
Adaptive Computing Systems

Brad L. Hutchings

Dept. of Electrical and Computer Eng.

Brigham Young University

What it is

- ◆ Using reprogrammable PLDs as compute devices.
 - implement custom datapath, hard-wired control.
 - Commercially available devices
 - » Xilinx, Altera, Atmel
 - ◆ fine-grained (1-bit) devices.
 - Research prototypes
 - » VT, UW, CMU, MIT, Berkeley, host of others.
 - ◆ coarser-grained (8-16 bit) devices.
- ◆ Key Idea: *custom architecture per application.*

What we know

Highest performance (>10x) can be achieved if:

- 👉 Applications exhibit significant data-level parallelism.
- 👉 Application data sets are large.
- 👉 Implementations are regular, concurrent, and deeply pipelined.
- 👉 Data elements are low resolution.
- 👉 Data are manipulated with logical or simple integer operations.
- 👉 Applications require only simple control.

What we've done

- ◆ Neural Networks
- ◆ Application-Specific Processors (DISC)
- ◆ Signal Processing
- ◆ Image Processing
- ◆ Automated Target Recognition
- ◆ Run-Time Reconfiguration
- ◆ Genetic Algorithms

General DISC Approach

- ◆ **Library-Based Approach**
 - Library contains application-specific hardware modules.
- ◆ **Software Control**
 - Sequencing, complex control, I/O controlled by software.
 - Software essentially “glues” modules to form an application.
- ◆ **Demand-Driven Execution**
 - Hardware modules loaded as required by the application.
- ◆ **Hardware Paging**
 - Idle hardware replaced with active modules.
- ◆ **Linear Hardware Space**
 - Hardware relocated at run-time to available FPGA space.

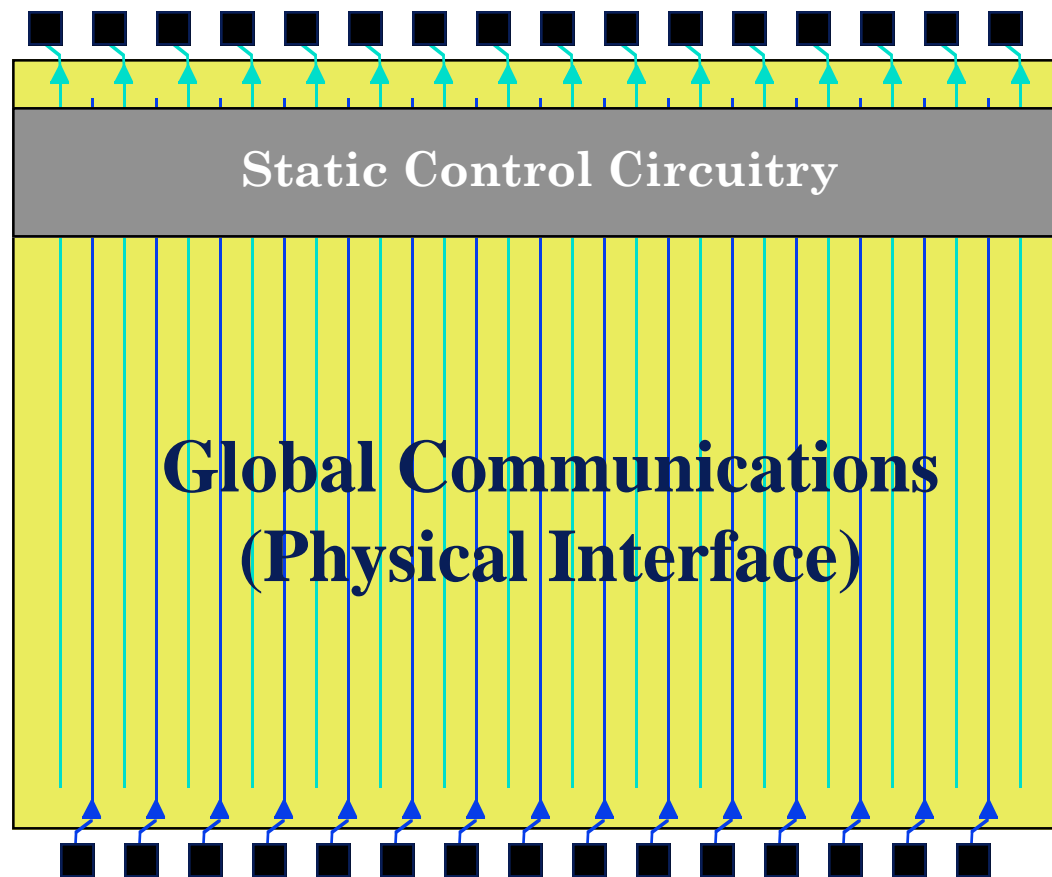
DISC: Prototype Implementation



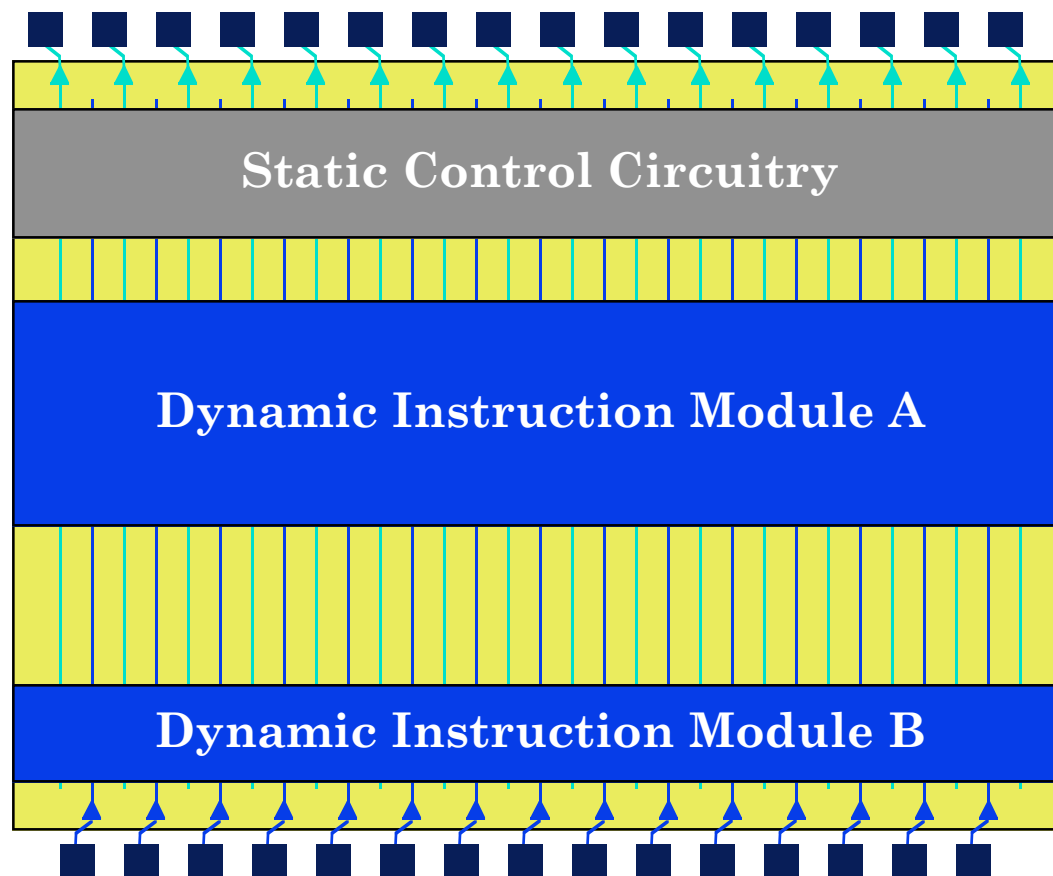
DISC: Prototype Implementation



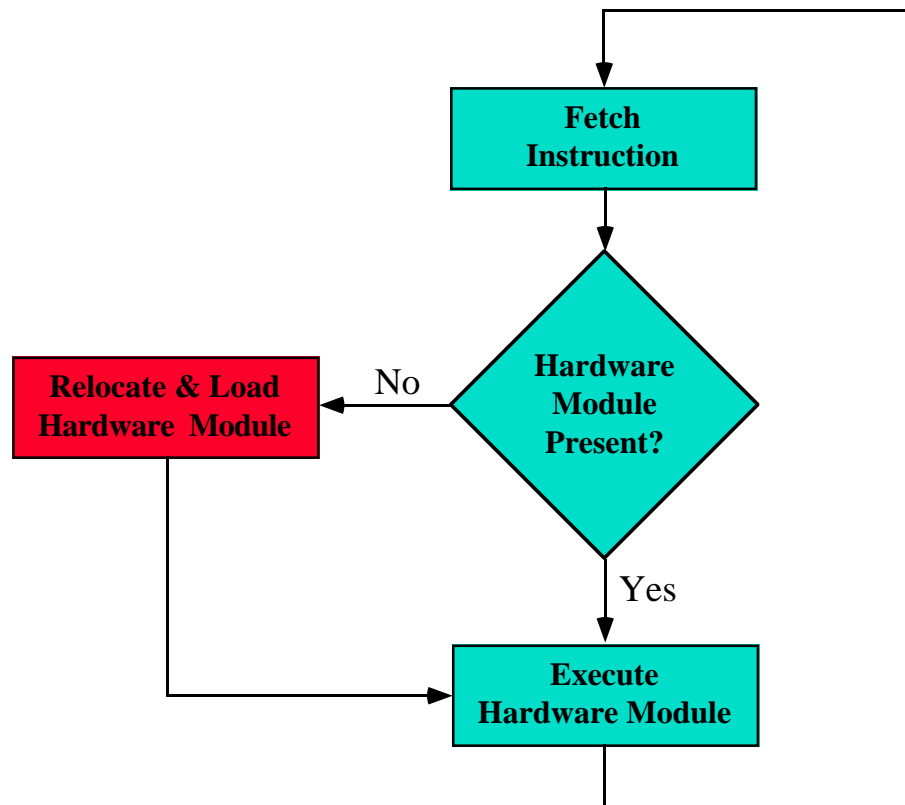
DISC: Prototype Implementation



DISC: Prototype Implementation

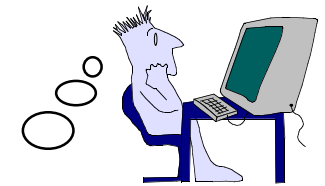


DISC Operation

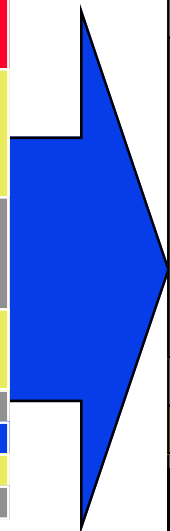


DISC Programming Model

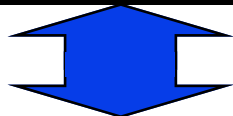
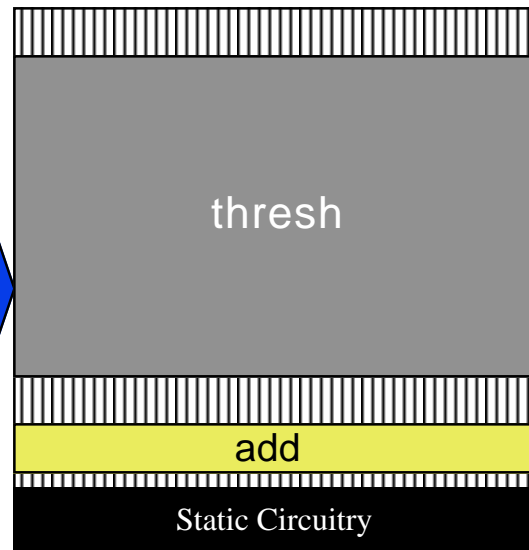
Application Programming



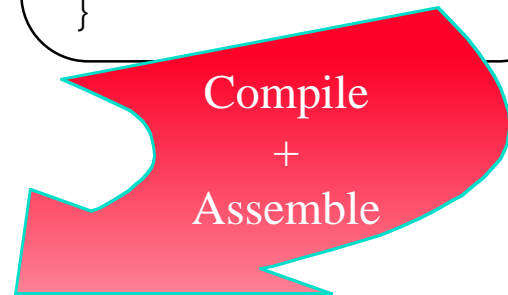
Instruction Library



DISC
(FPGA)



```
main() {  
    ...  
    while(...) {  
        hist(image1);  
        median(image1,t);  
        → thresh(t,image1);  
        copy(image1,i2);  
        erode(image1);  
        diff(i2, image1);  
    }  
}
```



Circuit Design



Image Morphology



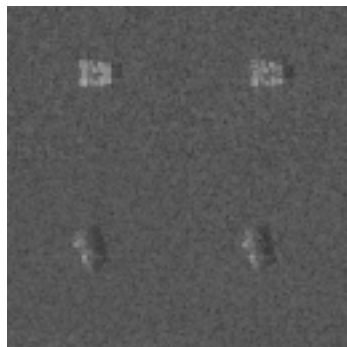
Original
Image



Outlined
Image



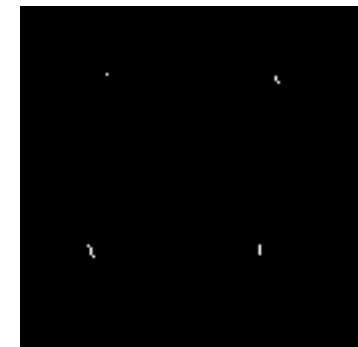
Skeletonized
Image



Original
Image



Processed
Image

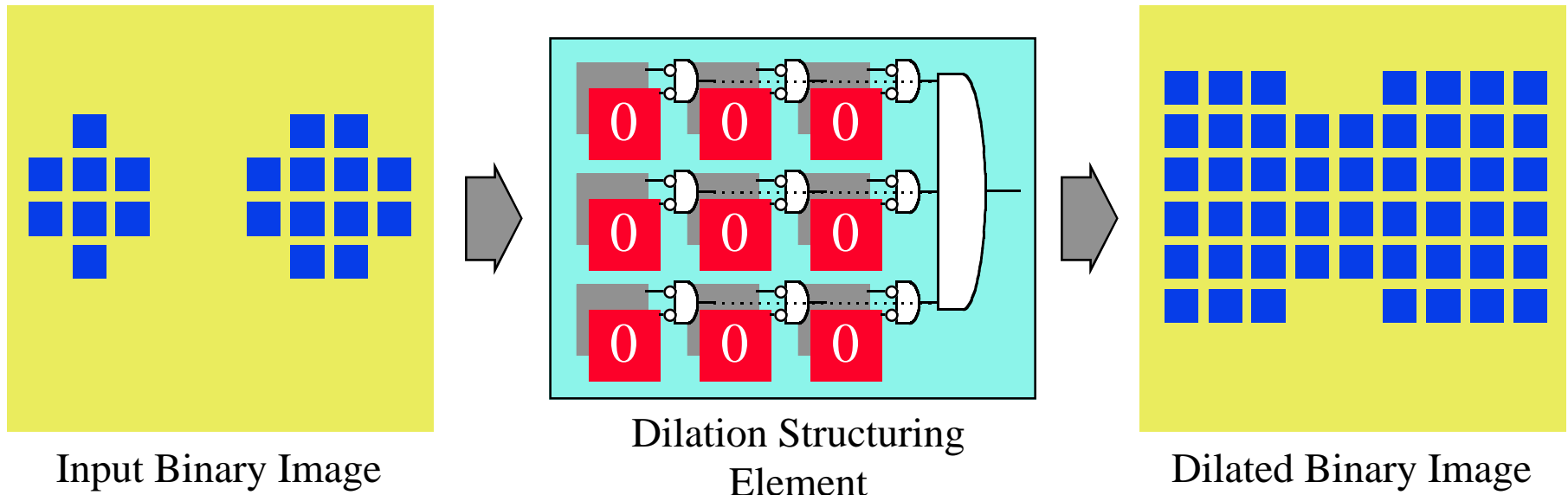


Processed
Image

Morphology Implementation Issues

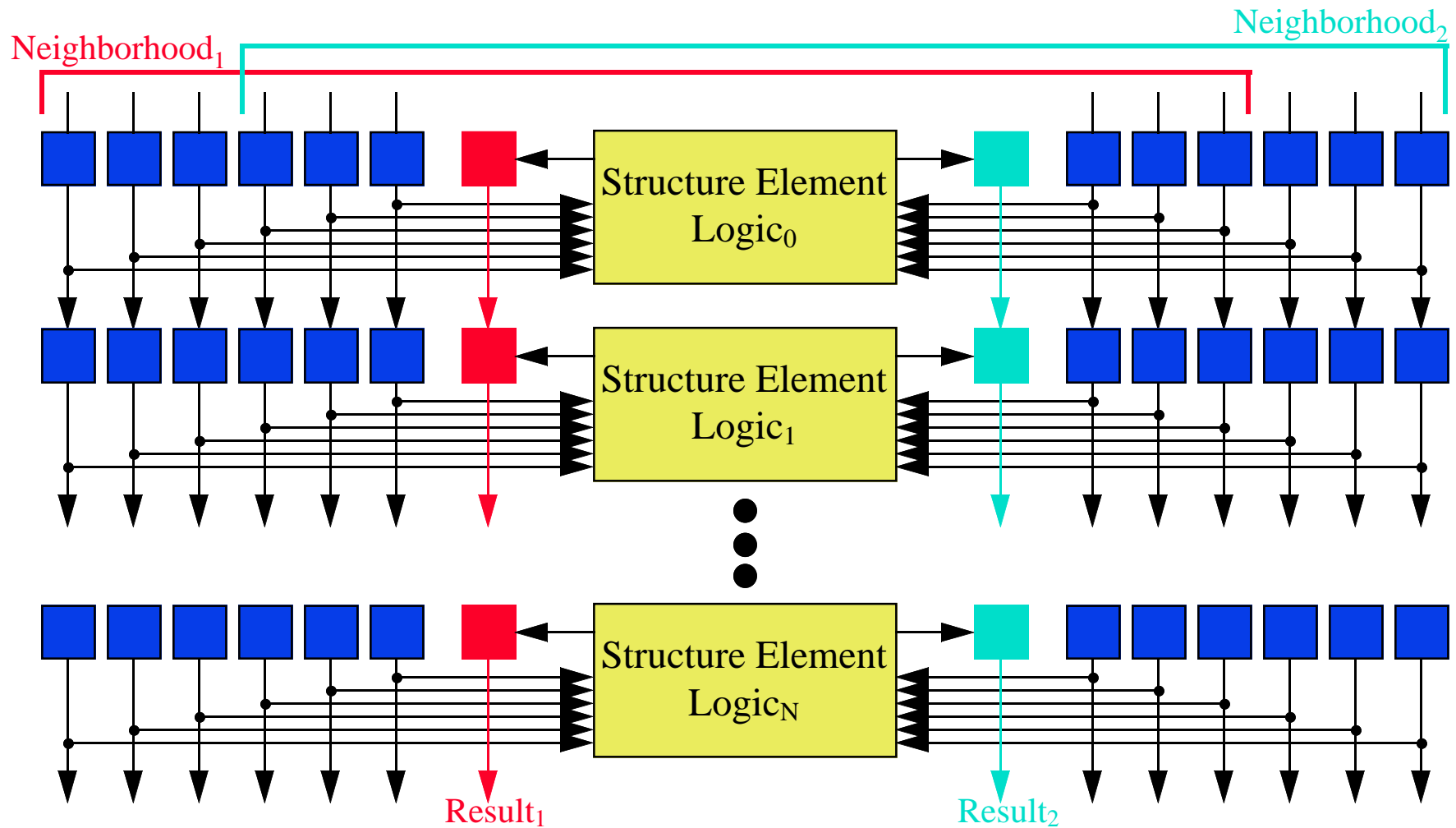
- ◆ Uses only simple logical operations.
- ◆ Extremely parallelizable.
- ◆ Can be deeply pipelined (1000s of stages).
- ◆ Must be programmable.
 - every morphology problem is different.

Morphology Operation



Dilation

Physical Realization



Projected Performance

- ◆ FPGA-based Morphology:

- » two bit-ops per row.
- » 400 rows per MCM.
- » 100-200 MHz
- » 800 bit-ops per clock cycle.
- » *80-160 billion bit-ops per second (BBOPS).*

- ◆ Compare to CPU-based Morphology:

- » Software optimized.
- » 486-66: 108 MBOPS, 1.63 bit-ops/clock cycle.
- » HP PA8000-160: 1.84 BBOPS, 11.51 bit-ops/clock cycle.

ATR and Correlation

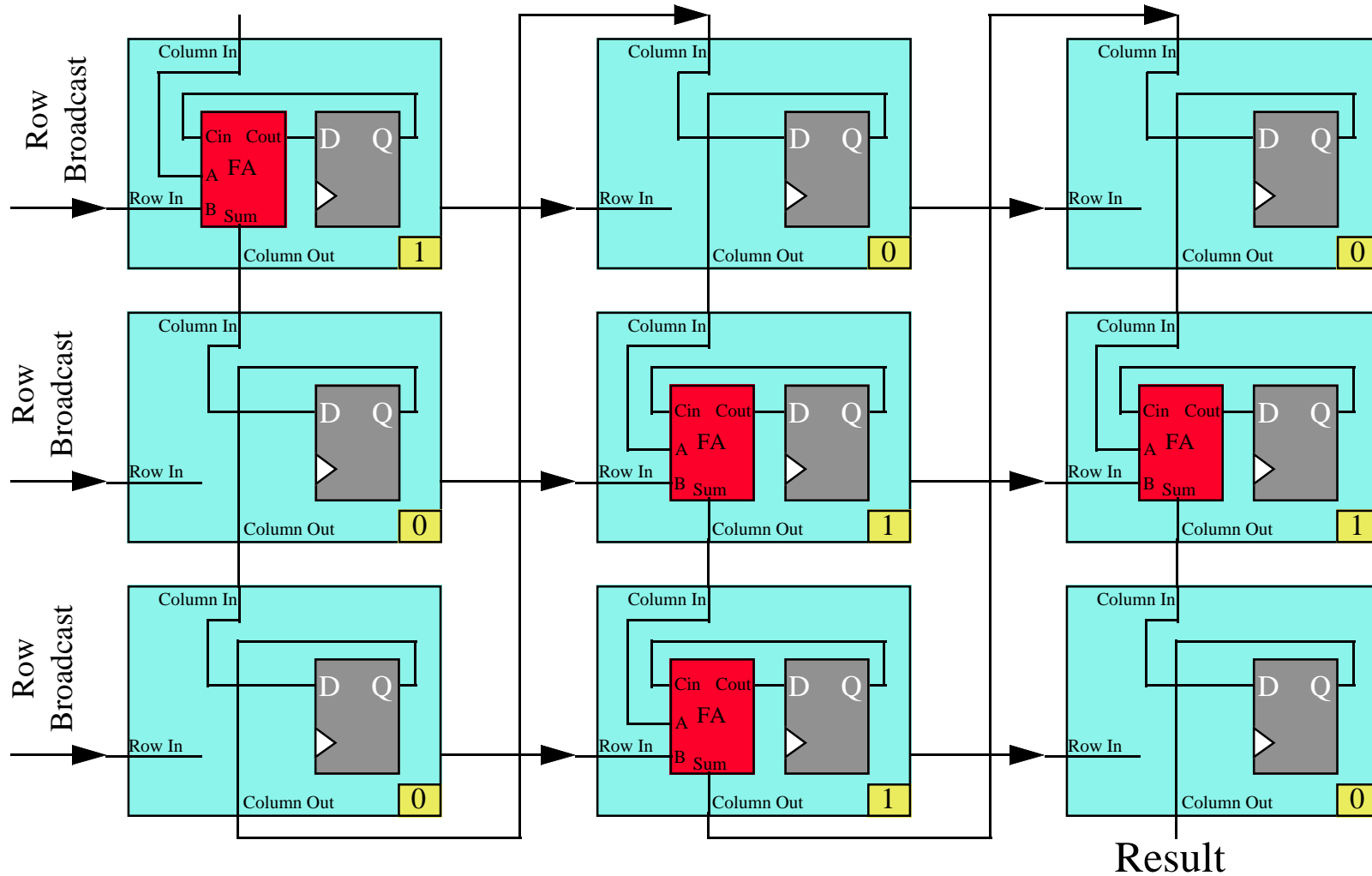
- ◆ Shape-sum is preliminary step of ATR.
- ◆ Shape-sum is (in part) correlation of:
 - $16 \times 16 \times 1$ -bit template with
 - $128 \times 128 \times 8$ -bit image.
- ◆ Correlation of **17K** templates (Chunky-SLD).
- ◆ Each template *can* be correlated in parallel.
- ◆ Question: **How to achieve significant parallelism at reasonable cost/size?**

Constant Propagation & RTR

◆ Key Ideas:

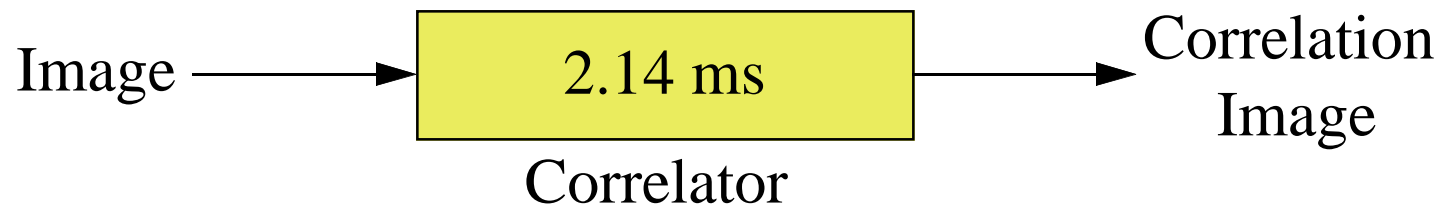
- 1. Decompose correlation into series of bit-serial adders.
- 2. Create specialized ‘1’ and ‘0’ bit-serial adder circuits.
- 3. At run-time, create a specialized shape-sum circuit by wiring these circuits for a specific template.
- 4. Partially reconfigure as necessary to create new shape-sum circuits.

Implementation (3x3 Example)



Projected Performance

- ◆ Image size: 128 x 128 x 8-bit.
- ◆ Template size: 16 x 16 x 1-bit.
- ◆ Clock rate: 100 MHz (200 MHz likely)
 - » 2.14 ms per template.
 - » 490 correlations per second/per correlator.
 - » 980 correlations per second/ per die.



Other Demonstrated Areas

- ◆ Image processing:
 - convolution, stereo imaging, filtering, edge detection, compression, etc.
- ◆ In general:
 - long-integer arithmetic, molecular biology, cryptography, high-energy physics, heat transfer, etc.

Primary Problems

- ◆ How broadly will this approach apply?
- ◆ How do we automate the implementation process?
- ◆ What device and system architectures are appropriate?