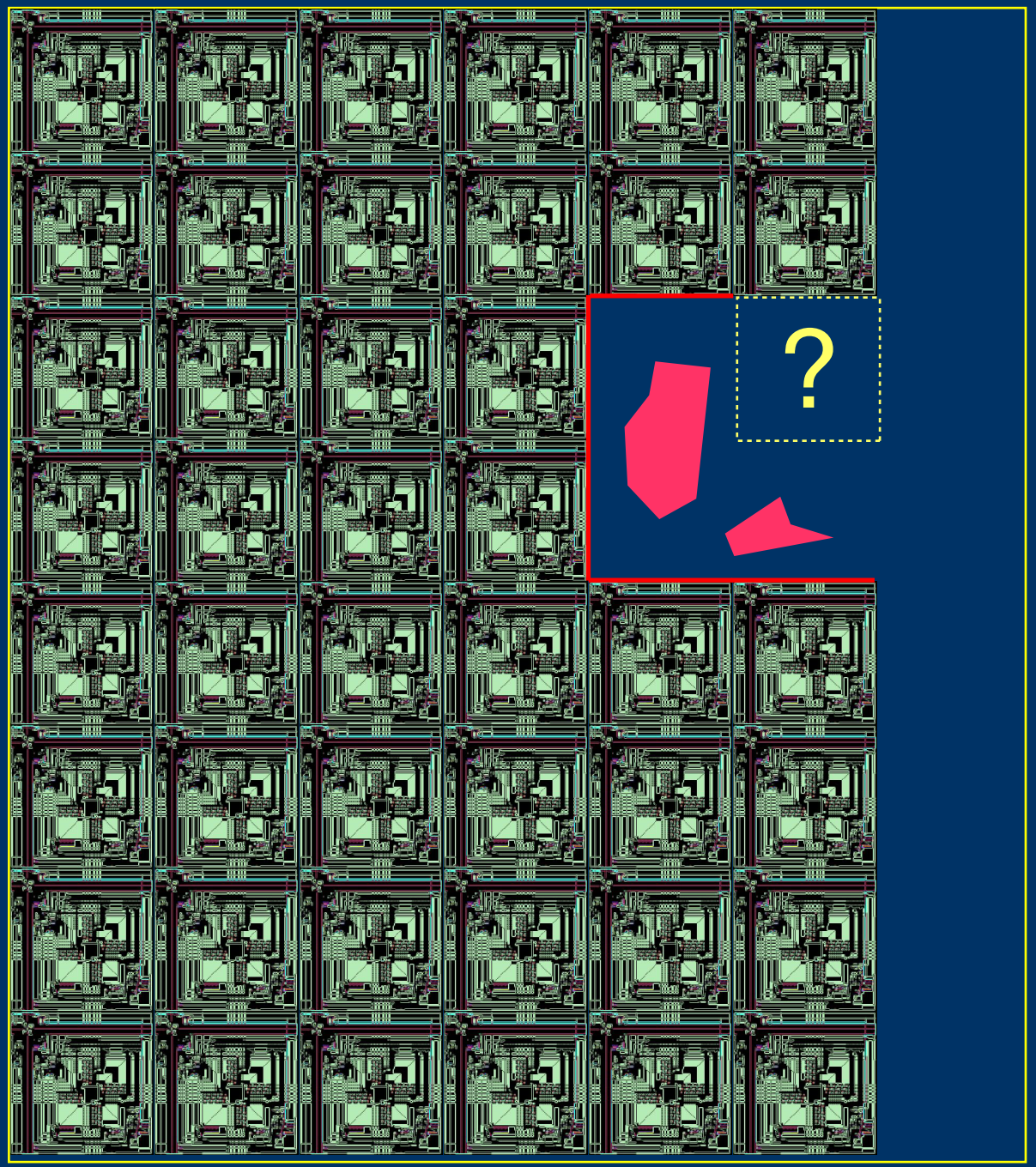


Build Phase:→

Test Phase: ↑

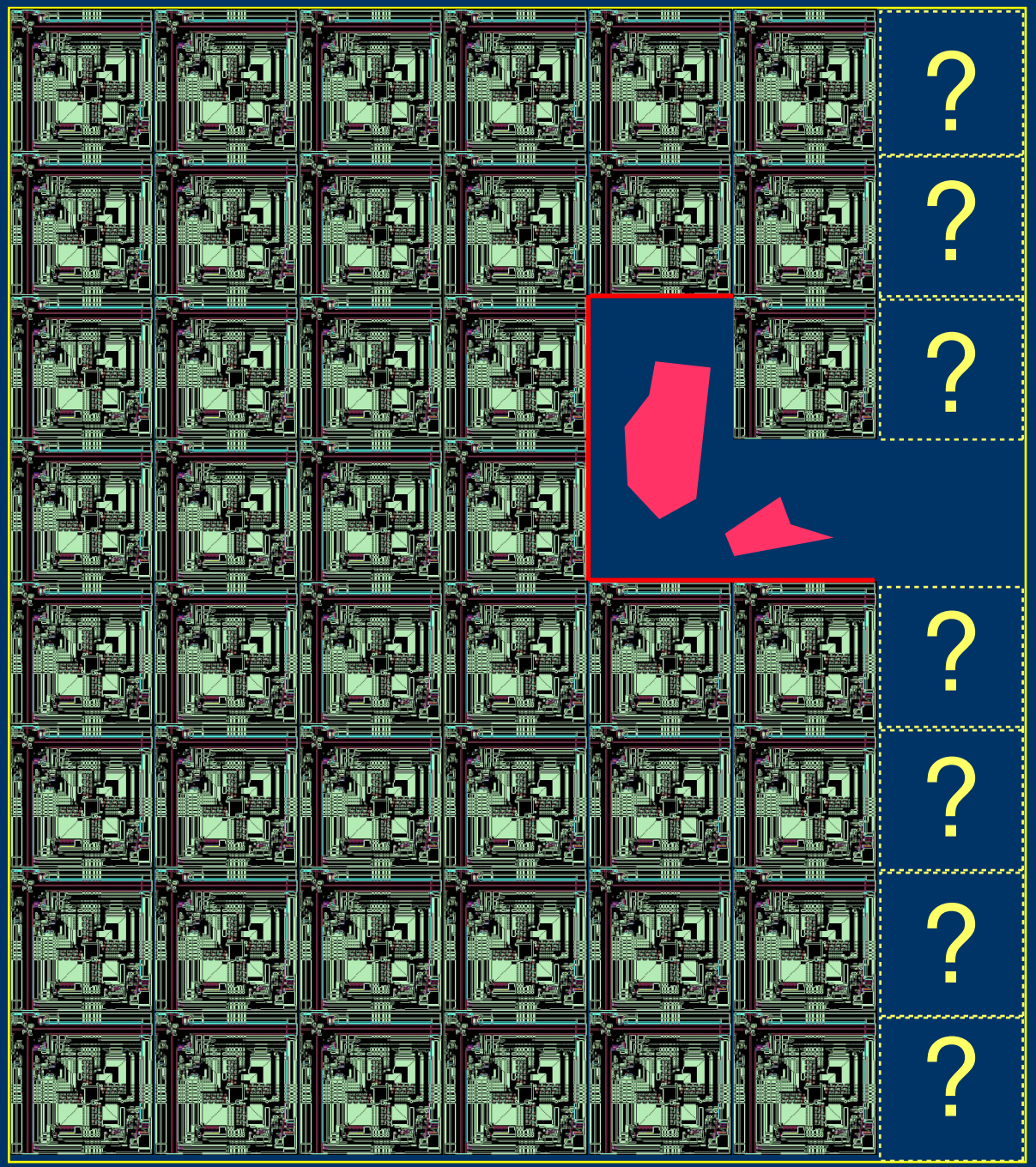
Build Phase: ↑

Test Phase: ↓



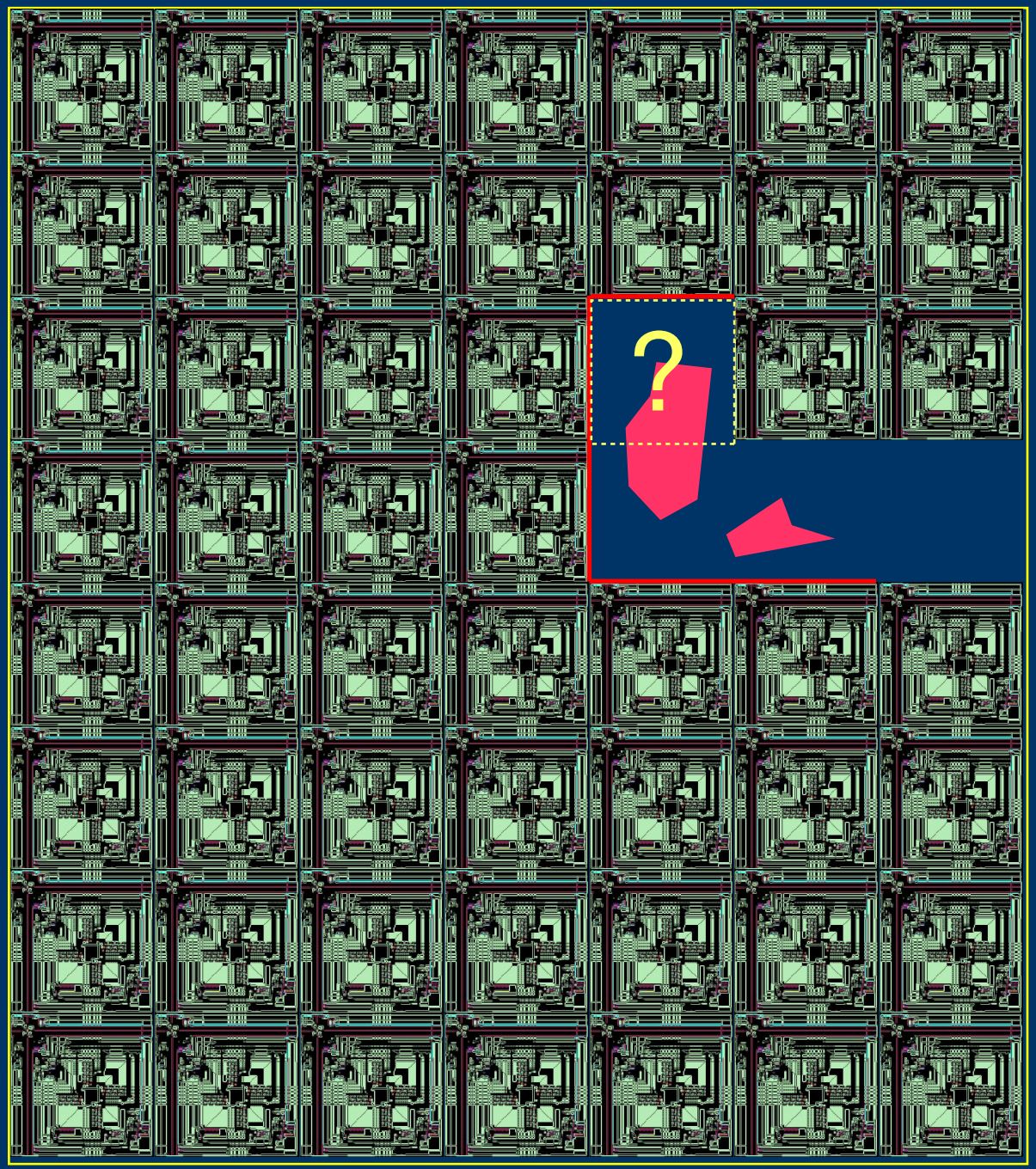
Build Phase:↓

Test Phase:→



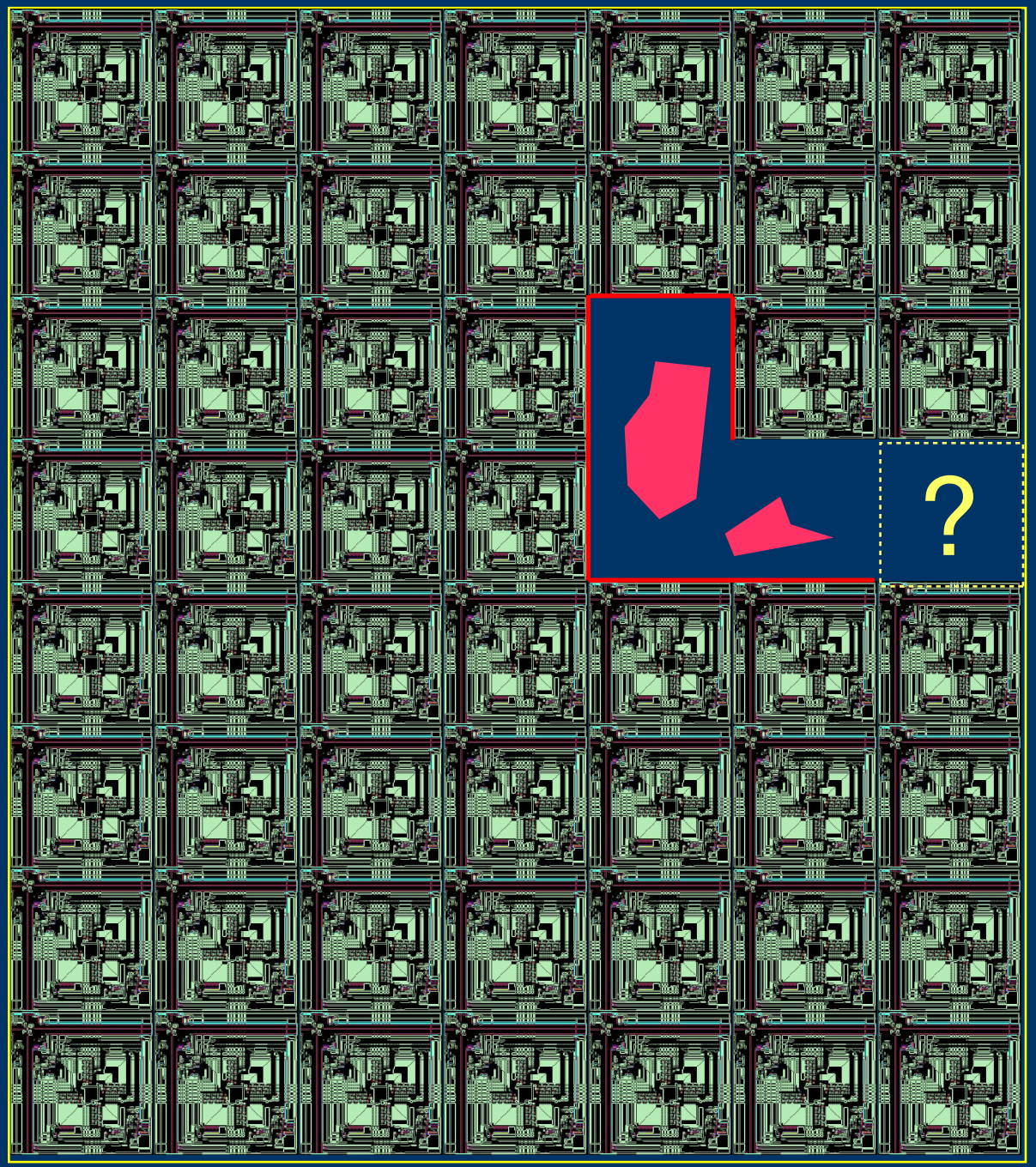
Build Phase: →

Test Phase: ←



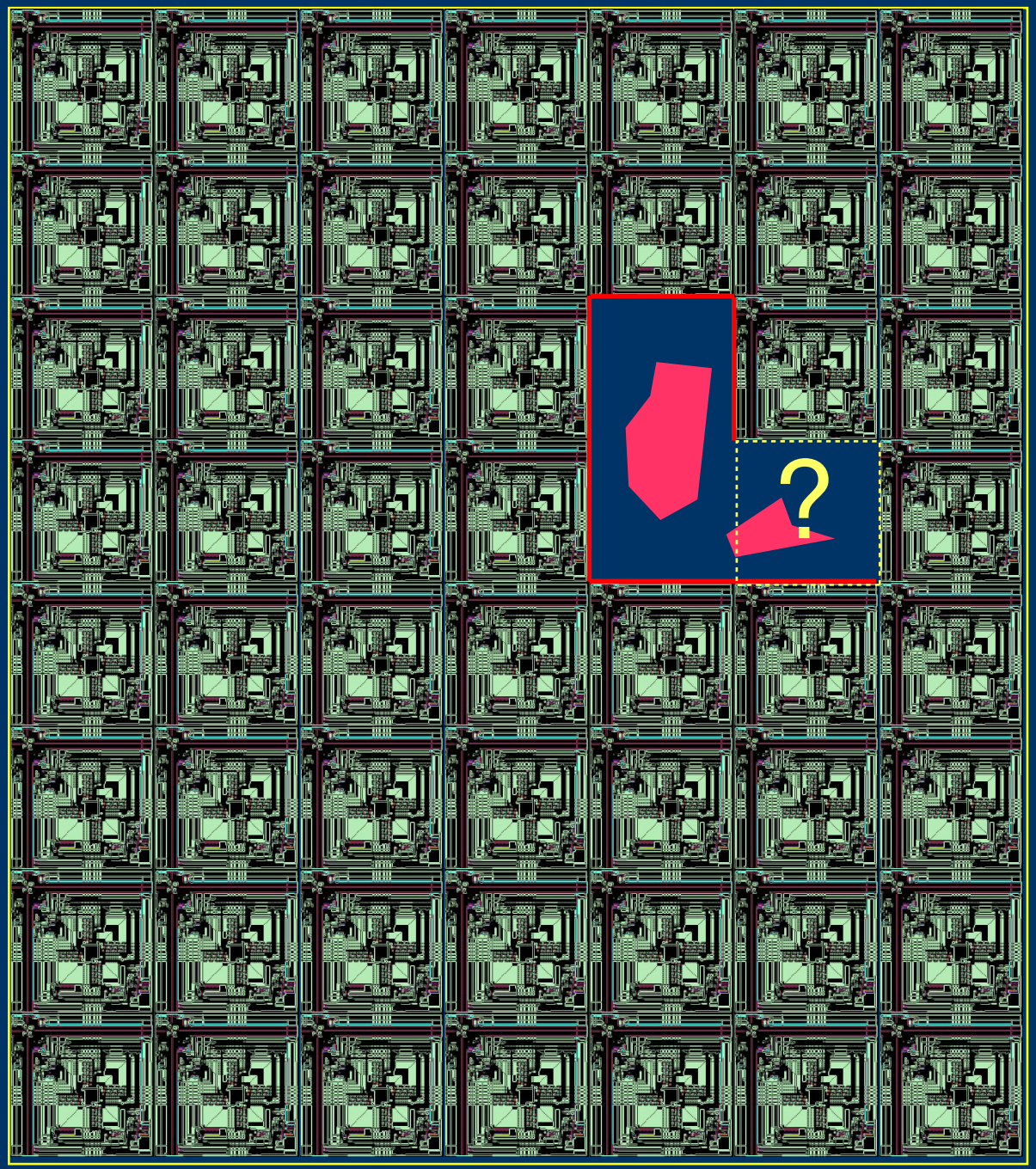
Build Phase:←

Test Phase: ↑



Build Phase: ↑

Test Phase:↓



Build Phase:↓

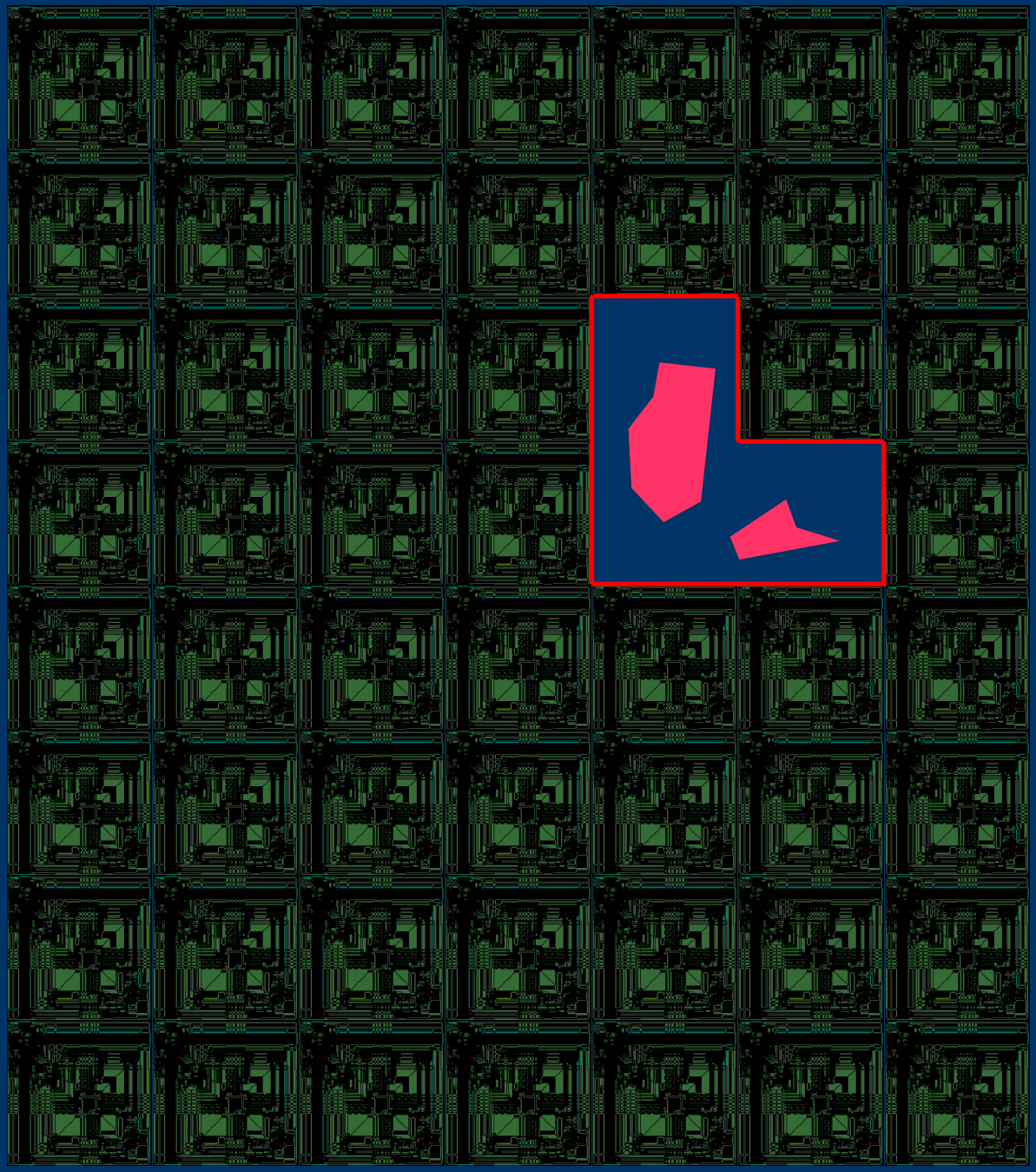
Test Phase: ←

Build Phase:←

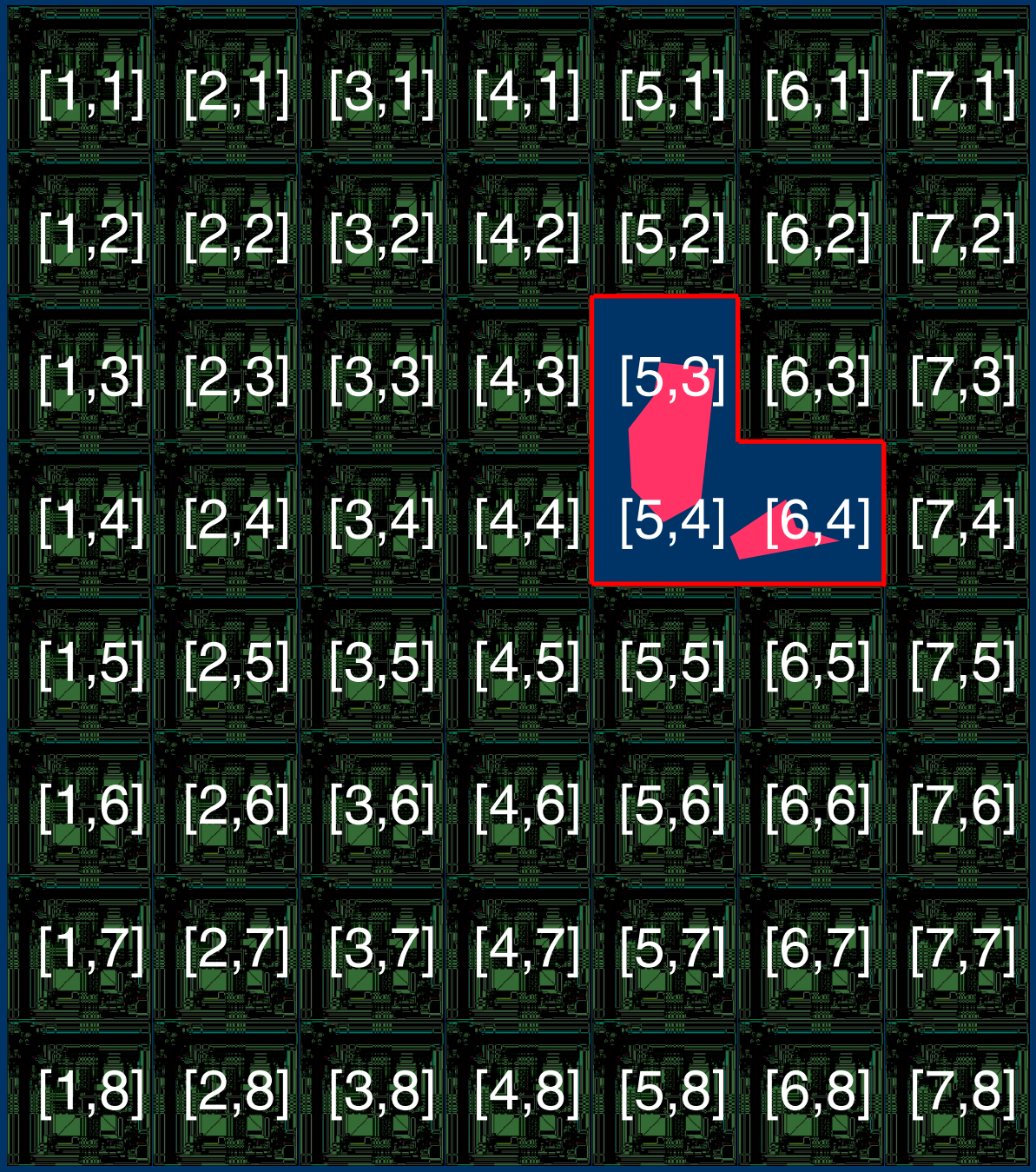
End Of Config String

GO→

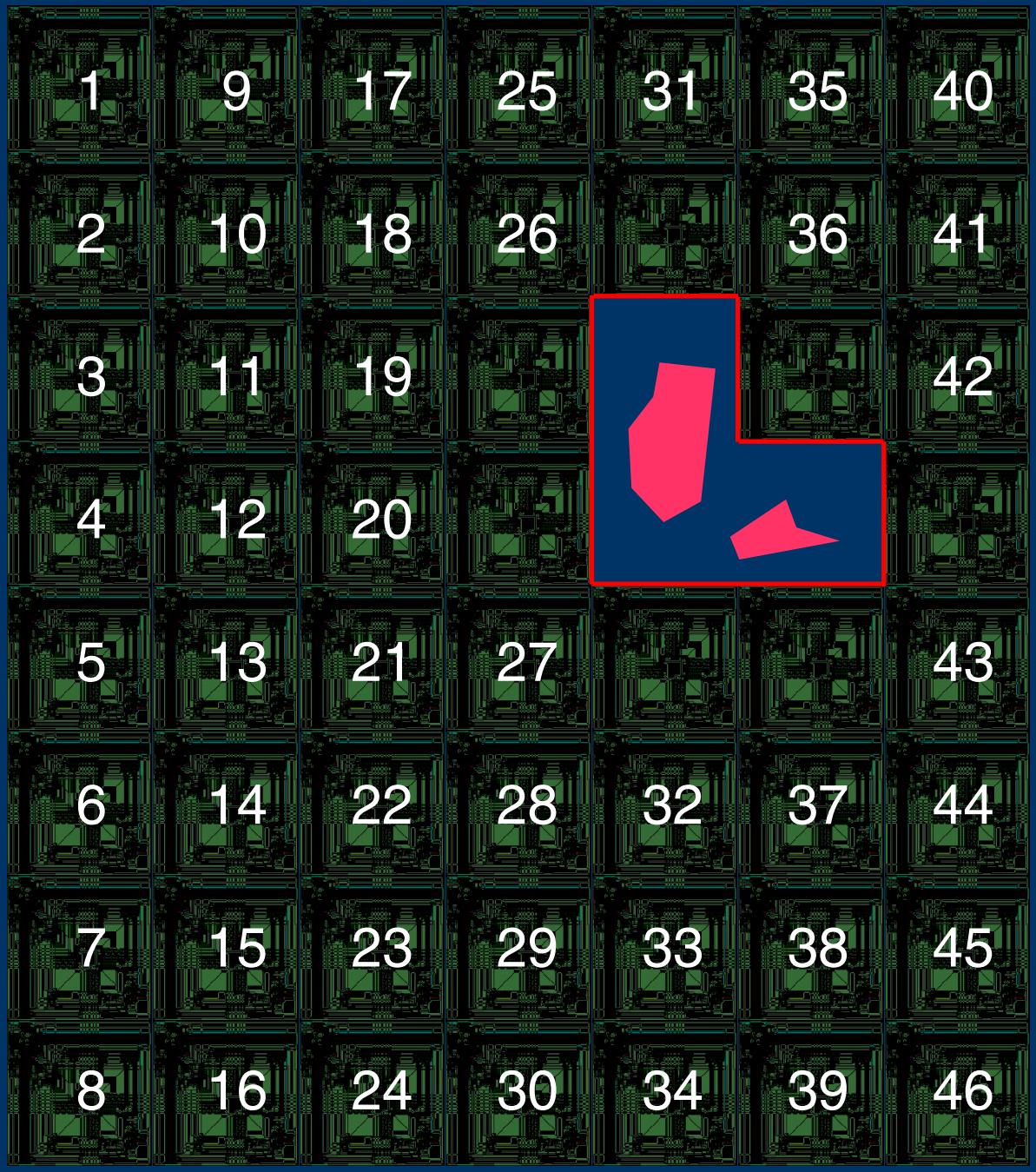
SELF-ORGANIZING PHASE:
Collective State Machine Begins Operation



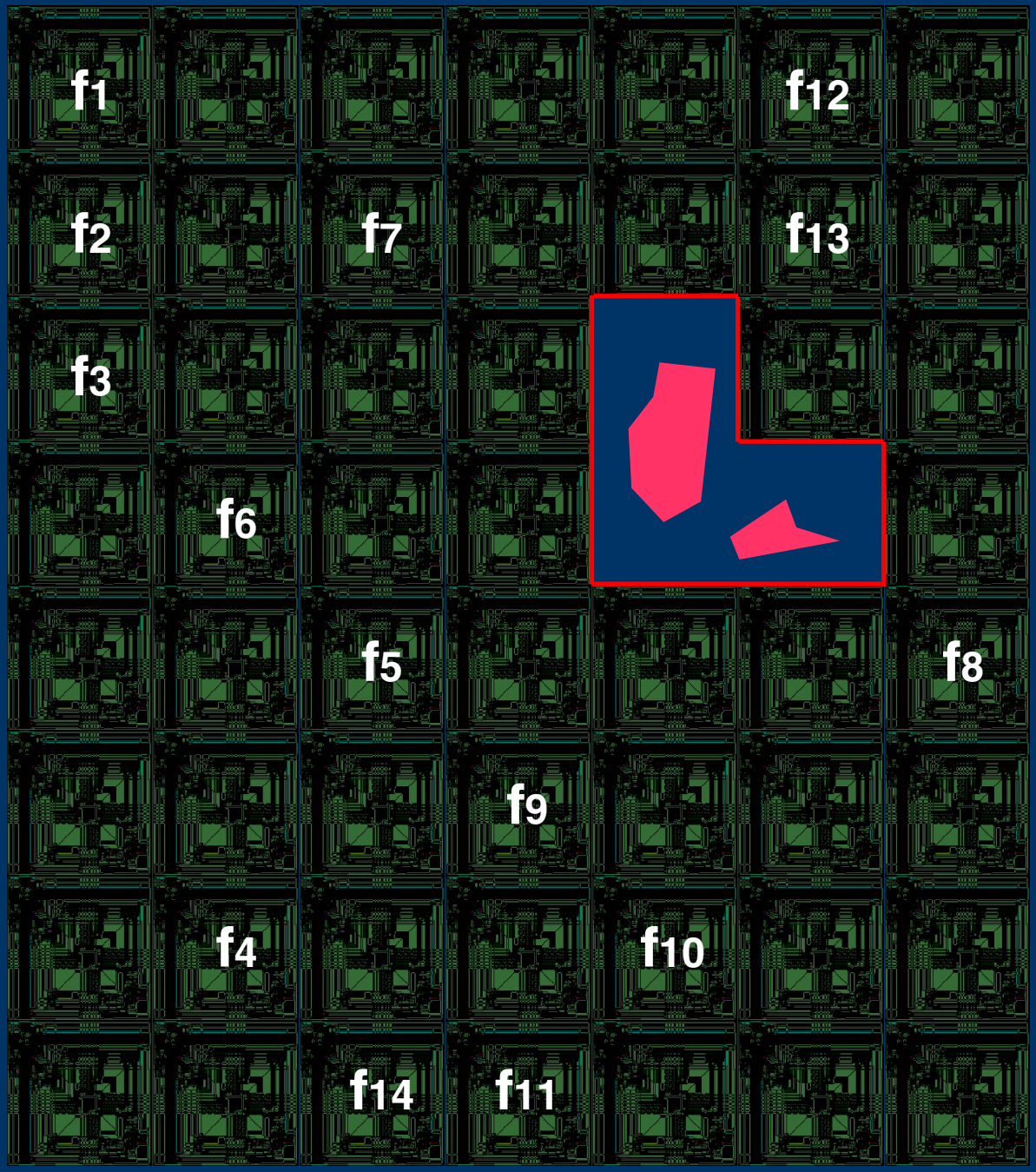
Static ID Assignment



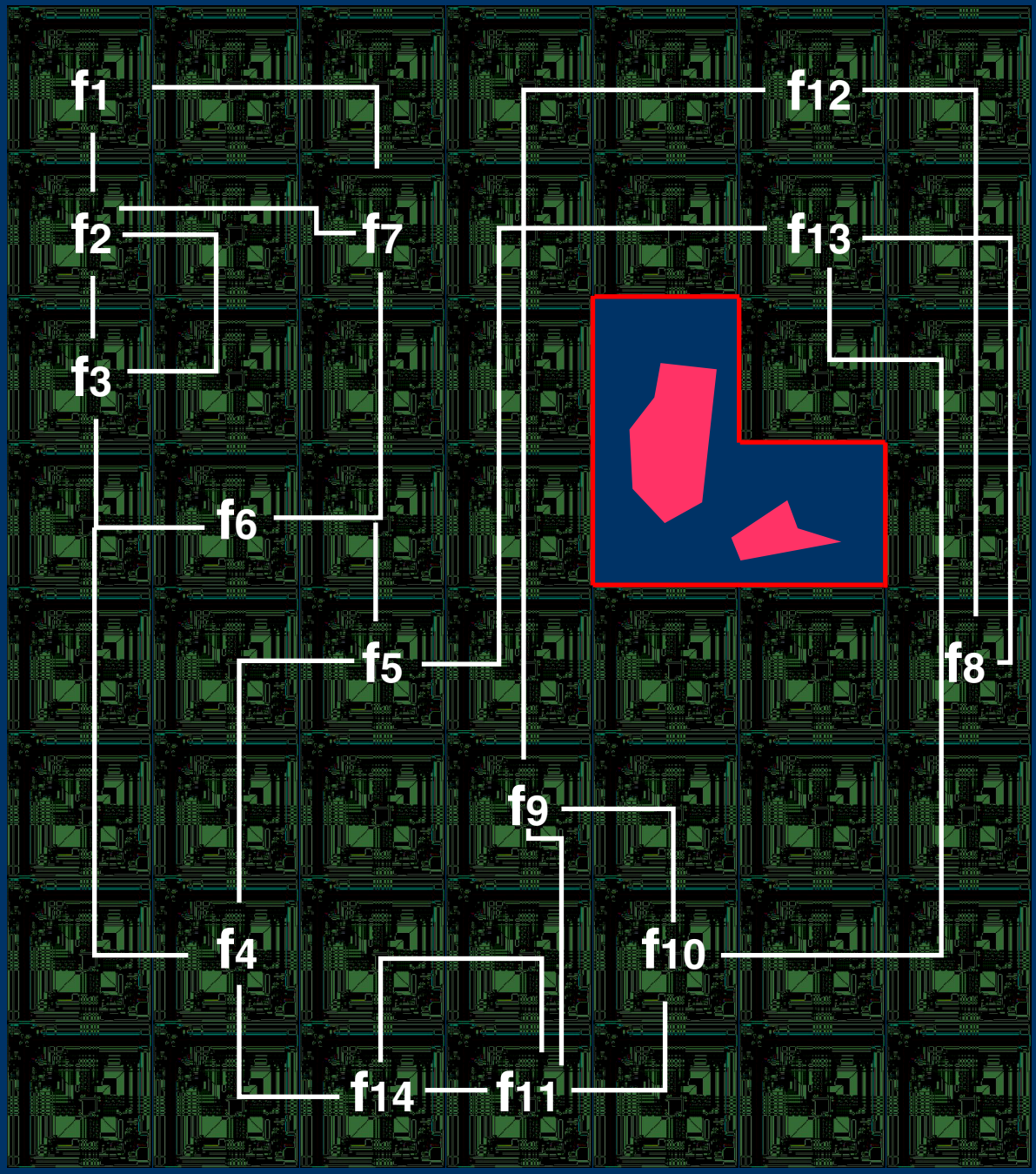
Dynamic ID Assignment



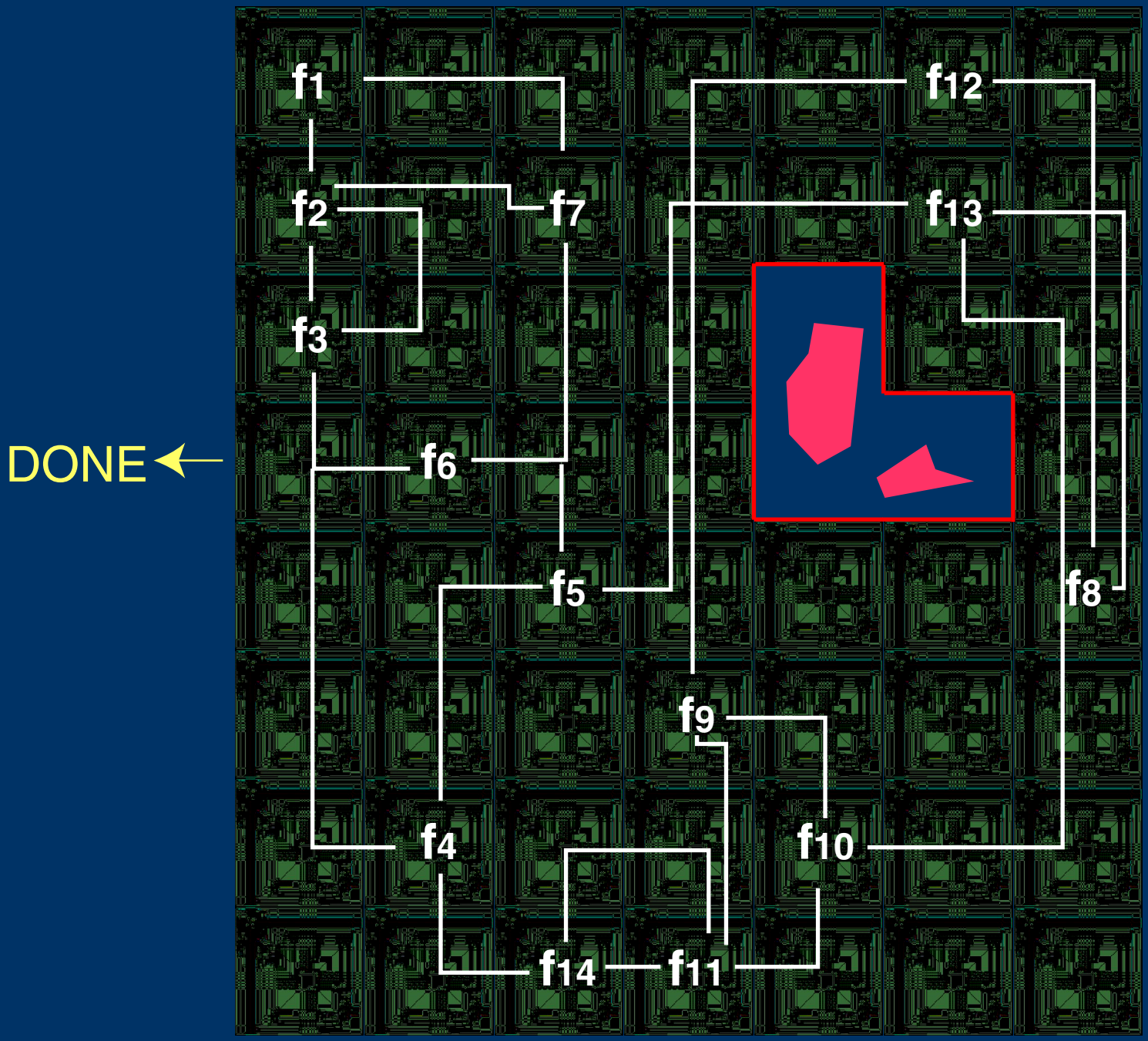
Supercell Differentiation



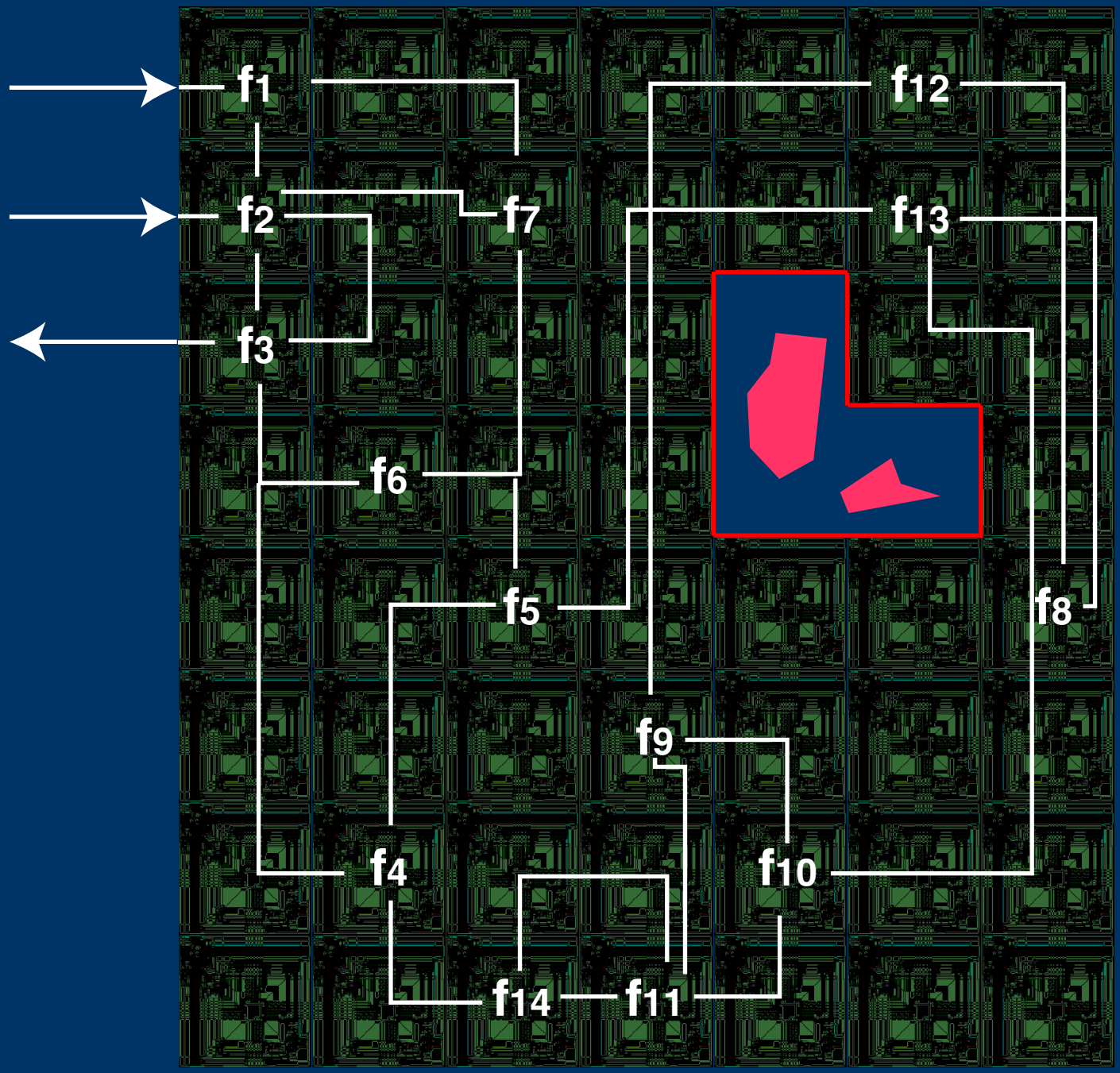
Inter-Cell Wiring



State Machine Completion



External I/O Connections



Key Points

- Minimal External Intervention
- Config string fixed per circuit
- No blessed HW except config string
- Final layout varies run-to-run
- Supercells only occupy good regions-thus they work perfectly
- Guard ring surrounds bad areas

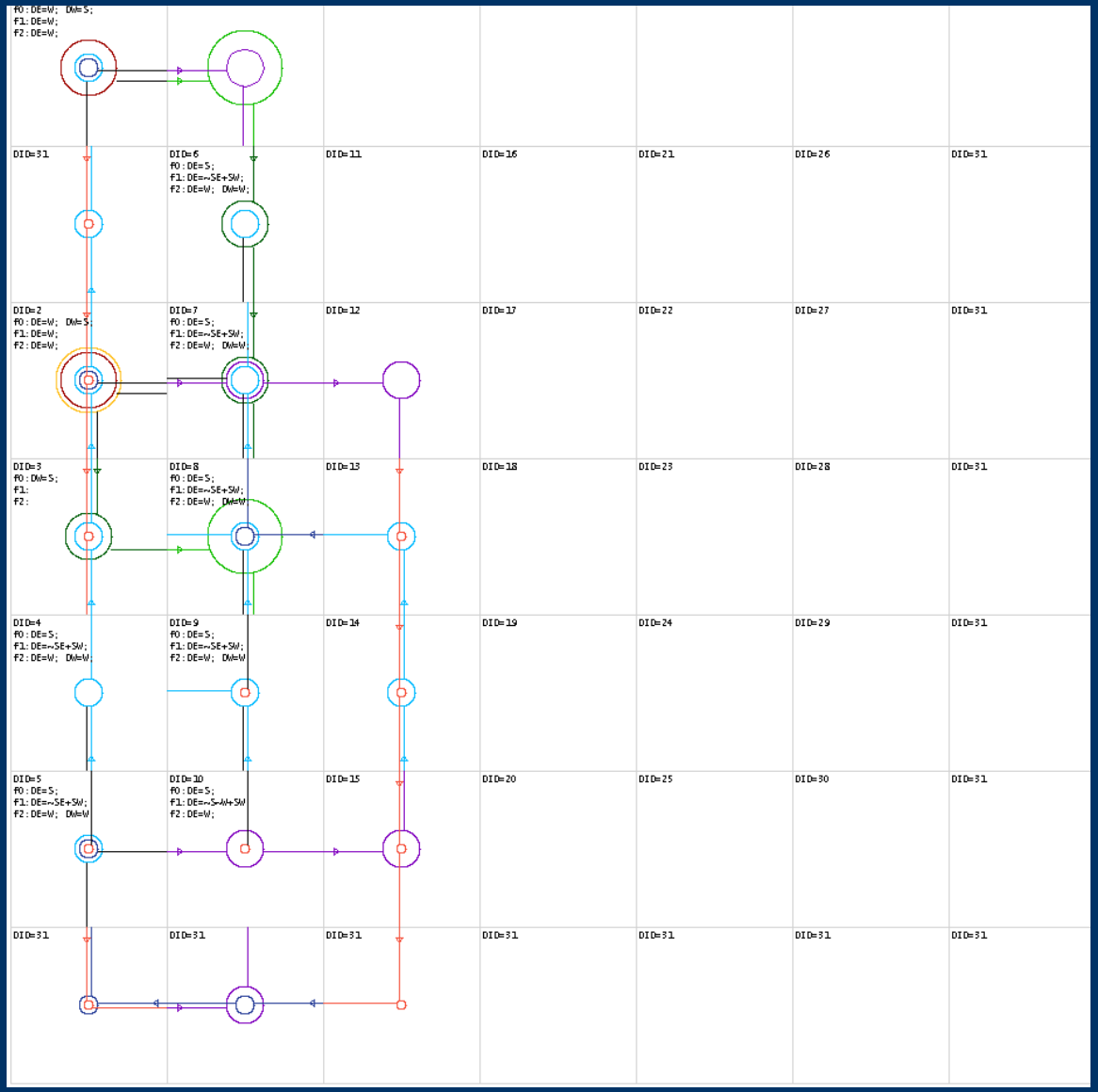
Test Setup

- Simulated Cell Matrix
- Simulated faults
  Variety of fault conditions
- Create test circuit genome
  Manual, but easy to automate
- Embed genome inside Supercell
- Create config string
- Send config string into empty
  Cell Matrix

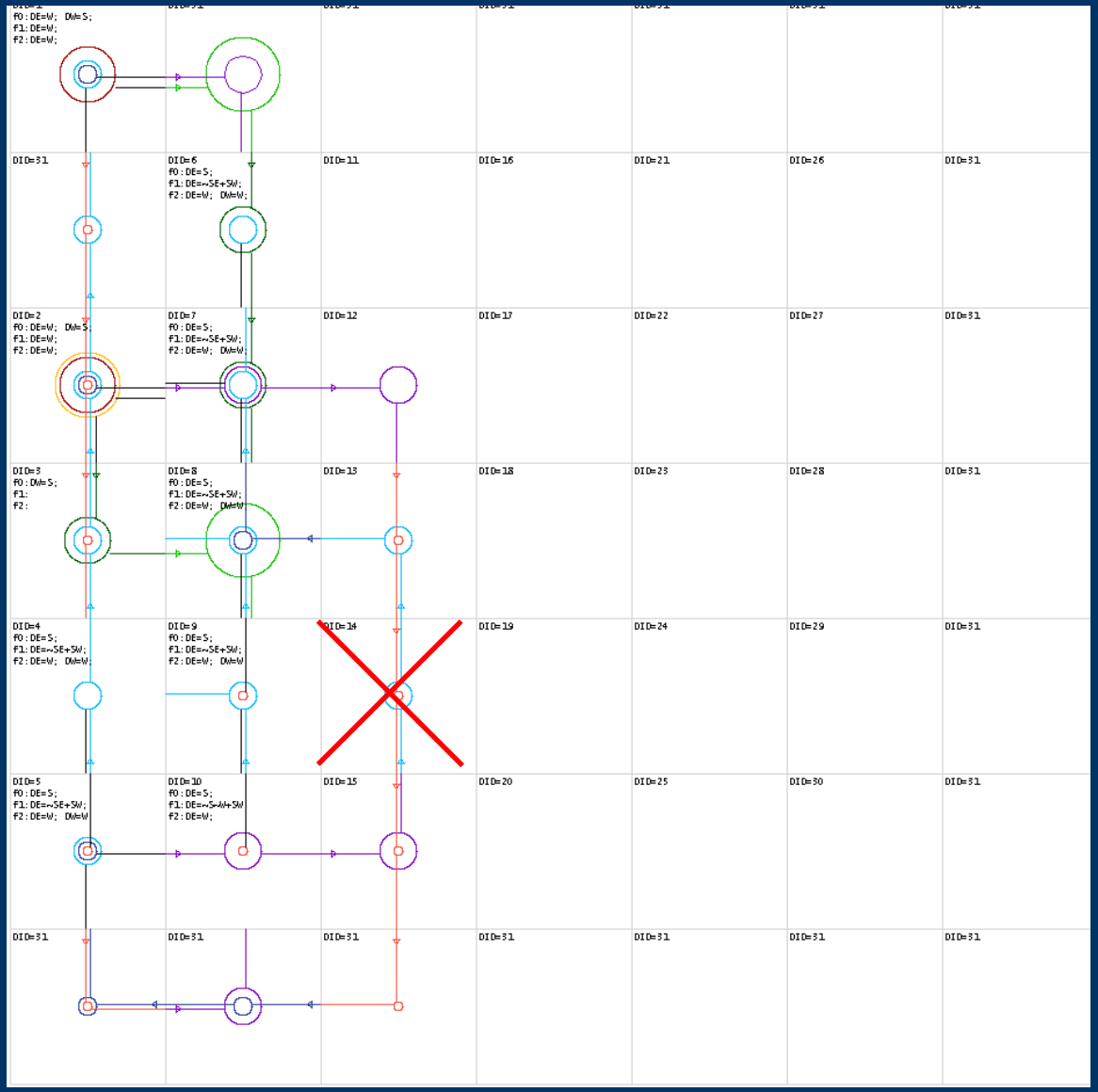
Analysis

- Test actual circuit behavior
  Always worked correctly
- Record system state
- Post-run analysis
  Graphical, human-readable form

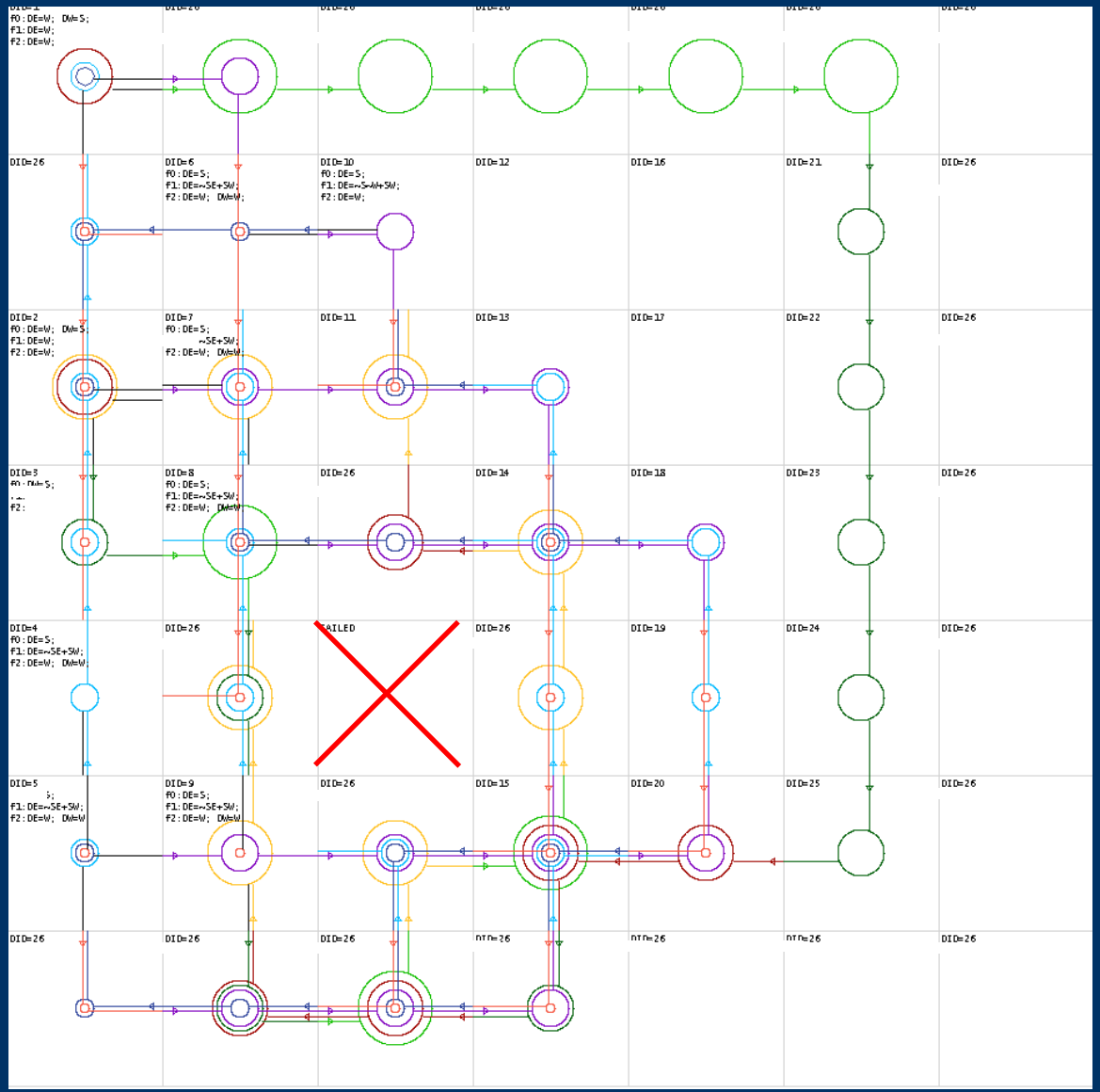
Layout Without Faults



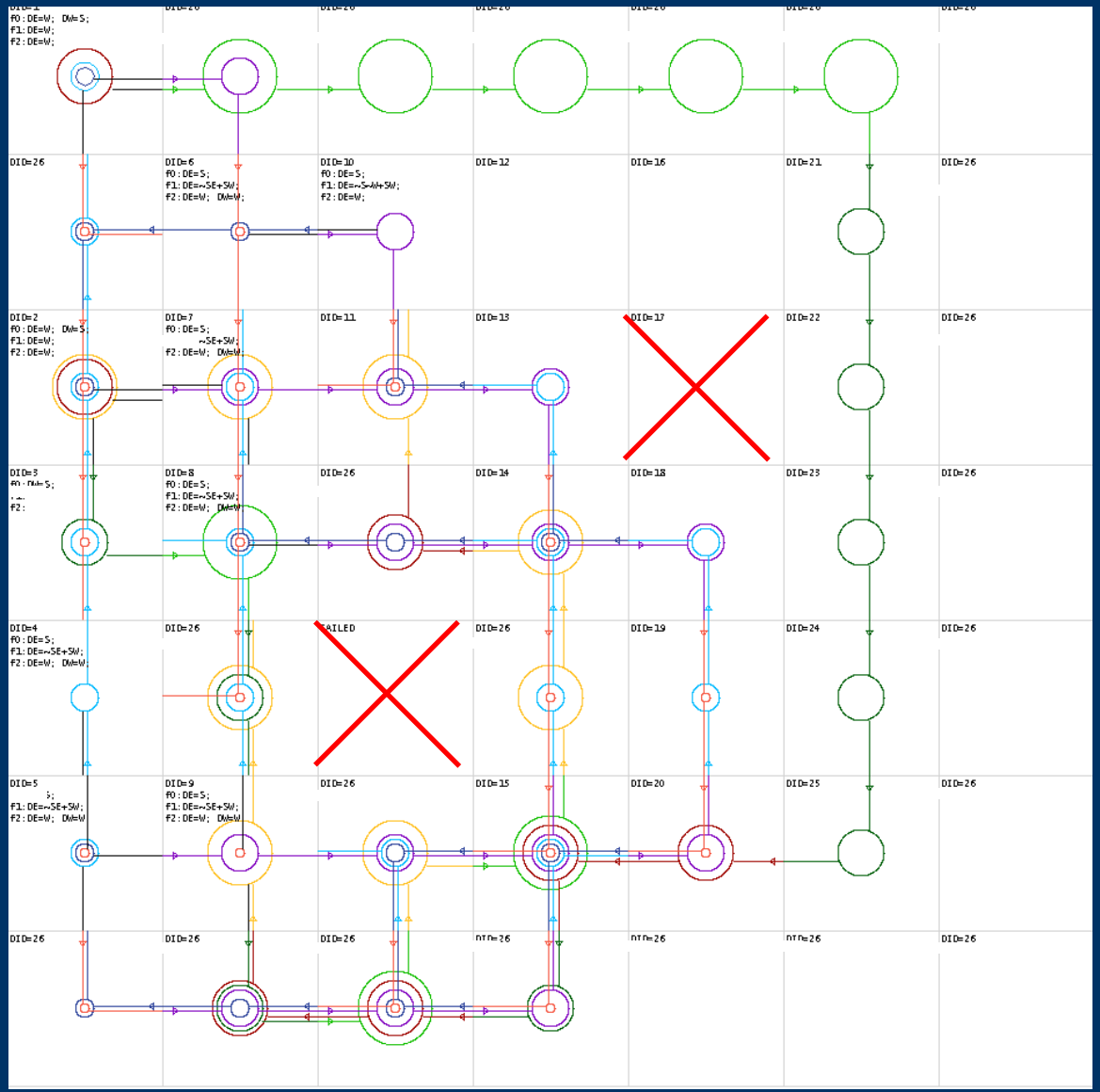
Layout Without Faults



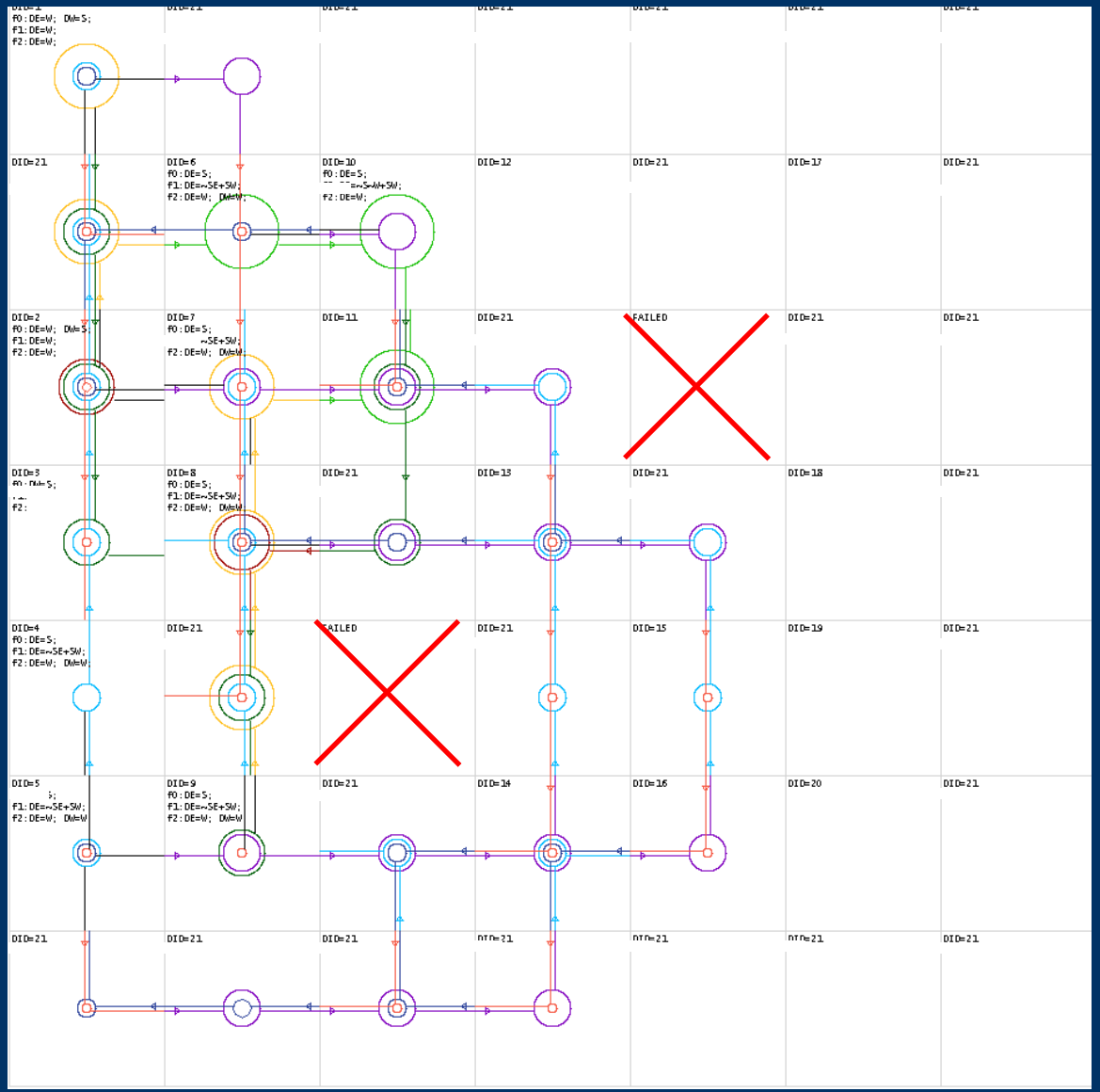
Layout Without Single Fault



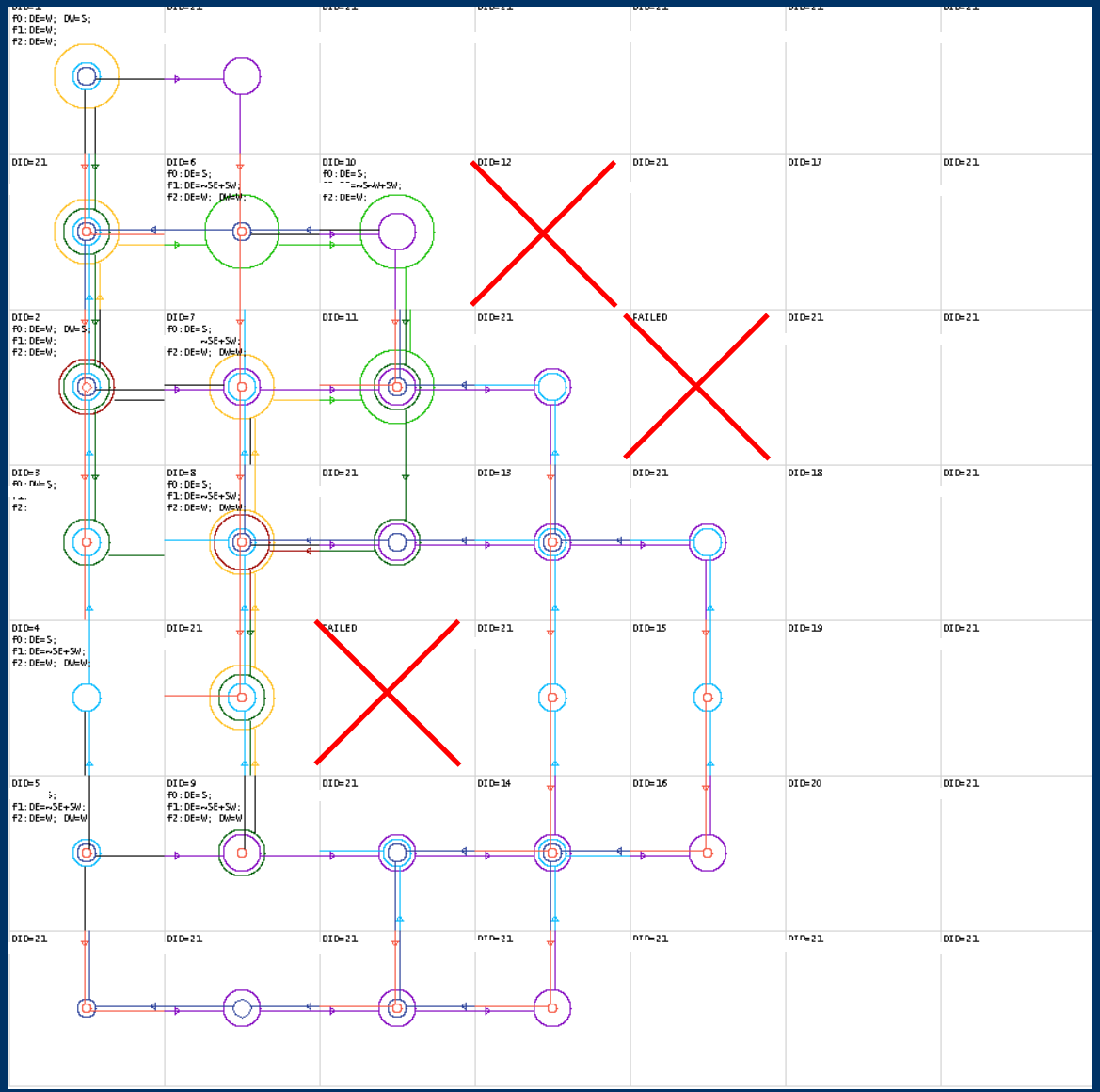
Layout Without Single Fault



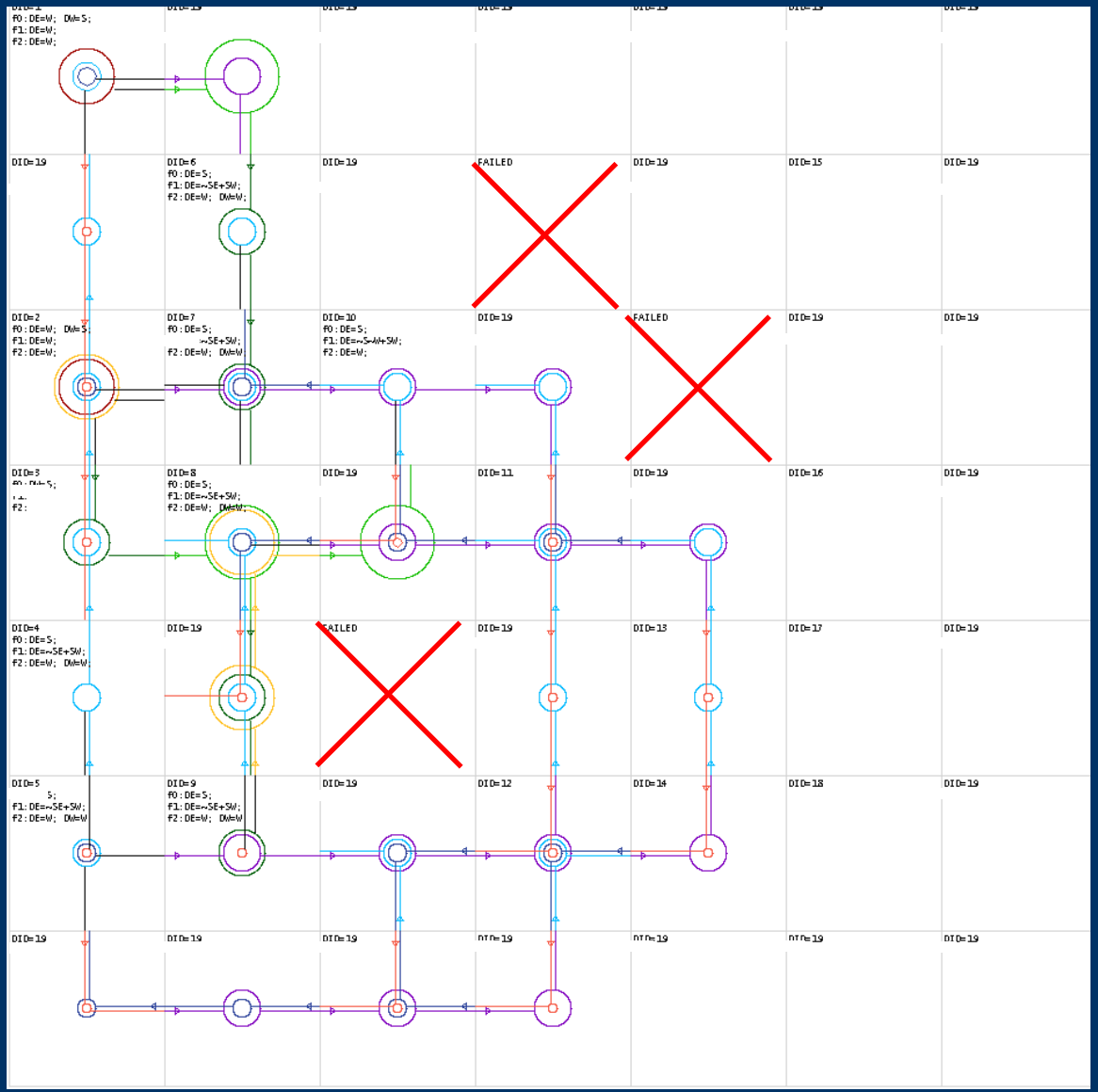
Layout Without Two Faults



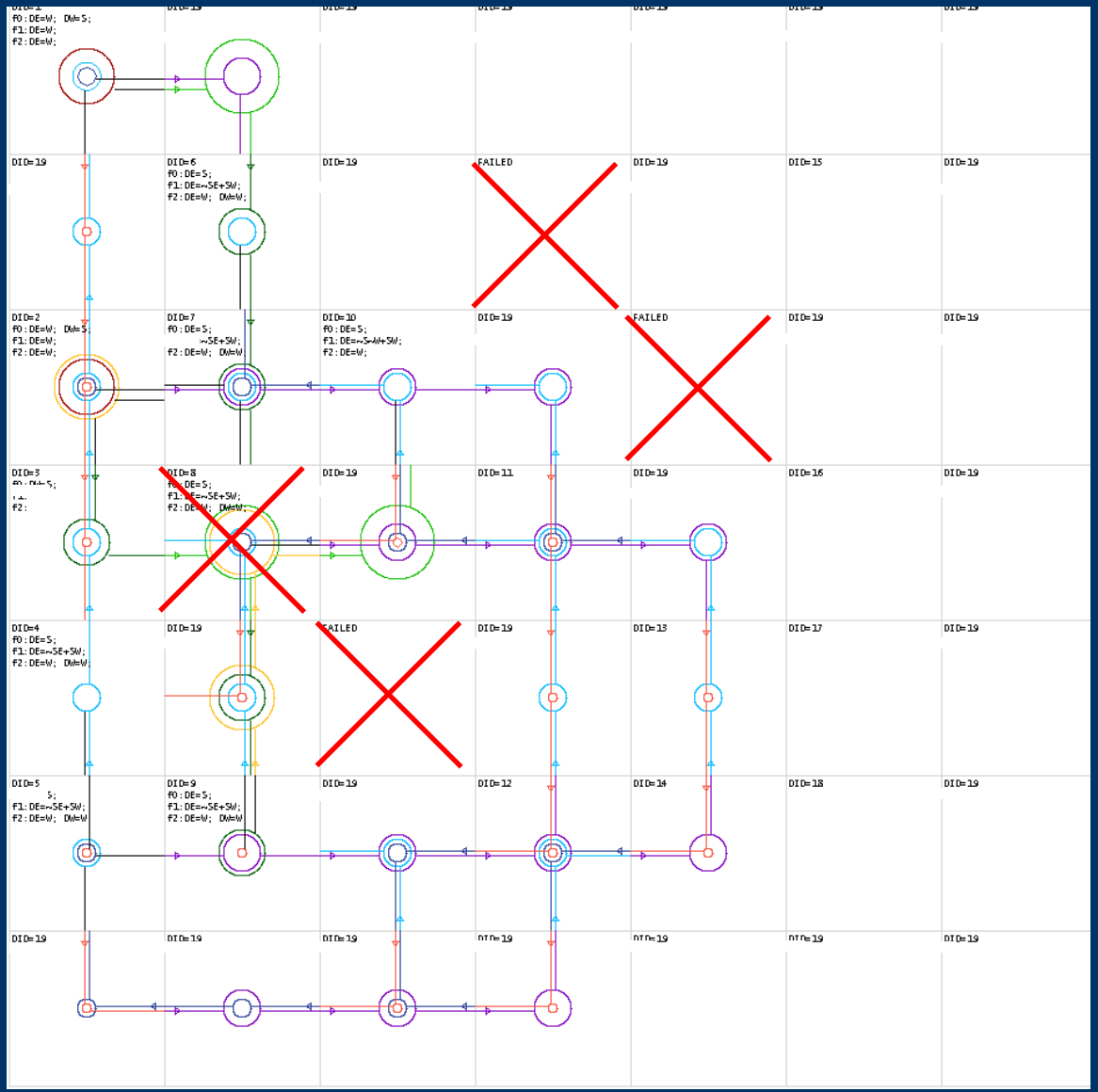
Layout Without Two Faults



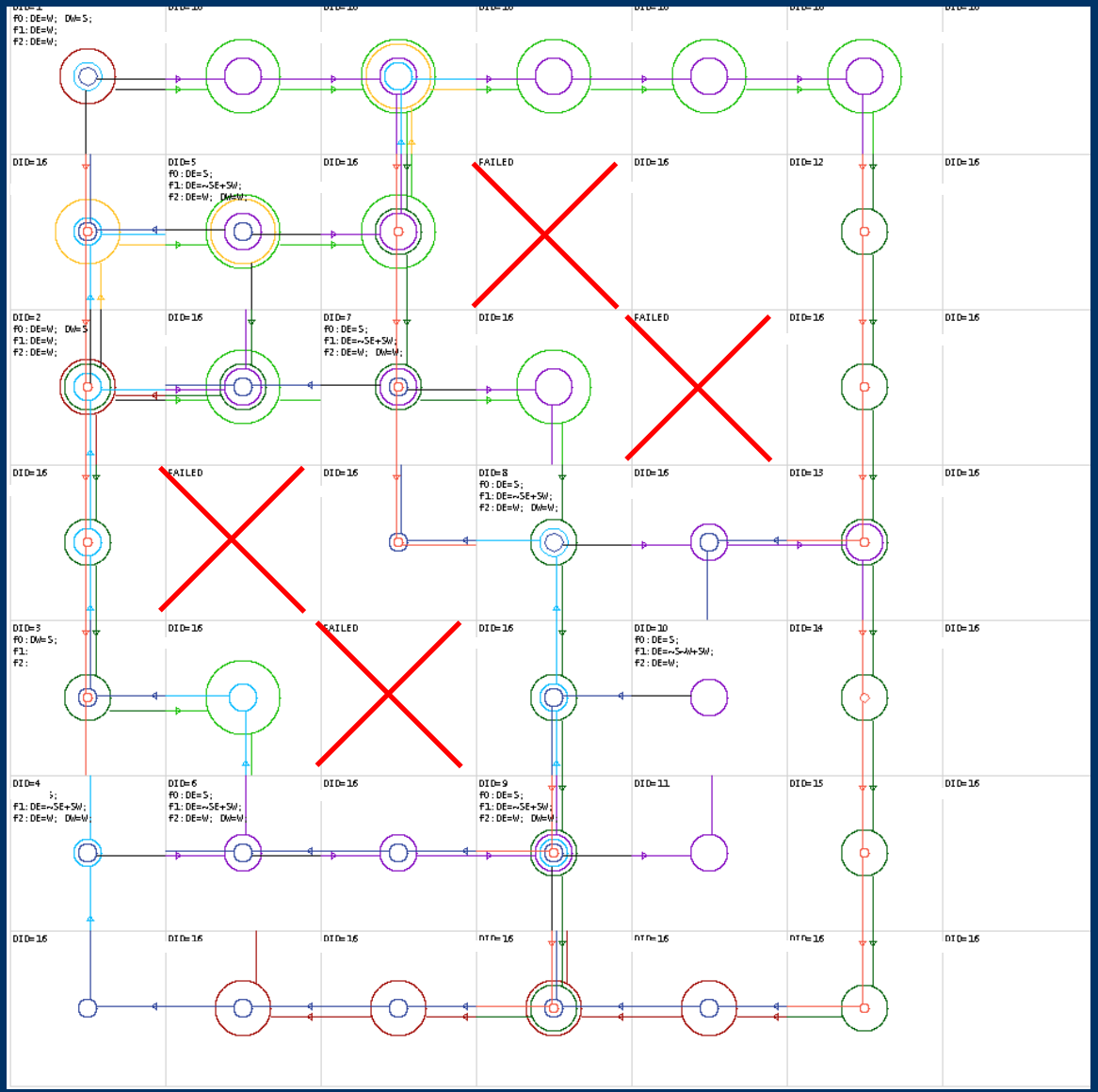
Layout Without Three Faults



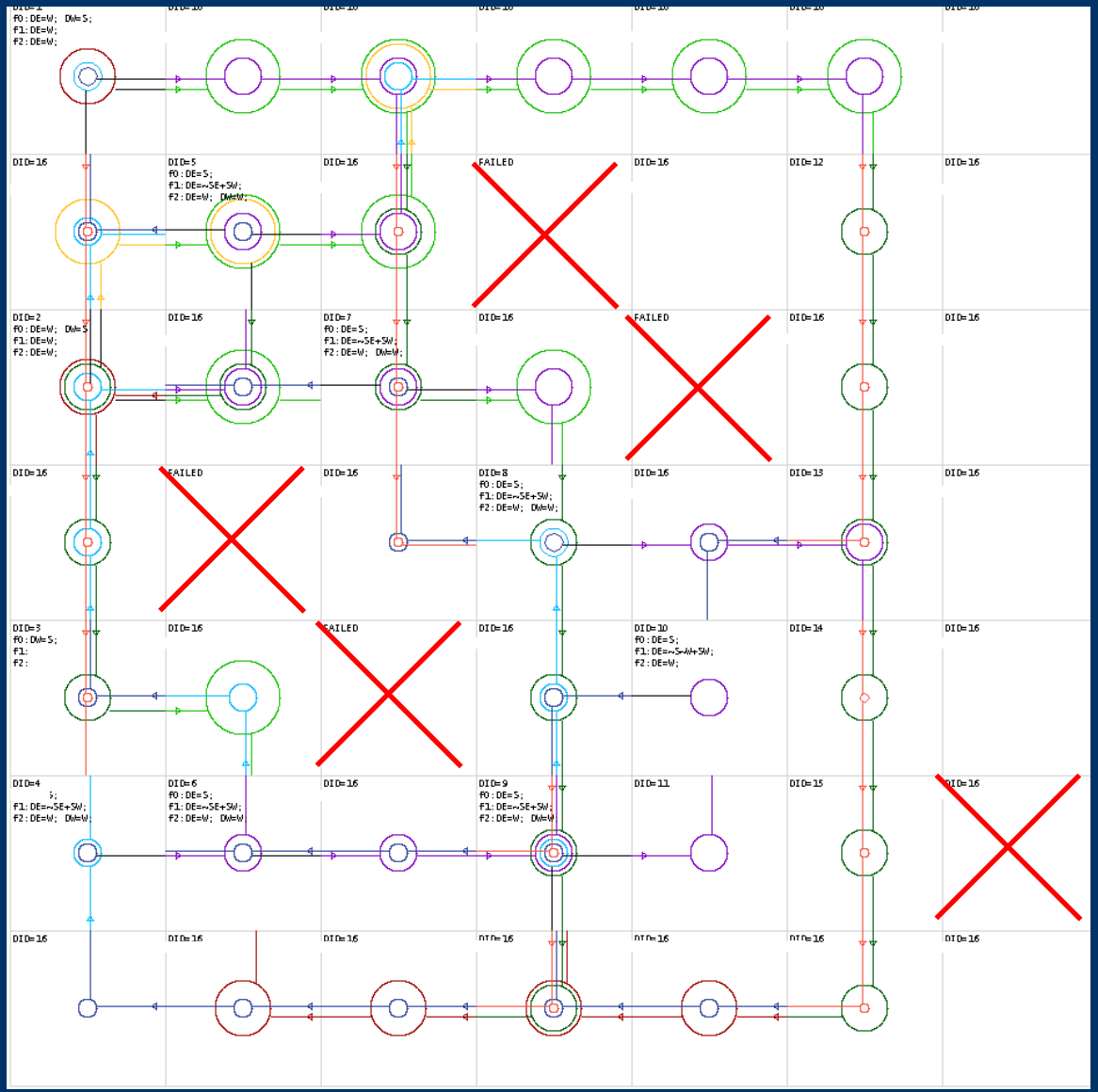
Layout Without Three Faults



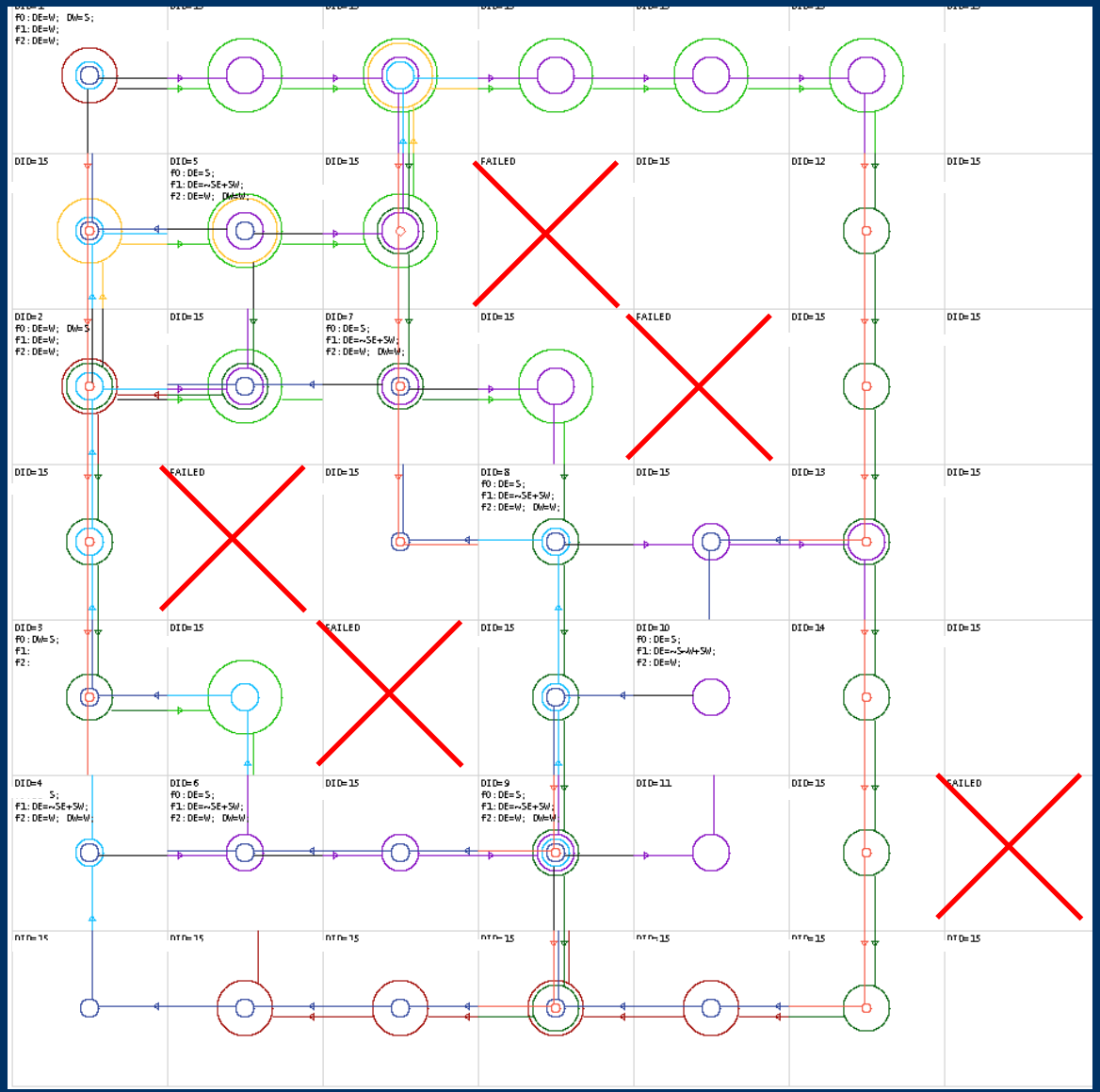
Layout Without Four Faults



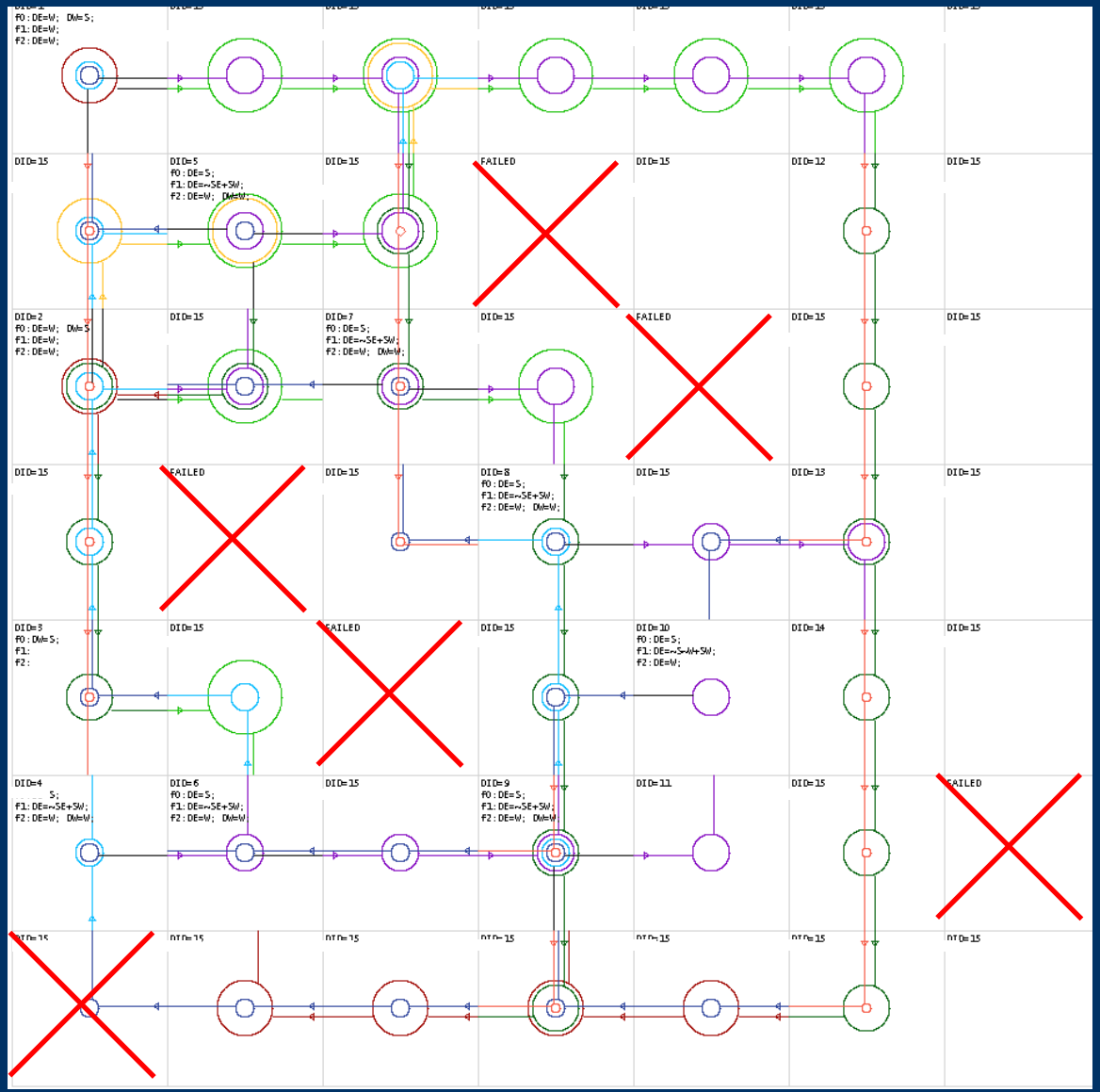
Layout Without Four Faults



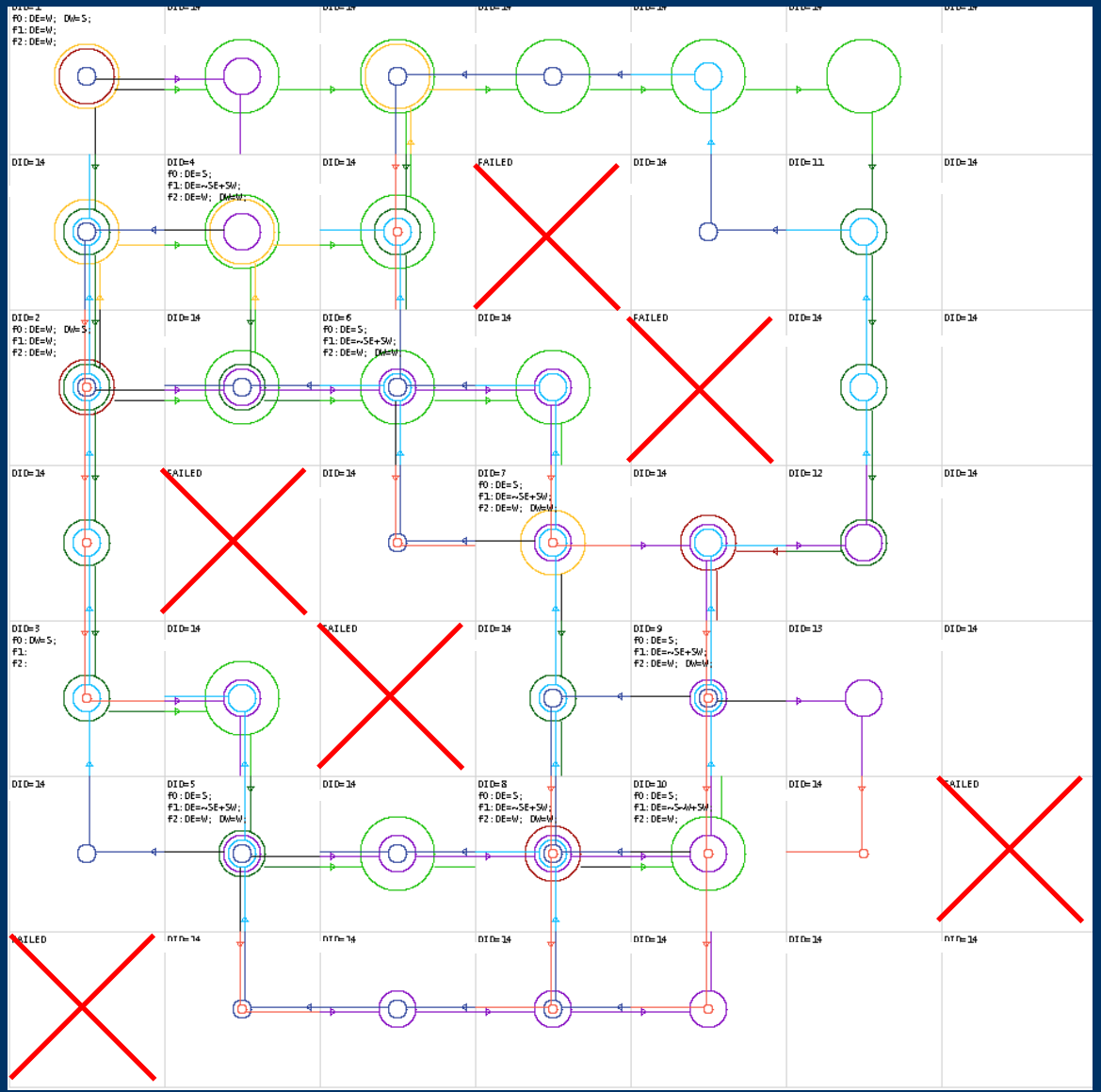
Layout Without Five Faults



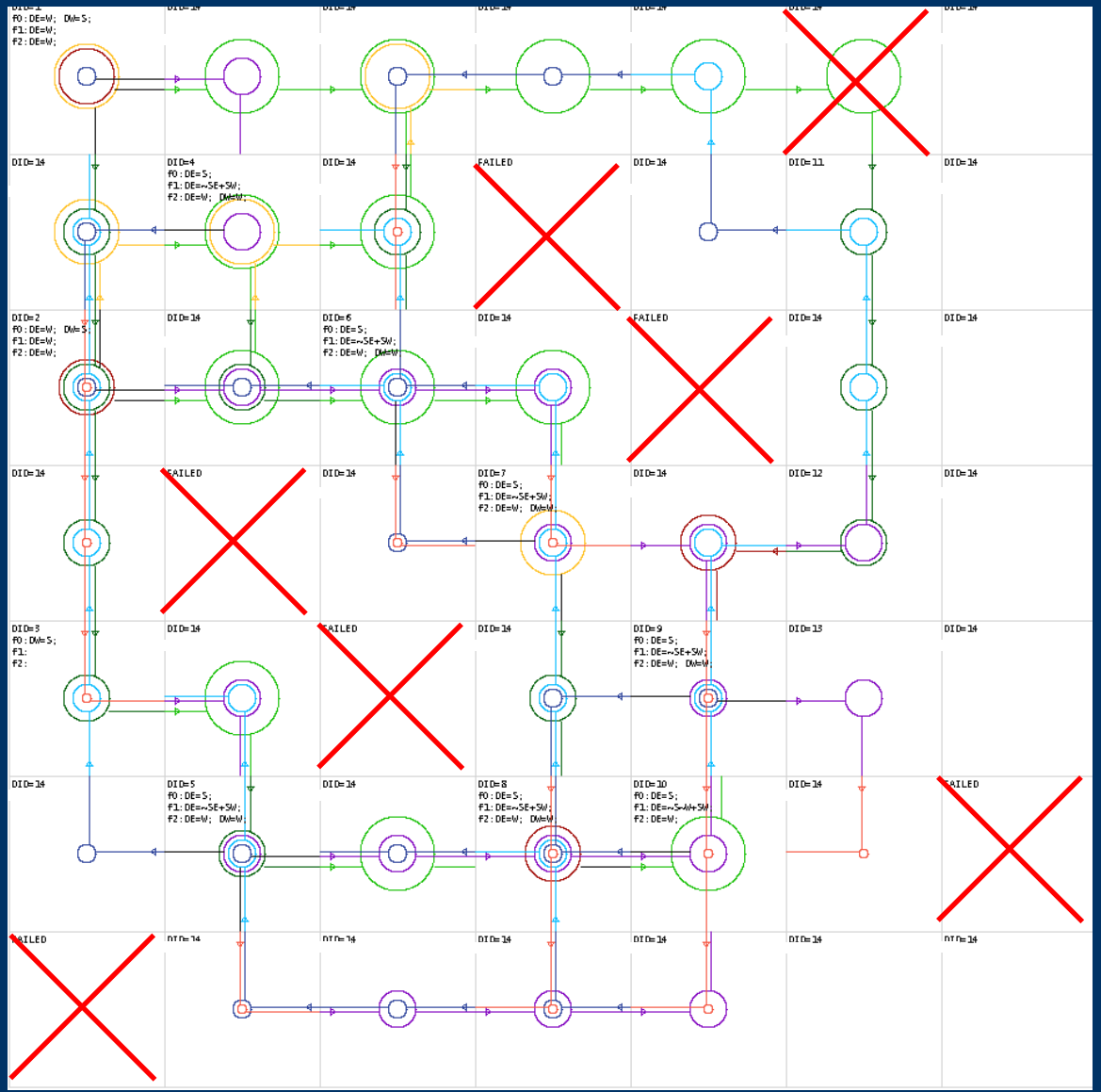
Layout Without Five Faults



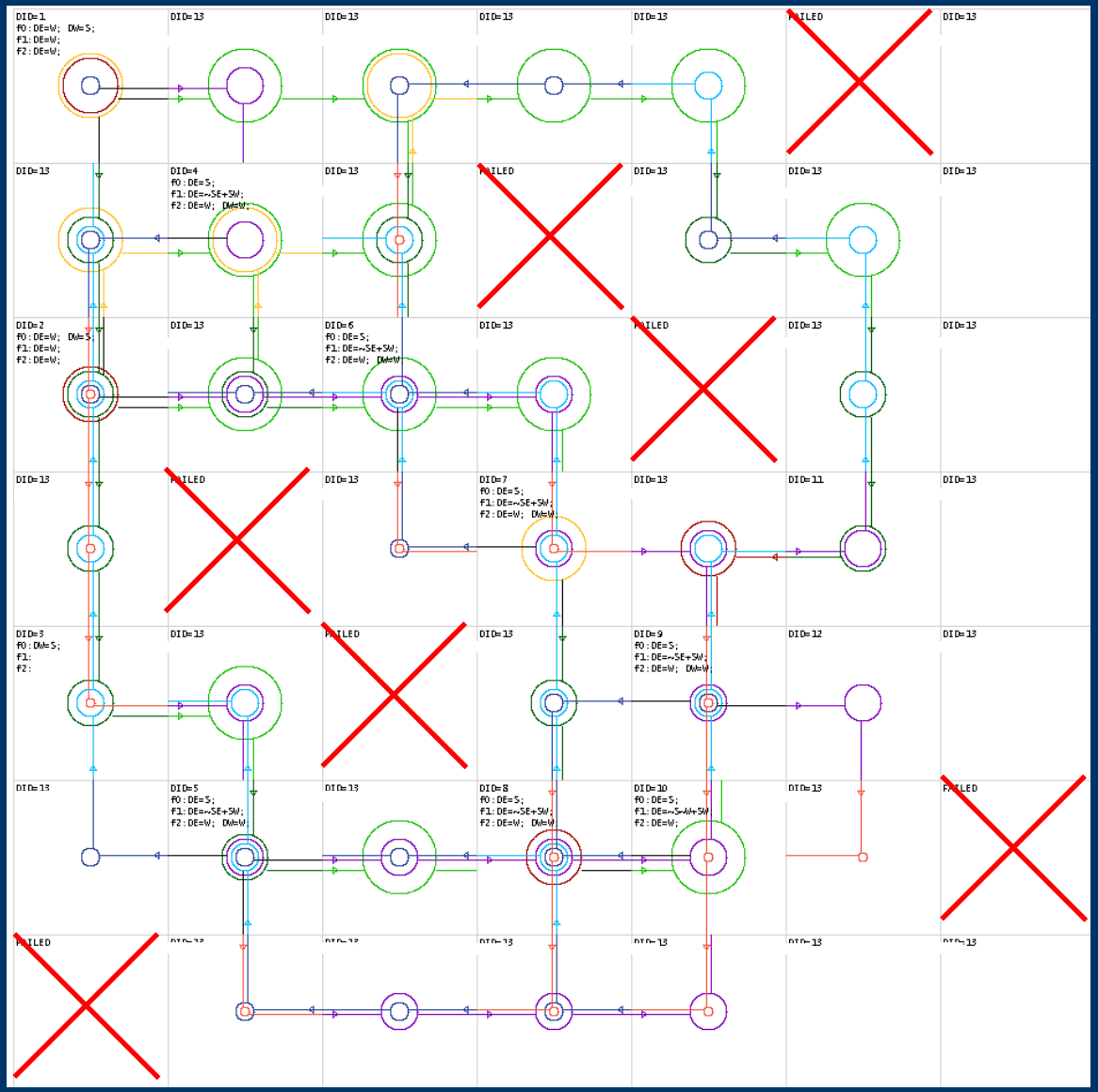
Layout Without Six Faults



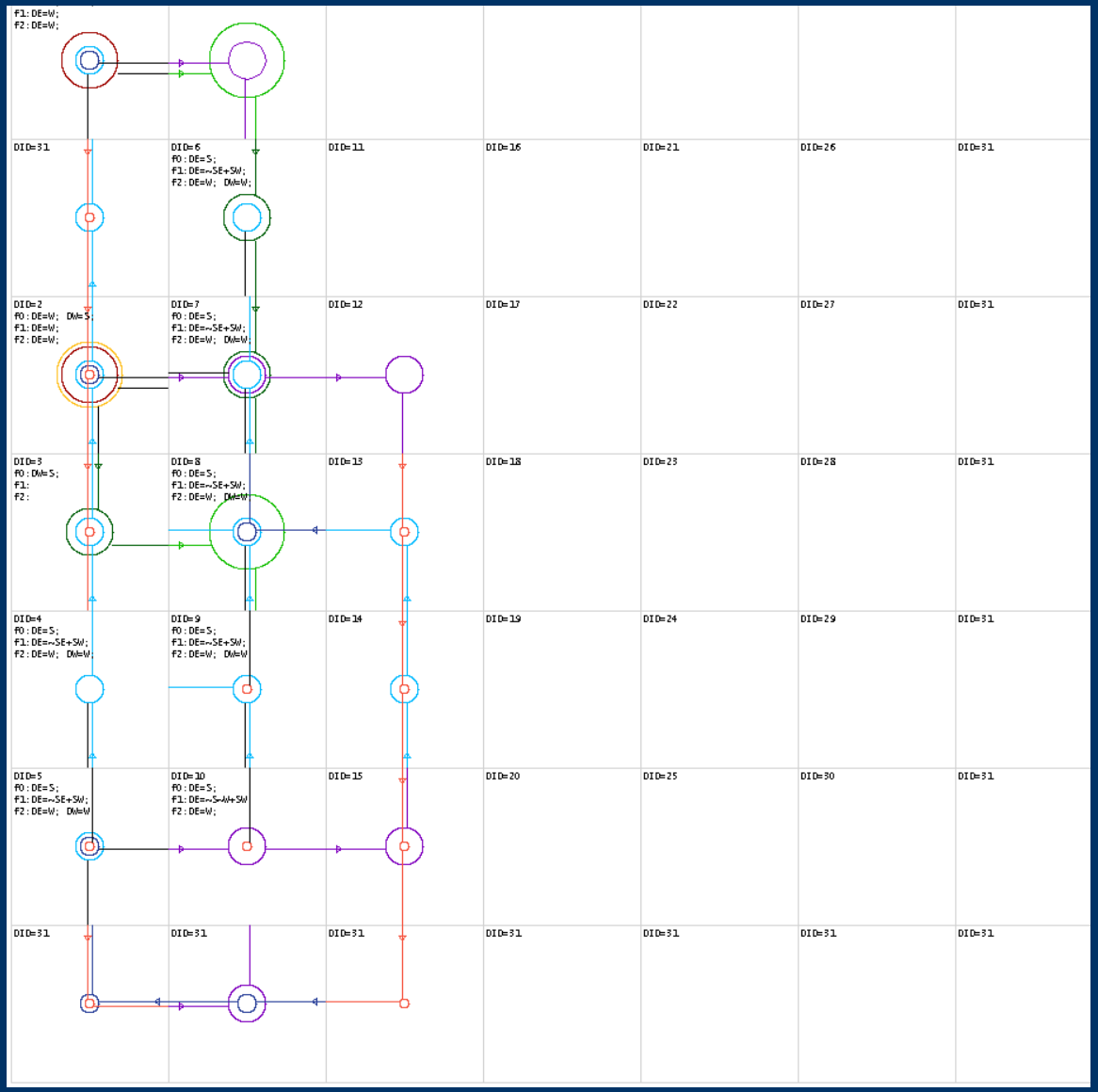
Layout Without Six Faults



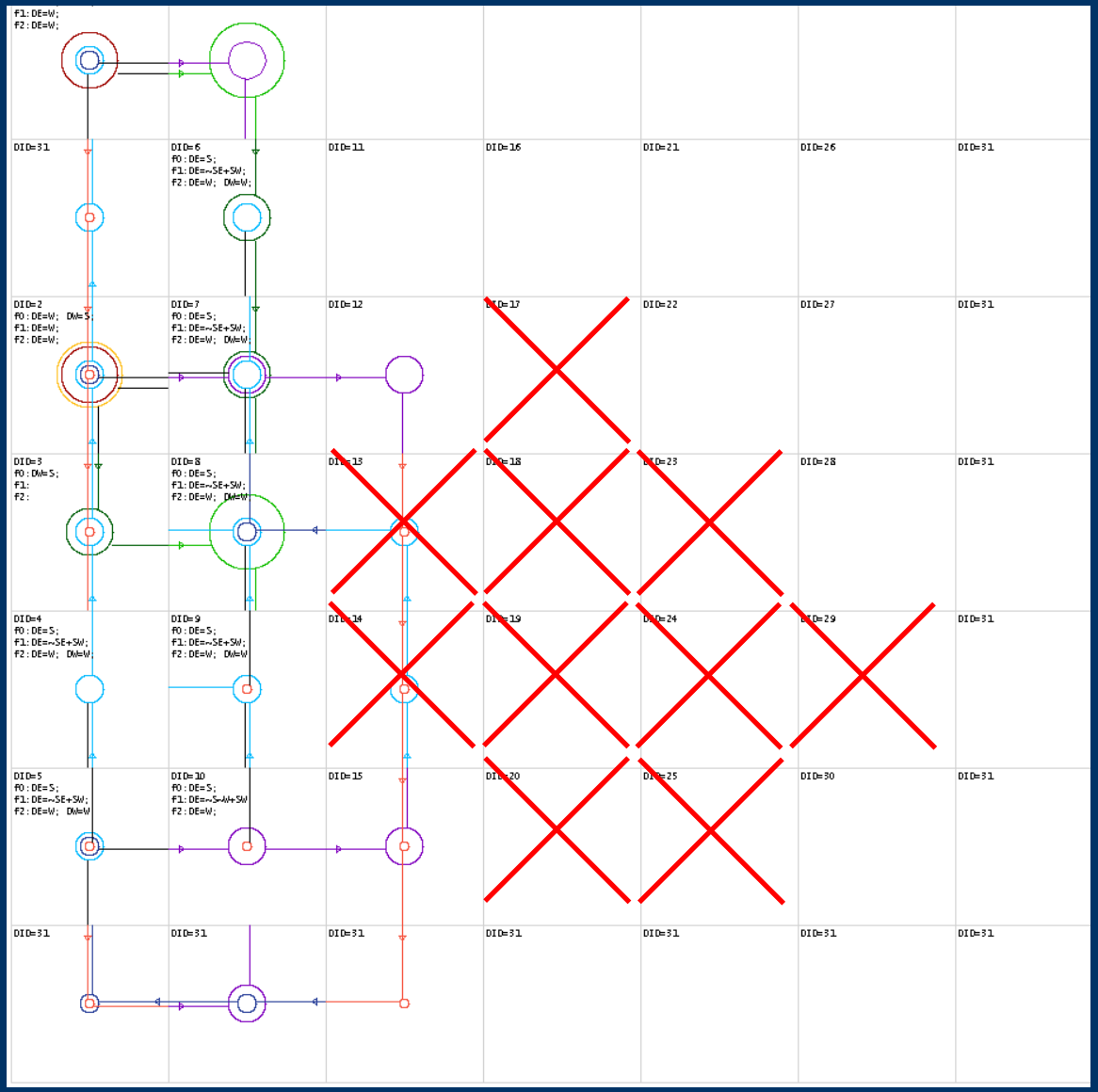
Fault in Every Row & Column



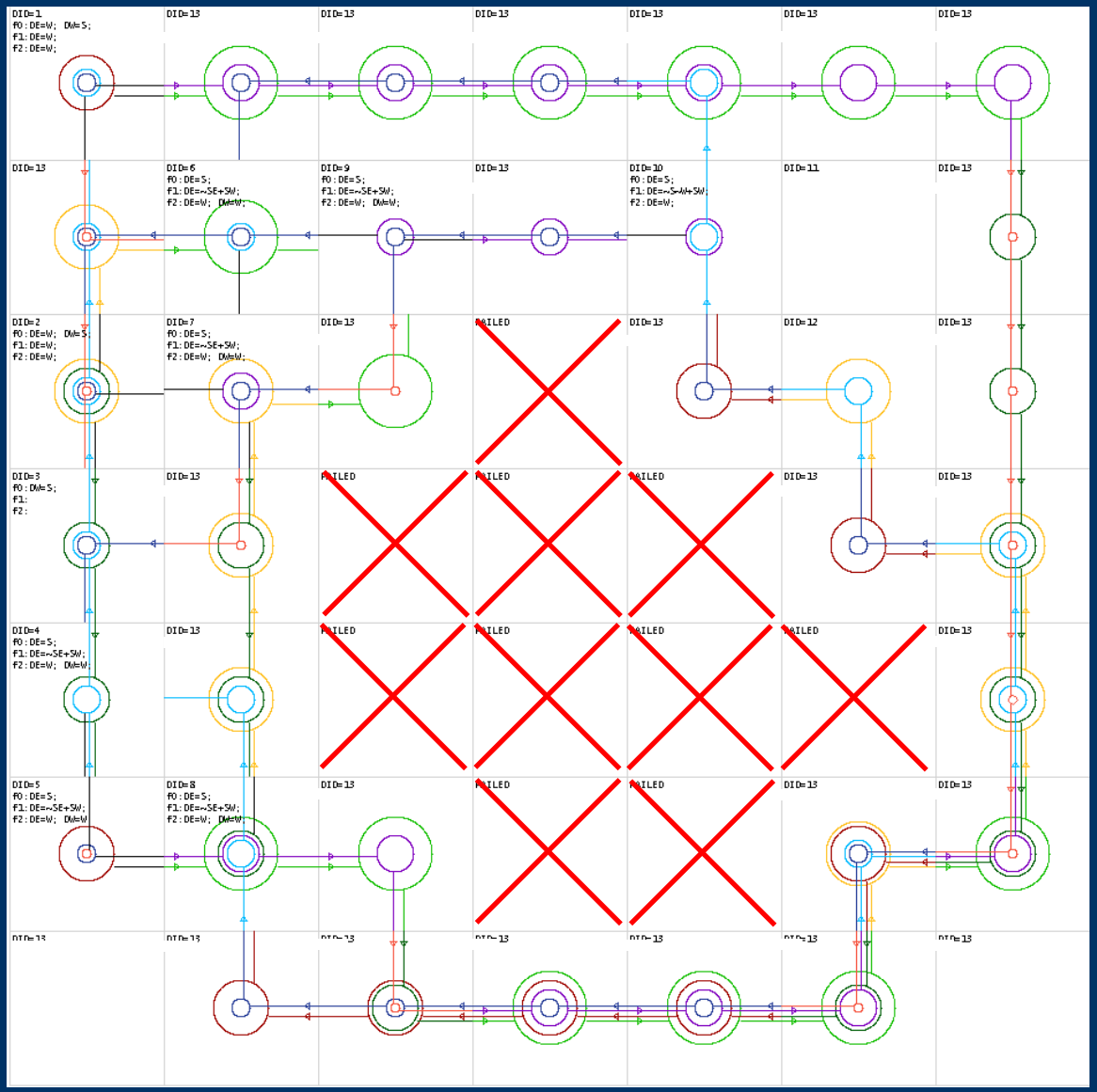
Layout Without Faults



Layout Without Faults



Single Large Fault Region (20% Faults)



Summary

- Target circuit succesfully implemented
- System operates autonomously:
  Fixed configuration string,
  Different resulting layouts
- Entire system defect resistant
  except config string storage/delivery
- Parallel test and configure-fast!
- Standard Cell Matrix architecture

Future Work

- Three-dimensional version
- Reduce Supercell size (down to 0?)
- Add fault **detection**
- Embed configuration string inside Supercell

FOR MORE INFORMATION

www.cellmatrix.com
nmacias@cellmatrix.com
1-540-357-1143
In US:1-877-473-0882