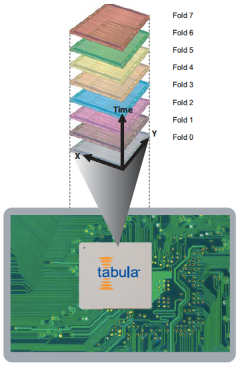
ESE534: Computer Organization

Day 22: April 9, 2012

Time Multiplexing

Tabula

- March 1, 2010
  - Announced new architecture
- We would say
  - w=1, c=8 arch.



[src: www.tabula.com]

Previously

- Saw how to pipeline **architectures**
  - specifically interconnect
  - talked about general case
- Saw how to **reuse** resources at maximum rate to do the *same* thing
- Saw demand-for and options-to-support data retiming

Today

- Multicontext
  - Review why
  - Cost
  - Packing into contexts
  - Retiming requirements
  - Some components
- [concepts we saw in overview week 2-3, we can now dig deeper into details]

How often is **reuse** of the *same* operation applicable?

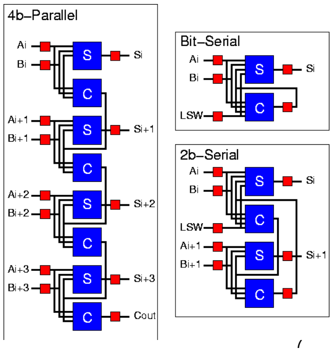
- In what cases can we exploit high-frequency, heavily pipelined operation?
- …and when can we not?

How often is **reuse** of the *same* operation applicable?

- Can we exploit higher frequency offered?
  - High throughput, feed-forward (acyclic)
  - Cycles in flowgraph
    - abundant data level parallelism [C-slow]
    - no data level parallelism
  - Low throughput tasks
    - structured (*e.g.* datapaths) [serialize datapath]
    - unstructured
  - Data dependent operations
    - similar ops [local control -- next time]
    - dis-similar ops

Structured Datapaths

- Datapaths: same *pinst* for all bits
- Can serialize and reuse the same data elements in succeeding cycles
- example: adder



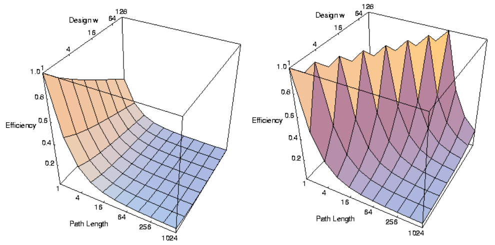
Preclass 1

- Sources of inefficient mapping W_{task}=4, L_{task}=4 to W_{arch}=1, C=1 architecture?

Preclass 1

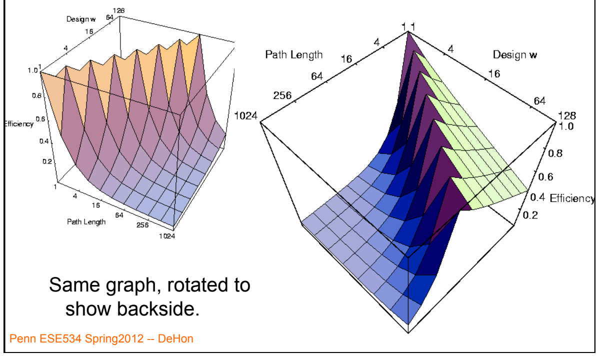
- How transform W_{task}=4, L_{task}=4 to run efficiently on W_{arch}=1, C=1 architecture?
- Impact on efficiency?

Throughput Yield



FPGA Model -- if throughput requirement is reduced for wide word operations, serialization allows us to reuse active area for same computation

Throughput Yield



Same graph, rotated to show backside.

Remaining Cases

- Benefit from multicontext as well as high clock rate
- *i.e.*
  - cycles, no parallelism
  - data dependent, dissimilar operations
  - low throughput, irregular (can’t afford swap?)

Single Context

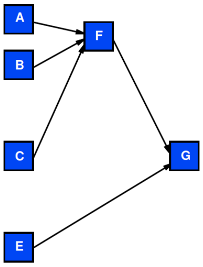
- When have:
  - cycles and no data parallelism
  - low throughput, unstructured tasks
  - dis-similar data dependent tasks
- Active resources sit idle most of the time
  - Waste of resources
- Cannot reuse resources to perform **different** function, only **same**

Resource Reuse

- To use resources in these cases
  - must direct to do different things.
- Must be able tell resources how to behave
- ➔ separate instructions (*pinsts*) for each behavior

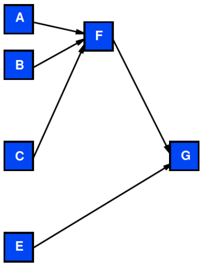
Preclass 2

- How schedule onto 3 contexts?



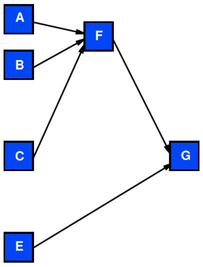
Preclass 2

- How schedule onto 4 contexts?

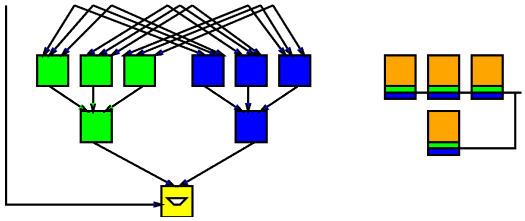


Preclass 2

- How schedule onto 6 contexts?

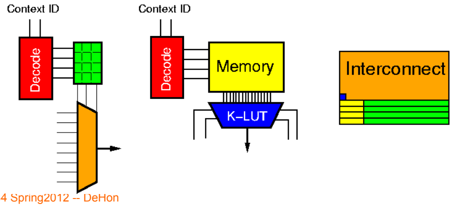


Example: Dis-similar Operations



Multicontext Organization/ Area

- A_{ctxt}≈$80K\lambda^2$
  - dense encoding
- A_{base}≈$800K\lambda^2$
- $A_{ctxt}:A_{base} = 1:10$

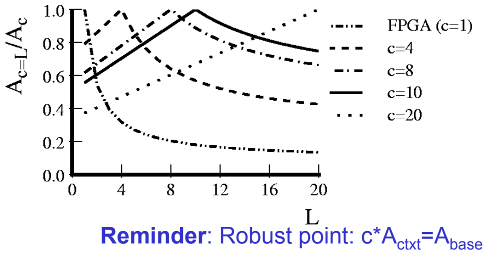


Preclass 3

- Area:
  - Single context?
  - 3 contexts?
  - 4 contexts?
  - 6 contexts?

Multicontext Tradeoff Curves

- Assume Ideal packing: $N_{active}=N_{total}/L$



Reminder: Robust point: $c*A_{ctxt}=A_{base}$

In Practice

Limitations from:

- Scheduling
- Retiming

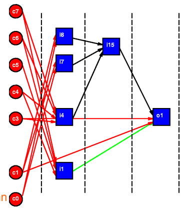
Scheduling

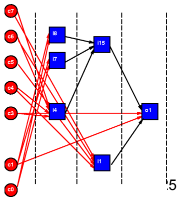
Scheduling Limitations

- N_A (**active**)
  - size of largest stage
- **Precedence**:
  can evaluate a LUT only after predecessors have been evaluated
  →cannot always, completely equalize stage requirements

Scheduling

- Precedence limits packing freedom
- Freedom do have
  - shows up as slack in network

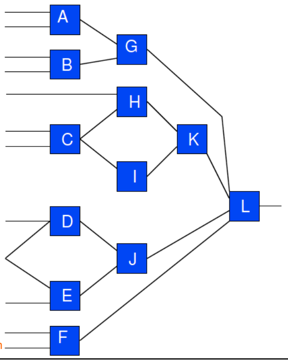




Scheduling

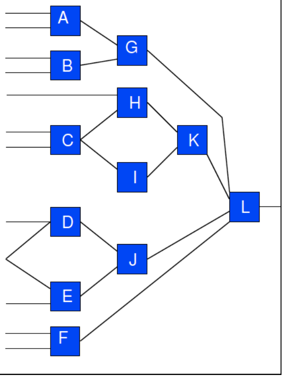
- Computing Slack:
  - ASAP (As Soon As Possible) Schedule
    - propagate depth forward from primary inputs
      - depth = 1 + max input depth
  - ALAP (As Late As Possible) Schedule
    - propagate distance from outputs back from outputs
      - level = 1 + max output consumption level
  - Slack
    - slack = L+1-(depth+level) [PI depth=0, PO level=0]

Work Slack Example



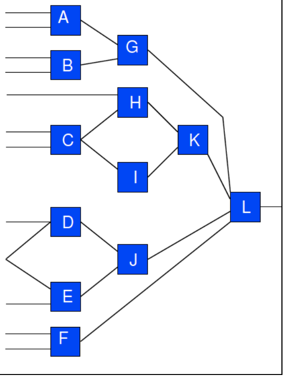
Preclass 4

- With precedence constraints, and unlimited hardware, how many contexts?



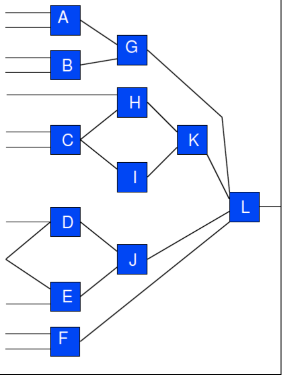
Preclass 5

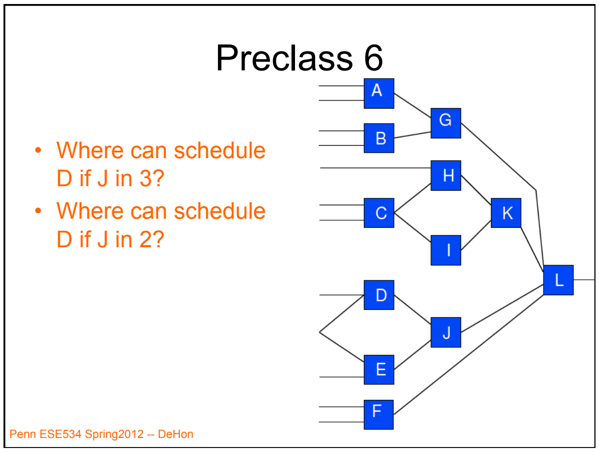
- **Without** precedence, how many compute blocks needed to evaluate in 4 contexts?



Preclass 6

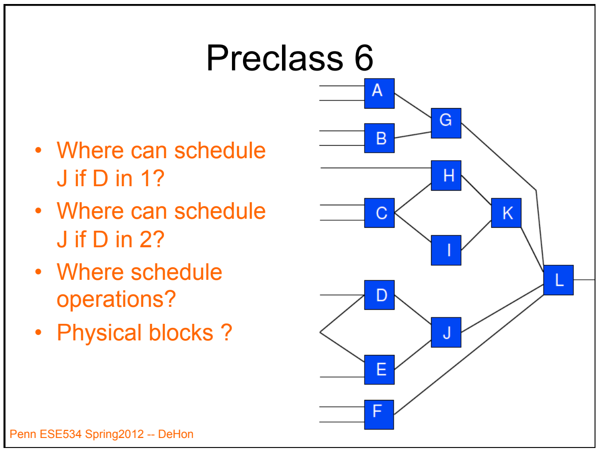
- Where can schedule?
  - J
  - D





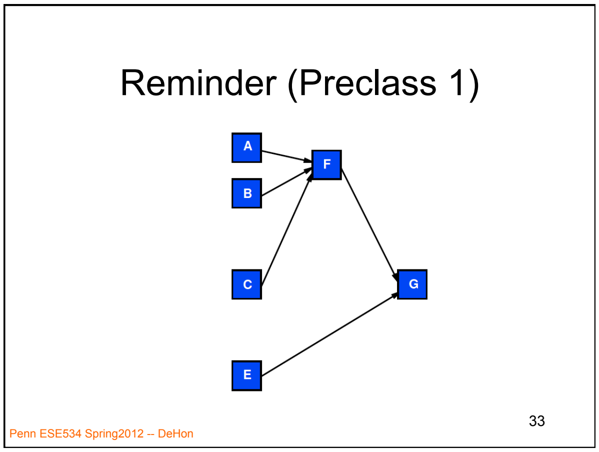
Preclass 6

- Where can schedule D if J in 3?
- Where can schedule D if J in 2?

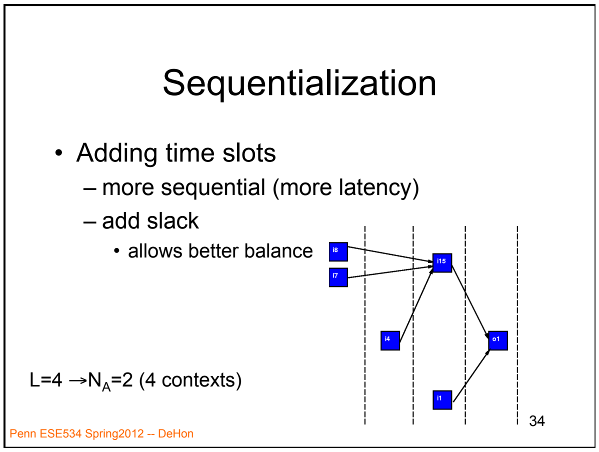


Preclass 6

- Where can schedule J if D in 1?
- Where can schedule J if D in 2?
- Where schedule operations?
- Physical blocks ?



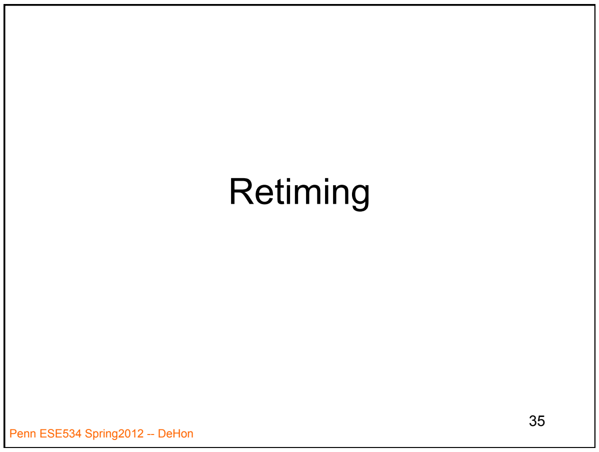
Reminder (Preclass 1)



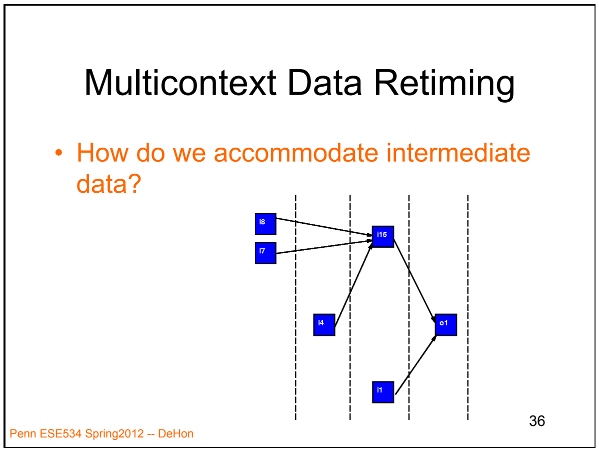
Sequentialization

- Adding time slots
  - more sequential (more latency)
  - add slack
    - allows better balance

L=4 →N_A=2 (4 contexts)



Retiming



Multicontext Data Retiming

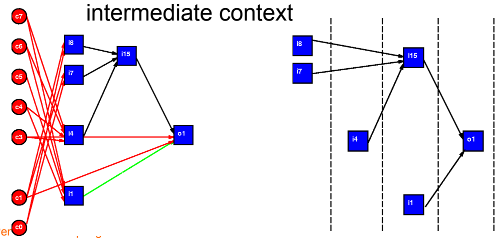
- How do we accommodate intermediate data?

Signal Retiming

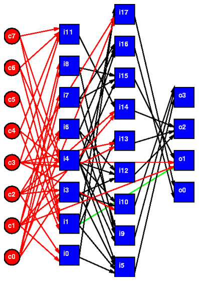
- Single context, non-pipelined
  - hold value on LUT Output (wire)
    - from production through consumption
  - Wastes wire and switches by occupying
    - for entire critical path delay L
    - not just for 1/L'th of cycle takes to cross wire segment
  - How show up in multicontext?

Signal Retiming

- Multicontext equivalent
  - need LUT to hold value for each intermediate context

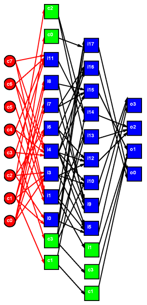


ASCII→Hex Example



Single Context: 21 LUTs @ $880K\lambda^2=18.5M\lambda^2$

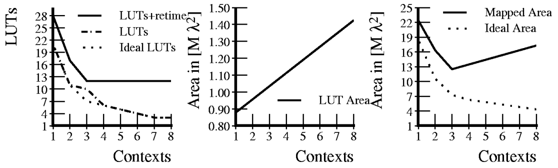
ASCII→Hex Example



Three Contexts: 12 LUTs @ $1040K\lambda^2=12.5M\lambda^2$

ASCII→Hex Example

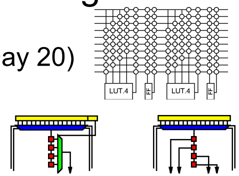
- All retiming on wires (active outputs)
  - saturation based on inputs to largest stage

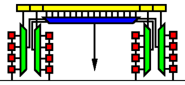


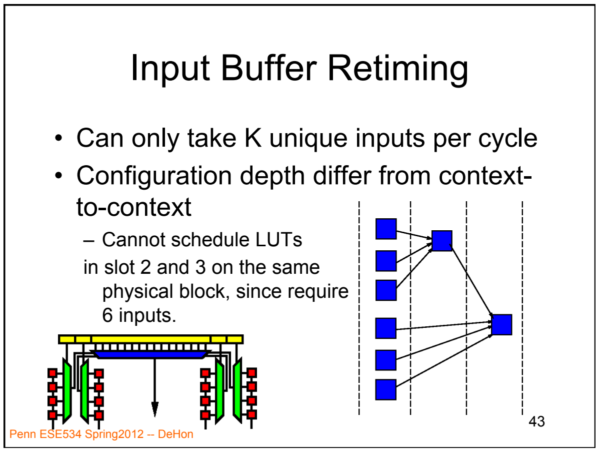
Ideal≡Perfect scheduling spread + no retime overhead

Alternate Retiming

- Recall from last time (Day 20)
  - Net buffer
    - smaller than LUT
  - Output retiming
    - may have to route multiple times
  - Input buffer chain
    - only need LUT every depth cycles

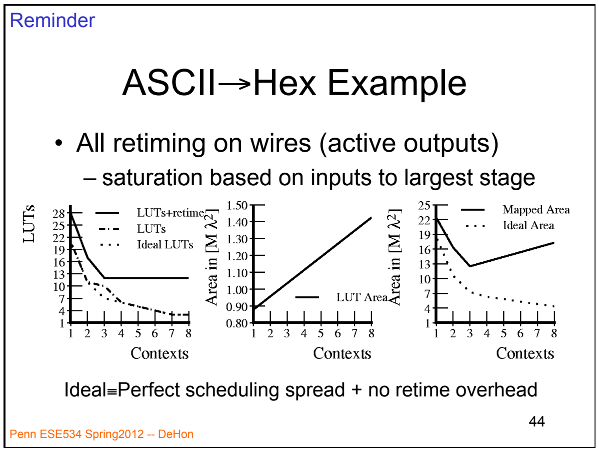






Input Buffer Retiming

- Can only take K unique inputs per cycle
- Configuration depth differ from context-to-context
  - Cannot schedule LUTs in slot 2 and 3 on the same physical block, since require 6 inputs.

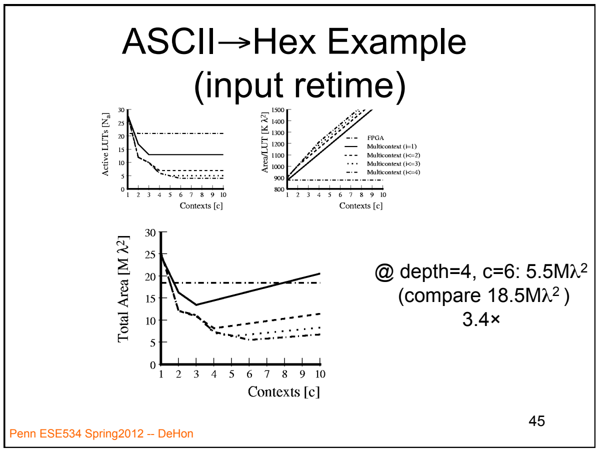


Reminder

ASCII→Hex Example

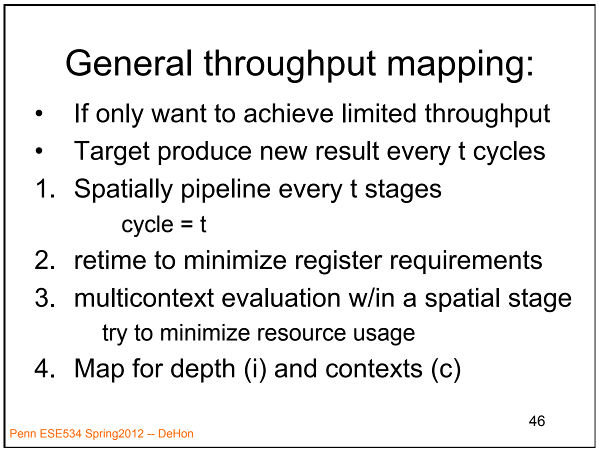
- All retiming on wires (active outputs)
  - saturation based on inputs to largest stage

Ideal≡Perfect scheduling spread + no retime overhead



ASCII→Hex Example (input retime)

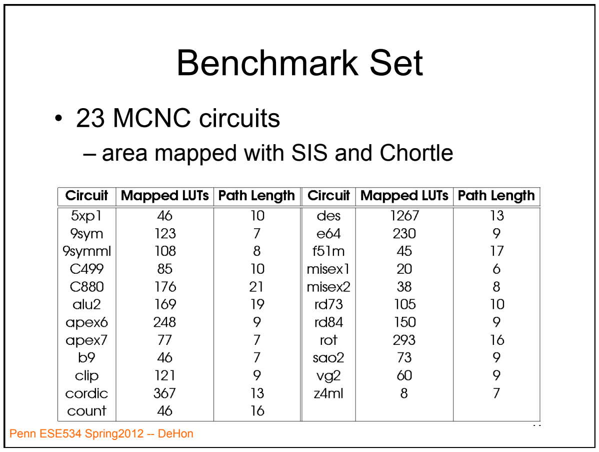
@ depth=4, c=6: 5.5Mλ^2
(compare 18.5Mλ^2)
3.4×



General throughput mapping:

- If only want to achieve limited throughput
- Target produce new result every t cycles

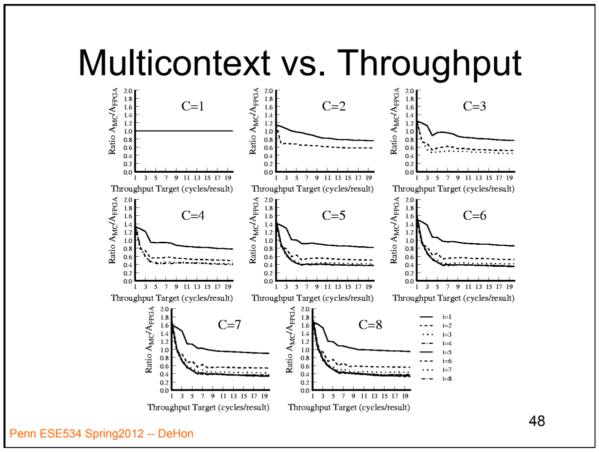
1. Spatially pipeline every t stages
   cycle = t
2. retime to minimize register requirements
3. multicontext evaluation w/in a spatial stage
   try to minimize resource usage
4. Map for depth (i) and contexts (c)



Benchmark Set

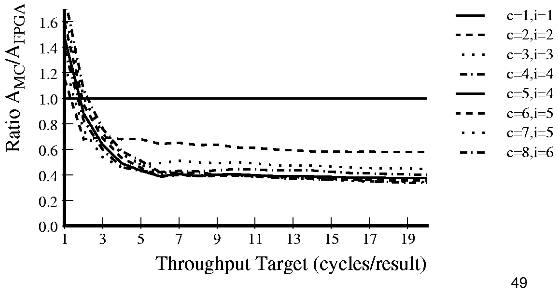
- 23 MCNC circuits
  - area mapped with SIS and Chortle

Circuit	Mapped LUTs	Path Length	Circuit	Mapped LUTs	Path Length
5xp1	46	10	des	1267	13
9sym	123	7	e64	230	9
9symml	108	8	f51m	45	17
C499	85	10	misex1	20	6
C880	176	21	misex2	38	8
alu2	169	19	rd73	105	10
apex6	248	9	rd84	150	9
apex7	77	7	rot	293	16
b9	46	7	sao2	73	9
clip	121	9	vg2	60	9
cordic	367	13	z4ml	8	7
count	46	16			



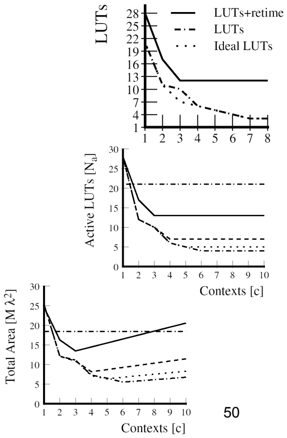
Multicontext vs. Throughput

Multicontext vs. Throughput



General Theme

- Ideal Benefit
  - *e.g.* Active=N/C
- Logical Constraints
  - Precedence
- Resource Limits
  - Sometimes bottleneck
- Net Benefit
- Resource Balance



Beyond Area

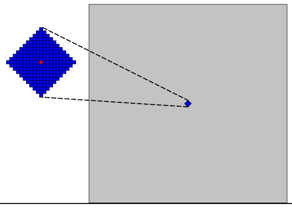
(Did not cover this section in class)

Only an Area win?

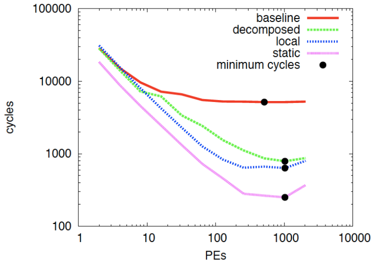
- If area were free, would we always want a fully spatial design?

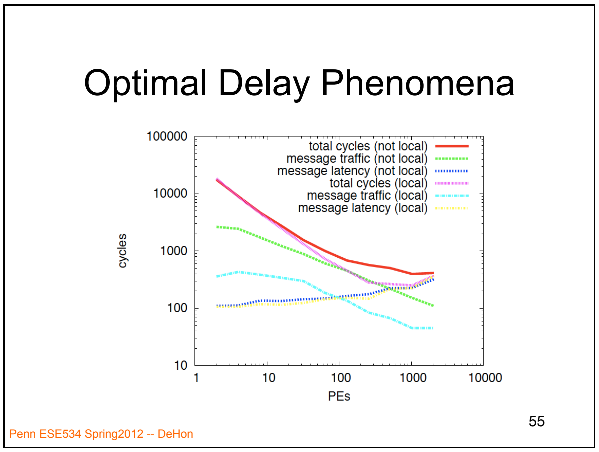
Communication Latency

- Communication latency across chip can limit designs
- Serial design is smaller → less latency

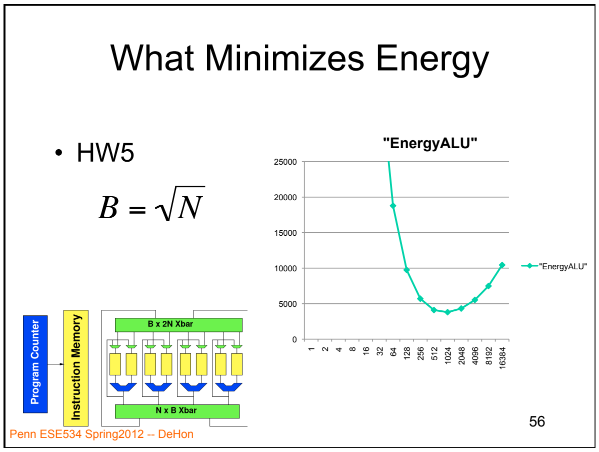


Optimal Delay for Graph App.





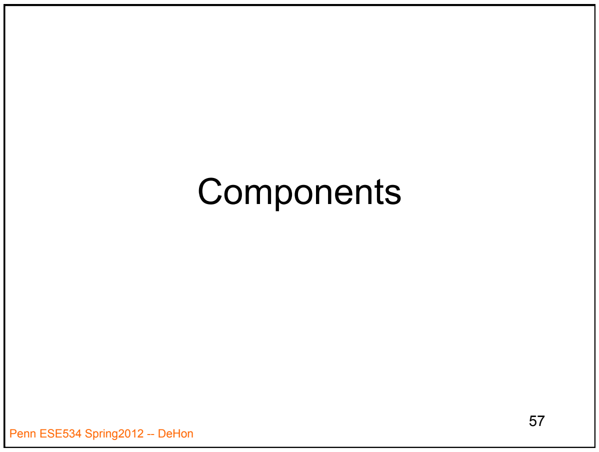
Optimal Delay Phenomena



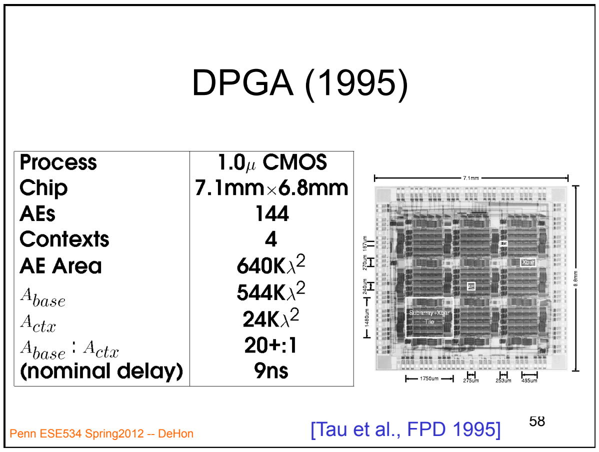
What Minimizes Energy

- HW5

$$B = \sqrt{N}$$



Components

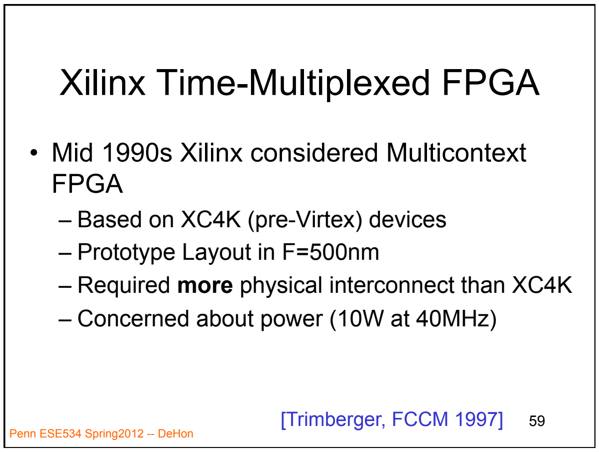


DPGA (1995)

Process	1.0μ CMOS
Chip	7.1mm$\times$6.8mm
AEs	144
Contexts	4
AE Area	640Kλ^2
A_{base}	544Kλ^2
A_{ctx}	24Kλ^2
$A_{base} : A_{ctx}$	20+:1
(nominal delay)	9ns

7.1mm
6.8mm
1750um

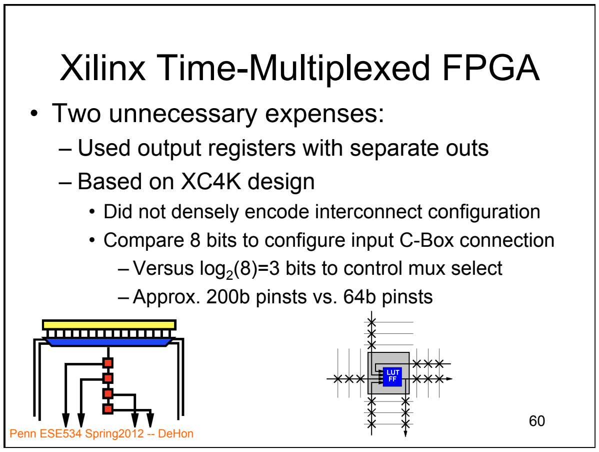
[Tau et al., FPD 1995]



Xilinx Time-Multiplexed FPGA

- Mid 1990s Xilinx considered Multicontext FPGA
  - Based on XC4K (pre-Virtex) devices
  - Prototype Layout in F=500nm
  - Required **more** physical interconnect than XC4K
  - Concerned about power (10W at 40MHz)

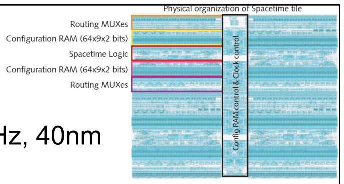
[Trimberger, FCCM 1997]



Xilinx Time-Multiplexed FPGA

- Two unnecessary expenses:
  - Used output registers with separate outs
  - Based on XC4K design
    - Did not densely encode interconnect configuration
    - Compare 8 bits to configure input C-Box connection
      - Versus $\log_2(8)=3$ bits to control mux select
      - Approx. 200b pinsts vs. 64b pinsts

Tabula


Physical organization of Spacetime tile
Routing MUXes
Configuration RAM (64x9x2 bits)
Spacetime Logic
Configuration RAM (64x9x2 bits)
Routing MUXes
Config RAM control & Clock control

- 8 context, 1.6GHz, 40nm
  - 64b pinsts
- Our model w/ input retime
  - 1Mλ^2 base
    - 80Kλ^2 / 64b pinst Instruction mem/context
    - 40Kλ^2 / input-retime depth
  - 1Mλ^2+8×0.12Mλ^2~=2Mλ^2 ➔ 4× LUTs (ideal)
    - Recall ASCIItoHex 3.4, similar for thput map
- They claim 2.8× LUTs

[MPR/Tabula 3/29/2009]

Admin

- No required reading for Wed.
  - Supplemental on web
- Office hours Tuesday
  - Good for questions about project

Big Ideas [MSB Ideas]

- Several cases cannot profitably reuse same logic at device cycle rate
  - cycles, no data parallelism
  - low throughput, unstructured
  - dis-similar data dependent computations
- These cases benefit from more than one instructions/operations per active element
- A_{ctxt}<< A_{active} makes interesting
  - save area by sharing active among instructions

Big Ideas [MSB-1 Ideas]

- Economical retiming becomes important here to achieve active LUT reduction
  - one output reg/LUT leads to early saturation
- c=4--8, I=4--6 automatically mapped designs roughly 1/3 single context size
- Most FPGAs typically run in realm where multicontext is smaller
  - How many for intrinsic reasons?
  - How many for lack of HSRA-like register/CAD support?