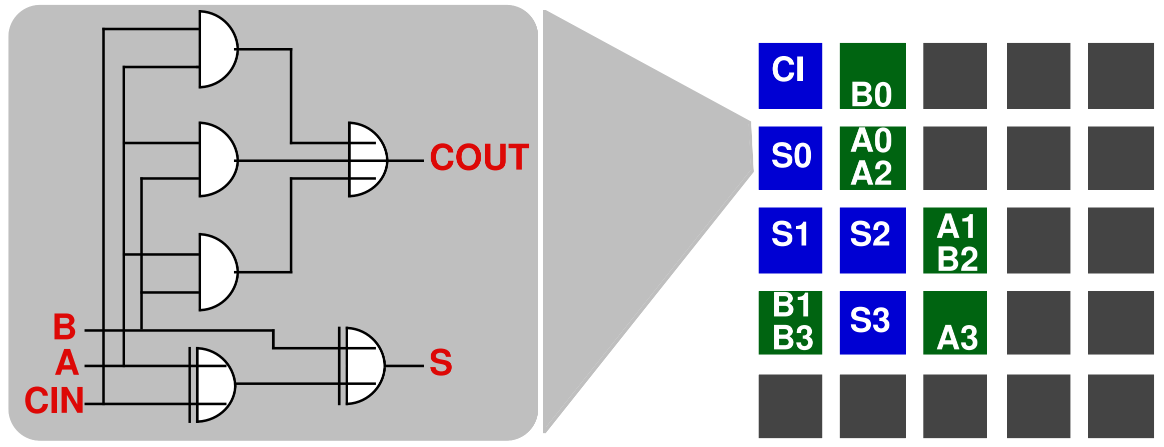
# Regular Datapaths on Field-Programmable Gate-Arrays

Andreas Koch
akoch@cs.berkeley.edu

## Overview

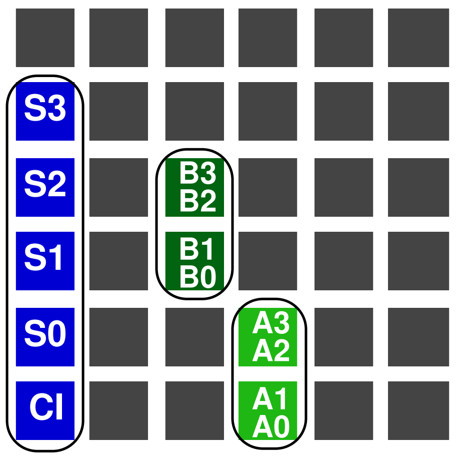
- Evolution of datapath implementation
  - Conventional
  - Module-based
  - Floorplanned modules
- Structured Design Implementation (SDI)
  - Enhanced module generators
  - Floorplanner
  - Compaction
- Experimental Results
- Summary

# Evolution of Datapath Implementation

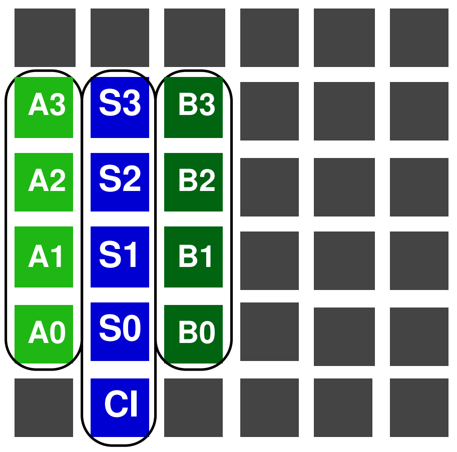


Gate-based

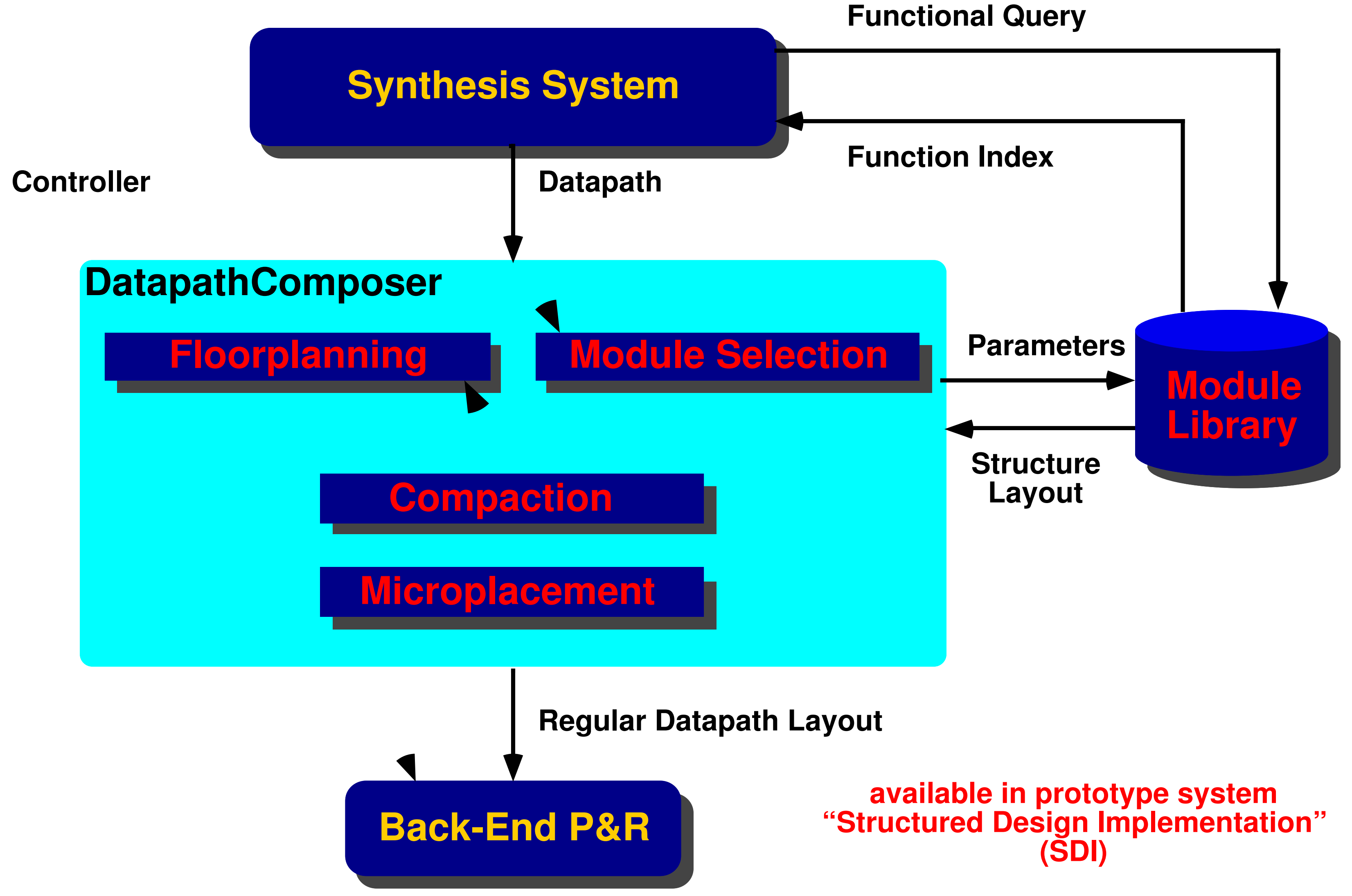
Module-based



Floorplanned Module-based



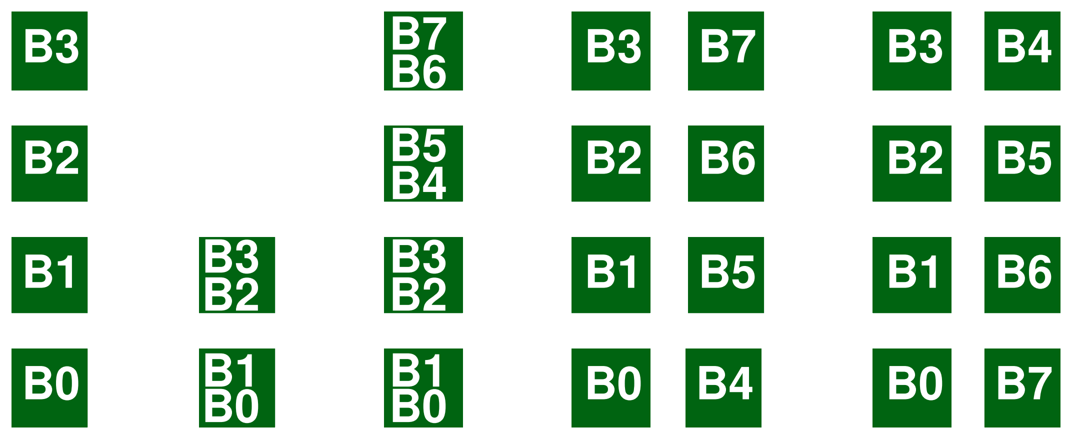
## System Overview



available in prototype system
“Structured Design Implementation”
(SDI)

# Module-Oriented Approach

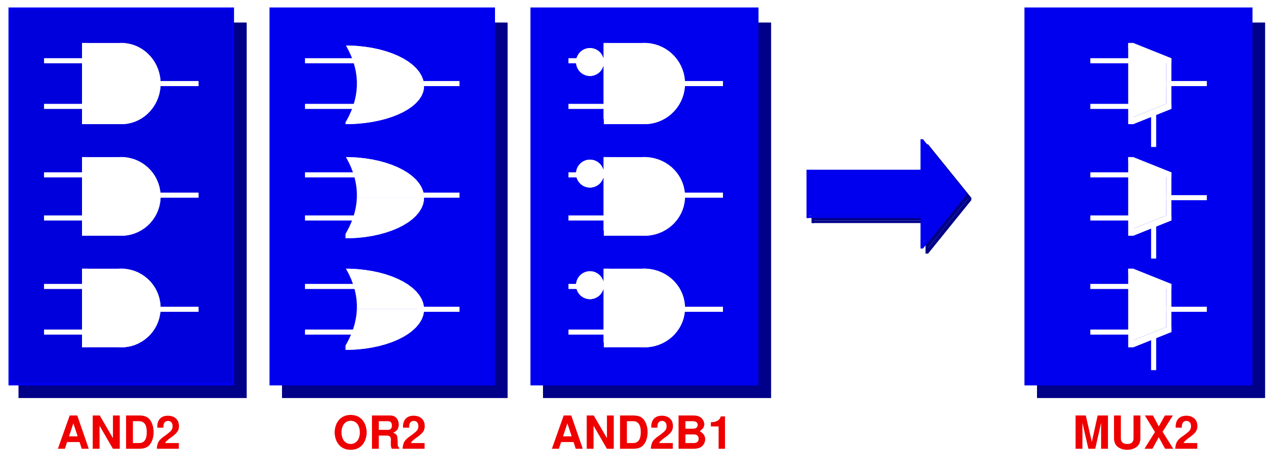
- Modules instead of gates: Add, Sub, Mul, RAM, ...
- Parameters: operand widths, data types
- Pre-partitioned and pre-placed circuits
- Alternate implementations may vary in
  - Tradeoffs: area, speed, latency, power, ...
  - Logical pitch (bits-per-logicblock)
  - Layout folding: linear, unidirectional, alternating



## Automatic Floorplanning & Selection

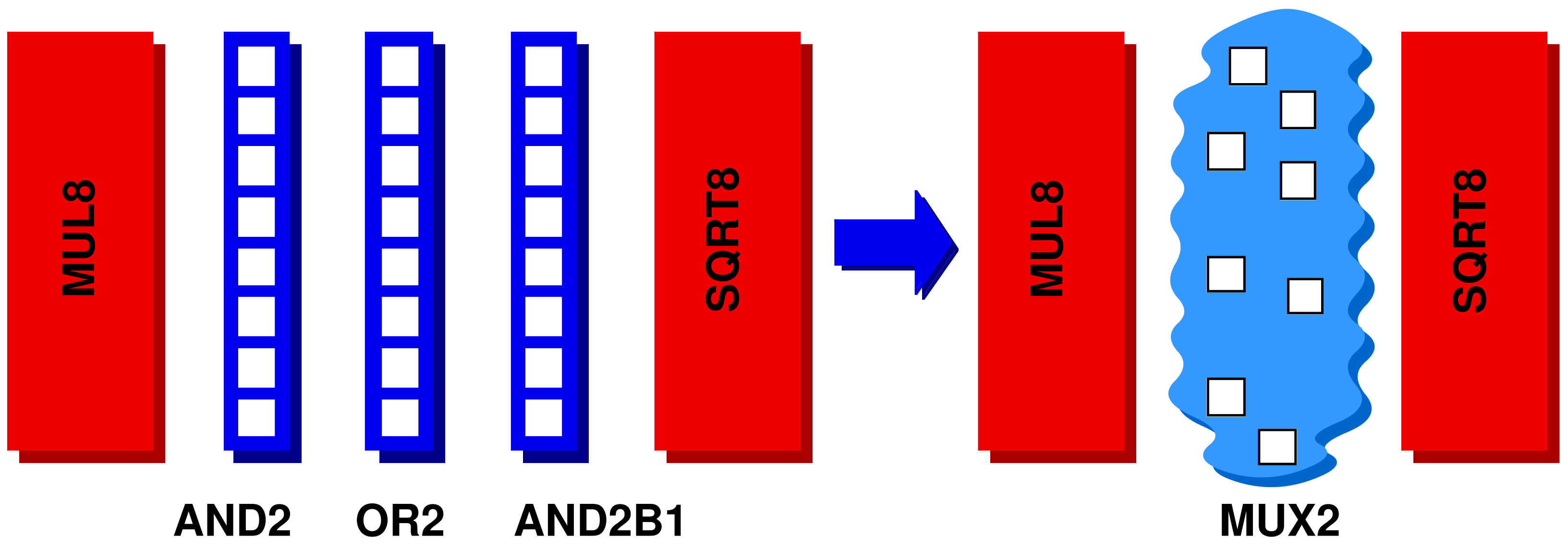
- **Optimizes entire data path**
  - Per-instance implementation selection
  - Intelligent module placement
  - ➤ Aligned significances, matched pitch
- **Evaluates**
  - Delay, Area, ...
  - Routability
  - Mergeability of adjacent modules
- **Generates**
  - Implementation assignment per instance
  - Linear placement of modules

## Coarse-Grained Logic Blocks



- Large and powerful logic blocks
  - Reduced programming overhead
- Inefficient mapping of simple functions
  - 2/3 area wasted
  - 1/2 speed lost
- Ratios deteriorate with increasing block size

# Currently: Simple Modules Dissolved

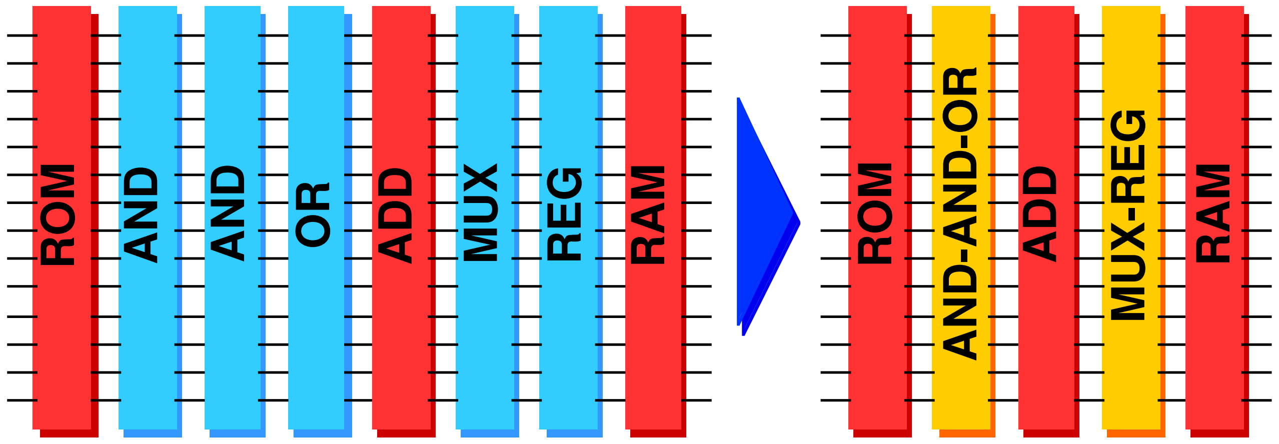


- No more pre-placement
- No more pre-partitioning
- Instead: unstructured
  - Logic optimization
  - Partitioning
  - Placement
- ➤ Irregular layout: lower performance

## New Approach in SDI

- **Compaction**
  - After floorplanning
  - Considers adjacent “simple” modules
  - Preserves bit-slices
- **Exploitation of regularity**
  - Extract circuit structure
  - Use recurring sub-circuits as new slices
- **Perform at bit-slice level**
  - Logic optimization
  - Partitioning
- **Perform at module level**
  - Regular placement within bit-slices

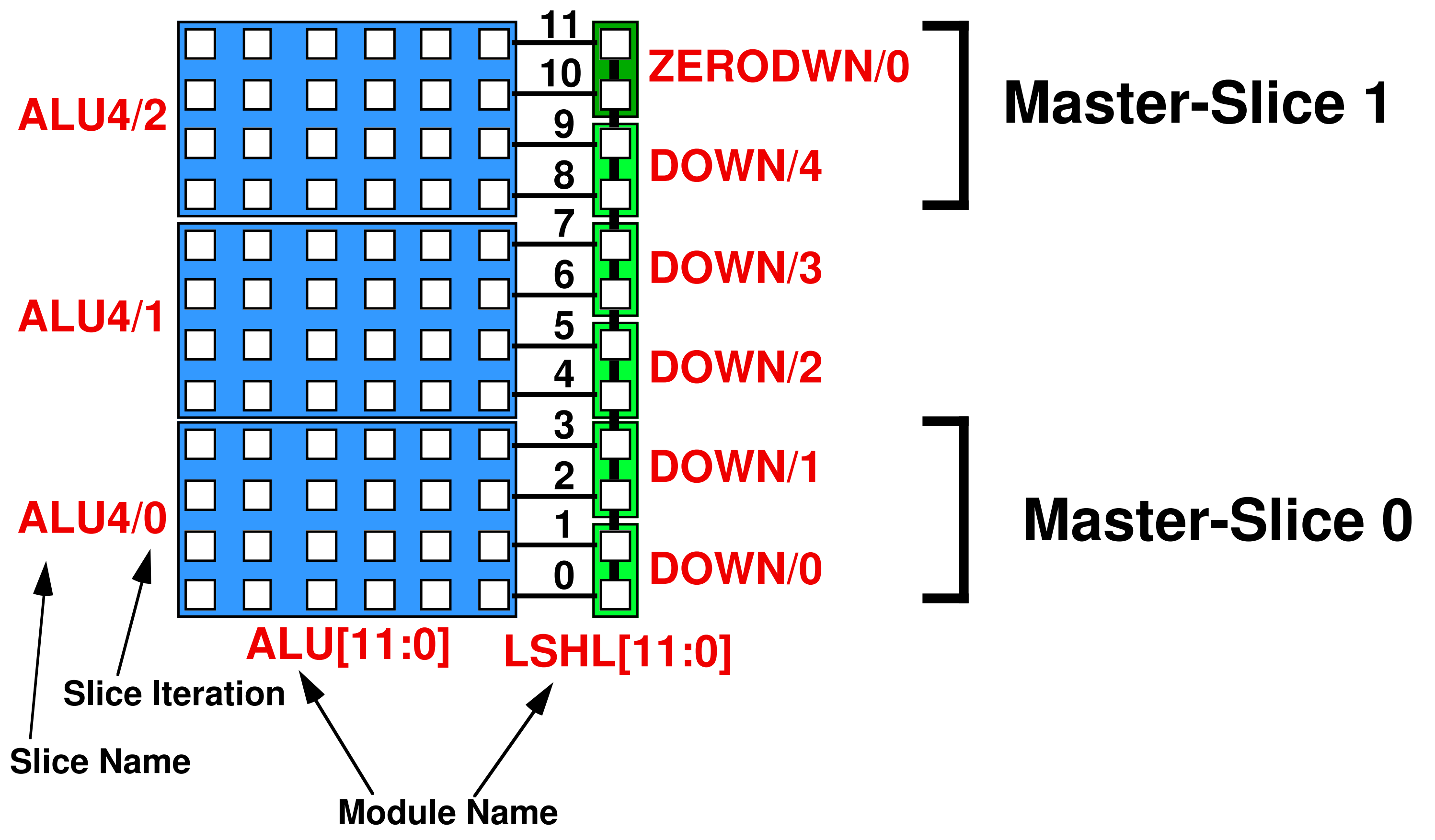
# Compactable Areas



- "Simple" combinational and sequential logic
  - Compactable
- FPGA-specific elements or complex modules
  - "Hard" modules
  - Not compactable
  - Delimit compactable areas
- Floorplanned topology left intact

## Structure Extraction/Regularity Analysis

- Determine bit-slices across module boundaries
- Find recurring sub-circuits



## Merging

- Performed on newly discovered master-slices (MS)
  - Across module boundaries
  - Within bit-slices
- ➤ Regularity is exploited and preserved

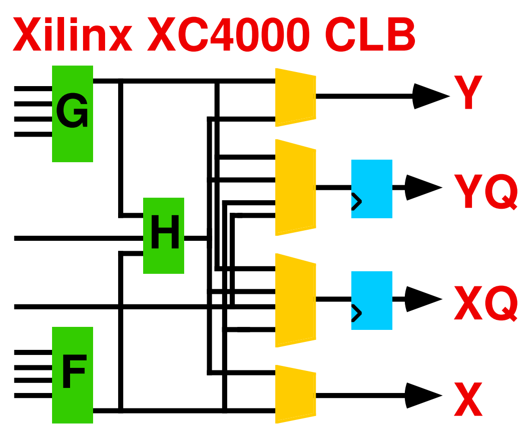
- Integrate standard tools
  - Logic optimization (SIS, ...)
  - Partitioning (MIS-PGA, FlowMap, TOS-TUM, ...)

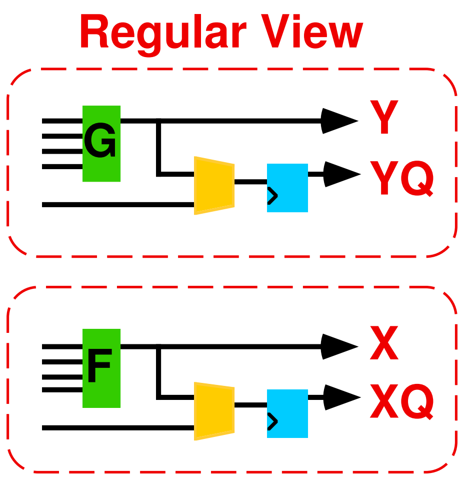
- ➤ Circuit now altered
  - Re-placement and -routing required

## Micro Placement

- **Places logic blocks within compacted module**
- **Exploit regularity by processing compacted MSs**
- **Timing-driven**
- **Two-phase placement**
  - **Horizontally**
    - Enable efficient control line routing
  - **Vertically**
    - Minimize delay
- **Regular view on target FPGA**

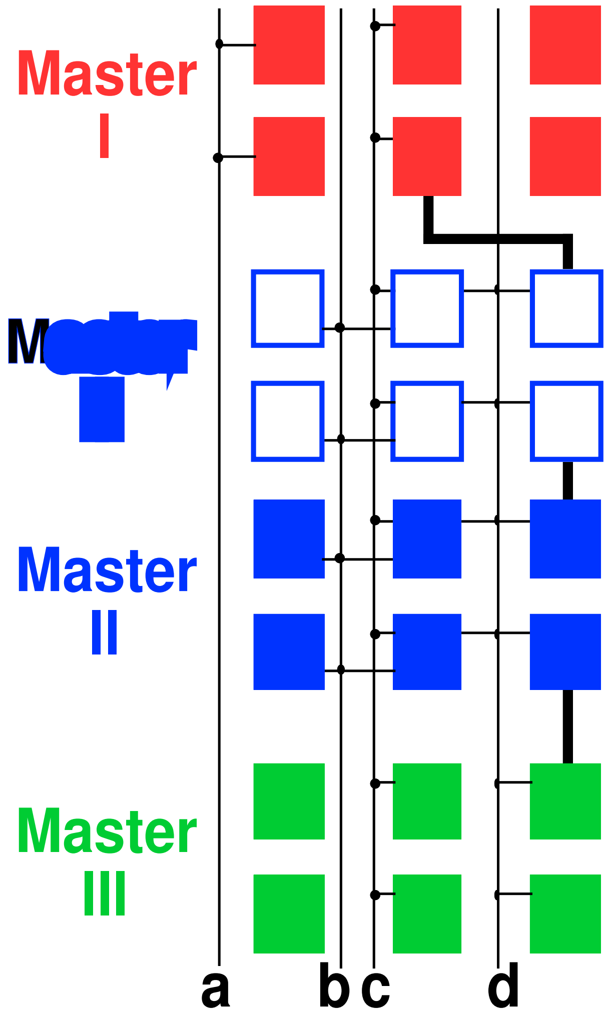
# Regular View on Logic Blocks





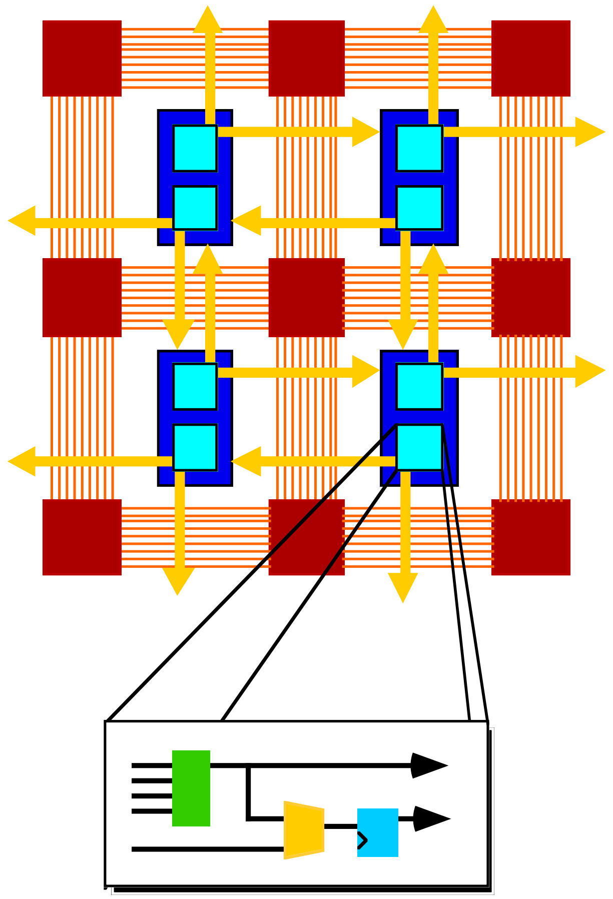
- XC4000 CLB inherently irregular
  - Two outputs, three LUTs
  - Limited flip-flop access
- ➤ Treat 1 CLB as 2 independent regular cells
  - Single 4-LUT plus flip-flop per cell
  - Unrestricted flip-flop access via H-block
- Placement performed on array of cells

# Horizontal Micro Placement



- Arranges cells in columns
- Simultaneously on all MSs
- Considers
  - Control signals on vertical long lines
  - Slice abutment
  - External connections
  - Delay estimates

# Vertical Micro Placement



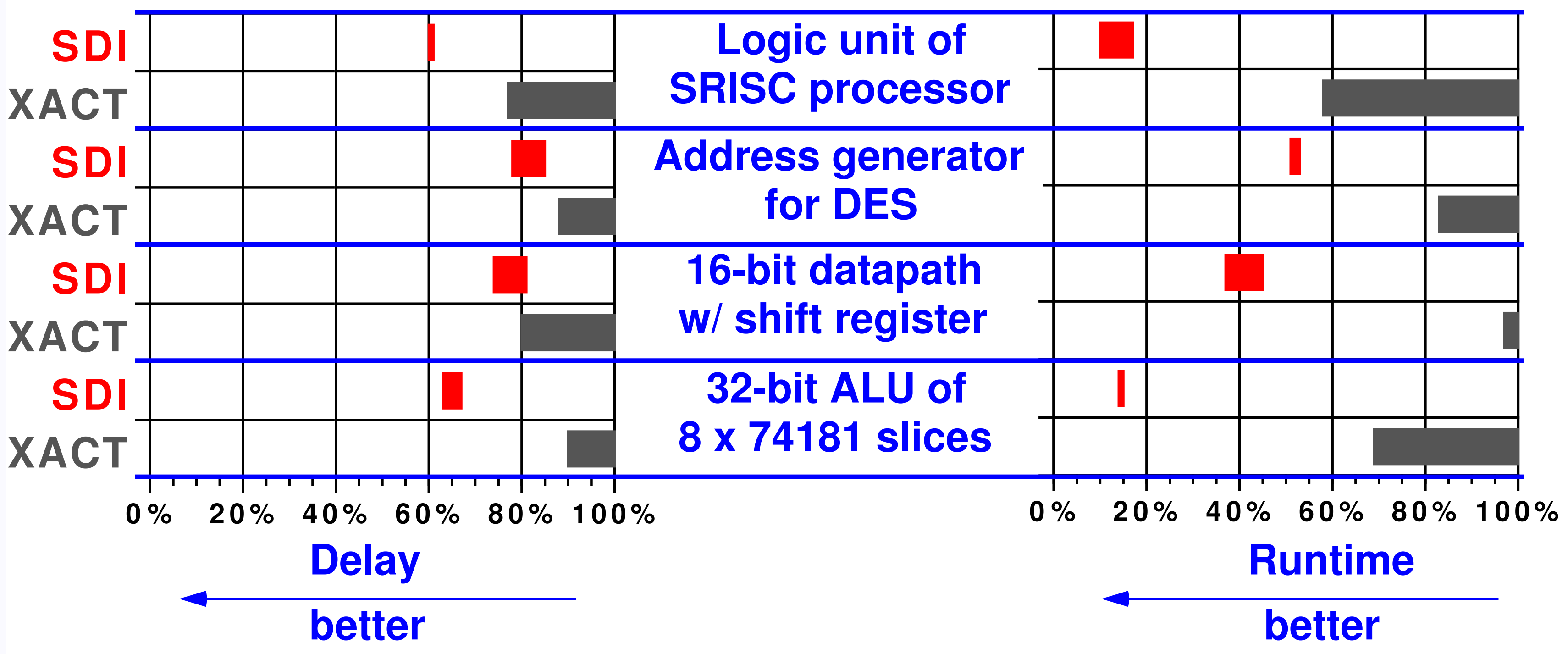
- Row-arrangement of cells
  - Packs cells into CLBs
- Separately for each MS
- Purely timing-driven
- Delay model includes
  - Position of CLB pins
  - Direct connections
  - Single-length lines
  - Switch matrices

## Design Integration

- **Assembly of**
  - **Linear datapath (by SDI)**
    - Unmodified “hard” modules
    - Compacted “soft” modules
  - **Controller (standard tools)**
- **Common final routing**
  - **Xilinx PPR**

- **Complete chip**
  - **Regular datapath**
  - **Irregular controller**

# Experimental Results



- SDI vs. standard approach (XACT 5.2.1)
  - Delays: shorter, less variance
  - Runtime: shorter (even w/o parallelism)

## Further Research

- Flexible open API between all system components
- Integration with HL synthesis tools
  - Automatic datapath recognition
- Automatic inter-module pipelining
- Increased emphasis on routability in
  - Partitioning
  - Micro placement
- Allow gradual generator refinement
  - Re-use compaction technology

## Summary

- Evolution of datapath design flow
- Structured Design Implementation (SDI)
  - Enhanced module generators
  - Floorplanner
  - Structure Extraction / Regularity Analysis
  - Compaction
- ➤ Reliable generation of fast circuits
- Further work

http://www.icsi.berkeley.edu/~akoch/research.html