CS184a: Computer Architecture (Structure and Organization)

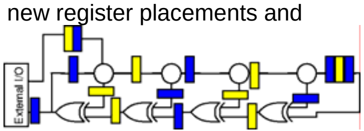
Day 20: February 27, 2005

Retiming 2:

Structures and Balance

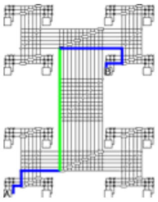
Last Time

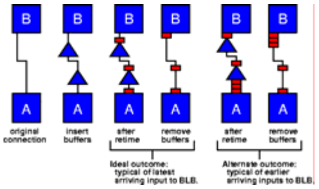
- Saw how to formulate and automate retiming:
  - start with network
  - calculate minimum achievable c
    - c = cycle delay (clock cycle)
  - make c-slow if want/need to make c=1
  - calculate new register placements and move



Last Time

- Systematic transformation for retiming
  - "justify" mandatory registers in design





Ideal outcome: typical of latest arriving input to BLB.

Alternate outcome: typical of earlier arriving inputs to BLB.

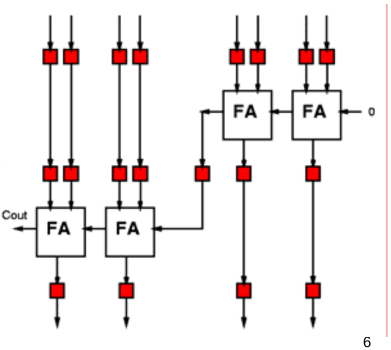
Today

- Retiming in the Large
- Retiming Requirements
- Retiming Structures

Retiming in the Large

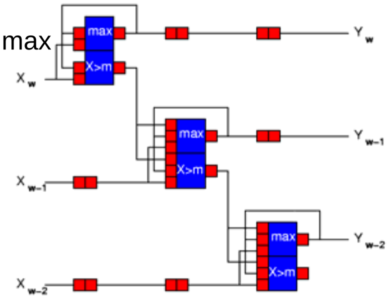
Align Data / Balance Paths

Day3:
registers
to align data



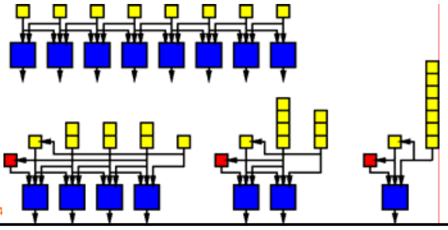
Systolic Data Alignment

- Bit-level max



Serialization

- Serialization
  - greater serialization → deeper retiming
  - **total:** same **per compute:** larger



Data Alignment

- For video (2D) processing
  - often work on local windows
  - retime scan lines
- E.g.
  - edge detect
  - smoothing
  - motion est.

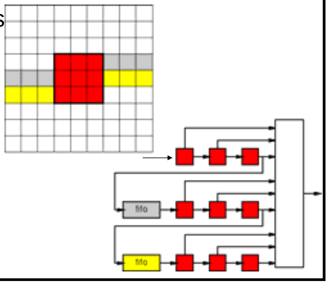
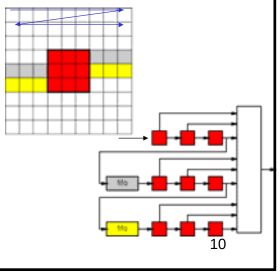


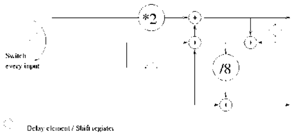
Image Processing

- See Data in raster scan order
  - adjacent, horizontal bits easy
  - adjacent, vertical bits
    - scan line apart

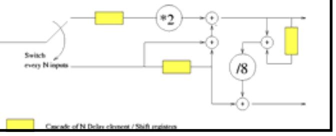


Wavelet

- Data stream for horizontal transform

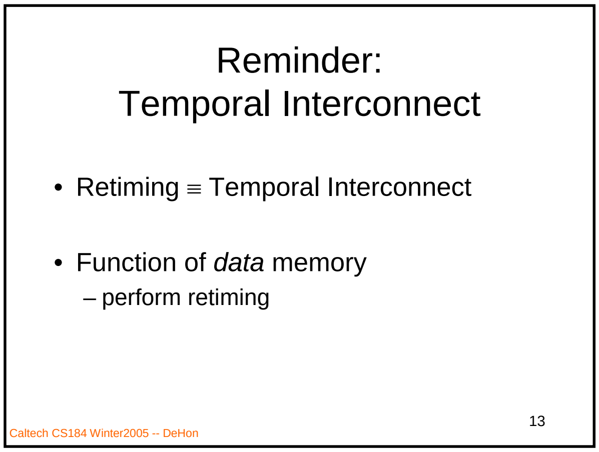


- Data stream for vertical transform
  - N=image width



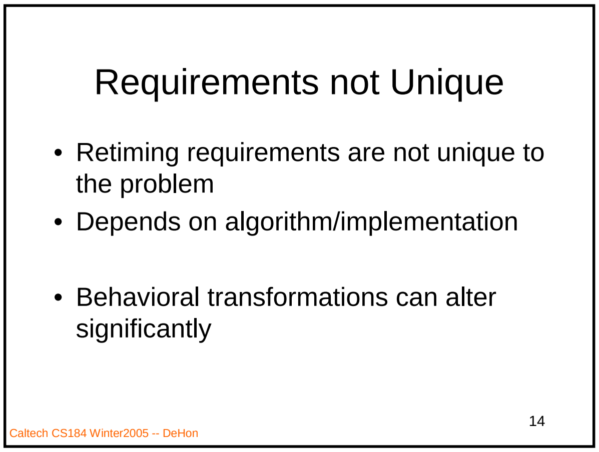
Retiming in the Large

- Aside from the local retiming for cycle optimization (last time)
- Many intrinsic needs to retime data for correct use of compute engine
  - some very deep
  - often arise from serialization



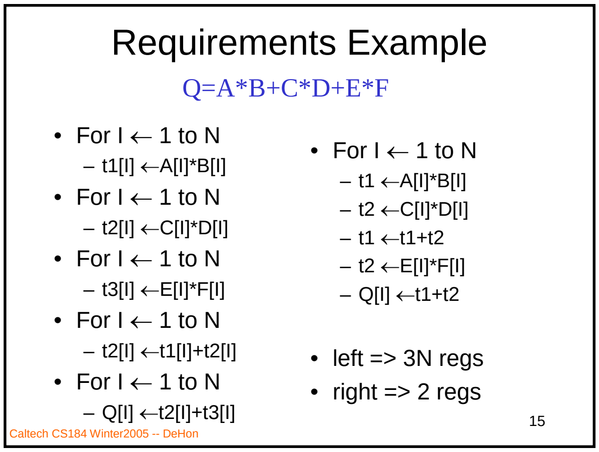
Reminder: Temporal Interconnect

- Retiming ≡ Temporal Interconnect
- Function of *data* memory
  - perform retiming



Requirements not Unique

- Retiming requirements are not unique to the problem
- Depends on algorithm/implementation
- Behavioral transformations can alter significantly



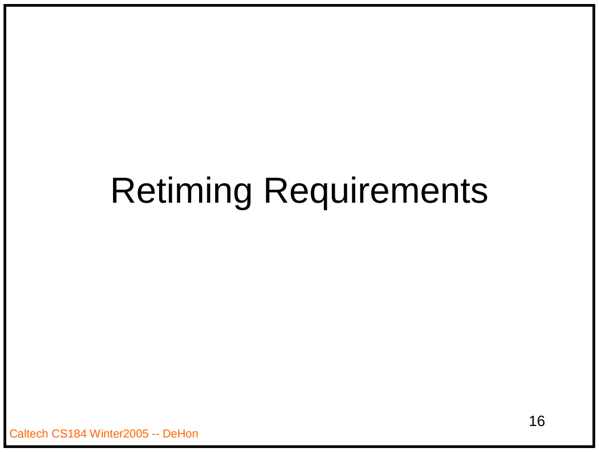
Requirements Example

Q=A*B+C*D+E*F

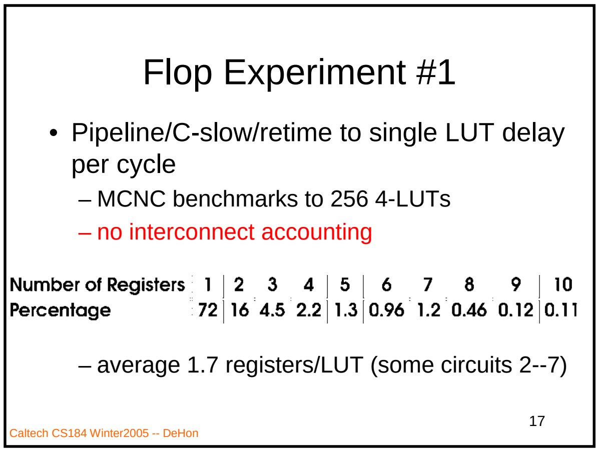
- For I ← 1 to N
  - t1[I] ←A[I]*B[I]
- For I ← 1 to N
  - t2[I] ←C[I]*D[I]
- For I ← 1 to N
  - t3[I] ←E[I]*F[I]
- For I ← 1 to N
  - t2[I] ←t1[I]+t2[I]
- For I ← 1 to N
  - Q[I] ←t2[I]+t3[I]

- For I ← 1 to N
  - t1 ←A[I]*B[I]
  - t2 ←C[I]*D[I]
  - t1 ←t1+t2
  - t2 ←E[I]*F[I]
  - Q[I] ←t1+t2

- left => 3N regs
- right => 2 regs



Retiming Requirements

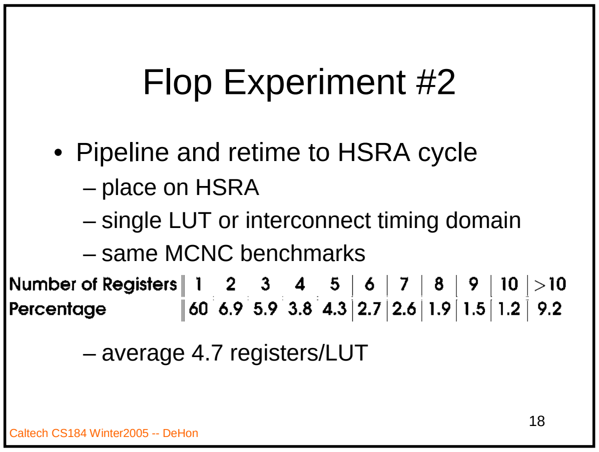


Flop Experiment #1

- Pipeline/C-slow/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)

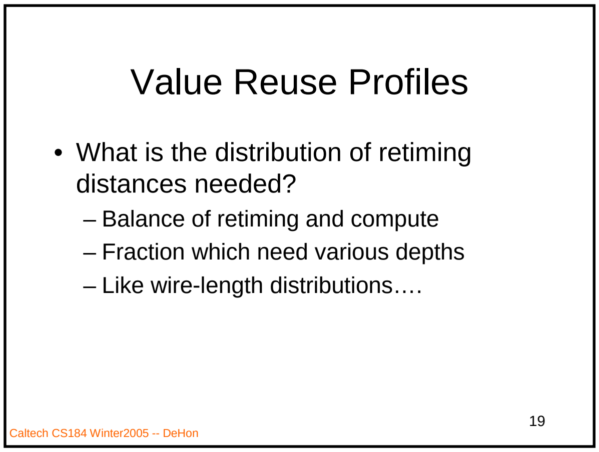


Flop Experiment #2

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

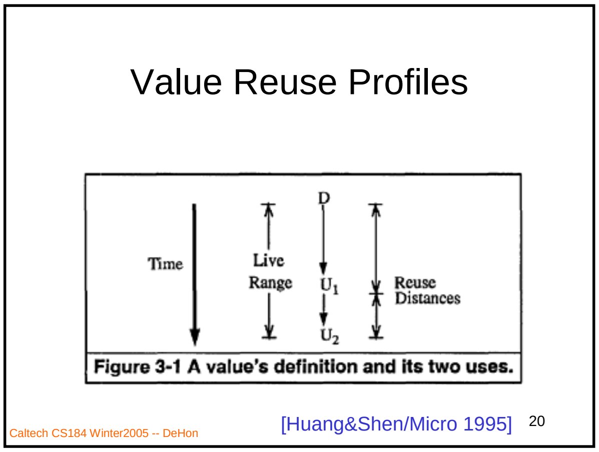
Number of Registers	1	2	3	4	5	6	7	8	9	10	>10
Percentage	60	6.9	5.9	3.8	4.3	2.7	2.6	1.9	1.5	1.2	9.2

- average 4.7 registers/LUT



Value Reuse Profiles

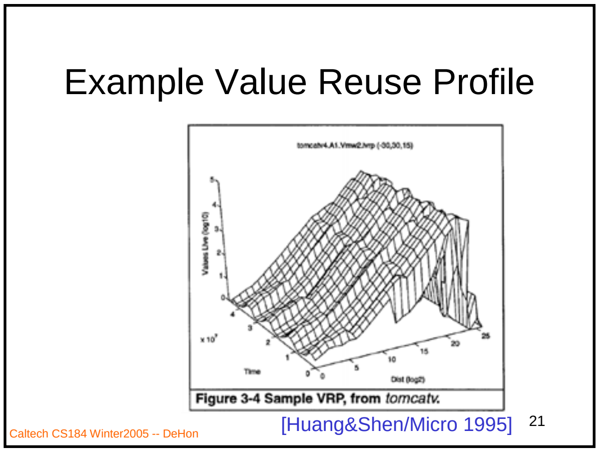
- What is the distribution of retiming distances needed?
  - Balance of retiming and compute
  - Fraction which need various depths
  - Like wire-length distributions….



Value Reuse Profiles

Figure 3-1 A value's definition and its two uses.

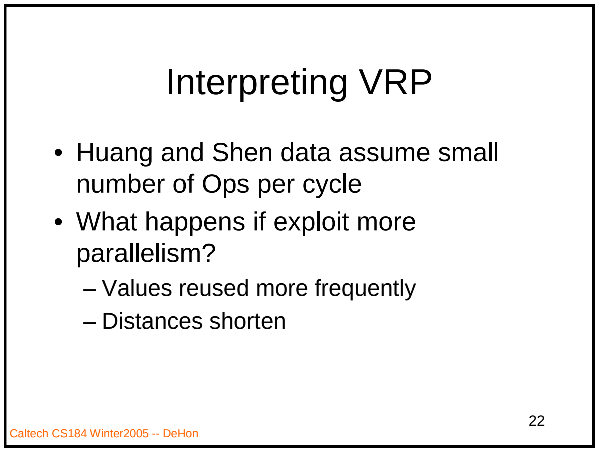
[Huang&Shen/Micro 1995]



Example Value Reuse Profile

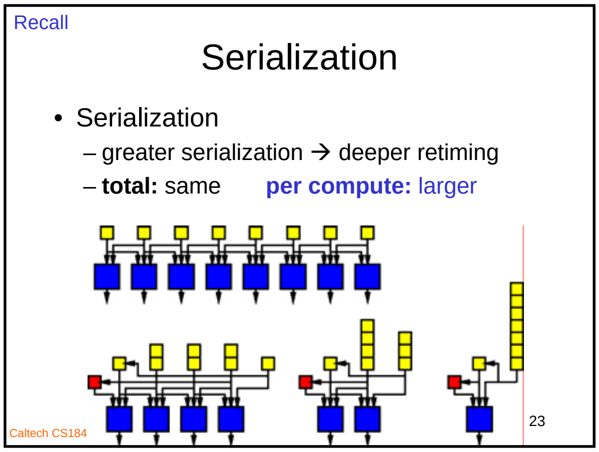
Figure 3-4 Sample VRP, from *tomcatv.*

[Huang&Shen/Micro 1995]



Interpreting VRP

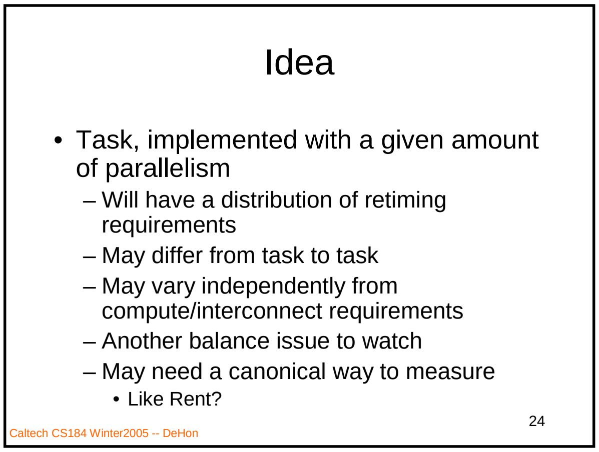
- Huang and Shen data assume small number of Ops per cycle
- What happens if exploit more parallelism?
  - Values reused more frequently
  - Distances shorten



Recall

Serialization

- Serialization
  - greater serialization → deeper retiming
  - **total:** same **per compute:** larger



Idea

- Task, implemented with a given amount of parallelism
  - Will have a distribution of retiming requirements
  - May differ from task to task
  - May vary independently from compute/interconnect requirements
  - Another balance issue to watch
  - May need a canonical way to measure
    - Like Rent?

Midpoint Admin

- Final Exercise

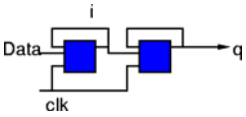
Retiming Structure

Structures

- How do we implement programmable retiming?
- Concerns:
  - Area: λ^2/bit
  - Throughput: bandwidth (bits/time)
  - Latency important when do not know when we will need data item again

Just Logic Blocks

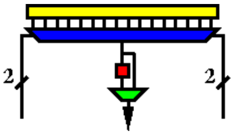
- Most primitive
  - build flip-flop out of logic blocks
    - I ←D*/Clk + I*Clk
    - Q ←Q*/Clk + I*Clk



  - Area: 2 LUTs (800K→1Mλ^2/LUT each)
  - Bandwidth: 1b/cycle

Optional Output

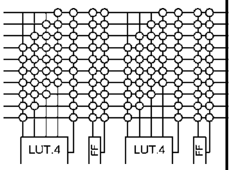
- Real flip-flop (optionally) on output



  - flip-flop: 4-5Kλ^2
  - Switch to select: ~ 5Kλ^2
  - Area: 1 LUT (800K→1Mλ^2/LUT)
  - Bandwidth: 1b/cycle

Separate Flip-Flops

- Network flip flop w/ own interconnect
  - + can deploy where needed
  - – requires more interconnect
  - + Vary LUT/FF ratio
    - Arch. Parameter



  - Assume routing ∝ inputs
    - 1/4 size of LUT
  - Area: 200Kλ^2 each
  - Bandwidth: 1b/cycle

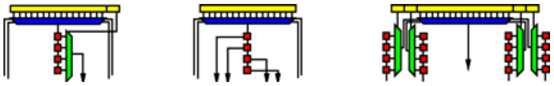
Deeper Options

- Interconnect / Flip-Flop is expensive
- How do we avoid?

Deeper

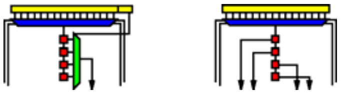
- Implication
  - don't need result on every cycle
  - number of regs>bits need to see each cycle
  - → lower bandwidth acceptable
    - → less interconnect

Deeper Retiming

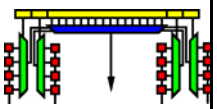


Output

- Single Output
  - Ok, if don't need other timings of signal
- Multiple Output
  - more routing

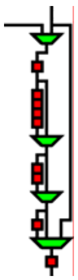


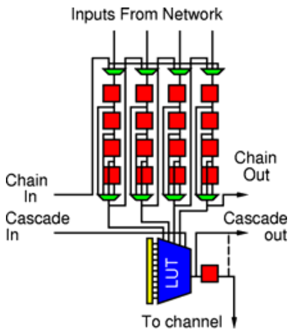
Input

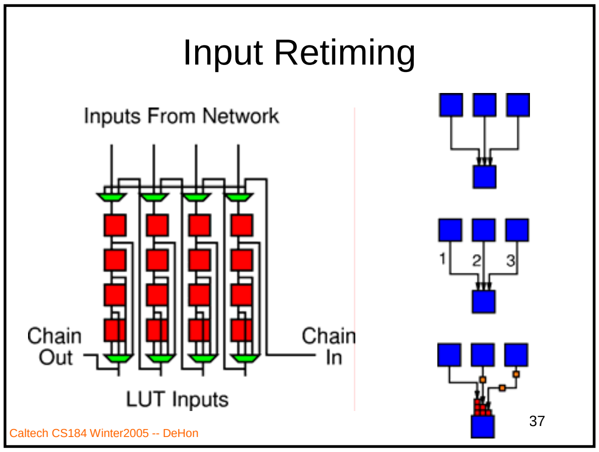


- More registers (K$\times$)
  - 7-10Kλ^2/register
  - 4-LUT => 30-40Kλ^2/depth
- No more interconnect than unretimed
  - ***open***: compare savings to additional reg. cost
  - Area: 1 LUT (1M+d*40Kλ^2) get Kd regs
    - d=4, 1.2Mλ^2
  - Bandwidth: K/cycle
    - 1/d th capacity

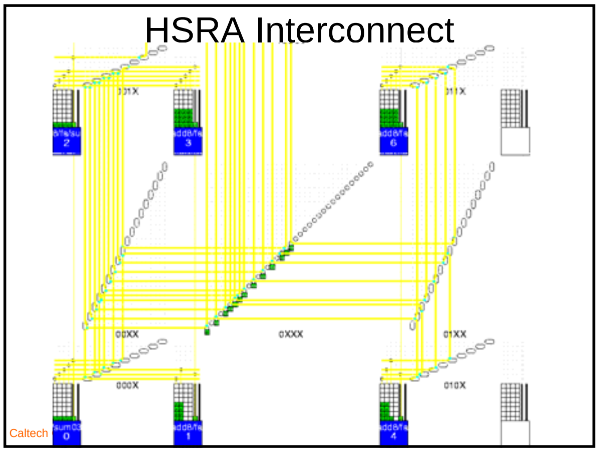
HSRA Input







Input Retiming



HSRA Interconnect

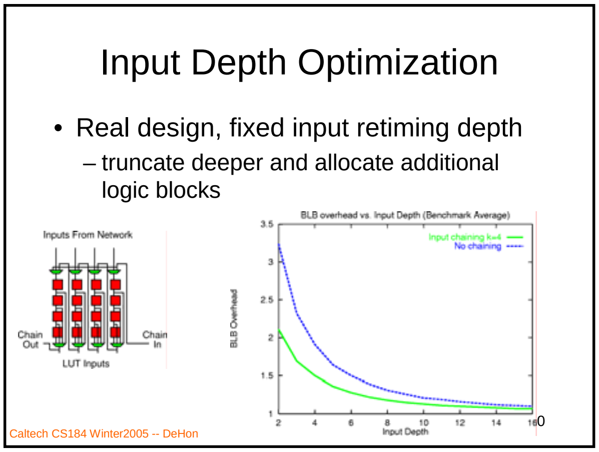
Recall

Flop Experiment #2

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

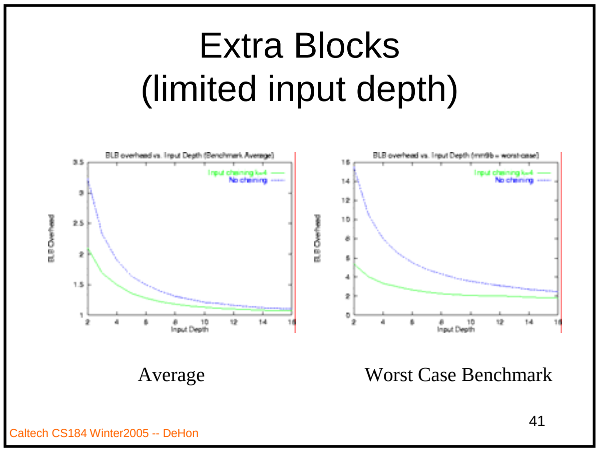
Number of Registers	1	2	3	4	5	6	7	8	9	10	>10
Percentage	60	6.9	5.9	3.8	4.3	2.7	2.6	1.9	1.5	1.2	9.2

  - average 4.7 registers/LUT



Input Depth Optimization

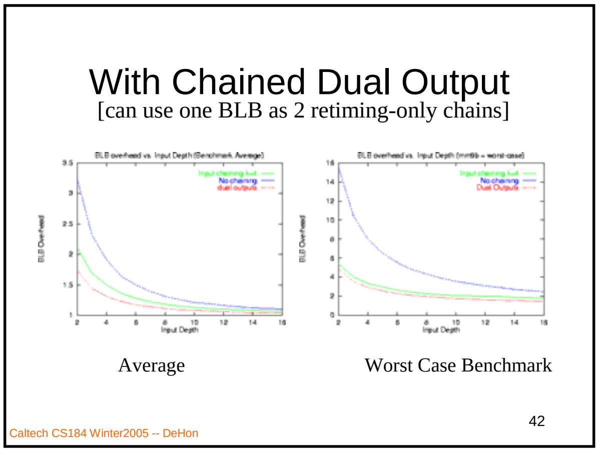
- Real design, fixed input retiming depth
  - truncate deeper and allocate additional logic blocks



Extra Blocks (limited input depth)

Average

Worst Case Benchmark



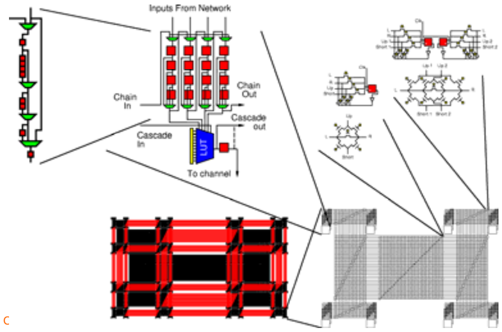
With Chained Dual Output

[can use one BLB as 2 retiming-only chains]

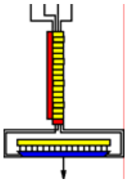
Average

Worst Case Benchmark

HSRA Architecture

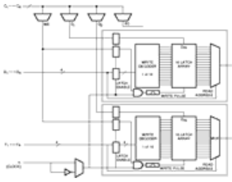


Register File



- From MIPS-X
  - 1Kλ^2/bit + 500λ^2/port
  - Area(RF) = (d+6)(W+6)(1Kλ^2+ports* 500λ^2)
- w>>6,d>>6 I+o=2 => 2Kλ^2/bit
- w=1,d>>6 I=o=4 => 35Kλ^2/bit
  - comparable to input chain
- More efficient for wide-word cases

Xilinx CLB



- Xilinx 4K CLB
  - as memory
  - works like RF
- Area: 1/2 CLB (640Kλ^2)/16≈40Kλ^2/bit
  - but need 4 CLBs to control
- Bandwidth: 1b/2 cycle (1/2 CLB)
  - 1/16 th capacity

Memory Blocks

- SRAM bit ≈ 1200λ^2 (large arrays)
- DRAM bit ≈ 100λ^2 (large arrays)
- Bandwidth: W bits / 2 cycles
  - usually single read/write
  - 1/2^A th capacity

Disk Drive

- Cheaper per bit than DRAM/Flash
  - (not MOS, no λ^2)
- Bandwidth: 60MB/s
  - For 4ns array cycle
    - ~2b/cycle@480Mb/s

Hierarchy/Structure Summary

- "Memory Hierarchy" arises from area/bandwidth tradeoffs
  - Smaller/cheaper to store words/blocks
    - (saves routing and control)
  - Smaller/cheaper to handle long retiming in larger arrays (reduce interconnect)
  - High bandwidth out of registers/shallow memories

	DRAM	SRAM	RF bit	FF/RF	RF× 1	XC	In FF	net FF	FF/LUT
λ^2	100	1200	2K	5K	40K	40K	75K	200K	800K
bw/cap.	$1/10^7$	$1/10^5$–10^3		1/100	1/100	1/16	1/4	1/1	1/1

Modern FPGAs

- Output Flop (depth 1)
- Use LUT as Shift Register (16)
- Embedded RAMs (16Kb)
- Interface off-chip DRAM (~0.1—1Gb)
- No retiming in interconnect
  - ….yet

Modern Processors

- DSPs have accumulator (depth 1)
- Inter-stage pipelines (depth 1)
  - Lots of pipelining in memory path…
- Reorder Buffer (4—32)
- Architected RF (16, 32, 128)
- Actual RF (256, 512…)
- L1 Cache (~64Kb)
- L2 Cache (~1Mb)
- L3 Cache (10-100Mb)
- Main Memory in DRAM (~10-100Gb)

Big Ideas [MSB Ideas]

- Tasks have a wide variety of retiming distances (depths)
- Retiming requirements affected by high-level decisions/strategy in solving task
- Wide variety of retiming costs
  - 100 $\lambda^2 \rightarrow 1M\lambda^2$
- Routing and I/O bandwidth
  - big factors in costs
- Gives rise to memory (retiming) hierarchy