Benefits and Applications of Self-Configurable Hardware: The Cell Matrix Architecture

Nicholas J. Macias
Cell Matrix Corporation

Overview of Talk

- High-Level Introduction
- Detailed Cell Behavior
- Benefits of the Architecture
- Applications of the system
- Status and future plans

HIGH LEVEL DESCRIPTION OF A CELL MATRIX

What is a Cell Matrix?

- Configurable Hardware:
  - Fixed Hardware Structure
  - Variable Functionality provided by subsequent *configuration* step

What is a Cell Matrix?

- Configurable Hardware
- Fine-Grained
  - Simple set of *cells* with very simple behavior
  - More complex behavior is built up from collections of cells
  - Cells are individually configurable via per-cell configuration memory

What is a Cell Matrix?

- Configurable Hardware
- Fine-Grained
- Cells are interconnected via fixed, nearest-neighbor topology
  - Cells exchange information with their neighbors

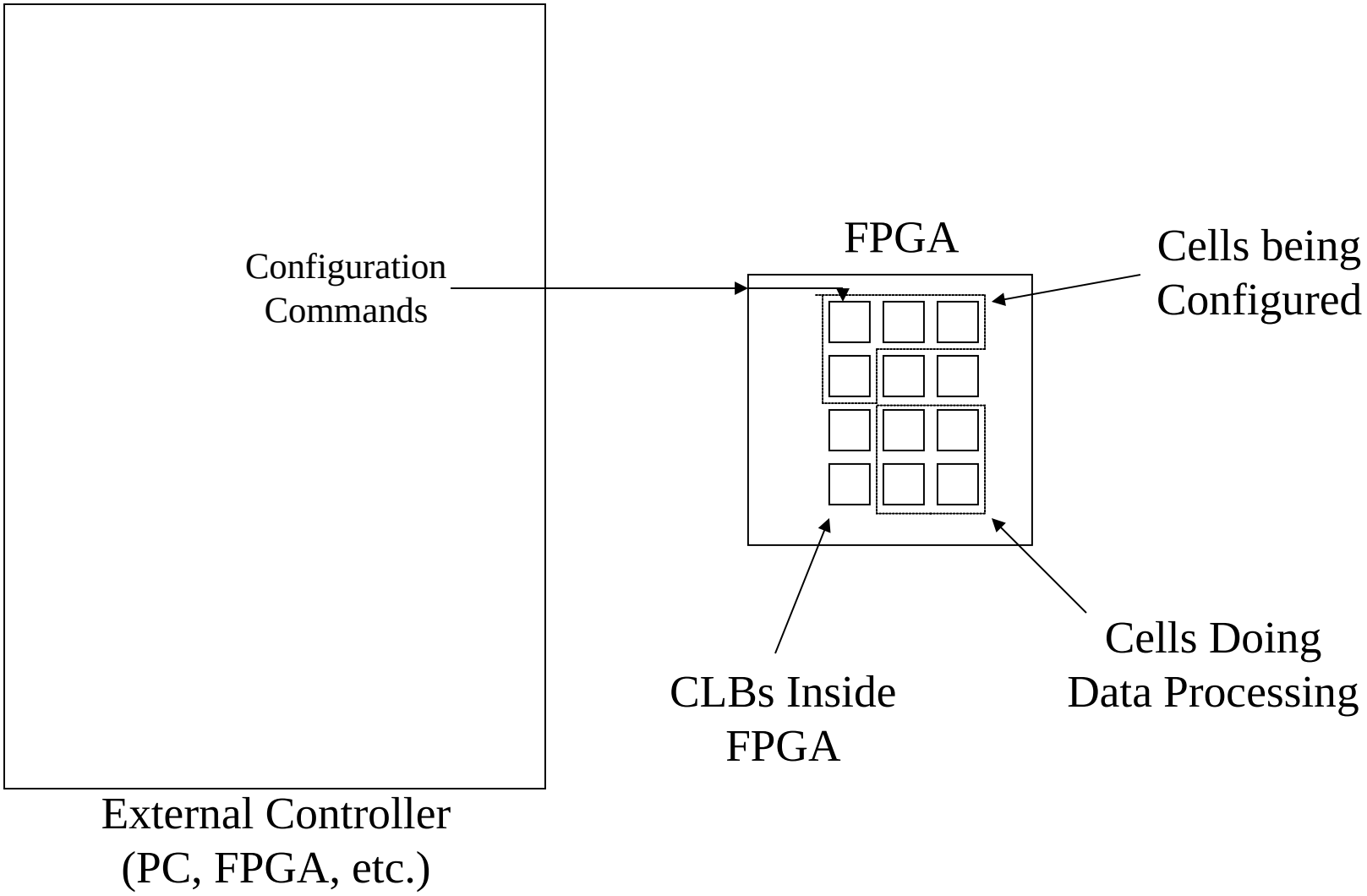
This Might Sound a Lot Like an FPGA

BUT…

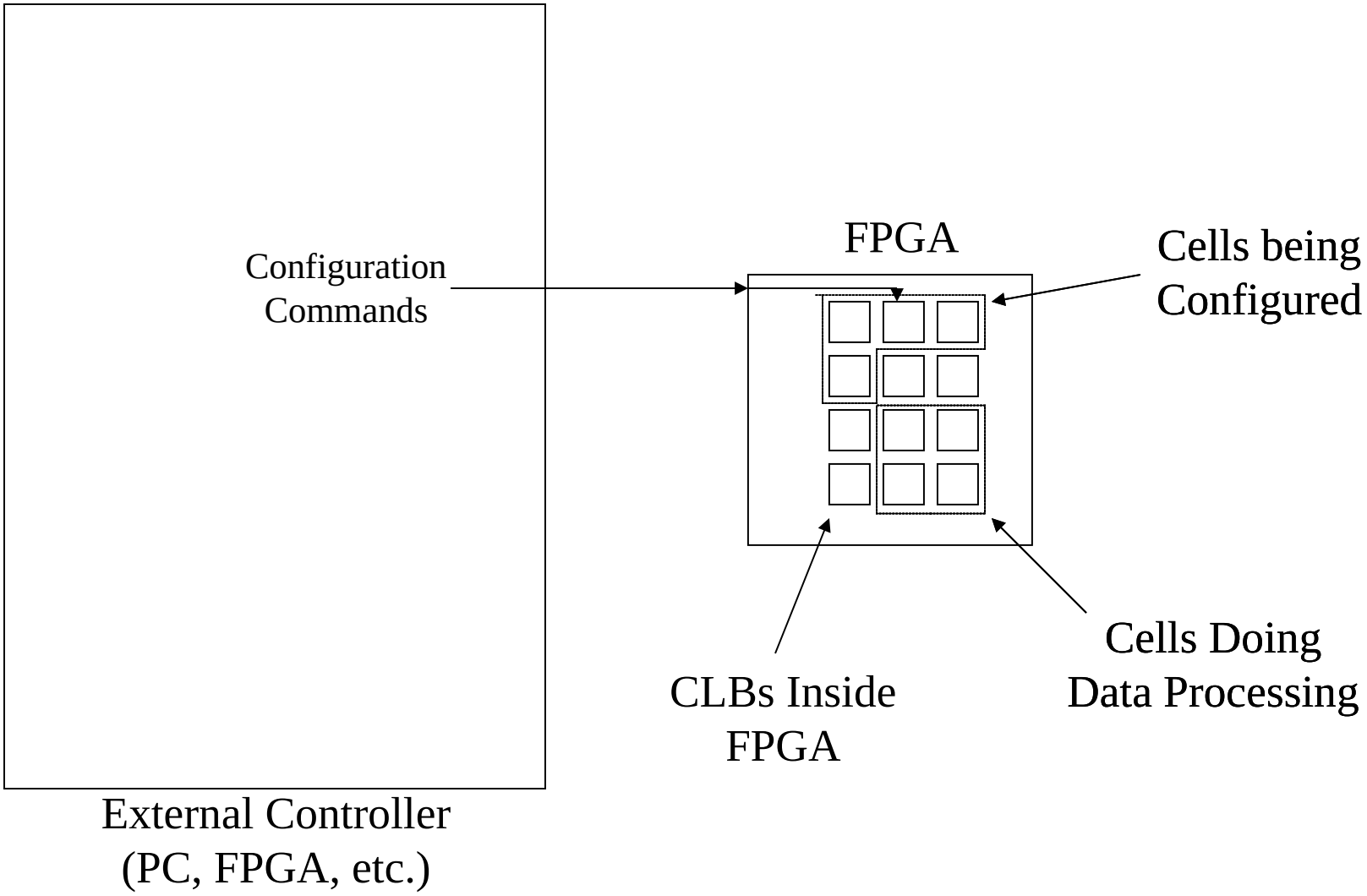
An FPGA is an **externally** configured device

- You send configuration information into the device, and the device responds accordingly.
- Configuration generation (external) is distinct from information processing (internal)

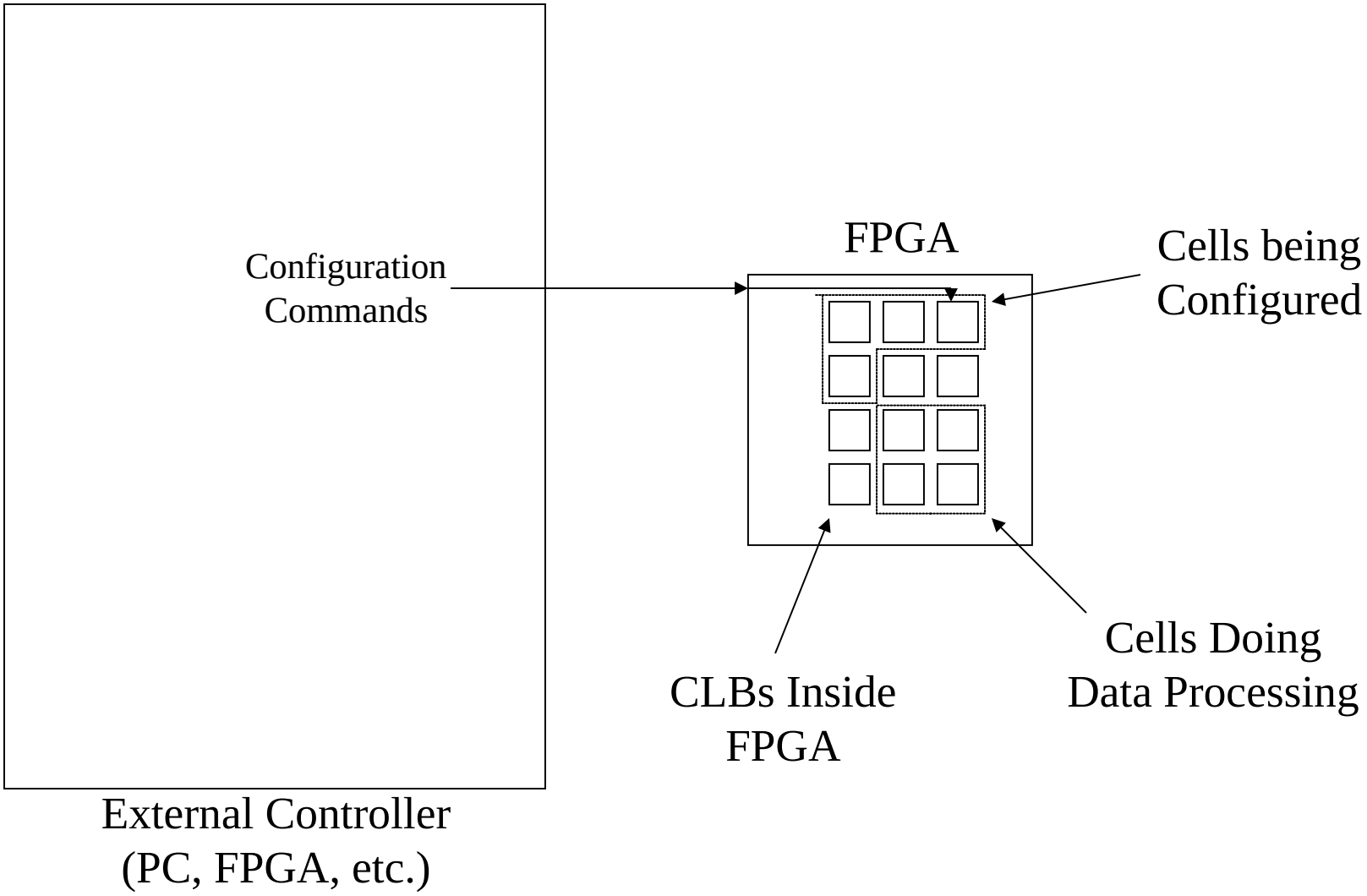
Cell Configuration in a Standard FPGA



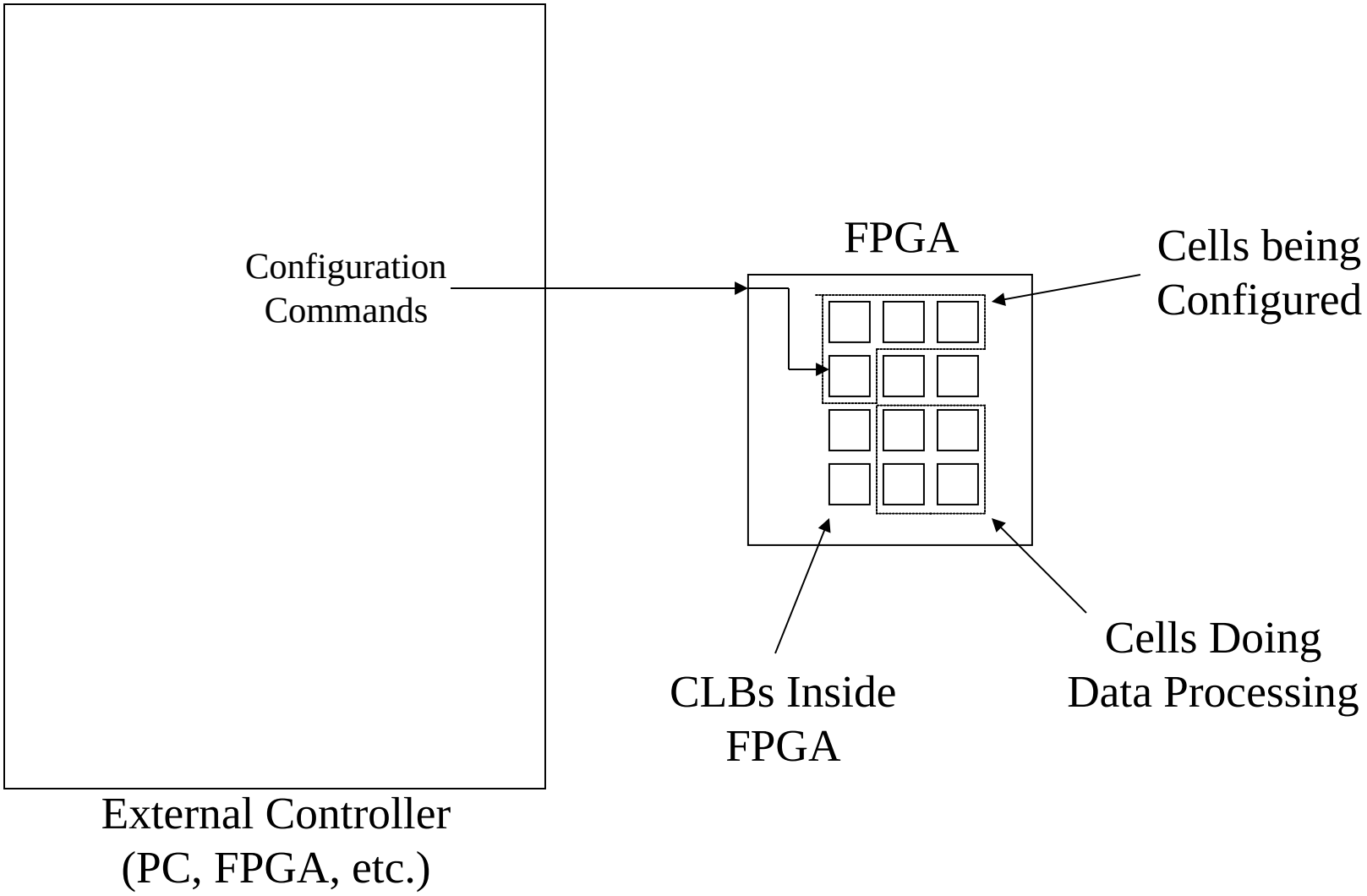
Cell Configuration in a Standard FPGA



Cell Configuration in a Standard FPGA



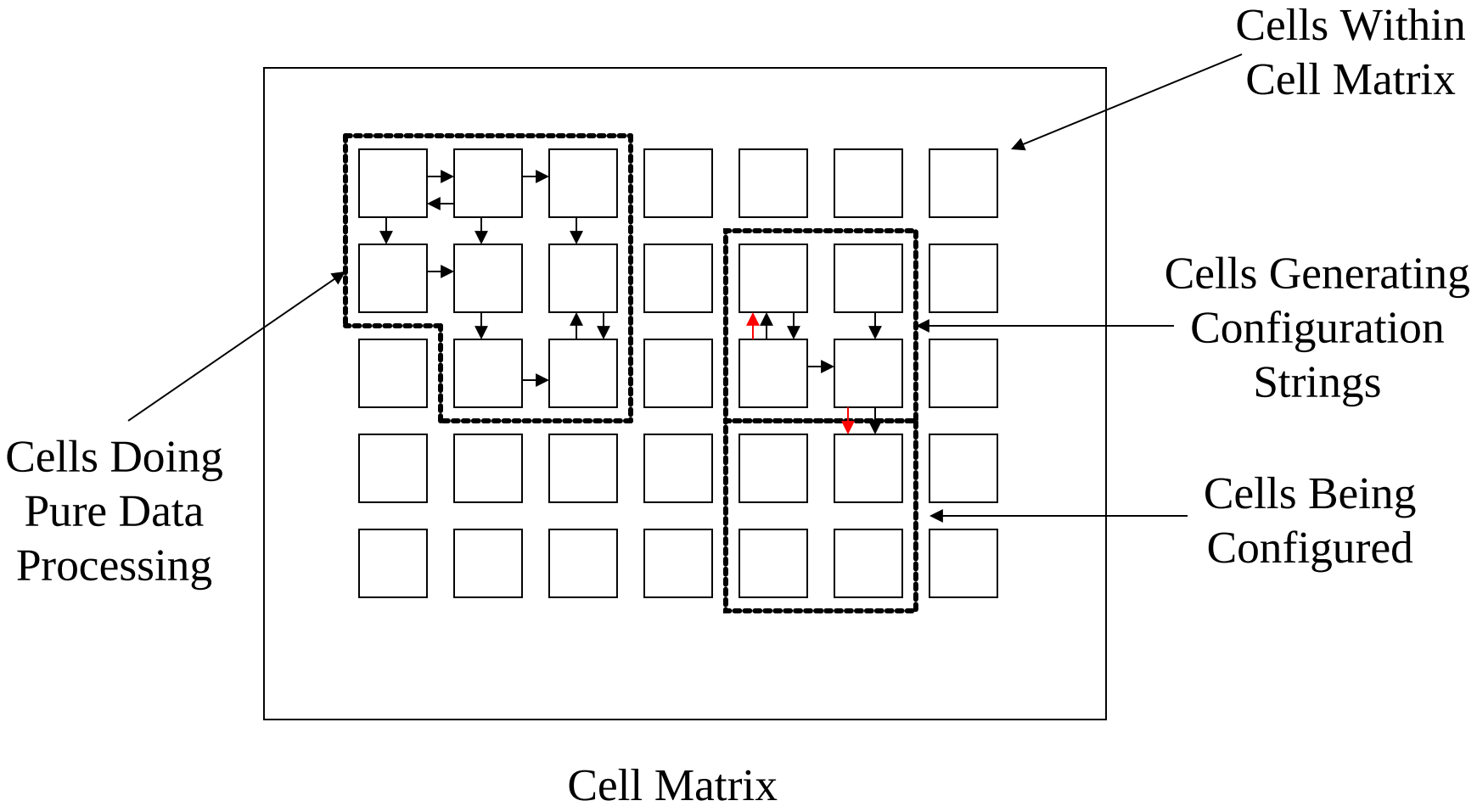
Cell Configuration in a Standard FPGA



A Cell Matrix is an **internally** configured device

- A cell within the matrix can write a neighboring cell's configuration memory (CODE)
  - This allows a cell to change a neighbor's behavior
- A cell can also read a neighbor's current configuration

Code and Data Processing Within a Cell Matrix



Information Flow Inside a Cell Matrix

- In addition to cells exchanging DATA with each other, cells also exchange CODE
- Fundamental difference from FPGA

- Also different from 2 FPGAs…

KEY CONCEPT

- Within a cell matrix, CODE and DATA are interchangeable
- Travel over same physical lines
- Indistinguishable from each other
- This is called Duality #1
  - Keep it in mind!

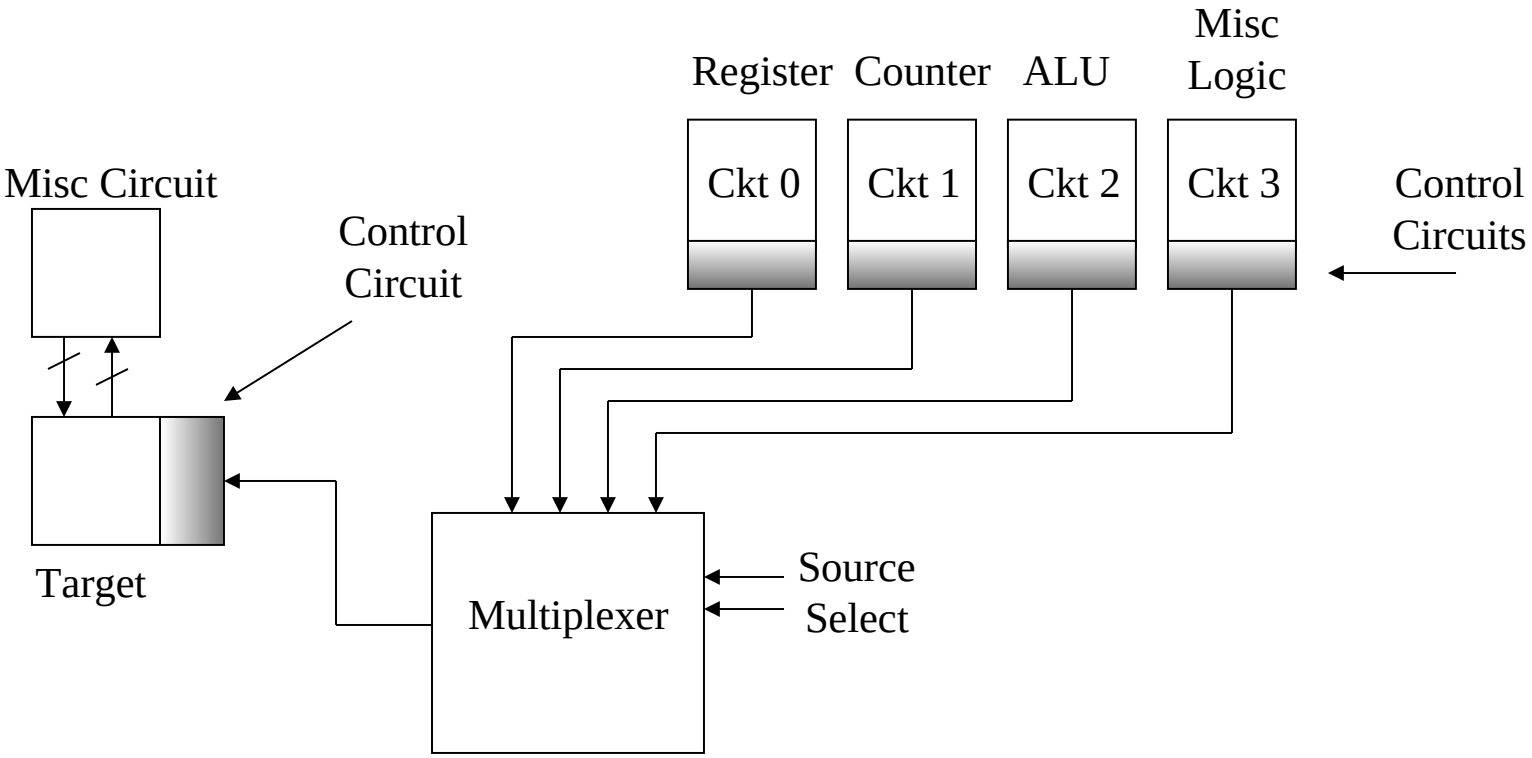
DUALITY #1

- Means you can build circuits which process **configuration** information in the same way they process data

DUALITY #1

- For example, you can store several configurations, select them via multiplexer, and so on, then use them to configure cells

Example


Register
Counter
ALU
Misc Logic
Ckt 0
Ckt 1
Ckt 2
Ckt 3
Control Circuits
Misc Circuit
Control Circuit
Target
Multiplexer
Source Select

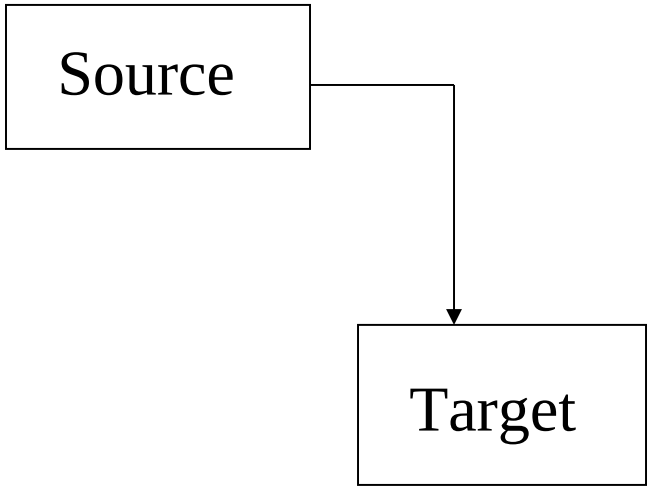
Stranger Examples

- Adder could be incrementing a configuration string!
- You could logically OR two configuration strings

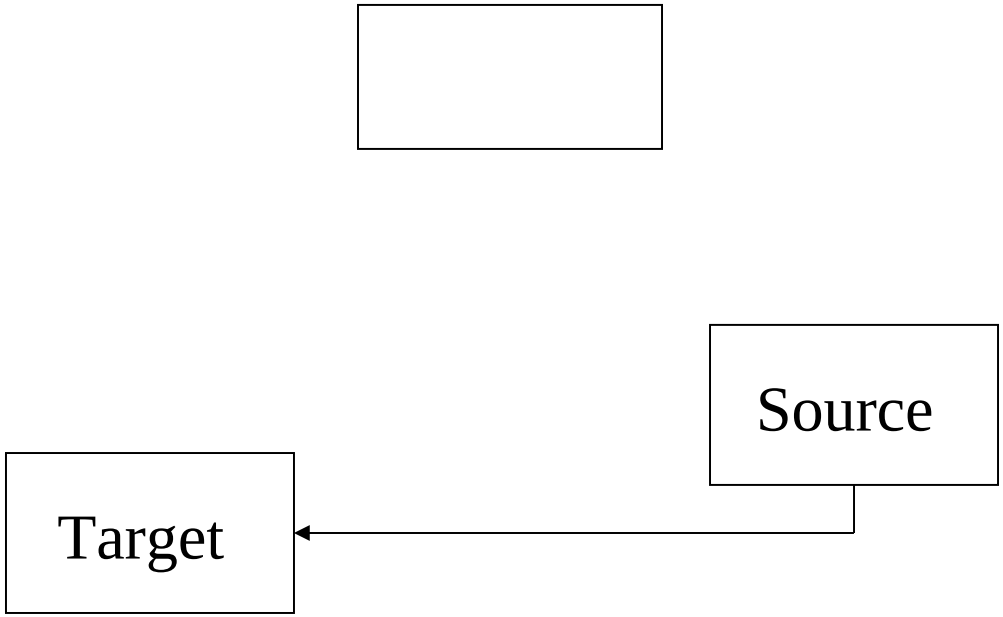
Duality #2

- The controlling cell and the cell being controlled are interchangeable
- No distinction between sources and targets
- **Probably most important feature!**

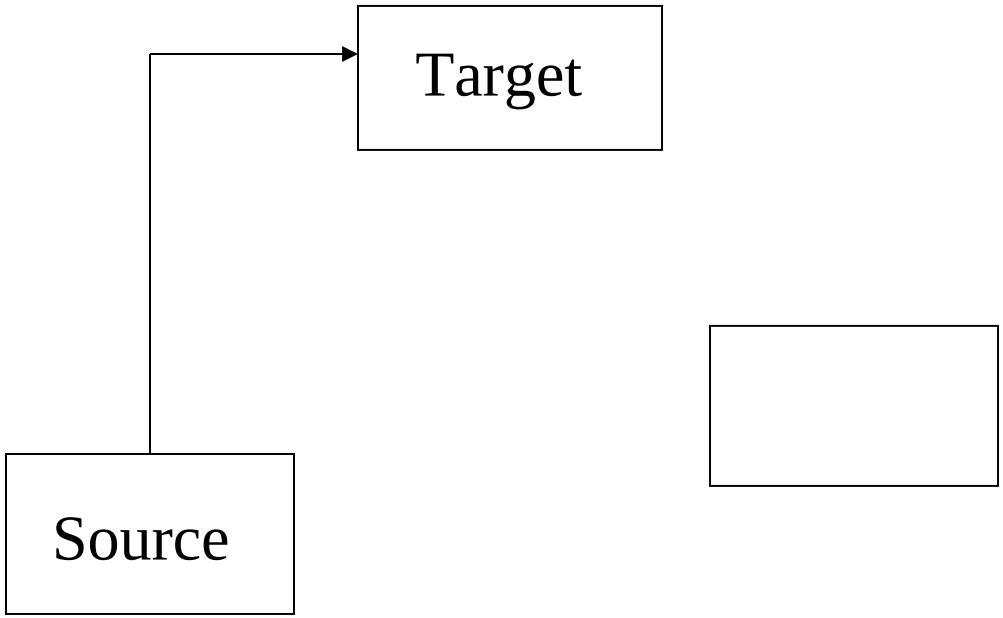
Example of Duality #2



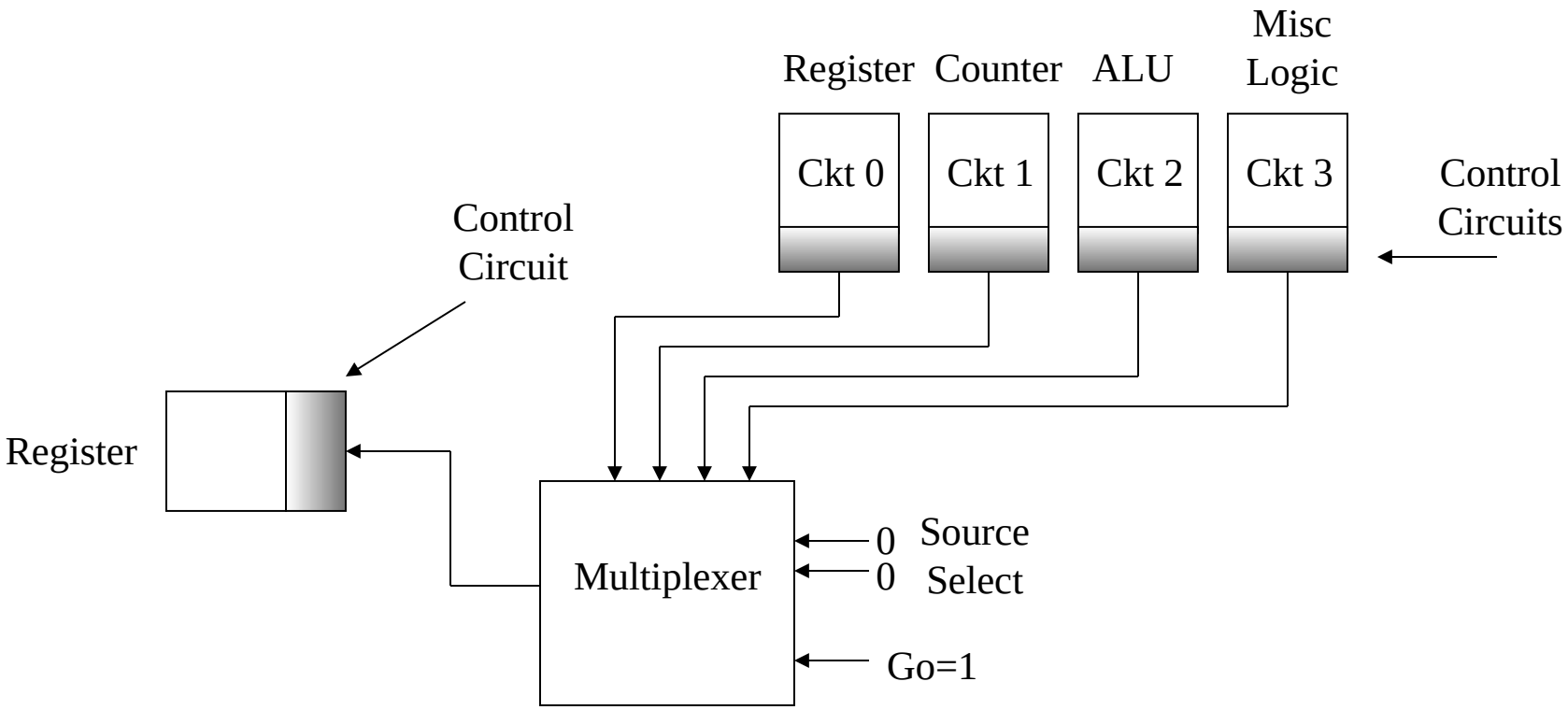
Example of Duality #2



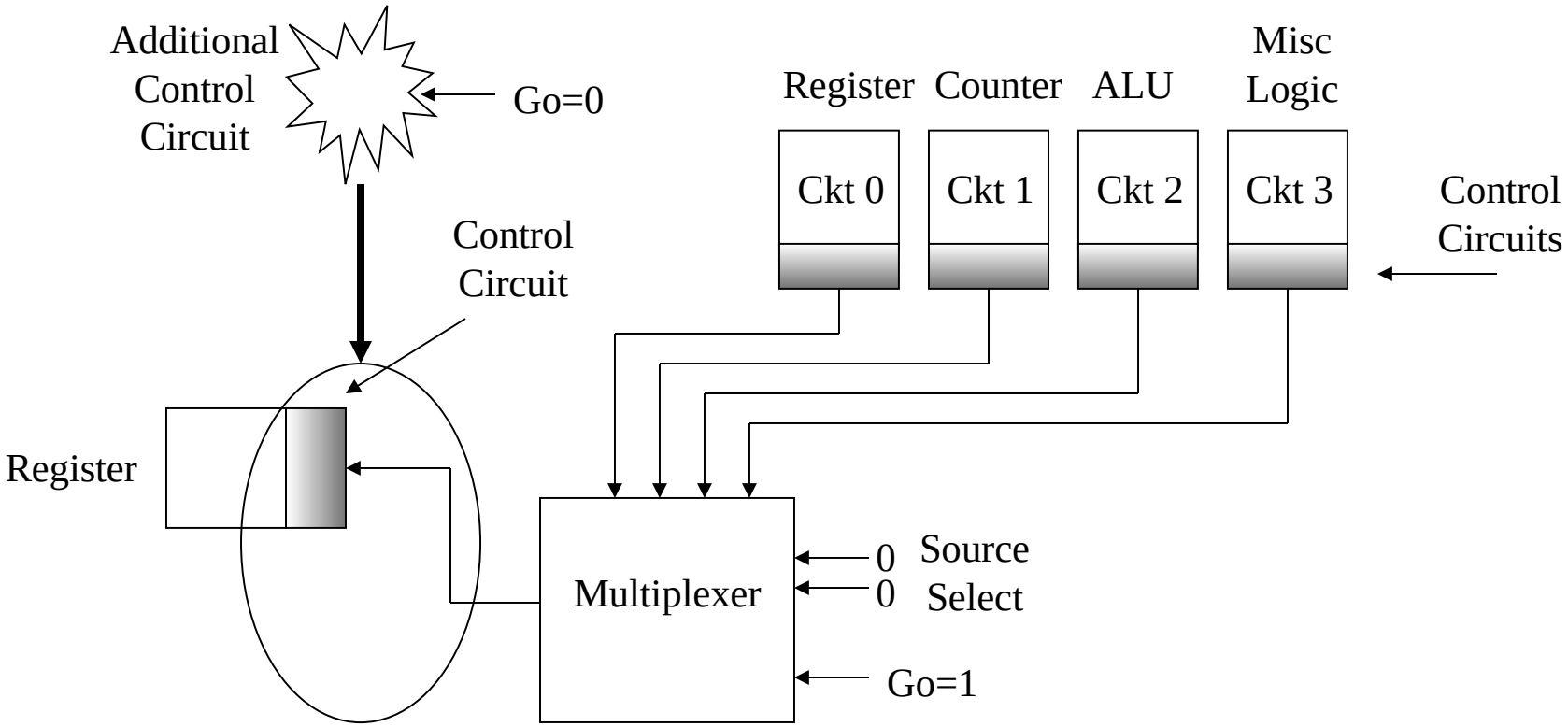
Example of Duality #2



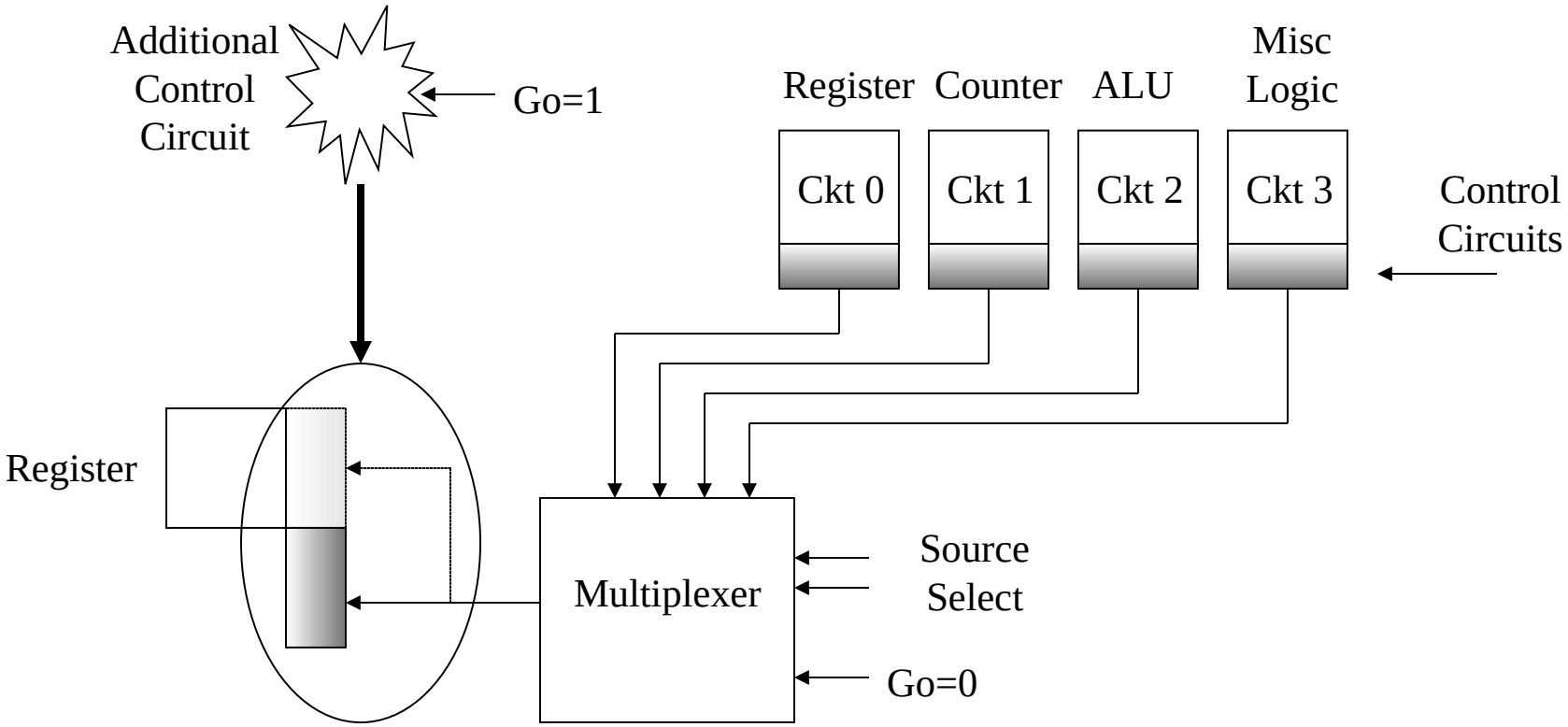
System-Level



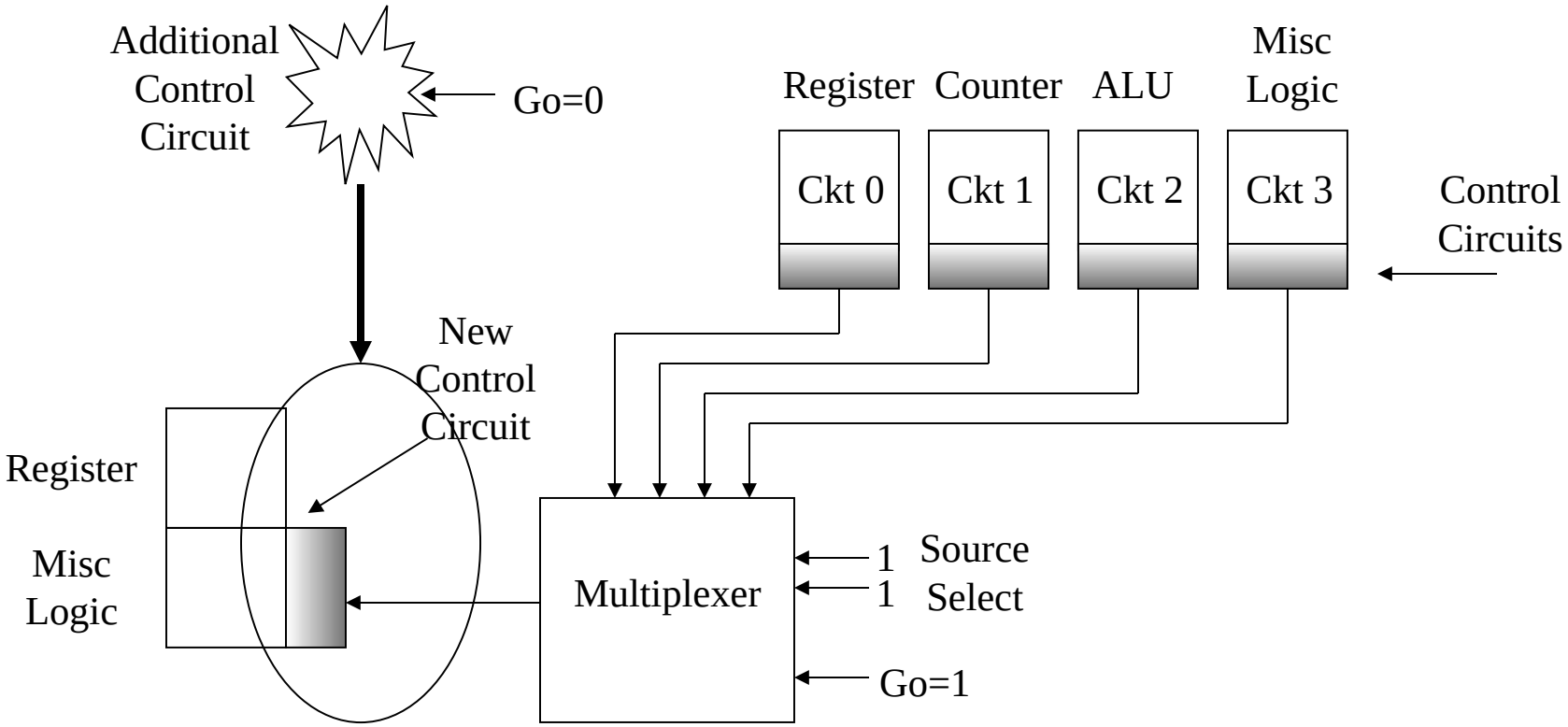
System-Level



System-Level

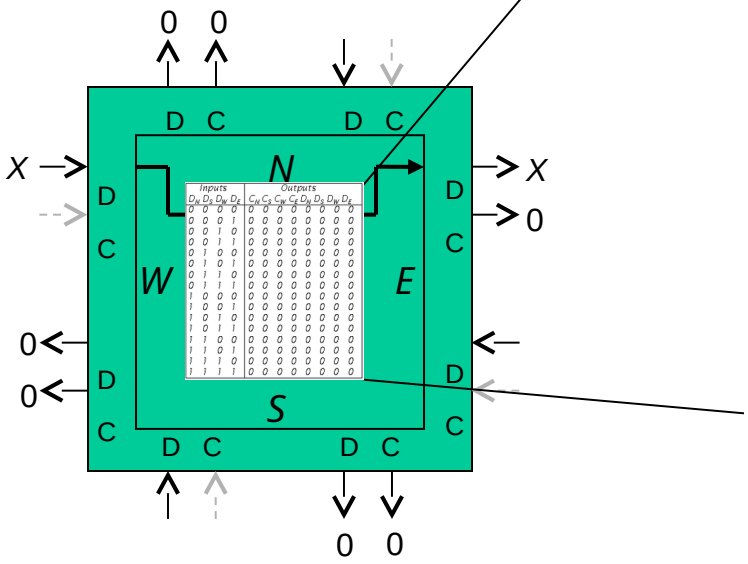


System-Level



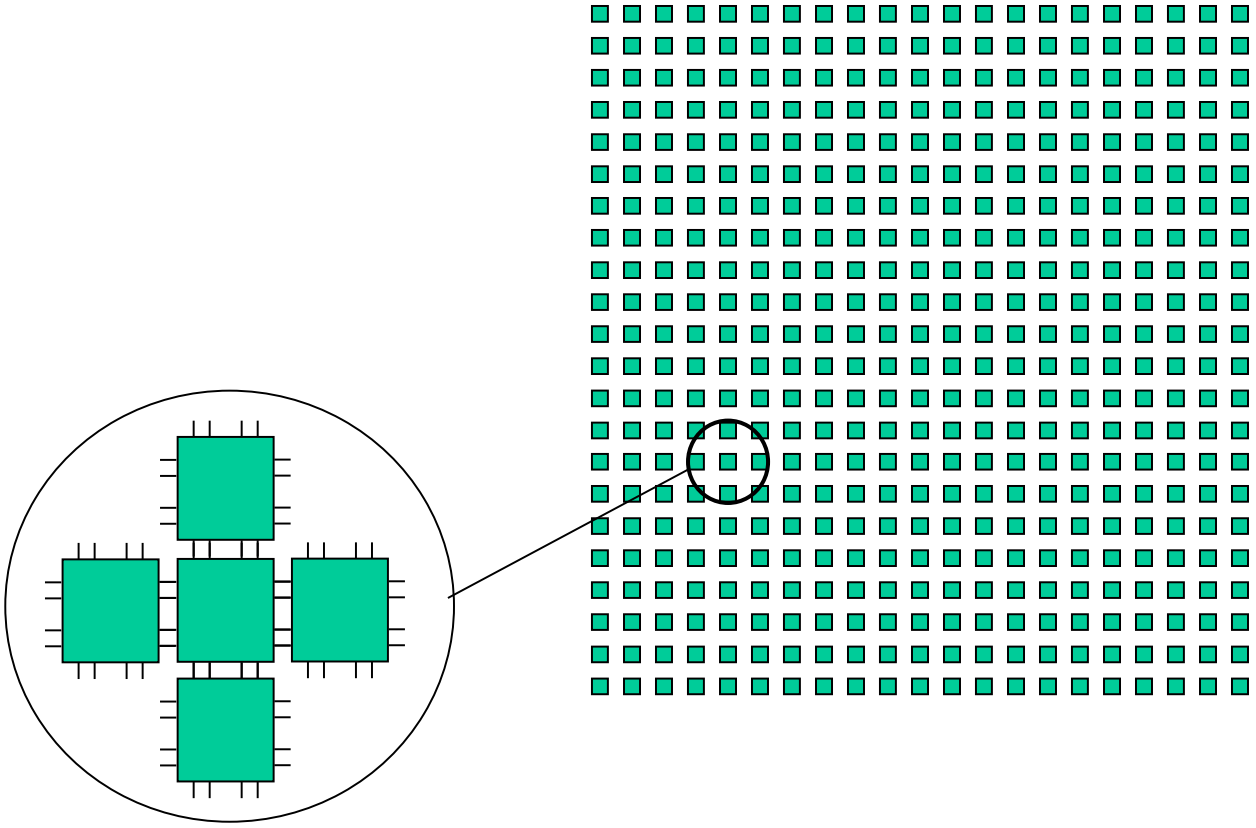
DETAILS OF CELL-LEVEL BEHAVIOR

Single Cell Behavior

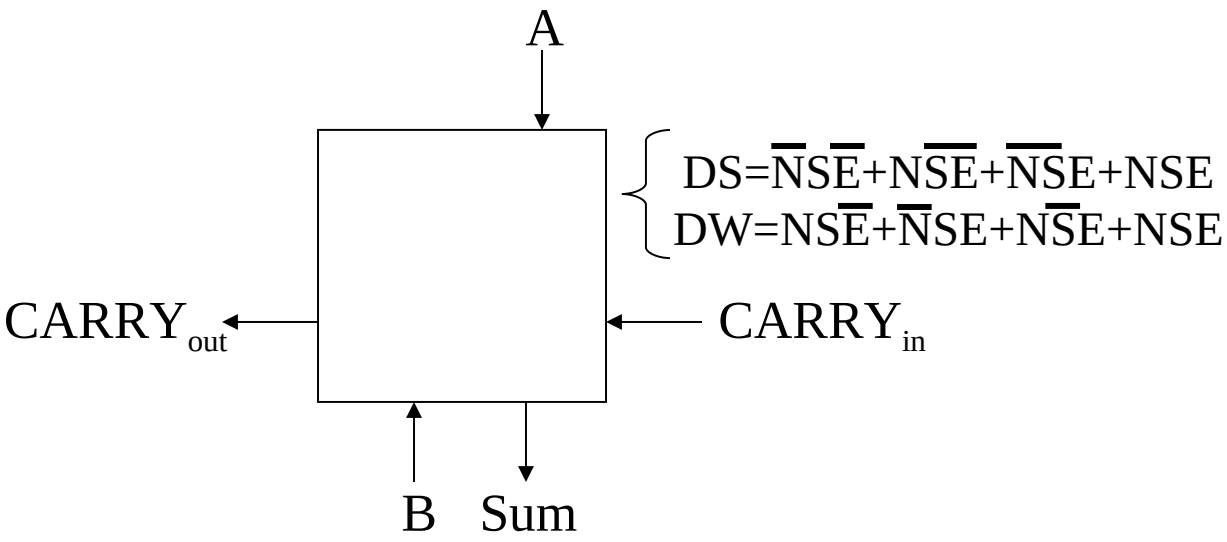


Inputs					Outputs							
D_N	D_S	D_W	D_E		C_N	C_S	C_W	C_E	D_N	D_S	D_W	D_E
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
0	0	0	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
0	0	1	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	
0	0	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	
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**	

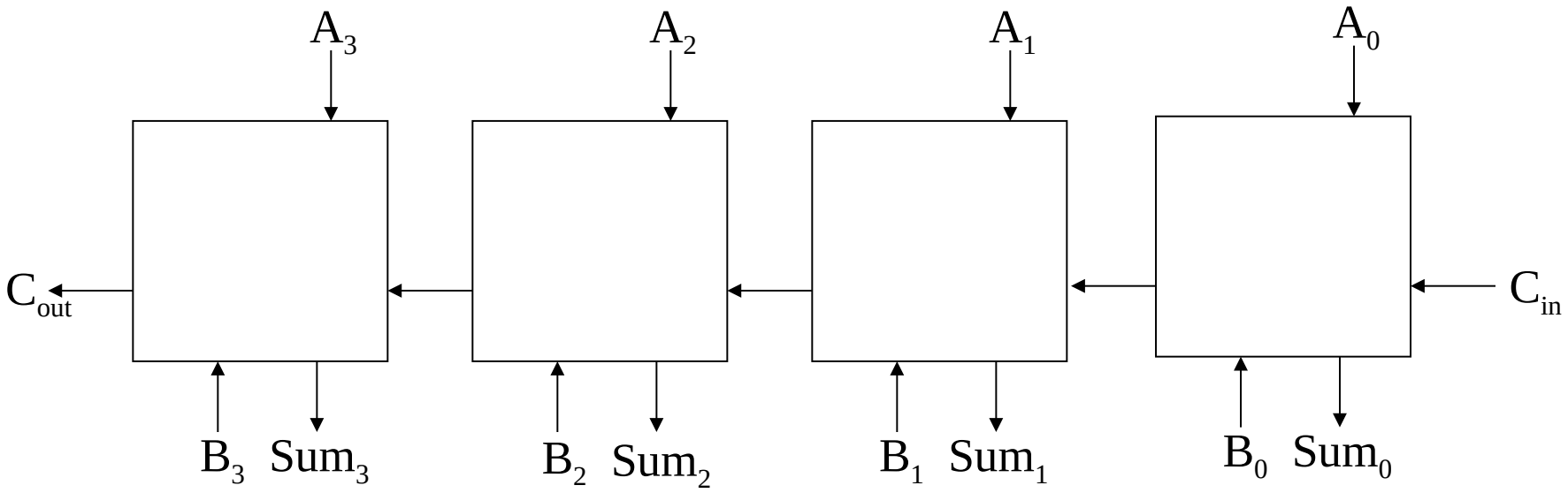
Cells are tiled in regular fashion
Nearest neighbor interconnect



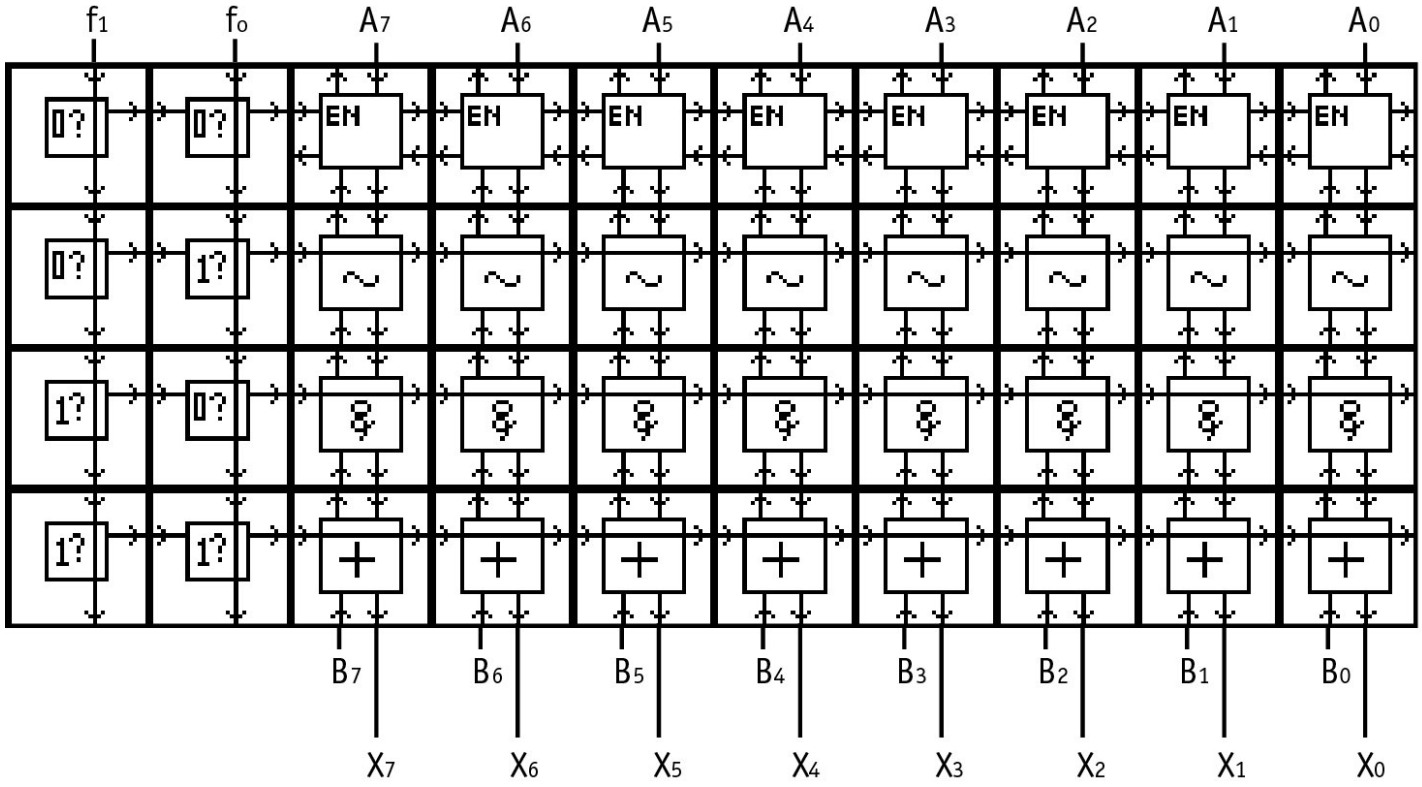
Example: One-Bit Full Adder



Example: Four-Bit Adder

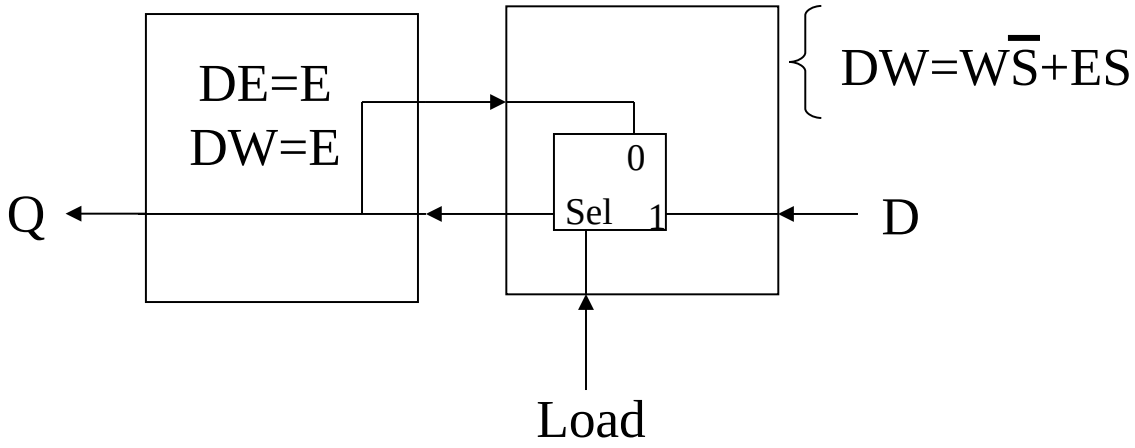


Example: ALU

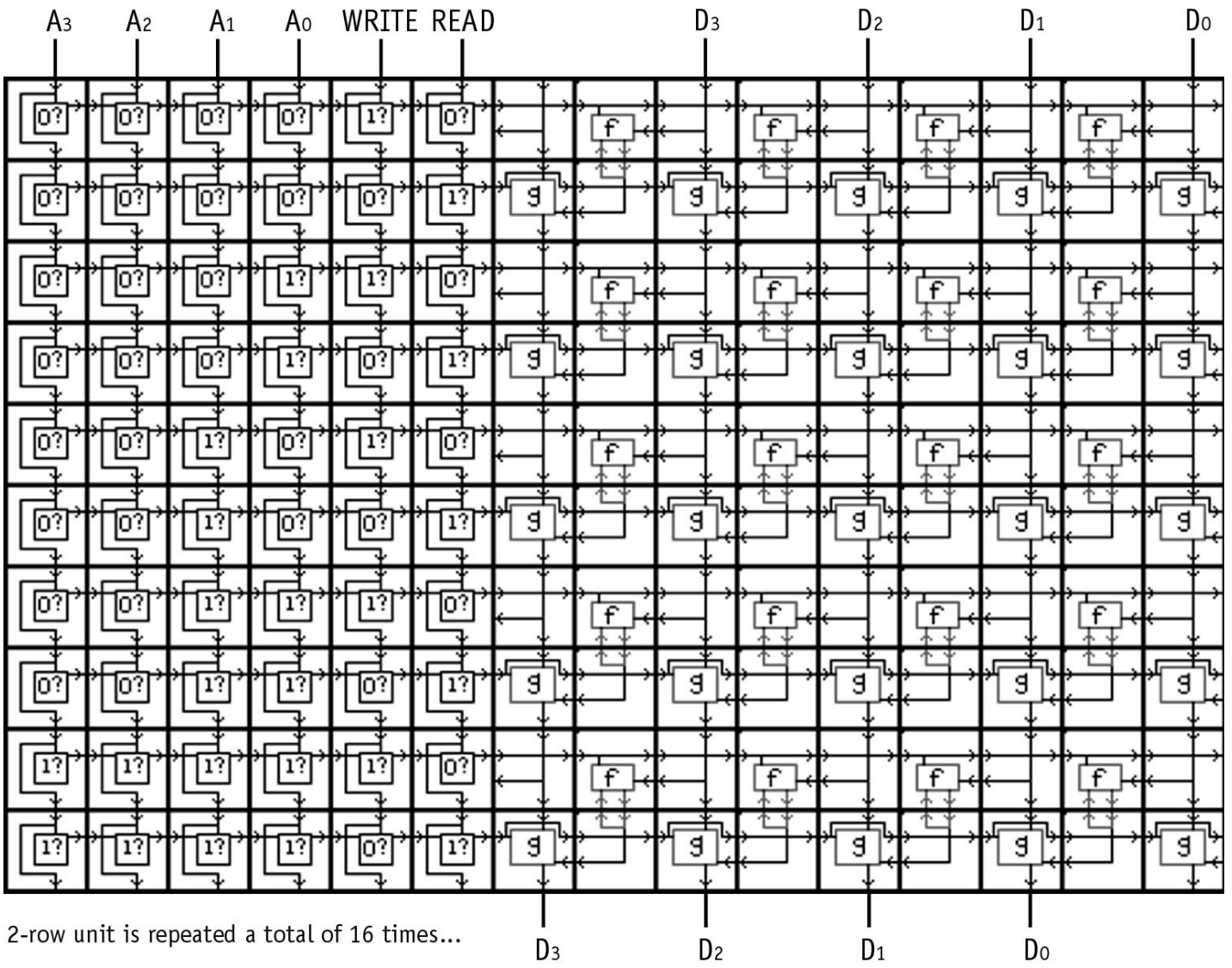


f_1	f_0	X
0	0	A + B
0	1	$\overline{A}$
1	0	A & B
1	1	A | B

Example: D-Flip Flop



Example: Memory



What About Code Processing?

- Each cell operates in one of two modes
- Above examples show cells in D mode, where they process information as Data
- A D-mode cell's configuration memory is being *executed* (treated as code)

C-Mode Processing

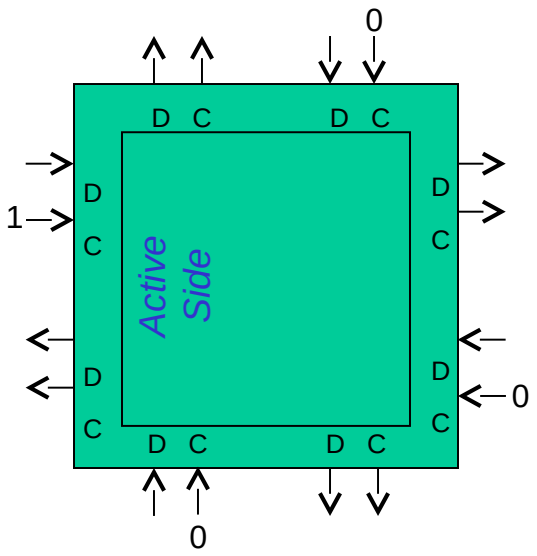
- When a cell is in C-mode, it is processing information as Code
- A C-mode cell's configuration memory is being *read and written* (treated as data)

Duality #3

- D-Mode Cell
  - Treats Information as Data
  - Treats Configuration Memory as Code
- C-Mode Cell
  - Treats Information as Code
  - Treats Configuration Memory as Data

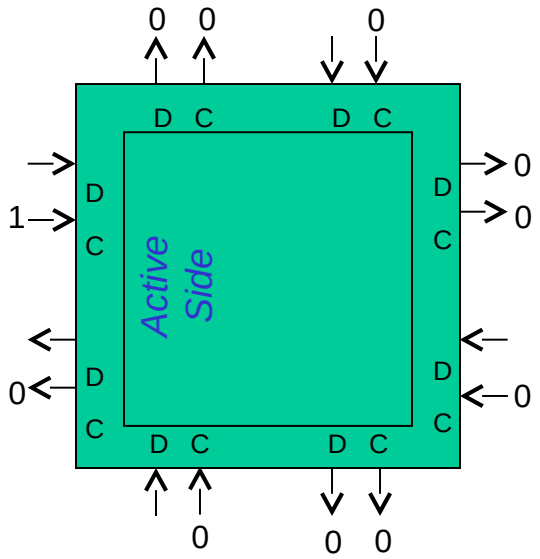
C-Mode Behavior

- If any C input=1, the cell enters C-Mode



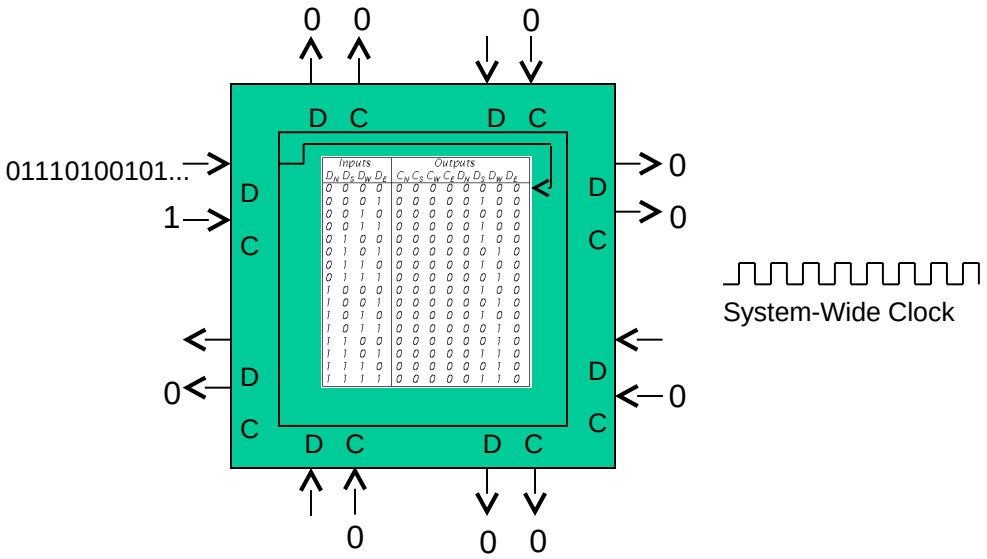
C-Mode Behavior

- All outputs are driven to 0 except on the Active Side



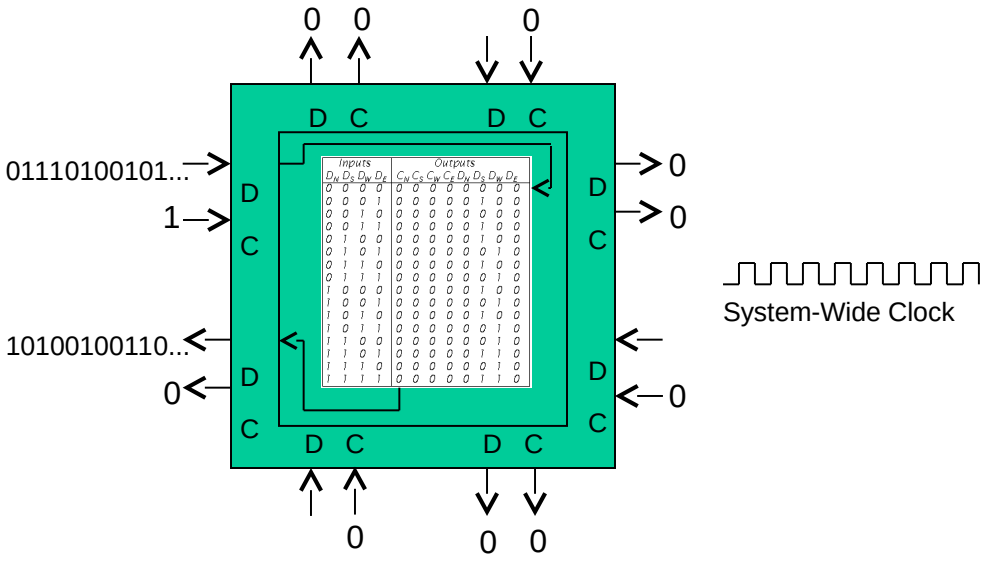
C-Mode Behavior

- D_{IN} from the active side is serially shifted into the cell's truth table



C-Mode Behavior

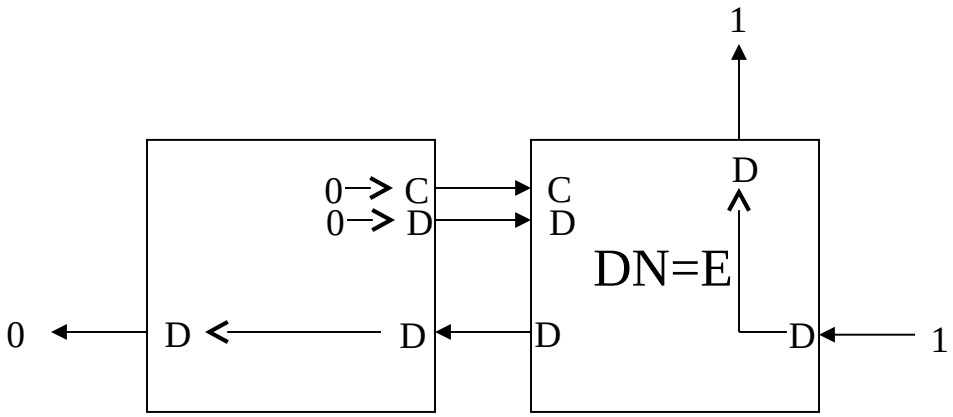
- As the truth table is shifted, it is sent serially to D_{OUT} on the active side



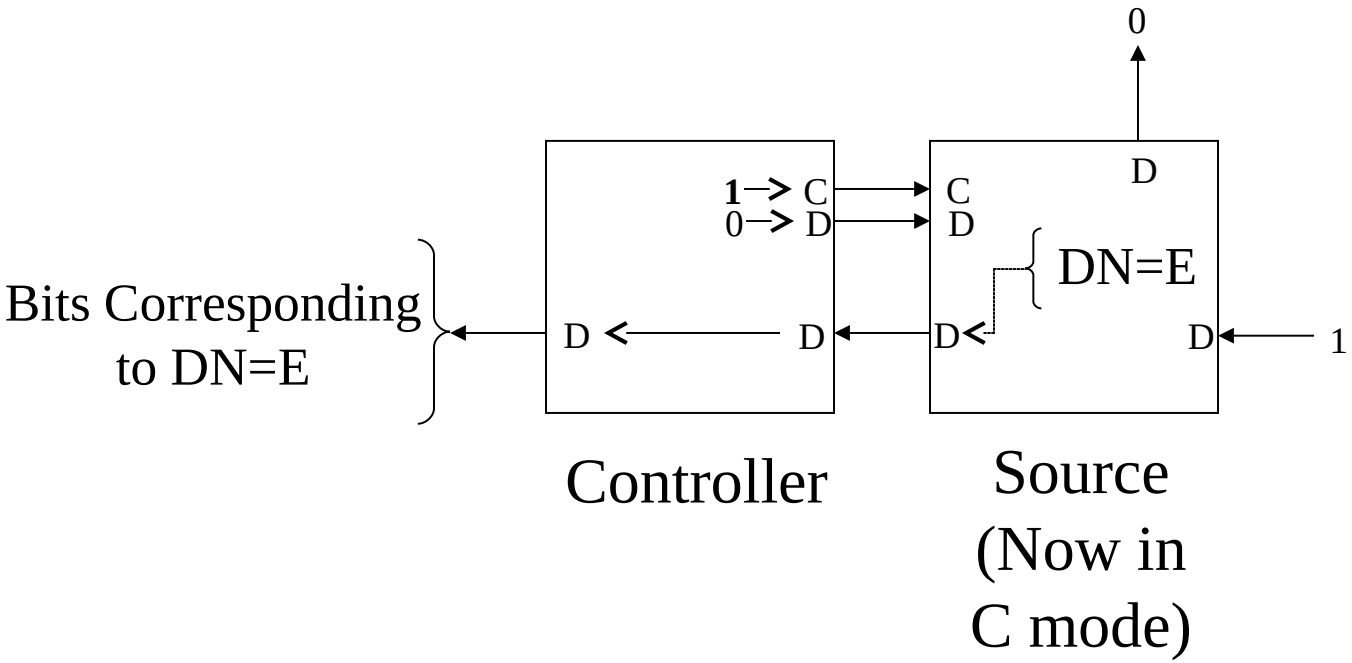
Allows a cell to read a neighbor's configuration

- Assert C output to a neighbor
- Read D output from that neighbor
- Supply new configuration via D output to that neighbor

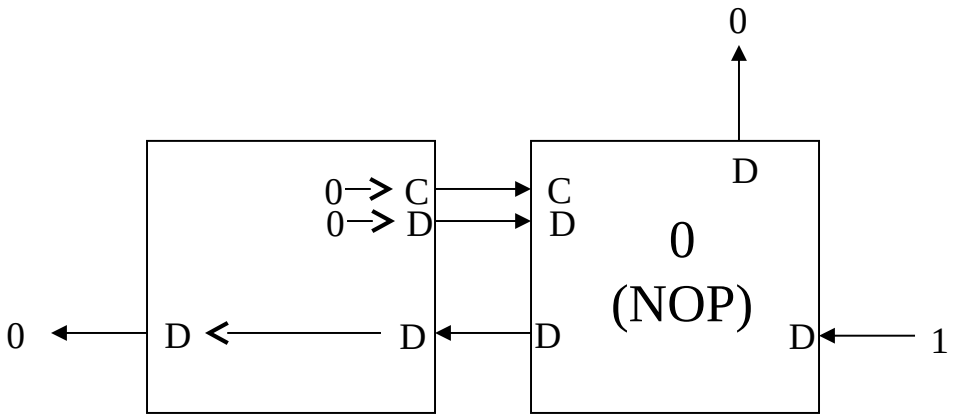
Cell Read



Cell Read


0
D
1 C
0 D
C
D
DN=E
Bits Corresponding
to DN=E
D
D
D
D
1
Controller
Source
(Now in
C mode)

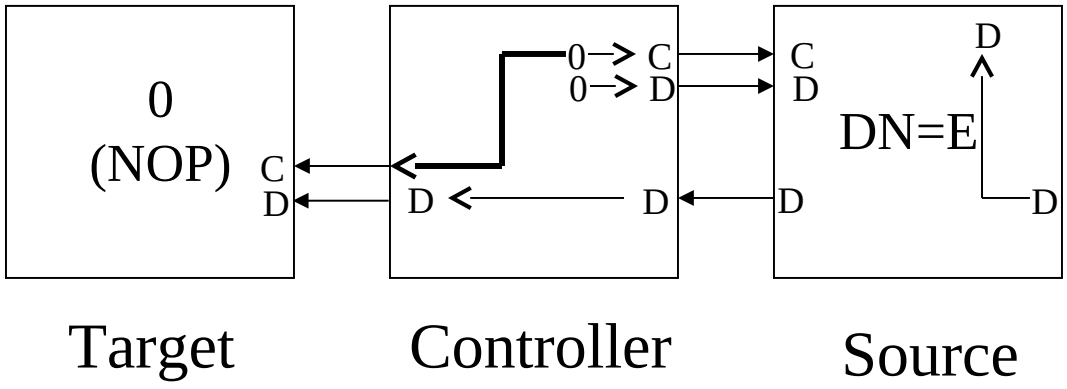
Final configuration



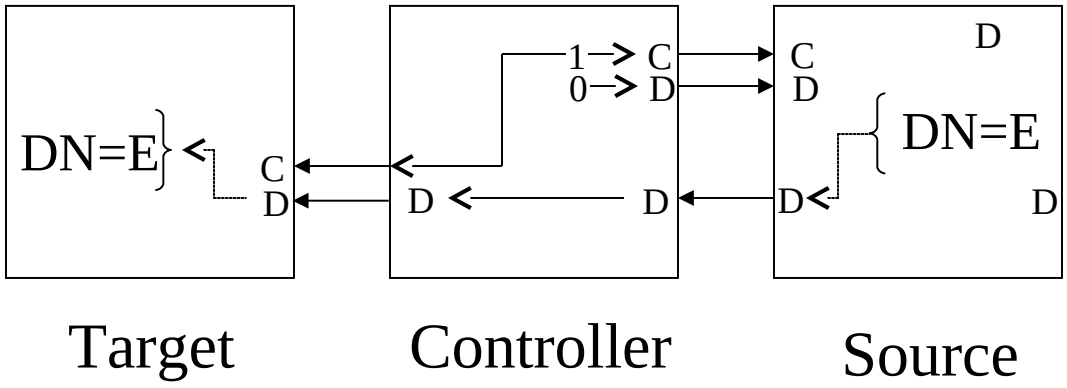
Configuration bits can be processed

Example: Cell Move

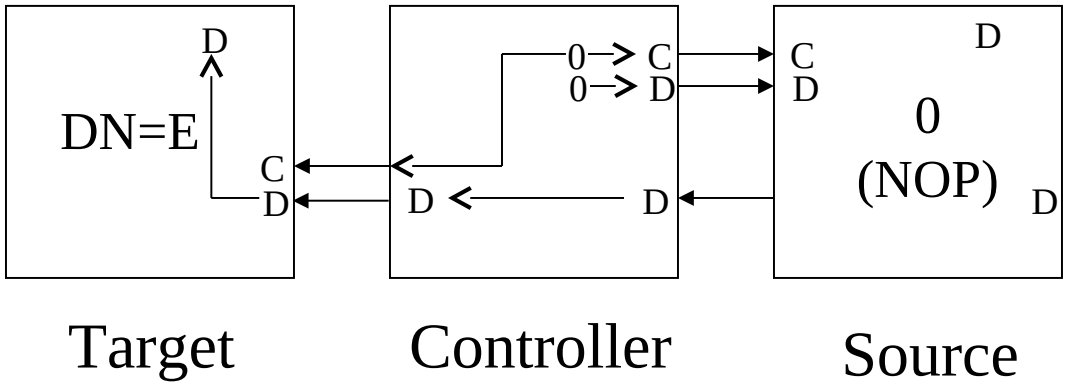
Cell Move



Source and Target are in C-Mode

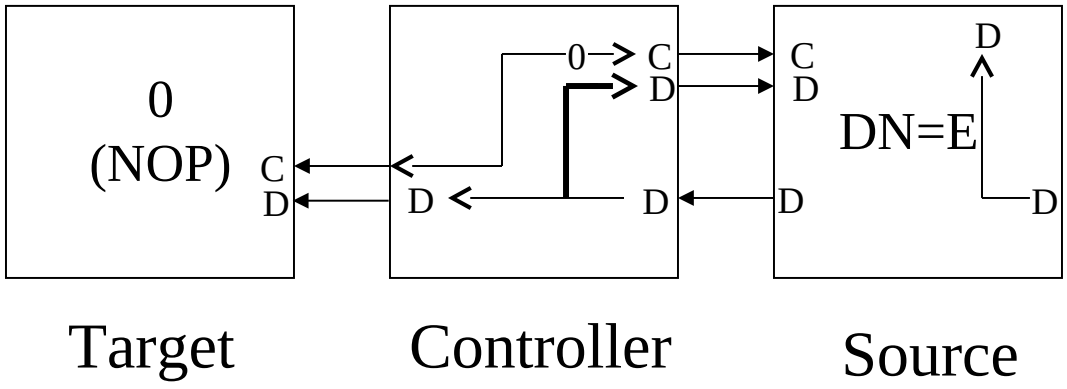


Final Configuration
Source has been moved!

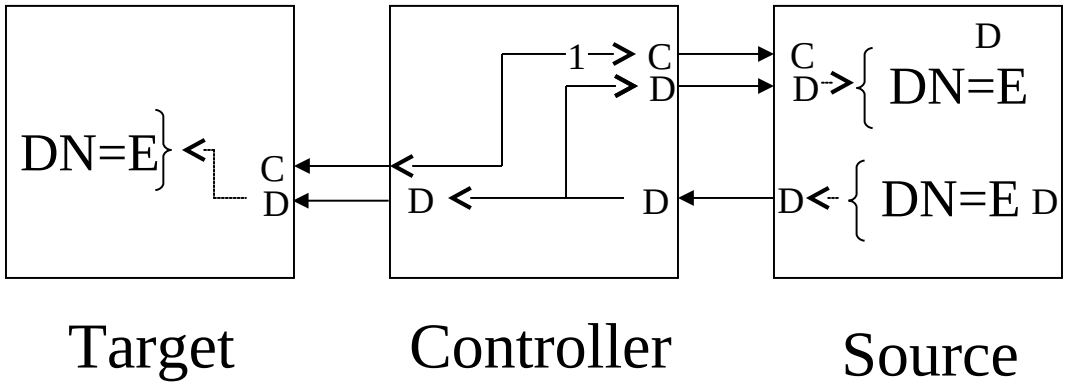


Minor modification for non-destructive read

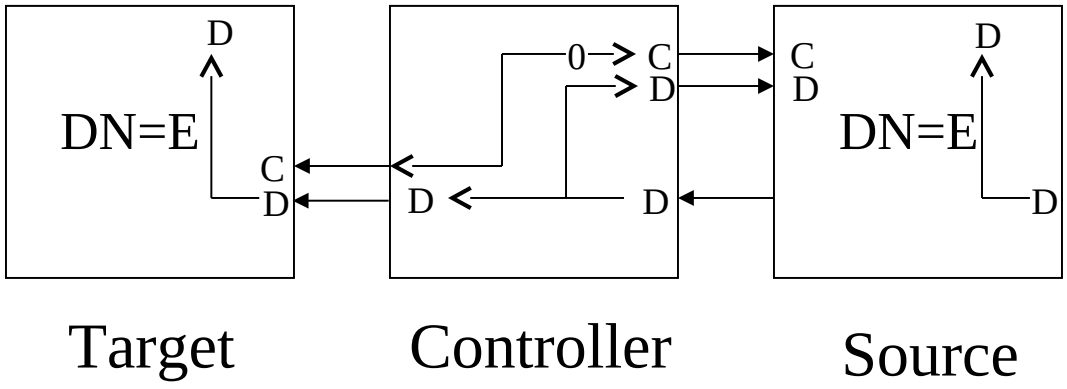
Cell Copy



Source and Target are in C-Mode



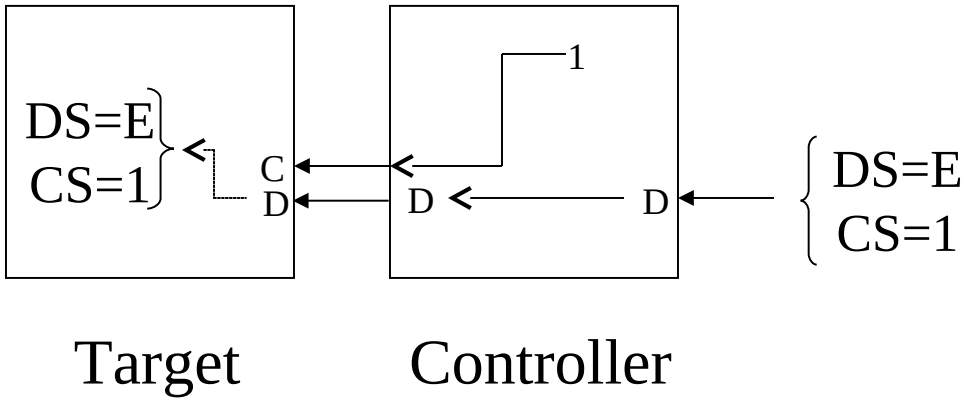
Final Configuration
Source has been copied



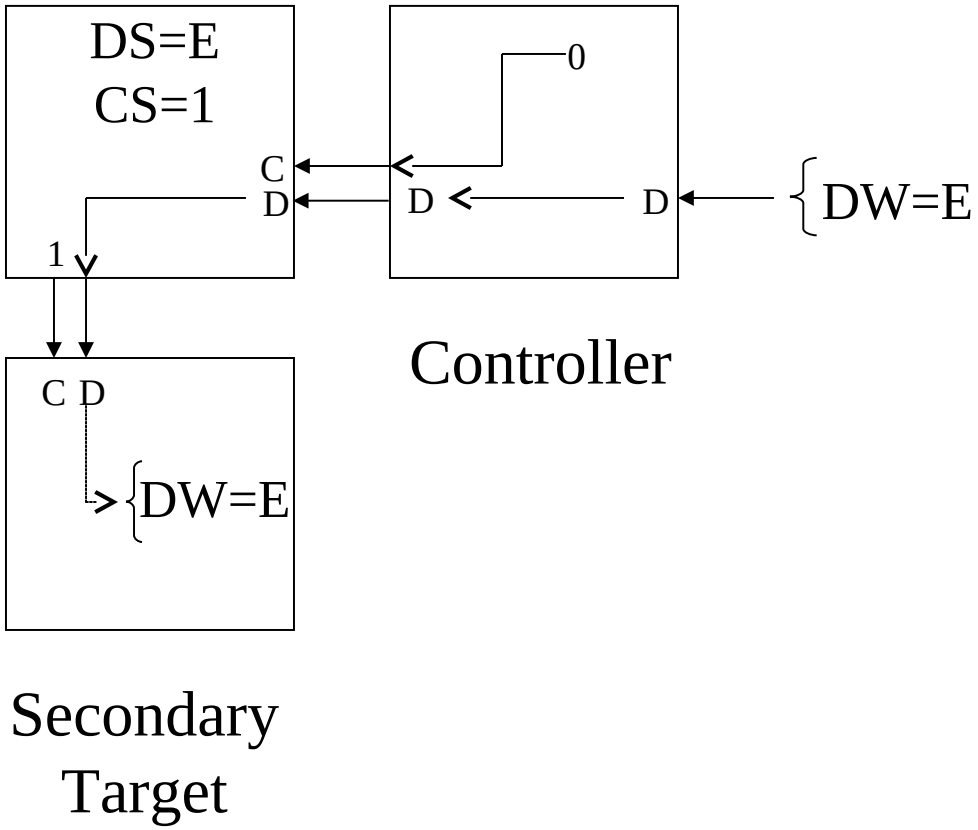
Using Duality #2 at the Cell Level

- Cells which are configured can subsequently configure other cells

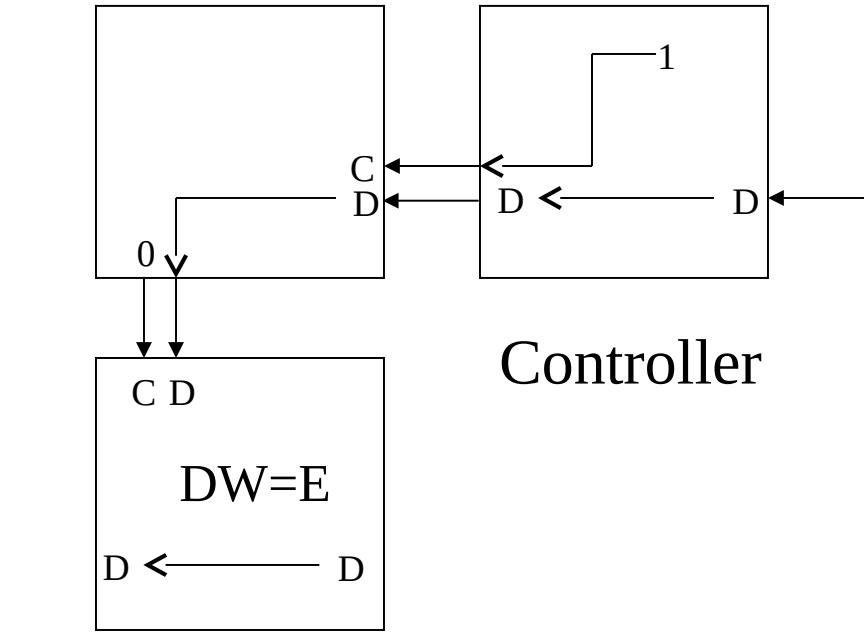
First, configure target



Next, target configures secondary target



Finally, target returns to C-mode



This allows cells to configure non-neighboring cells

- Very important capability
- Configurations generally require close cooperation among groups of cells

This allows cells to configure non-neighboring cells

- Very important capability
- Configurations generally requires close cooperation among groups of cells

- But why bother?

BENEFITS OF THE CELL MATRIX ARCHITECTURE

Benefits of the Architecture

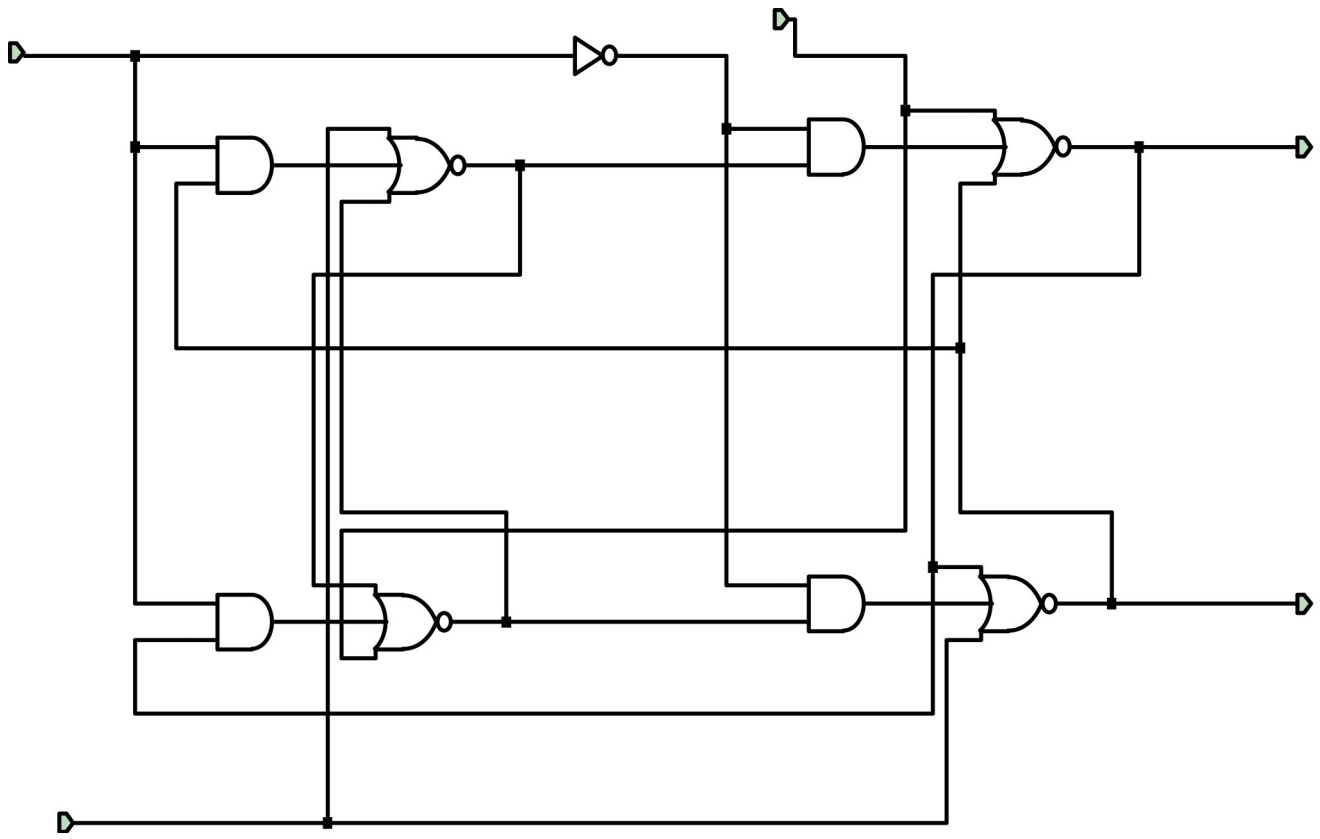
- Speed

Benefits of the Architecture

- Parallelism

Benefits of the Architecture

- Parallelism
  - Hardware is parallel



Benefits of the Architecture

- Parallelism
  - Hardware is parallel
  - Fine-grained design

Benefits of the Architecture

- Parallelism
  - Hardware is parallel
  - Fine-grained design
  - Can custom design parallel hardware to your particular problem

Benefits of the Architecture

- Parallelism
- Size

Benefits of the Architecture

- Parallelism
- Size
  - High cell count

Benefits of the Architecture

- Parallelism
- Size
  - High cell count → Lots of parallelism

Benefits of the Architecture

- Parallelism
- Size
  - High cell count → Lots of parallelism
  - Still need to be smart about how you use HW

Benefits of the Architecture

- Parallelism
- Size
  - High cell count → Lots of parallelism
  - Still need to be smart about how you use HW
  - What is a high cell count?

Some Big Numbers

- # of gates in large FPGA:

Some Big Numbers

- # of gates in large FPGA:1 Million

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:50 Million

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:50 Million
- # of gates in the world:

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:50 Million
- # of gates in the world:< 10^{20}

(J. Tour, MNT 2000)

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:50 Million
- # of gates in the world:< 10^{20}
- # of carbon atoms in a pencil:

Some Big Numbers

- # of gates in large FPGA:1 Million
- # of gates in large ASIC:50 Million
- # of gates in the world:< 10^{20}
- # of carbon atoms in a pencil:>10^{23}

Avogadro Machine:
On order of 10^{23} switches

Order 10^{21} Cells

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support
  - Scalable

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support
  - Scalable
  - Fault Tolerant
    - Only local interactions
    - Very few shared signals
    - Homogeneous hardware

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support
  - Scalable
  - Fault Tolerant
  - Also can be 3-D architecture

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support
  - Scalable
  - Fault Tolerant
  - Also can be 3-D architecture
  - Good features for nanotechnology

Benefits of the Architecture

- Parallelism
- Size
- Manufacturing Support

- Adds up to Massively Parallel Data Processing

QUESTION

How do you control a billion trillion cells?

IDEA

- Cell Matrix supports massively parallel data processing
- Code and Data are interchangeable
- Can we do massively parallel *code* processing?

YES!

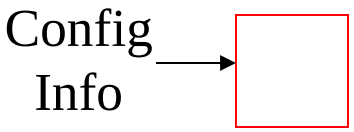
APPLICATIONS OF THE CELL MATRIX ARCHITECTURE

Example: Massively Parallel Configuration
A Space Filling Circuit

- Array of *space filling blocks* (multi-cell)
- Block will pass config info to bottom and right
- If adjacent to another block, pass as data
- Otherwise configure cells

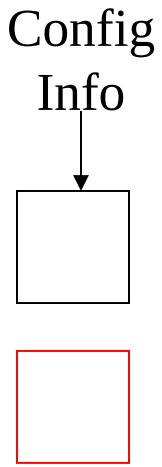
A Space Filling Circuit

- Configure first space filling block



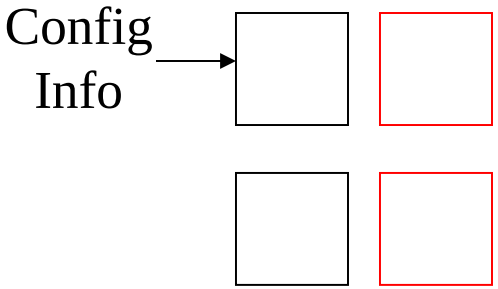
A Space Filling Circuit

- Use that block to configure a second block



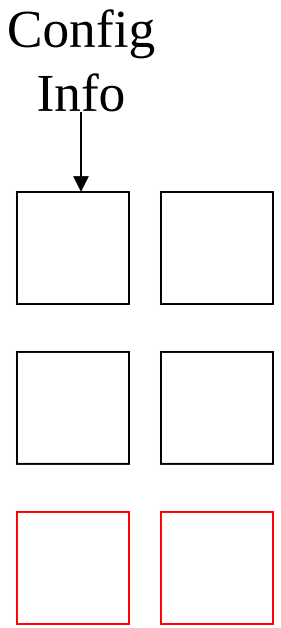
A Space Filling Circuit

- Use those blocks to configure two more



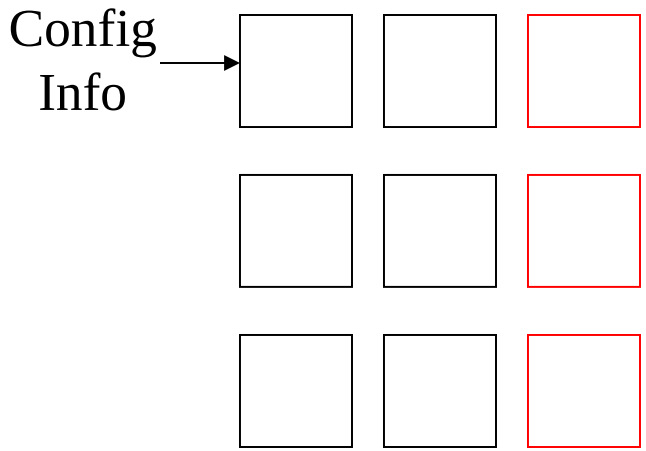
A Space Filling Circuit

- Use those blocks to configure two more



A Space Filling Circuit

- Use those blocks to configure three more



Example: Massively Parallel Configuration

Space Filling Circuit

Growth Rate

- Step 0: 1 block
- Step 1: 2 blocks
- Step 2: 4 blocks
- Step 3: 6 blocks
- Step 4: 9 blocks
- Step (even) n:$((n+2)/2)^2$ blocks

Even faster on 3-D system

- After n steps

$$\#\ \text{blocks} = ((n+3)/3)^3$$

How fast is this?

- Want to configure 10^{19} blocks

How fast is this?

- Want to configure 10^{19} blocks
- Assume 1 μsec per block

How fast is this?

- Want to configure 10^{19} blocks
- Assume 1 μsec per block
- Sequential FPGA requires 10^{19} μsec

How fast is this?

- Want to configure 10^{19} blocks
- Assume 1 μsec per block
- Sequential FPGA requires 10^{19} μsec

316880 Years

How fast is this?

- Want to configure 10^{19} blocks
- Assume 1 μsec per block
- Sequential FPGA requires 10^{19} μsec

316880 Years

(and 10.57 months)

How about on a 3-D cell matrix?

- $((n+3)/3)^3 = 10^{19}$

How about on a 3-D cell matrix?

- $((n+3)/3)^3 = 10^{19}$
- n=6.46 million steps

How about on a 3-D cell matrix?

- $((n+3)/3)^3 = 10^{19}$
- n=6.46 million steps
- each step takes 1 μsec

How about on a 3-D cell matrix?

- $((n+3)/3)^3 = 10^{19}$
- n=6.46 million steps
- each step takes 1 μsec
- Total time=6.46 sec!

This is the difference between an FPGA and a cell matrix

FPGA	Cell Matrix
316880 years	6.46 seconds

Space filling circuit is just one example

- Other parallel configuration techniques are possible
- Differentiation also possible

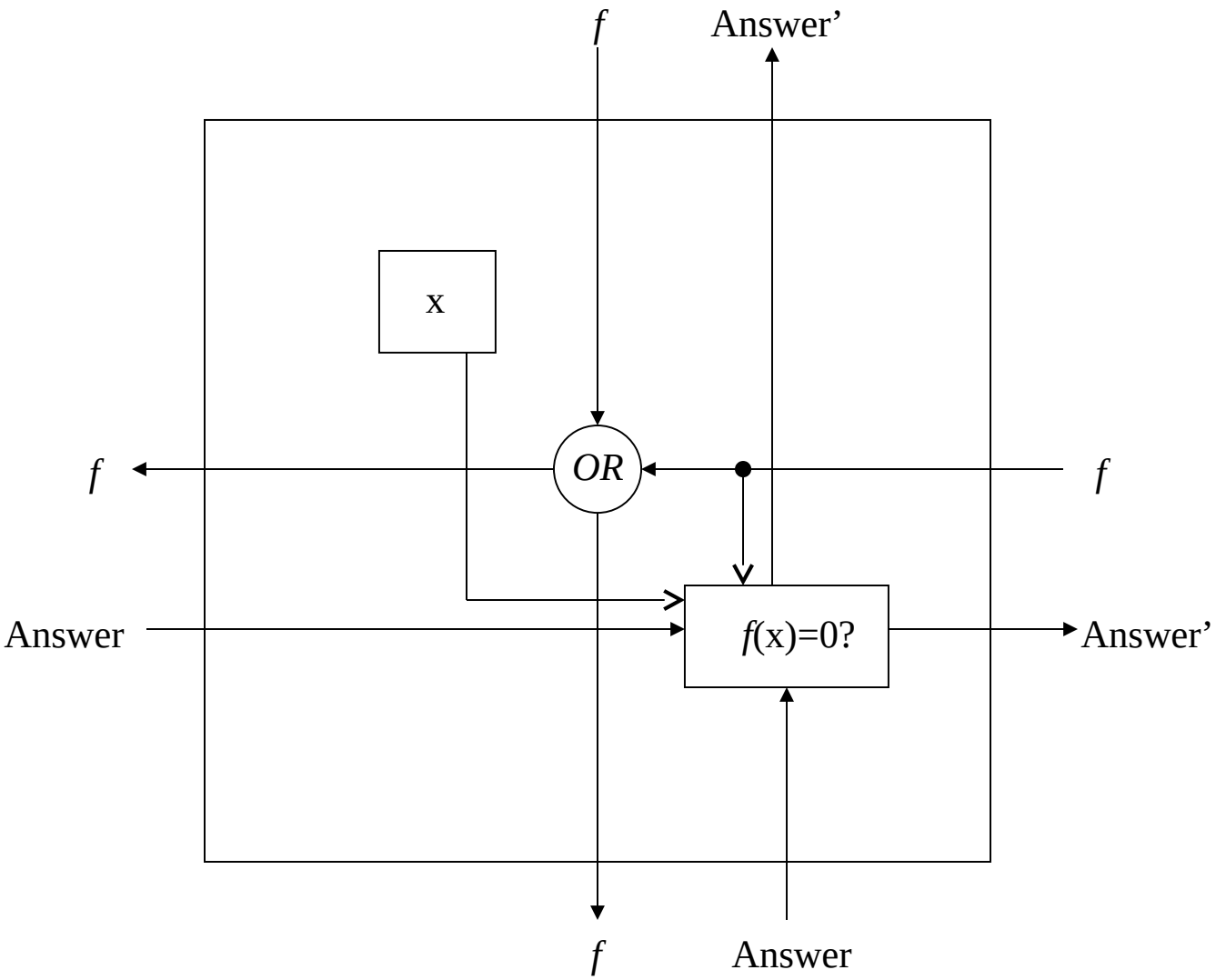
What else can we build?

- Space filling circuit is just a carrier
- Piggyback useful circuits on top of it

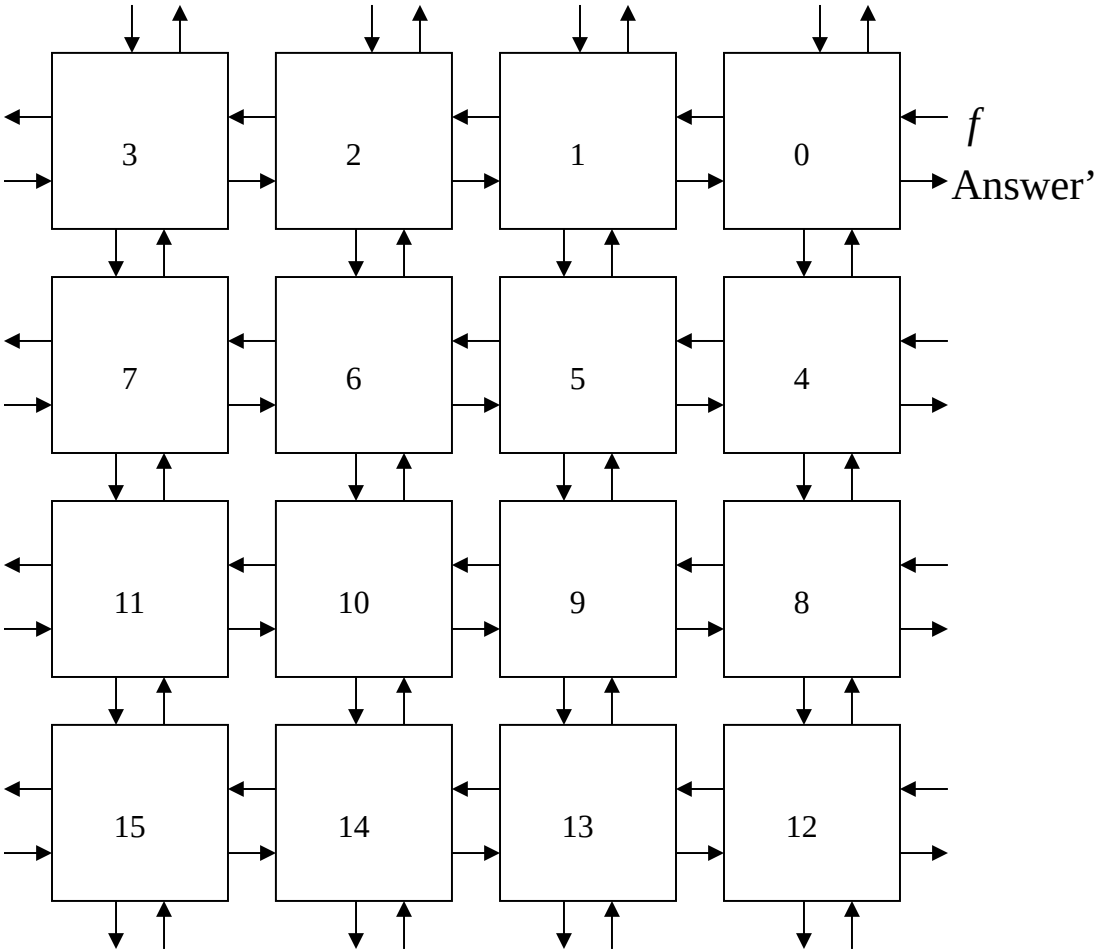
Example: Large Search Problem

- Want to find value which solves some problem
- Build a basic search block which tests one particular value
- Tile these in an array, each block testing its own value
- Propagate inputs and answer back through array

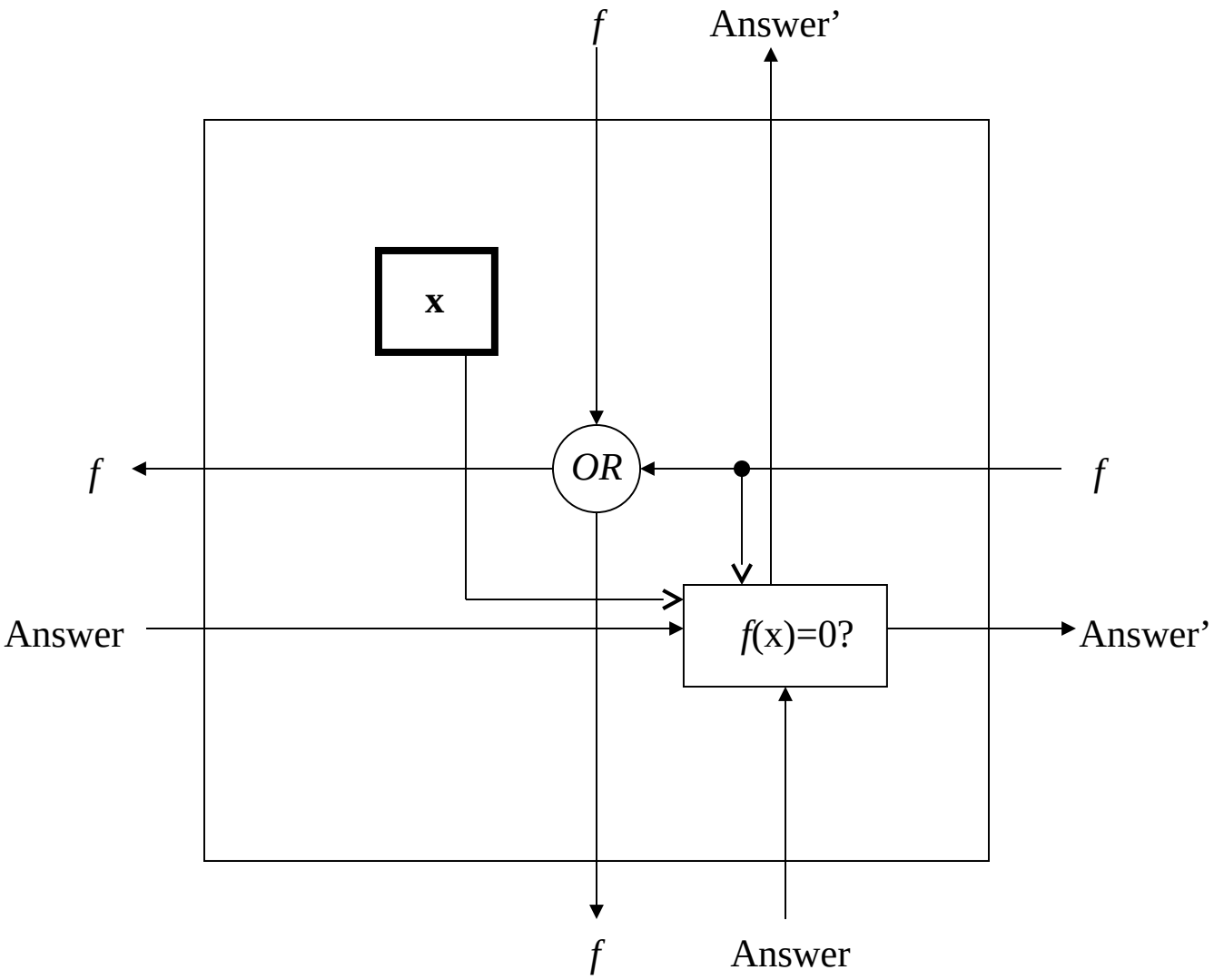
Basic Search Block
Solve *f*(x)=0



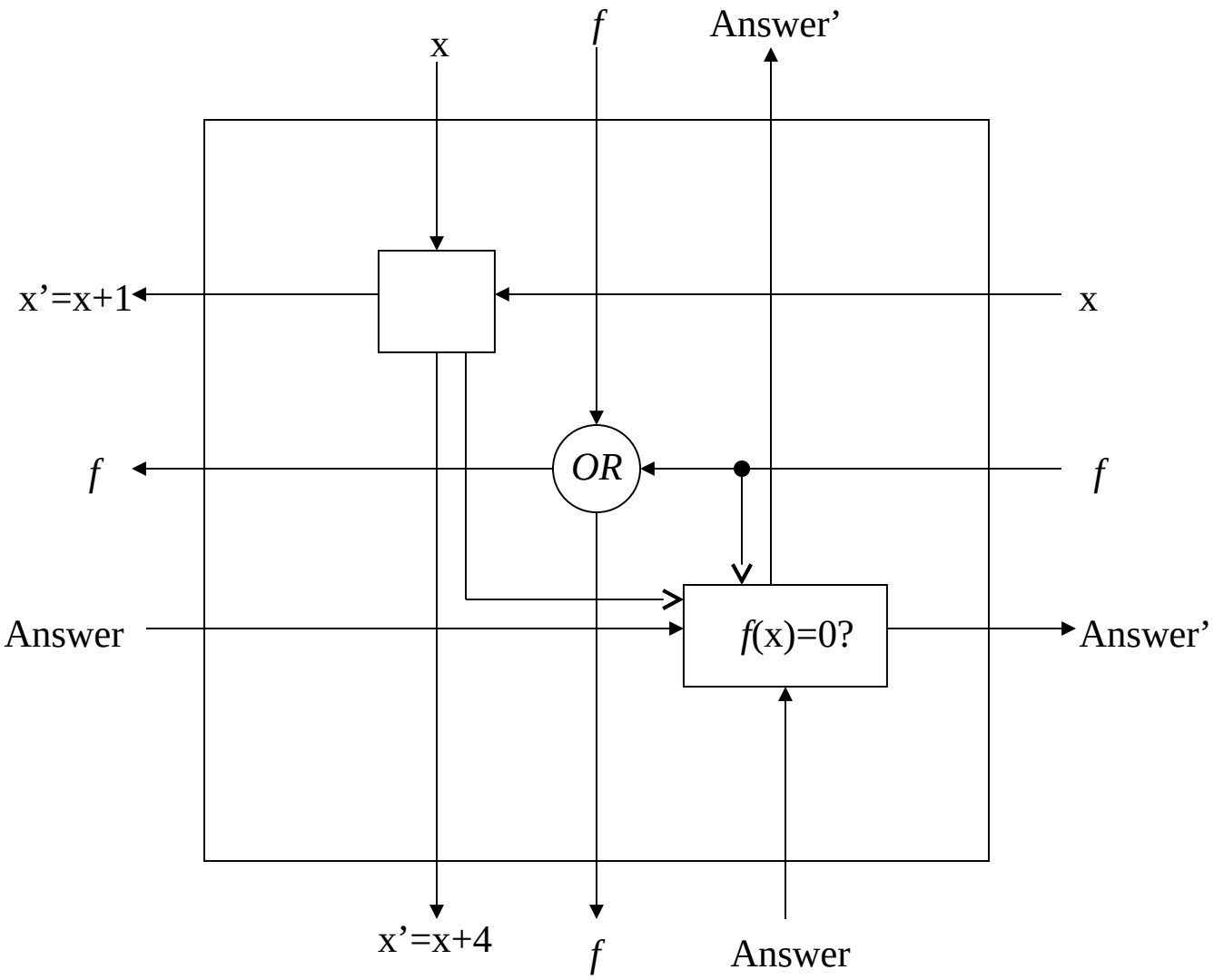
Array of blocks



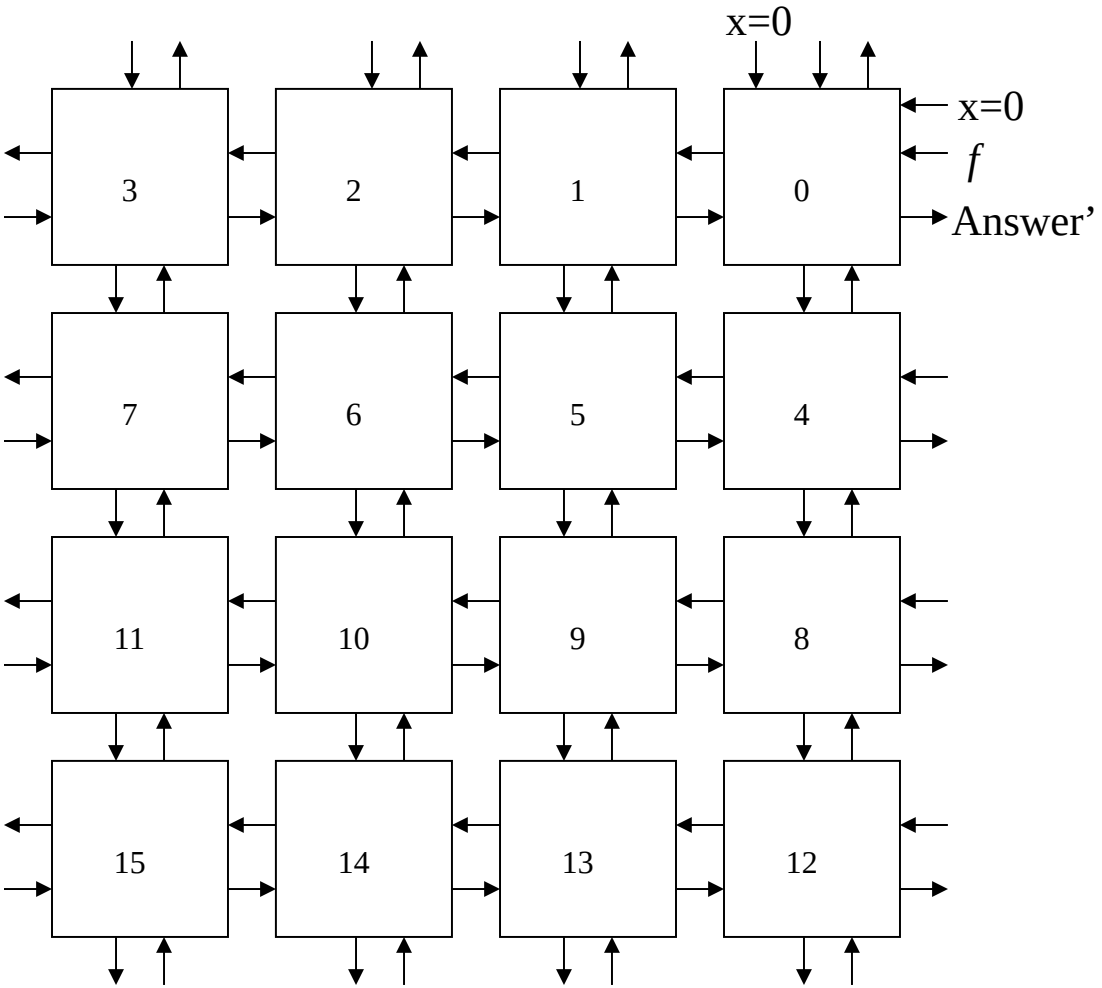
Would like uniform block



Now all blocks are identical



Array of identical blocks



Allows entire search space to be tested in a single iteration

- Well suited to problems that have:
  - Extremely large search space
  - Chaotic behavior
  - Complex algorithm

Good example

56-bit DES Cracking

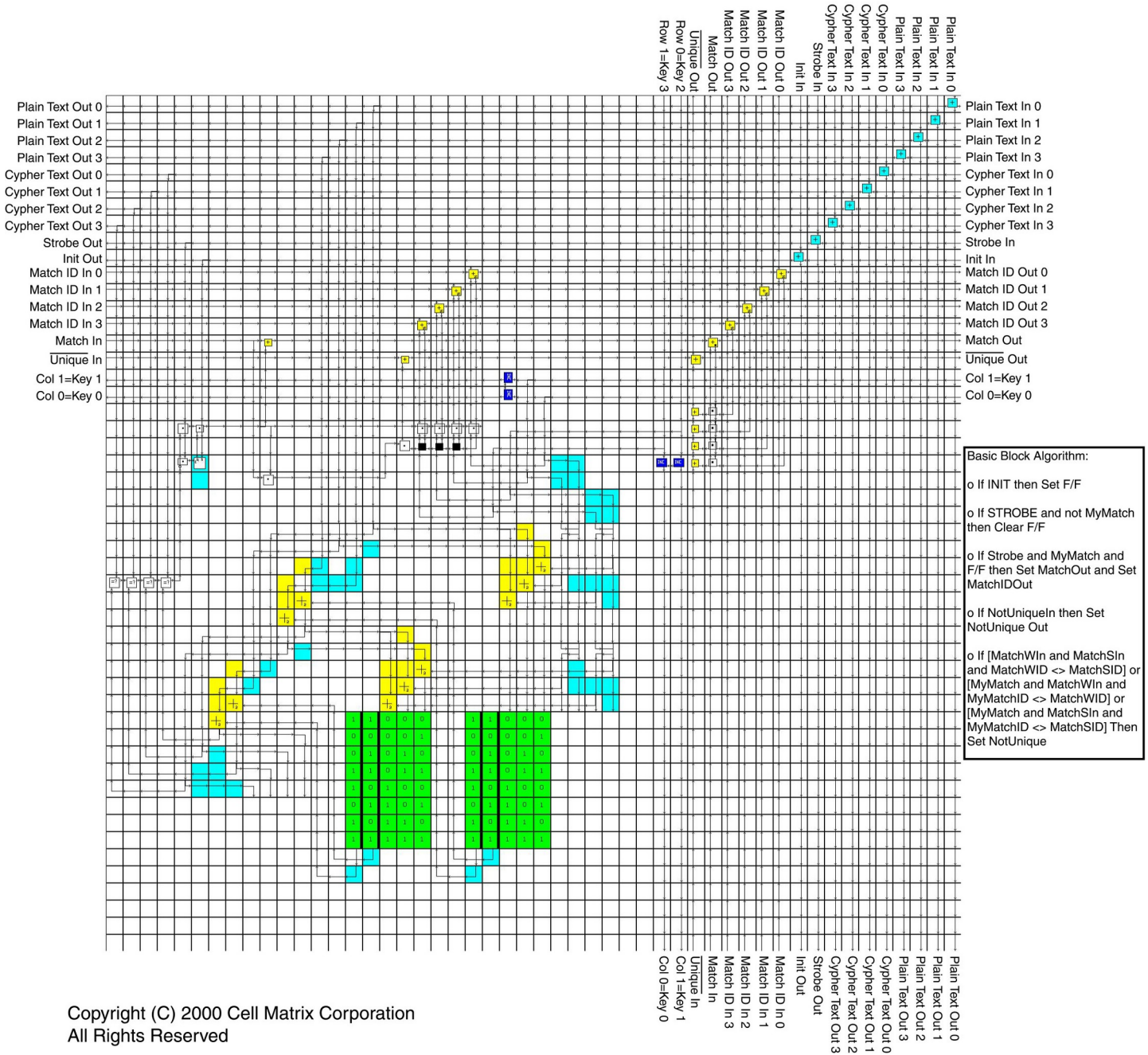
DES Cracking

- Large search problem
- Build 2^{56} search blocks
- Distribute known plain/cipher text
- Each block tests one unique key
- Correct key propagates out of array

DES Cracking

- Can easily fit all 2^{56} blocks on an Avogadro machine
- Can *configure* 2^{56} blocks in a few seconds
- Can **crack** a text pair in sub-millisecond time
- Fast enough for 256K (DSL)

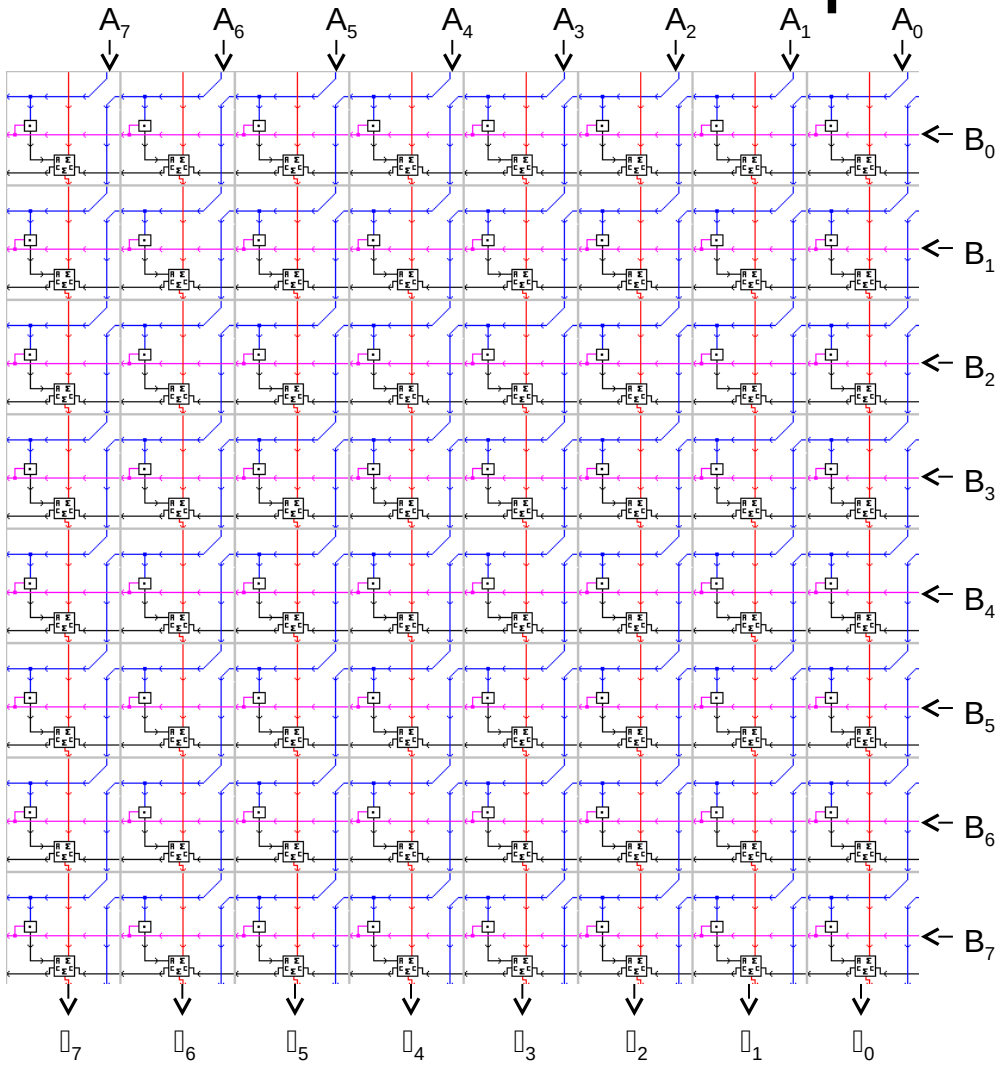
4-Bit DES Key Finding Block



Can re-think old problems to find new algorithms

- Such as wide-bit arithmetic using wide-bit integer processors

Combinatorial Multiplier

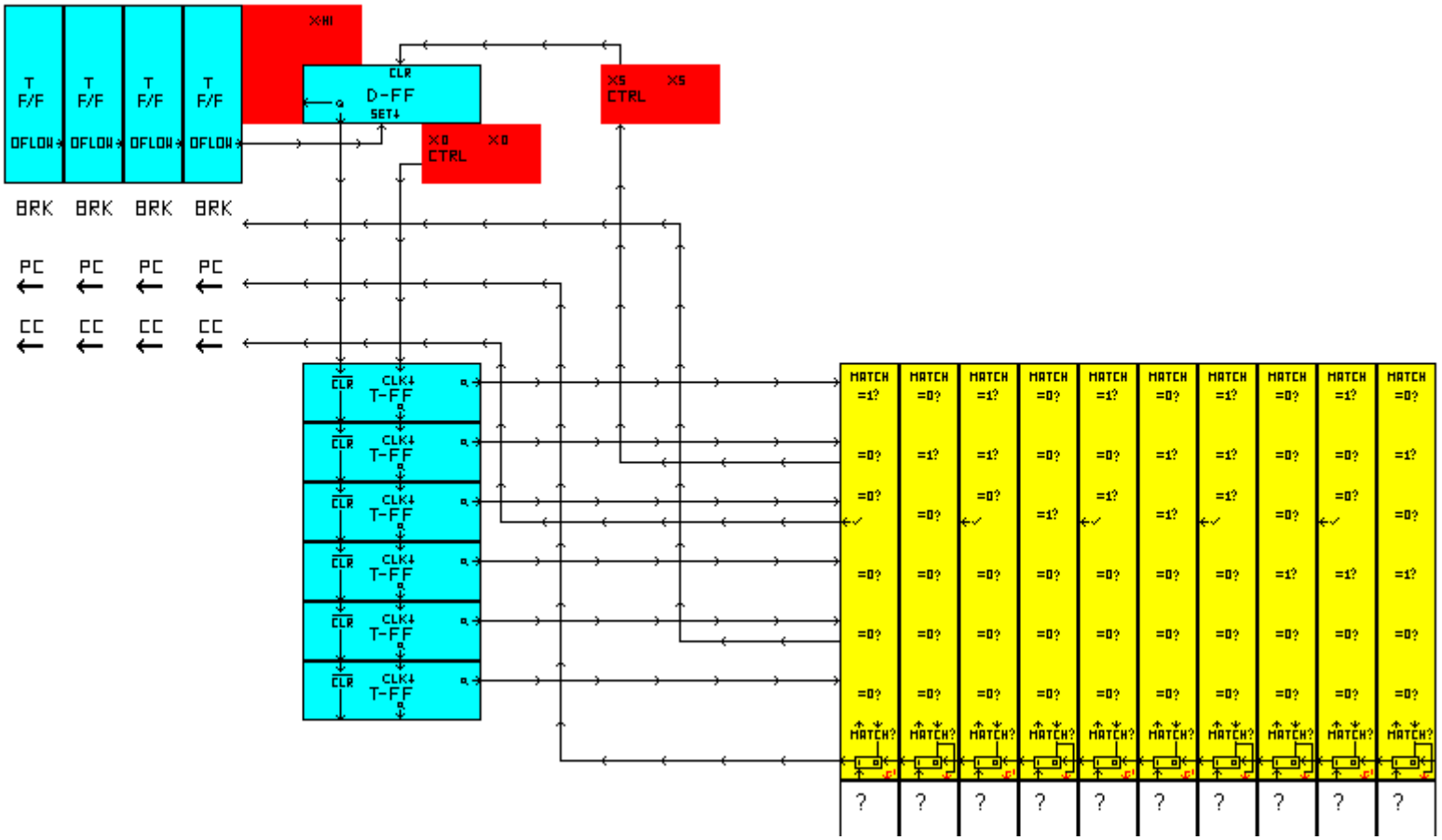


Other Useful Parallel Circuits

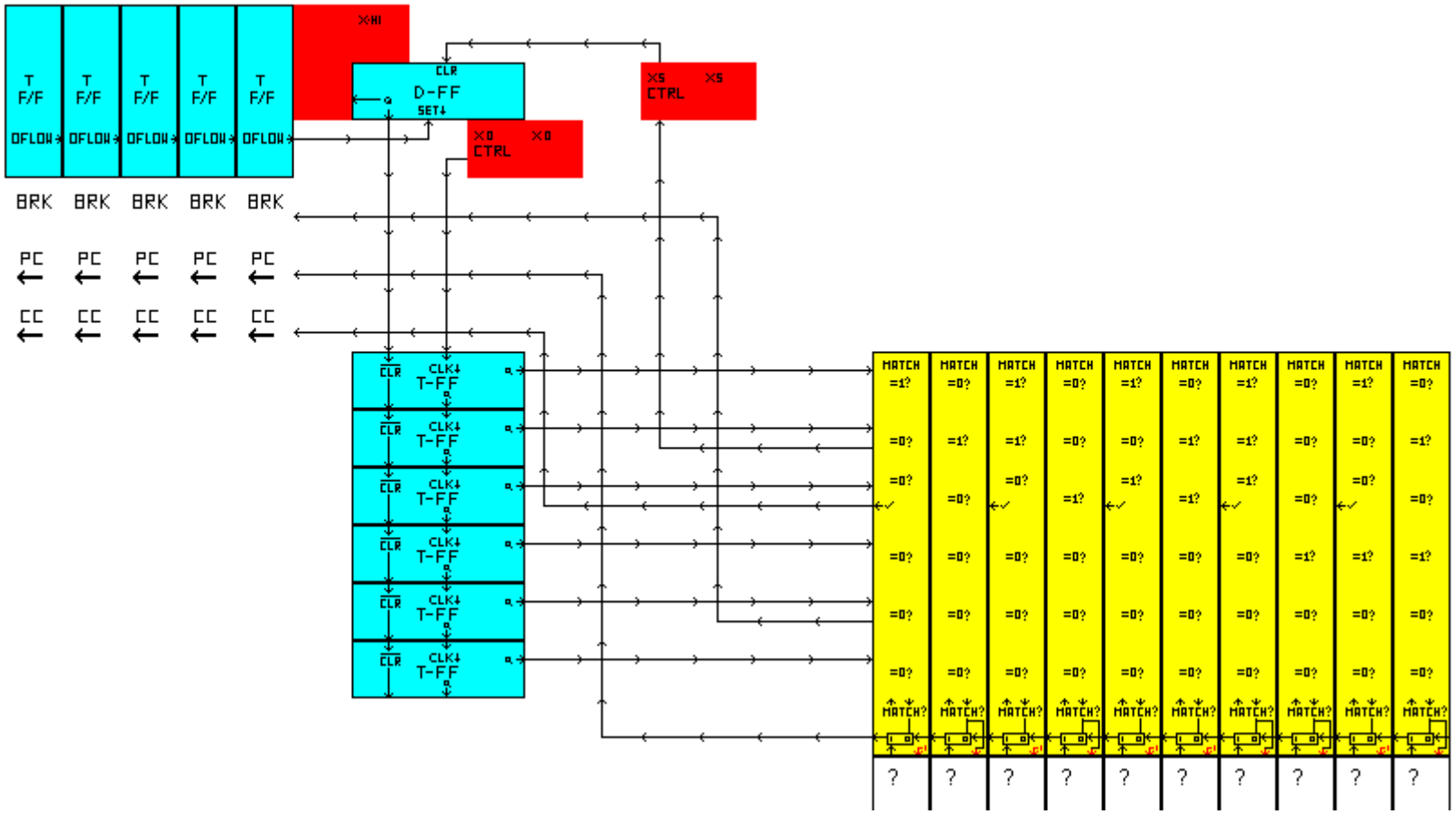
- Combinational nxn matrix multiplier
- APL machine
- Finite element method w/dynamic mesh refinement
- Simulation of physical processes

Can Use Cell Matrix-Specific Features to Develop New, Unique Circuits

Expanding Counter



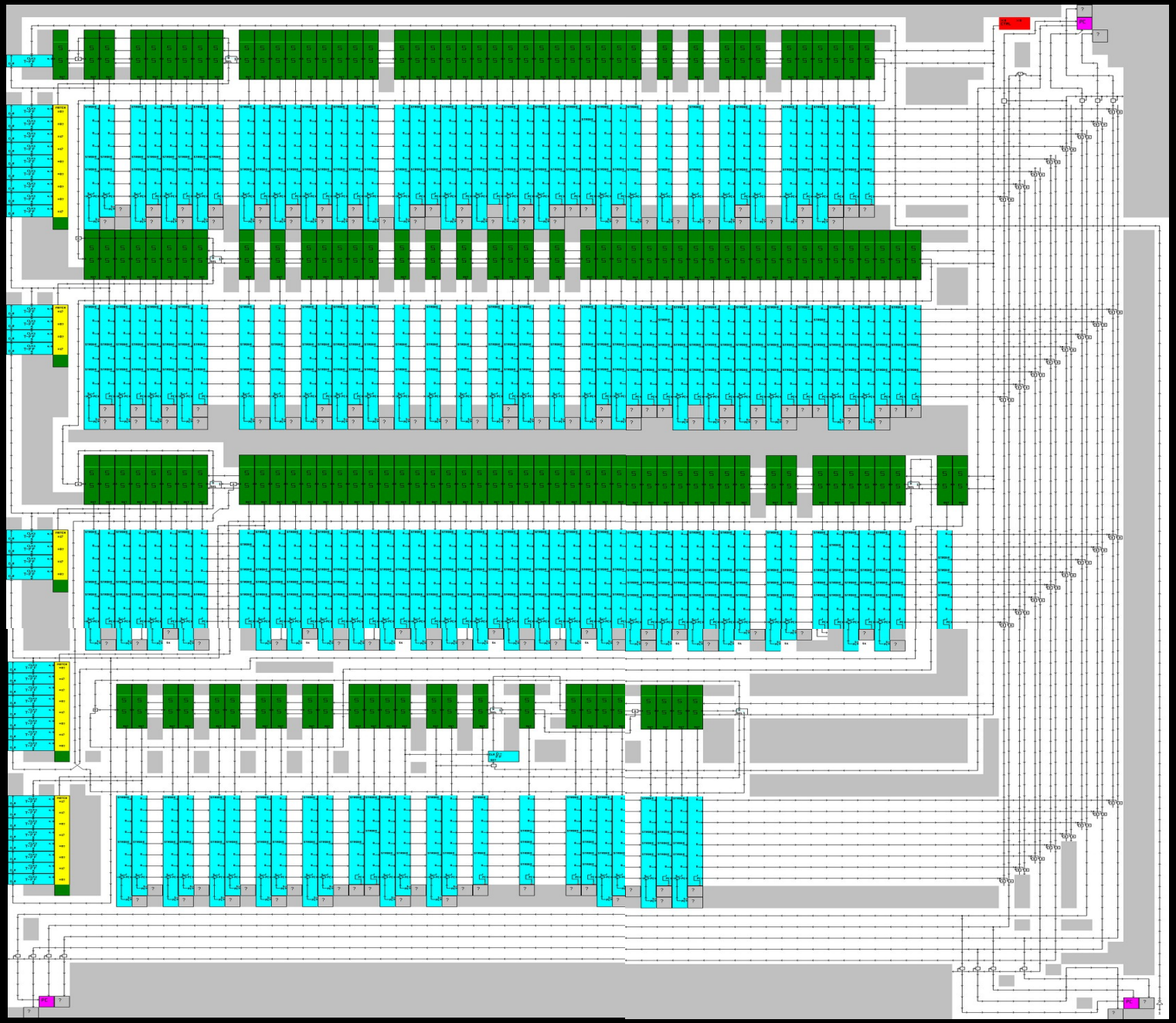
Expanding Counter



This is an example of autonomous, self-modifying hardware!

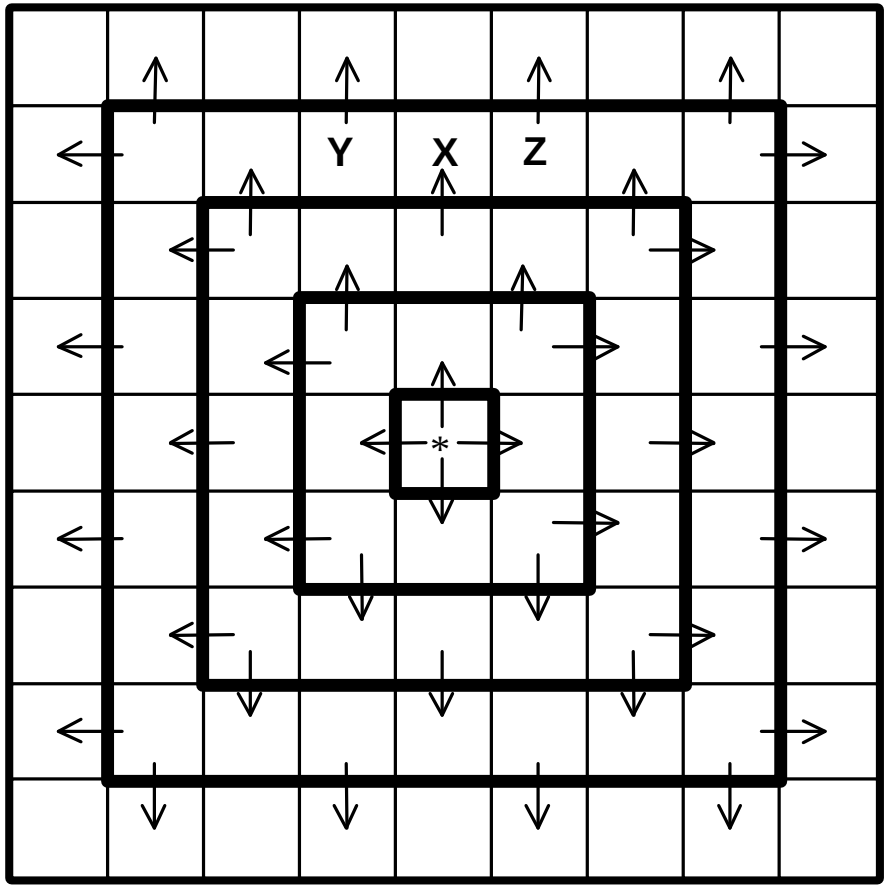
Leads to Interesting Possibilities

- Virtual Hardware
- Hardware Swapping
- Architecture Tuning



Evolvable Hardware

Ringed GA



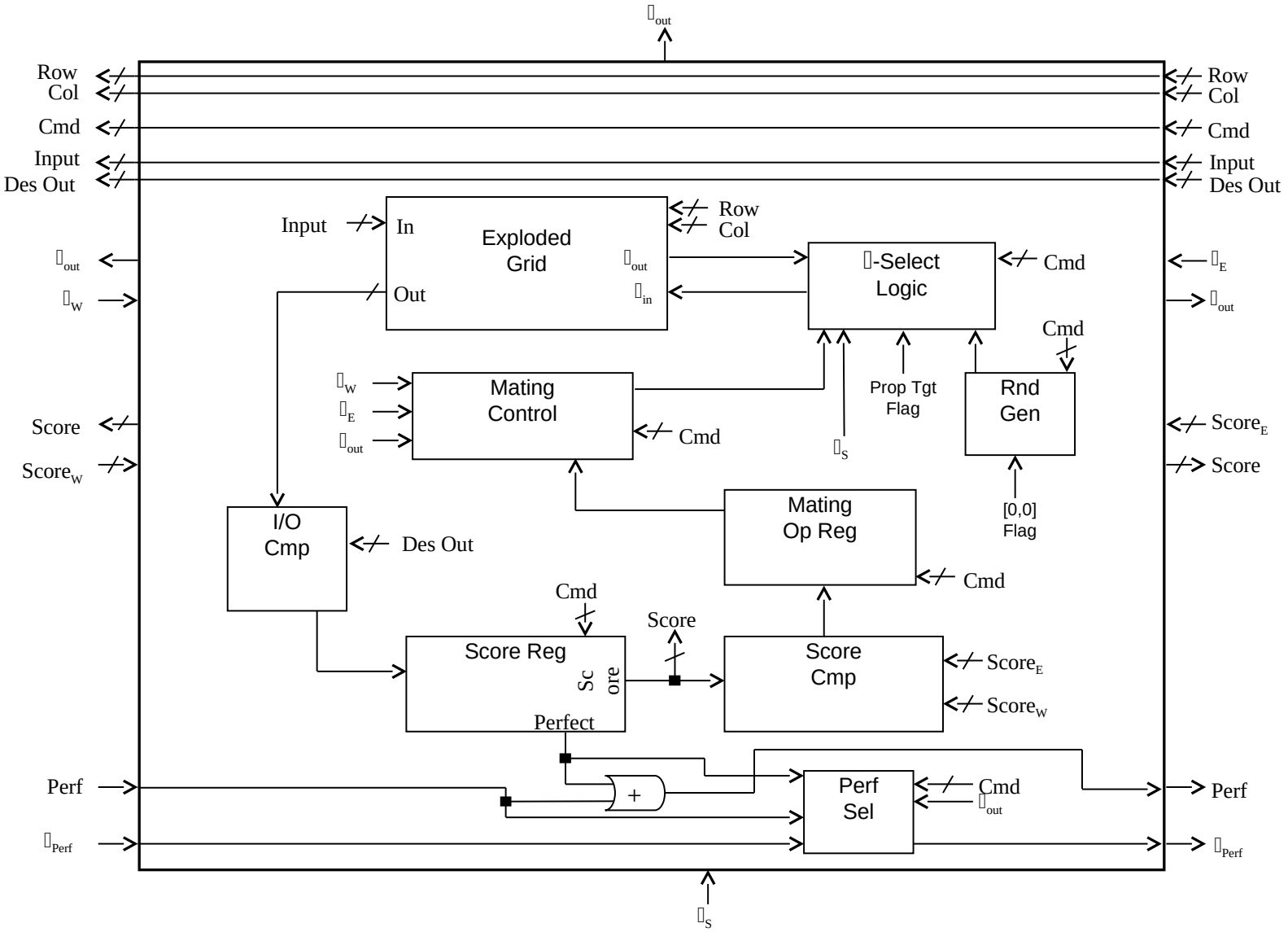


Figure 6. Single Individual for the RGA. Configuration shown and comments below are for individual along top of a ring and not in a corner. Actual evolving circuit is stored in the Exploded Grid. Row, Col, Cmd, Input and Des Out come from global controller, and are sent in parallel to all individuals. Cmd selects the RGA operation, Row and Col access individual PIG cells within the individual's circuit, Input is a test input, Des Out is the desired output from the circuit. All other I/O lines connect to immediately neighboring individuals. lines carry configuration information. Perf indicates an individual along the row has achieved a perfect score. Configuration of leftmost perfect individual appears along $_{Perf}$.

Fault Tolerance

- Prior to building blocks of cells, can run self-tests on region inside matrix
- Then mark bad blocks and avoid internally
- Guardwalls
- Student working on utilizing bad block info

STATUS AND FUTURE WORK

Current Status

- Cell Matrix Corporation
  (www.cellmatrix.com)

Current Status

- Cell Matrix Corporation
- Patents issued (US #5,886,537) and pending

Current Status

- Cell Matrix Corporation
- Patents issued and pending
- Software toolset
  - Simulators
  - Layout Tools
  - Viewers
  - Debuggers
  - Compilers

Current Status

- Cell Matrix Corporation
- Patents issued and pending
- Software toolset
- Spreading the word

Current Status

- Cell Matrix Corporation
- Patents issued and pending
- Software toolset
- Spreading the word
- NASA SBIR

Current Status

- Cell Matrix Corporation
- Patents issued and pending
- Software toolset
- Spreading the word
- NASA SBIR
- Collaborations

Current Status

- Cell Matrix Corporation
- Patents issued and pending
- Software toolset
- Spreading the word
- NASA SBIR
- Collaborations
- Students

Future Work
More Circuits

- Different parallel configuration schemes
- Numerous specific problems
- Libraries

Future Work
Application to Neural Networks

- Possible collaboration with Hugo de Garis (Starlab)
- Populations of neural networks
- Autonomous evolution

Future Work
Software Development

- 3-D Toolset
- Networked simulator
- Better debugging
- Better viewing
- Higher-level tools

Future Work
Evolvable Hardware

- Preliminary D-mode work done
- Evolving C-mode circuits?
- Hard to control, but can be done

Future Work
Hardware Fabrication

- FPGA implementation-up to 10,000 cells
- ASIC-Up to one million?
  - Need a foundry
- 3-D Manufacturing: ???

Future Work Nanotechnology

- Unlimited potential
- Initial contacts
- Want to focus nanotechnology research

Future Work
Looking for students

- Lots of good research work
- Hardware (implementation, fault tolerance)
- Software (tool development)
- Cell Matrix (circuit development, methodologies, etc.)

Acknowledgements

- Lisa Durbeck-Cell Matrix Expert, CEO
- Murali D. Raju-Co-inventor
- Lawrence B. Henry III-Co-inventor
- Hugo de Garis
- Jim Pearn (www.artificialbrains.com)
- Dimitri Yatsenko

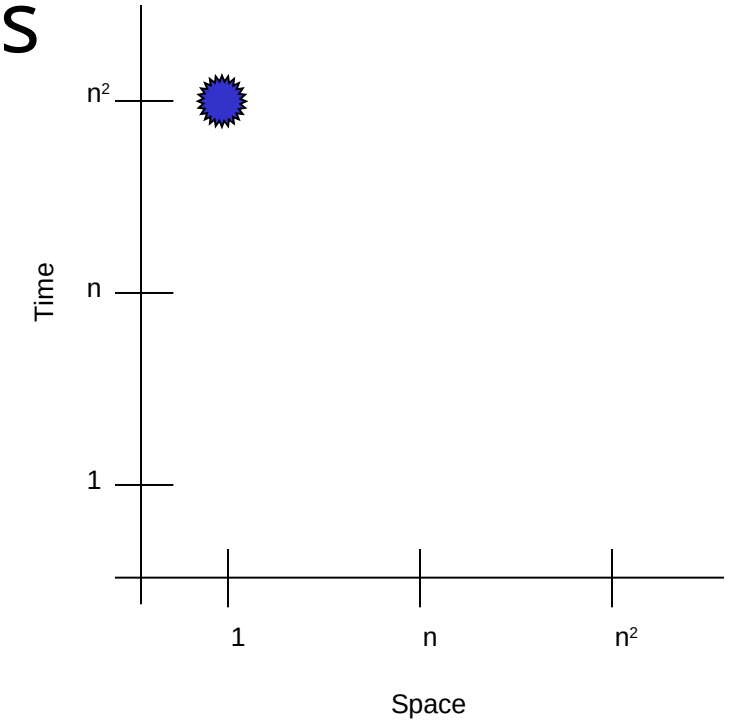
For More Information

- Cell Matrix Corporation (www.cellmatrix.com)
- nmacias@cellmatrix.com
- US (877) 473-0882

  (801) 414-8900 (Nick)
- Come talk to me!

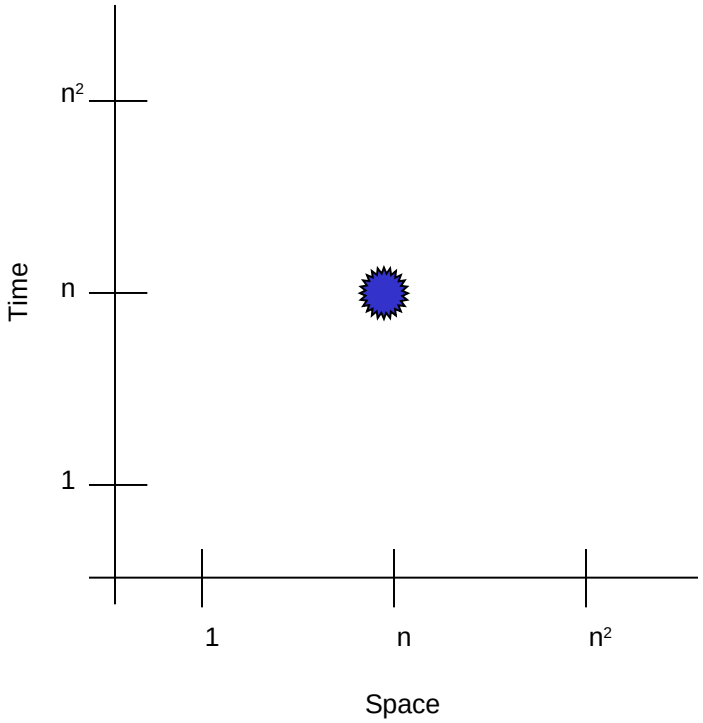
Space Vs. Time

- Fixed hardware uses
  O(1) space and
  O(n^2) time



Space Vs. Time

- Reconfigurable HW used to build n-bit accumulators and logic uses O(n) space and O(n) time



Space Vs. Time

- Reconfigurable HW used to build an nxn multiplication array uses $O(n^2)$ space and $O(1)$ time