

ESE680-002 (ESE534): Computer Organization

Day 21: April 2, 2007
Time Multiplexing



Previously

- Saw how to pipeline **architectures**
 - specifically interconnect
 - talked about general case
- Including how to map to them
- Saw how to **reuse** resources at maximum rate to do the *same* thing

Today

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

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

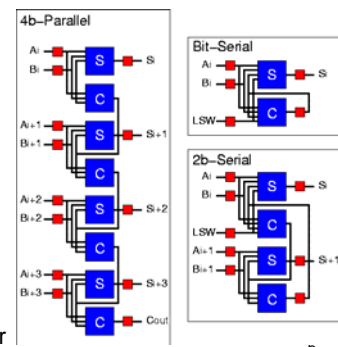
- In what cases can we exploit high-frequency, heavily pipelined operation?
- ...and when can we not?

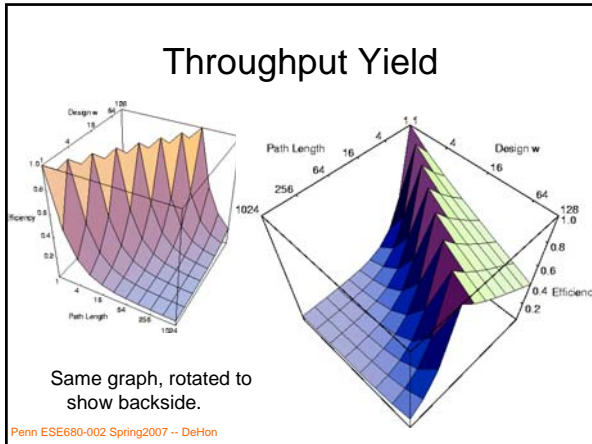
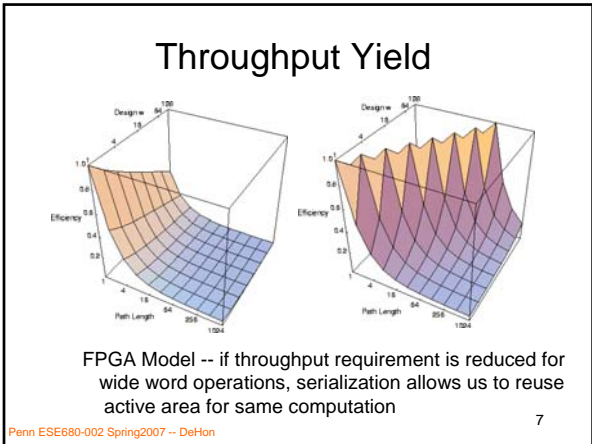
How often is **reuse** of the *same* operation applicable?

- Can we exploit higher frequency offered?
 - High throughput, feed-forward (acyclic)
 - abundant data level parallelism [C-slow]
 - no data level parallelism
 - Cycles in flowgraph
 - 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

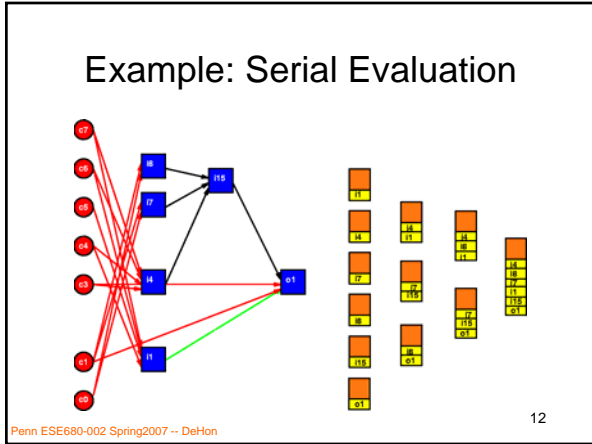




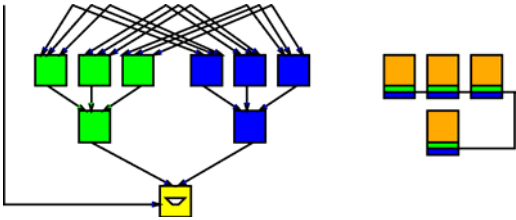
- ### Remaining Cases
- Benefit from **multicontext** as well as high clock rate
 - *i.e.*
 - cycles, no parallelism
 - data dependent, dissimilar operations
 - low throughput, irregular (can't afford swap?)
- 9

- ### 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**
- 10

- ### 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
- 11



Example: Dis-similar Operations

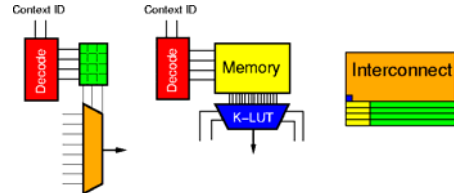


Penn ESE680-002 Spring2007 -- DeHon

13

Multicontext Organization/Area

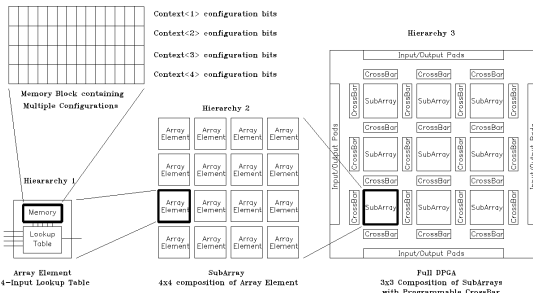
- $A_{\text{ctxt}} \approx 80K\lambda^2$
- dense encoding
- $A_{\text{base}} : A_{\text{ctxt}} = 1:10$
- $A_{\text{base}} \approx 800K\lambda^2$



Penn ESE680-002 Spring2007 -- DeHon

14

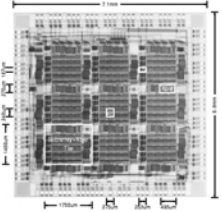
Example: DPGA Prototype



Penn ESE680-002 Spring2007 -- DeHon

Example: DPGA Area

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

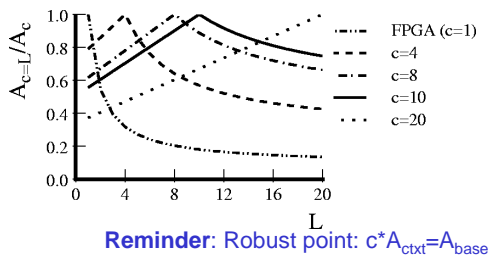


Penn ESE680-002 Spring2007 -- DeHon

16

Multicontext Tradeoff Curves

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



Penn ESE680-002 Spring2007 -- DeHon

17

In Practice

Limitations from:

- Scheduling
- Retiming

Penn ESE680-002 Spring2007 -- DeHon

18

Scheduling Limitations

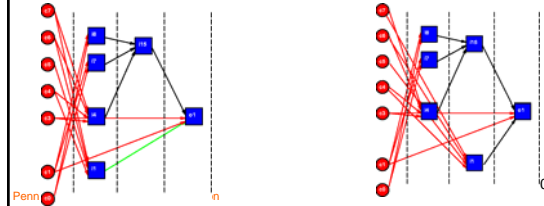
- N_A (active)
 - size of largest stage
- **Precedence:**
 - can evaluate a LUT only after predecessors have been evaluated
 - cannot always, completely equalize stage requirements

Penn ESE680-002 Spring2007 -- DeHon

19

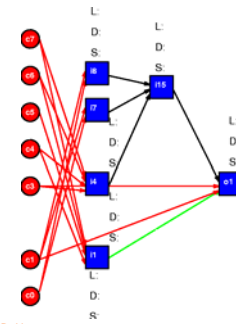
Scheduling

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



Penn

Slack Example



Penn ESE680-002 Spring2007 -- DeHon

22

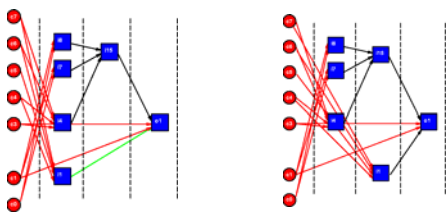
Scheduling

- **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]

Penn ESE680-002 Spring2007 -- DeHon

21

Allowable Schedules



Active LUTs (N_A) = 3

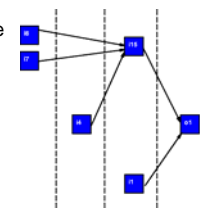
23

Penn ESE680-002 Spring2007 -- DeHon

Sequentialization

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

$L=4 \rightarrow N_A=2$ (4 or 3 contexts)



Penn ESE680-002 Spring2007 -- DeHon

24

Multicontext Scheduling

- “Retiming” for multicontext
 - **goal**: minimize peak resource requirements
 - resources: logic blocks, retiming inputs, interconnect
- NP-complete
- list schedule, anneal

Penn ESE680-002 Spring2007 -- DeHon

25

Multicontext Data Retiming

- How do we accommodate intermediate data?
- Effects?

Penn ESE680-002 Spring2007 -- DeHon

26

Signal Retiming

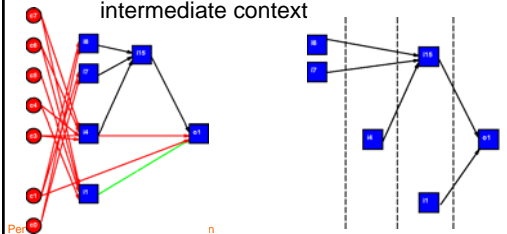
- Non-pipelined
 - hold value on LUT Output (wire)
 - from production through consumption
 - Wastes wire and switches by occupying
 - for entire critical path delay L
 - not just for 1/L'th of cycle takes to cross wire segment
 - How show up in multicontext?

Penn ESE680-002 Spring2007 -- DeHon

27

Signal Retiming

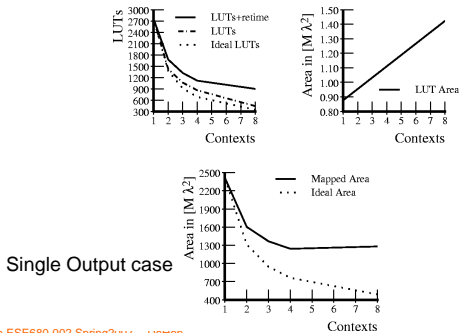
- Multicontext equivalent
 - need LUT to hold value for each intermediate context



Penn ESE680-002 Spring2007 -- DeHon

28

DES Latency Example

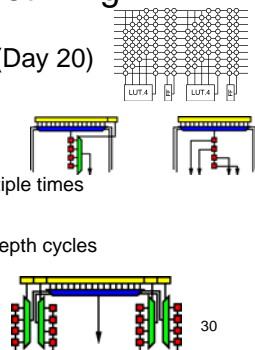


Penn ESE680-002 Spring2007 -- DeHon

29

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



Penn ESE680-002 Spring2007 -- DeHon

30

Input Buffer Retiming

- Can only take K unique inputs per cycle
- Configuration depth differ from context-to-context

Penn ESE680-002 Spring2007 -- DeHon

31

ASCII→Hex Example

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

Penn ESE680-002 Spring2007 -- DeHon

32

ASCII→Hex Example

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

Penn ESE680-002 Spring2007 -- DeHon

33

ASCII→Hex Example

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

Ideal=Perfect scheduling spread + no retime overhead

Penn ESE680-002 Spring2007 -- DeHon

34

ASCII→Hex Example (input retime)

@ depth=4, c=6: $5.5M\lambda^2$ (compare $18.5M\lambda^2$)

Penn ESE680-002 Spring2007 -- DeHon

35

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
retime (list schedule) to minimize resource usage
4. Map for depth (i) and contexts (c)

Penn ESE680-002 Spring2007 -- DeHon

36

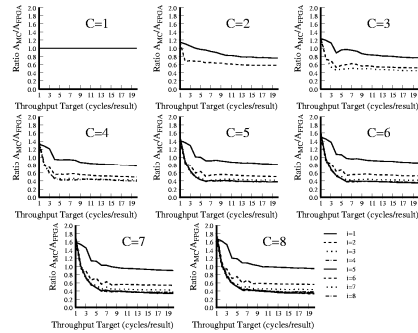
Benchmark Set

- 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			

Penn ESE680-002 Spring2007 -- DeHon

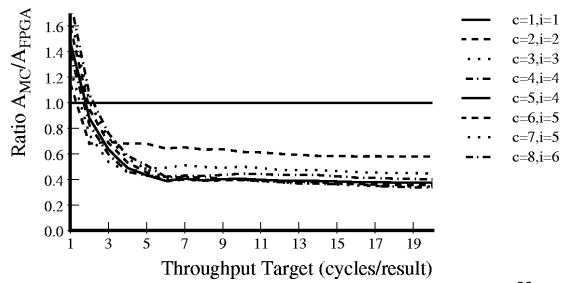
Multicontext vs. Throughput



Penn ESE680-002 Spring2007 -- DeHon

38

Multicontext vs. Throughput

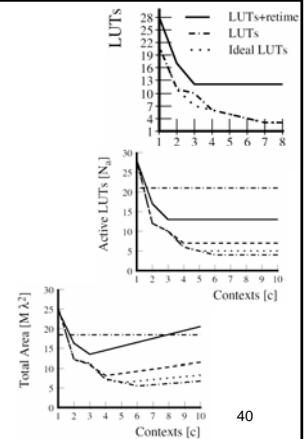


Penn ESE680-002 Spring2007 -- DeHon

39

General Theme

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



Penn ESE680-002 Spring2007 -- DeHon

40

Admin

- Assignment 8 due Wednesday
- Still swapping around lectures
 - Note reading for Wed. online
- Final Exercise out Wednesday
 - ...on everything...
 - ...but includes time multiplexing

Penn ESE680-002 Spring2007 -- DeHon

41

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
- $A_{\text{ctxt}} \ll A_{\text{active}}$ makes interesting
 - save area by sharing active among instructions

Penn ESE680-002 Spring2007 -- DeHon

42

Big Ideas [MSB-1 Ideas]

- Economical retiming becomes important here to achieve active LUT reduction
 - one output reg/LUT leads to early saturation
- $c=4-8$, $l=4-6$ automatically mapped designs 1/2 to 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?

Penn ESE680-002 Spring2007 -- DeHon

43