HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array

William Tsu, Kip Macy, Atul Joshi, Randy Huang,
Norman Walker, Tony Tung, Omid Rowhani, Varghese George,
John Wawrzynek, and André DeHon

BRASS Project
University of California at Berkeley

Myth

FPGAs inherently run at an order of magnitude lower clock rates than microprocessors.

What's in a Clock Cycle

- FPGA cycle times are elusive
  - cycle not defined by architecture
  - varies almost continuously based on routing
  - makes timing difficult
- Processor cycles are well defined
  - cycle defined by architecture
  - all operations quantized to this cycle
  - for all applications => run processor at cycle

Defining a Cycle

- Pick a target clock cycle
- Define what happens in a clock cycle based on that
  - how much computation
  - how much interconnect
- Assemble computation by combining cycles
  - ...you were paying for the delay anyway...

Don't Believe It!

- Example: XC4000XL-09 (0.35μm)
  - Minimum clock low/high 2.3ns → 4.6ns cycle
  - Composing:
    - clock→Q 1.5ns
    - interconnect budget 1.5ns
    - logic→clock setup 1.6ns

      4.6ns

Also: Von Herzen FPGA97, XC3100-09 → 4ns

Cycle Comparison

Design	Feature	Cycle
XC4000XL-09	0.35μm	4.6 ns
A10K100A-1	0.35μm	5.0 ns
Strong Arm	0.35μm	5.0 ns
Alpha(96)	0.35μm	2.3 ns
SPARC	0.35μm	3.0 ns
Pentium	0.35μm	3.3 ns
Alpha(97)	0.35μm	1.7 ns
HSRA	0.40μm	4.0 ns

FPGA cycles comparable to contemporary microprocessors.

Outline

- FPGA cycle times
- Why low frequency?
- Architecture and CAD for high frequency
- HSRA
- Experiments
- Assessment

Why FPGA designs run slowly?

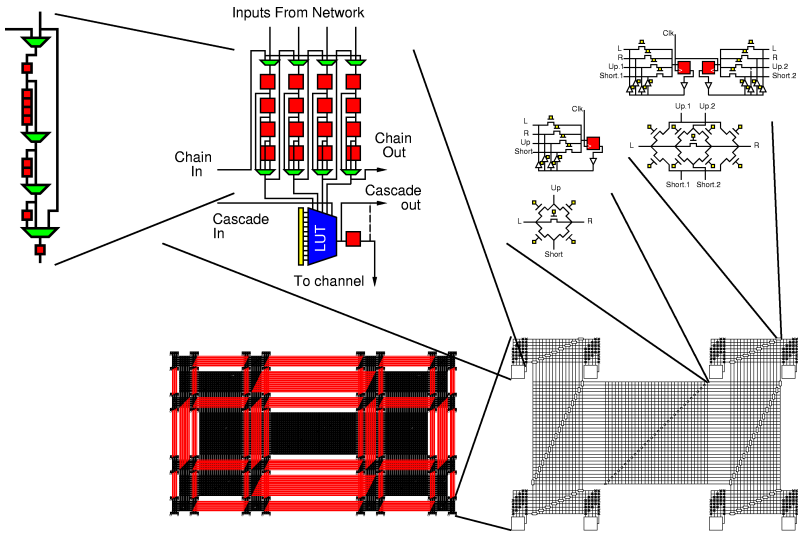
Few designs run at 200+MHz...

1. Limited application/user requirements
2. Cyclic data dependencies
3. Poor tool support
4. Long interconnect delays
5. Pipelining expensive?

HSRA

- High-Speed, Hierarchical Synchronous Reconfigurable Array
- Attacks architecture and CAD impediments
  - pipeline the interconnect (4)
  - balance retiming resources (5)
  - CAD for auto retiming (3)

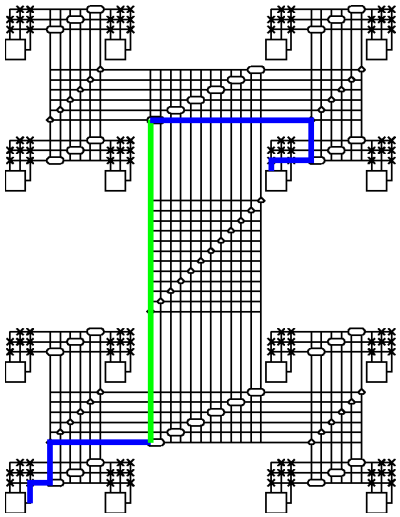
HSRA Architecture



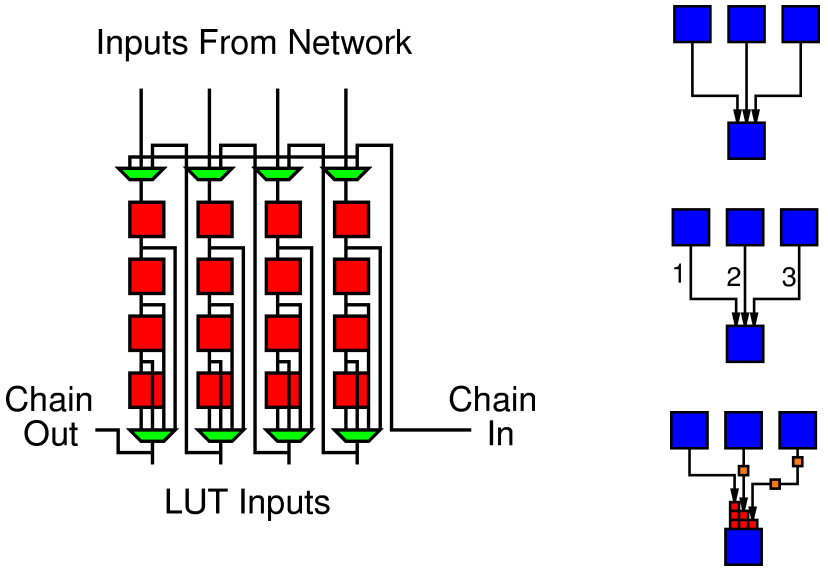
HSRA

- 5-LUT with 5th input hardwired to neighbor
  - (can be used 4-input, 2-output LUT w/ some restrictions)
- Flip-flop bank on inputs for retiming
- Hierarchical Interconnect
- Fixed clock cycle (0.4μm = 4ns)
- Pipelined Interconnect

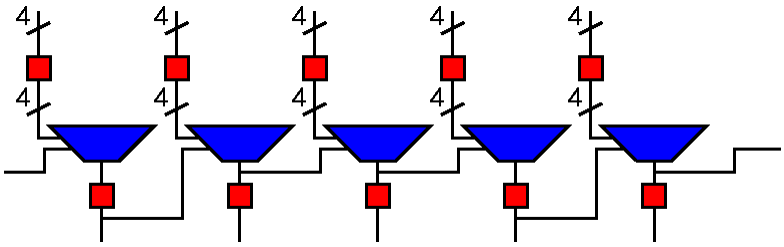
Pipelined Interconnect



Input Retiming



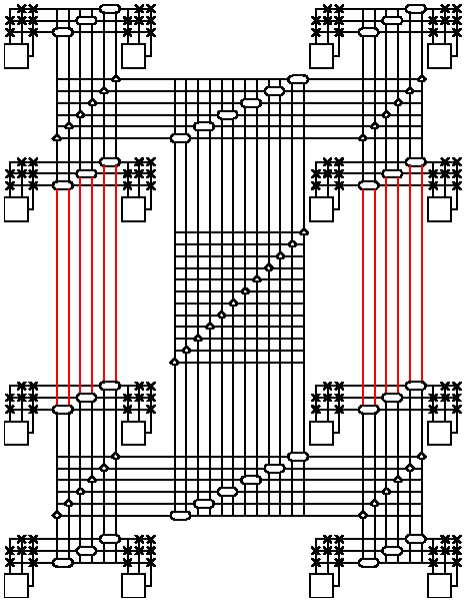
Balancing Logic Evaluation Cycle (BLB Cascade Timing)



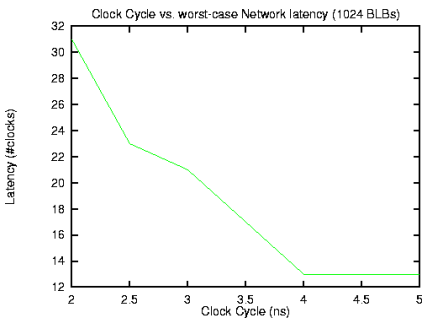
Hierarchical Interconnect

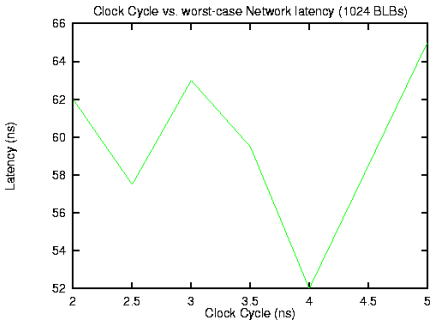
Fat-Tree/Fat-Pyramid inspired network; Geometric bandwidth growth toward root.

(Parameterized growth allows exploration/tuning.
=>Our recent study suggests p=0.6 good for “random logic”)



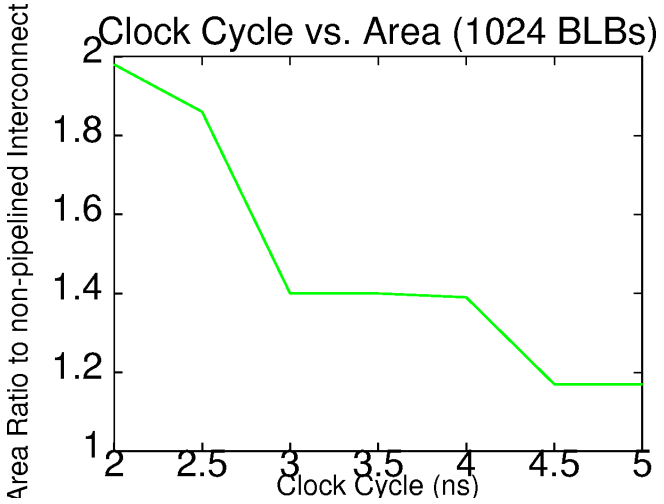
What Cycle?





Data from 0.4µm DRAM Process

Area vs. Cycle



Flop Experiment #1

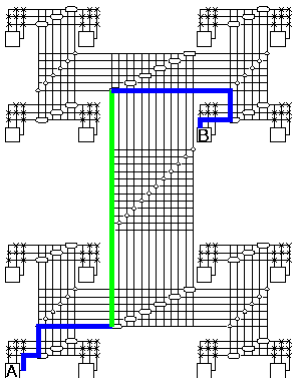
- Pipeline and retime to single LUT delay per cycle
  - MCNC benchmarks to 256 4-LUTs
  - no interconnect accounting

Number of Registers	1	2	3	4	5	6	7	8	9	10
Percentage	72	16	4.5	2.2	1.3	0.96	1.2	0.46	0.12	0.11

  - average 1.7 registers/LUT (some circuits 2--7)

HSRA Retiming

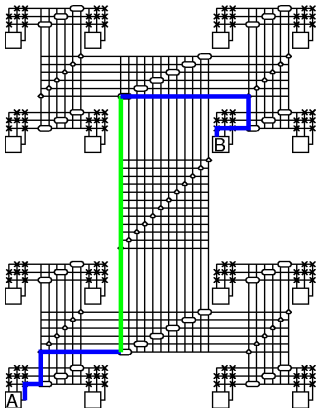
- One additional twist to retiming task
  - long, pipelined interconnect
    - ⇒ need more than one register on paths

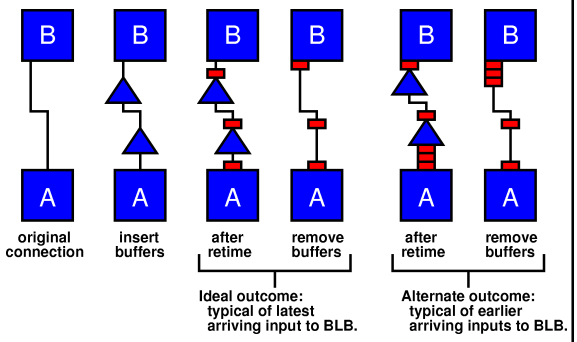


Accommodating HSRA Interconnect Delays (CAD)

- Add “logical” buffers to LUT→LUT path to match interconnect register requirements
- Reduces HSRA retiming to existing retiming problem
- Retime to C=1 as before
- Buffer chains force enough registers to cover interconnect delays

Add Interconnect Delays





Flop Experiment #2

- Pipeline and retime to HSRA cycle
  - place on HSRA
  - single LUT or interconnect domain
  - same MCNC benchmarks

Number of Registers	1	2	3	4	5	6	7	8	9	10	>10
Percentage	60	6.9	5.9	3.8	4.3	2.7	2.6	1.9	1.5	1.2	9.2

  - average 4.7 registers/LUT

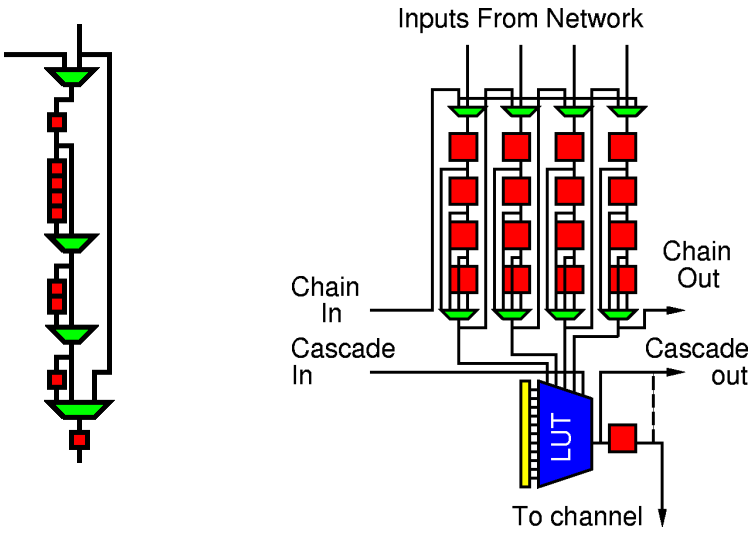
Design Question

- How deep should we make input retiming register bank?
  - Most inputs need only one (60%)
  - Some inputs need very deep (>10)
  - Average Input depth: 4.7

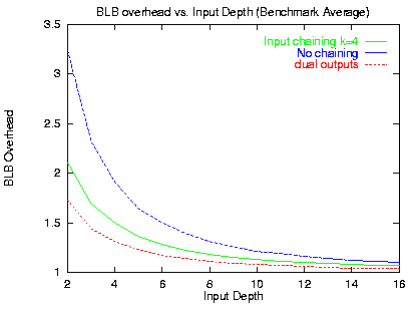
Limit Input Depth

- Experiment limiting input depths
- For each output -> input pair
  - calculate delay
  - get regs
  - if (regs-delay) > input_regs
    - allocate retiming buffer(s) to cover regs
    - share among sinks if possible

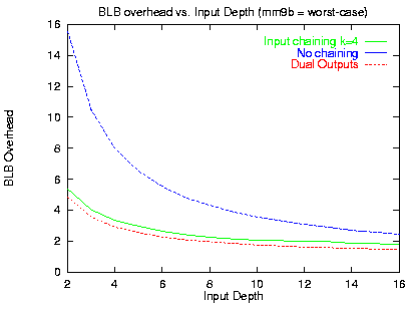
HSRA Input



Extra Blocks
(limited input depth)



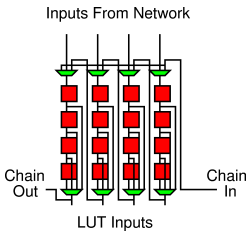
Average

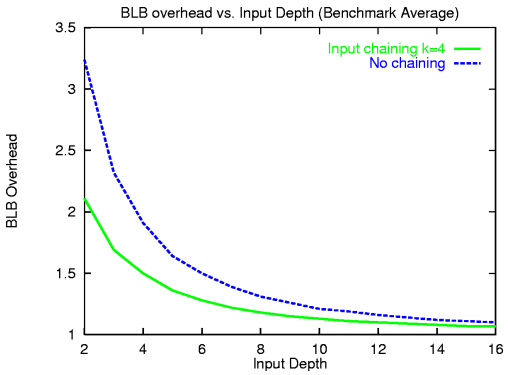


Worst Case Benchmark

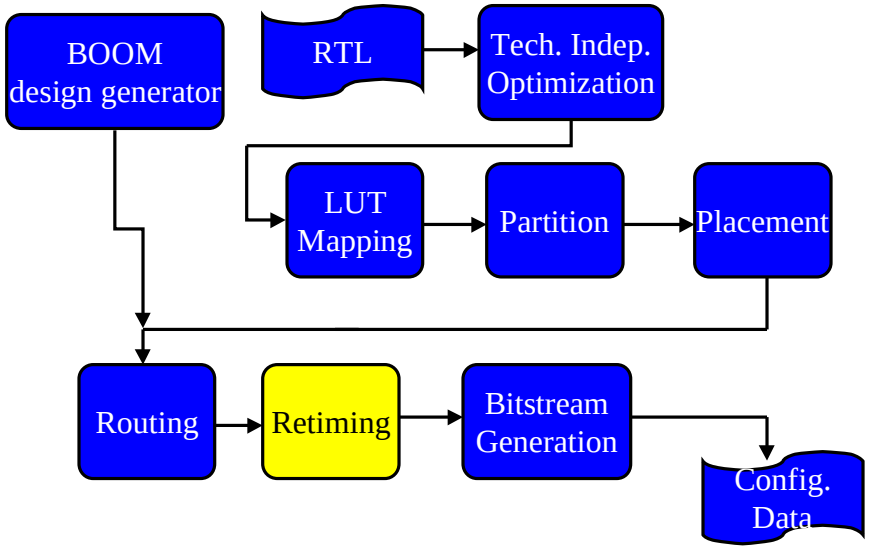
Input Depth Optimization

- Real design, fixed input retiming depth
  - truncate deeper and allocate additional logic blocks

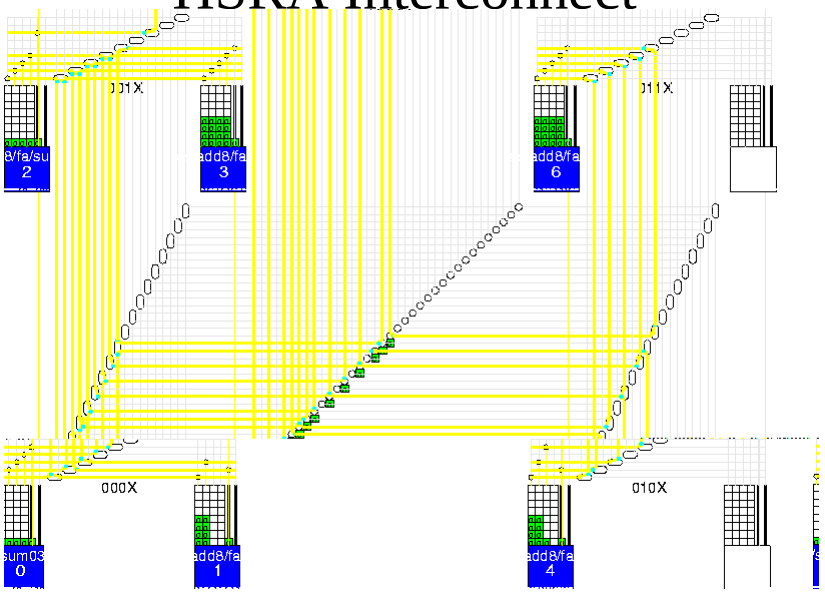




HSRA CAD Flow



HSRA Interconnect



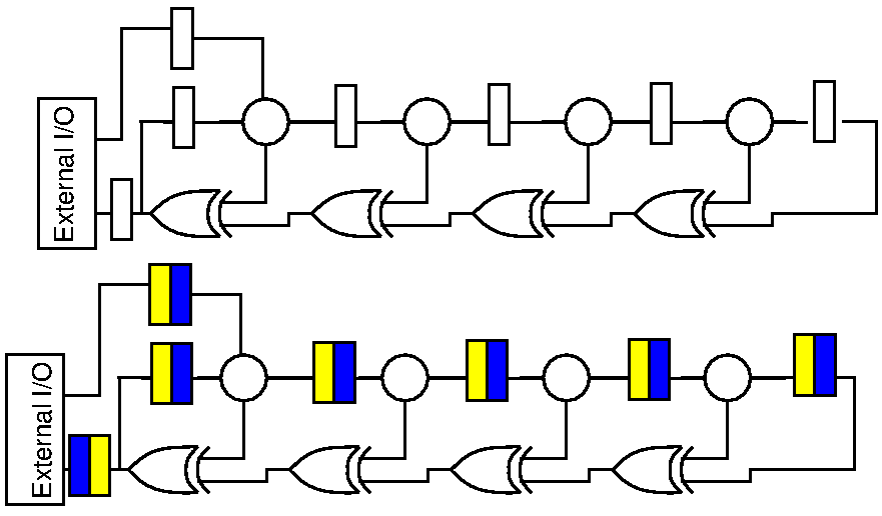
Mapping => Retiming

- Exploit technique developed for Systolic Arrays (Leiserson)
- Retime
  - find a legal movement of registers to improve circuit performance (area)
- For HSRA: retime to fully pipeline design
  - match HSRA cycle
  - justify / cover interconnect delays

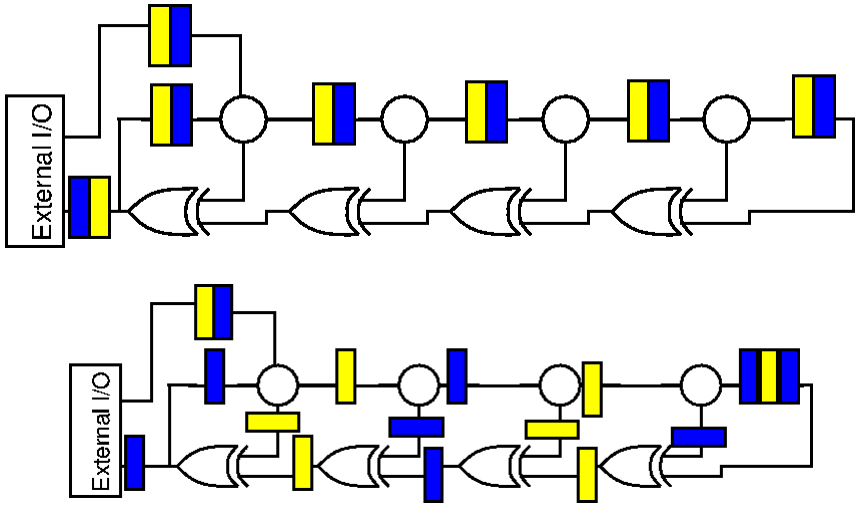
HSRA Retiming

- Automatic Mapping Attack
  - pipeline as far as possible
  - find resulting cycle, C
  - make C-slow
  - final retime
    - to distribute C-slow registers

Cycle => C-slow



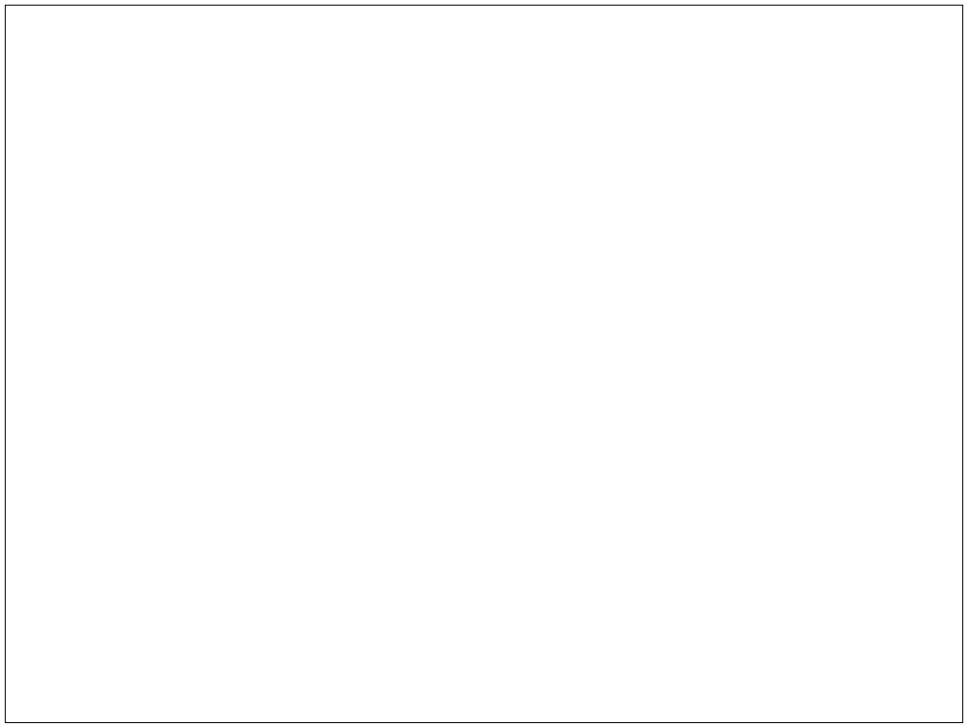
Retimed 2-Slow Cycle



C-Slow applicable?

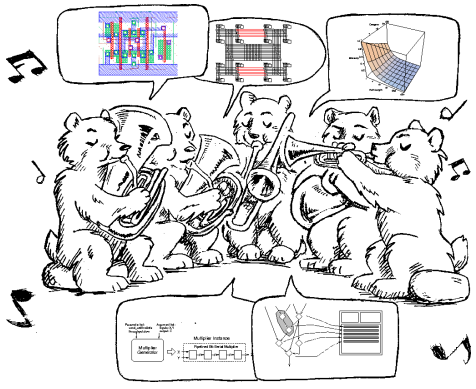
- Available parallelism
  - solve C identical, independent problems
    - e.g. process packets (blocks) separately
    - e.g. independent regions in images
- Commutative operators
  - e.g. max example

Assessment



Summary

- No inherent reasons for FPGAs/RC arrays to run slower than microprocessors
- Current FPGAs lack architectural and CAD support to reliably achieve high clock rates
- HSRA demonstrates how to attack problems
  - retiming balance
  - interconnect pipelining
  - automated retiming



Berkeley Reconfigurable Architectures Software and Systems (BRASS)

<http://www.cs.berkeley.edu/projects/brass/>