CS184a: Computer Architecture (Structure and Organization)

Day 11: January 31, 2005
Compute 1: LUTs

Previously

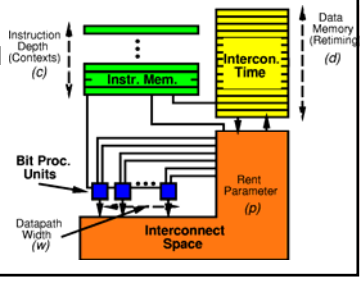
- Instruction Space Modeling
  - huge range of densities
  - huge range of efficiencies
  - large architecture space
  - modeling to understand design space
- Empirical Comparisons
  - Ground cost of programmability

Today

- Look at Programmable Compute Blocks
- Specifically LUTs Today
- Recurring theme:
  - define parameterized space
  - identify costs and benefits
  - look at typical application requirements
  - compose results, try to find best point

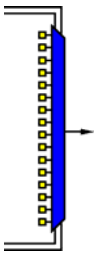
Compute Function

- What do we use for "compute" function
- Any Universal
  - NANDx
  - ALU
  - LUT

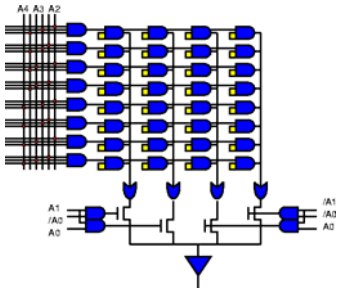


Lookup Table

- Load bits into table
  - 2^N bits to describe
  - → 2^{2^N} different functions
- Table translation
  - performs logic transform



Lookup Table

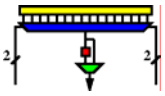


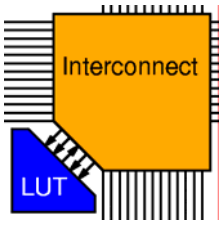
We could...

- Just build a large memory = large LUT
- Put our function in there
- What's wrong with that?

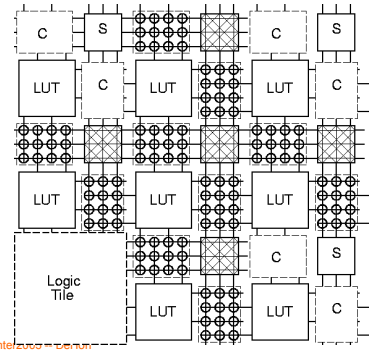
FPGA = Many small LUTs

Alternative to one big LUT





Toronto FPGA Model



What's best to use?

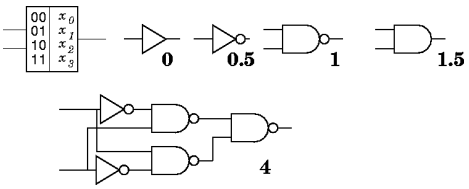
- Small LUTs
- Large Memories

- …small LUTs or large LUTs
- …or, how big should our memory blocks used to peform computation be?

Start to Sort Out: Big vs. Small Luts

- Establish equivalence
  - how many small LUTs equal one big LUT?

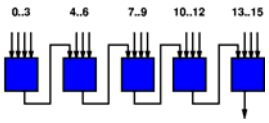
"gates" in 2-LUT ?



How Much Logic in a LUT?

- Lower Bound?
  - Concrete: 4-LUTs to implement M-LUT?
- Not use all inputs?
  - 0 … maybe 1
- Use all inputs?
  - (M-1)/3

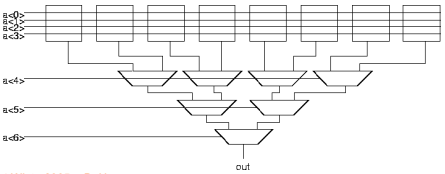
(M-1)/k for K-lut



example M-input AND
•cover 4 ins w/ first 4-LUT,
•3 more and cascade input with each additional

How much logic in a LUT?

- Upper Upper Bound:
  - M-LUT implemented w/ 4-LUTs
  - M-LUT $\leq 2^{M-4}+(2^{M-4}-1) \leq 2^{M-3}$ 4-LUTs



How Much?

- Lower Upper Bound:
  - 2^{2^M} functions realizable by M-LUT
  - Say Need n 4-LUTs to cover; compute n:
    - strategy count functions realizable by each
    - $(2^{2^4})^n \geq 2^{2^M}$
    - $n\log(2^{2^4}) \geq \log(2^{2^M})$
    - $n2^4\log(2) \geq 2^M\log(2)$
    - $n2^4 \geq 2^M$
    - $n \geq 2^{M-4}$

How Much?

- Combine
  - Lower Upper Bound
  - Upper Lower Bound
  - (number of 4-LUTs in M-LUT)

$$2^{M-4} \leq n \leq 2^{M-3}$$

Memories and 4-LUTs

- For the **most complex** functions
  - an M-LUT has ~2^{M-4} 4-LUTs

◊ SRAM 32Kx8 λ=0.6μm
  - 170Mλ^2 (21ns latency)
  - $8*2^{11}$ =16K 4-LUTs

◊ XC3042 λ=0.6μm
  - 180Mλ^2 (13ns delay per CLB)
  - 288 4-LUTs

- Memory is 50+x denser than FPGA
  - …and faster

Memory and 4-LUTs

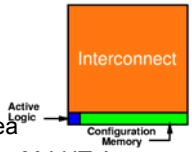
- For "regular" functions?

◊ 15-bit parity
  - entire 32Kx8 SRAM
  - 5 4-LUTs
    - (2% of XC3042 ~ 3.2Mλ^2~1/50th Memory)

◊ 7b Add
  - entire 32Kx8 SRAM
  - 14 4-LUTs
    - (5% of XC3042, 8.8Mλ^2~1/20th Memory)

LUT + Interconnect

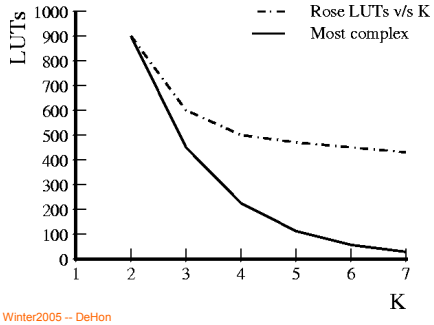
- Interconnect allows us to exploit **structure** in computation
- Already know
  - LUT Area << Interconnect Area
  - Area of an M-LUT on FPGA >> M-LUT Area
- …but most M-input functions
  - complexity << 2^M



Different Instance, Same Concept

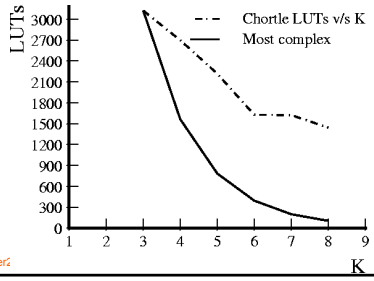
- Most general functions are huge
- Applications exhibit **structure**
- Exploit structure to optimize "common" case

LUT Count vs. base LUT size



LUT vs. K

- DES MCNC Benchmark
  - moderately irregular



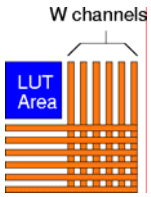
Toronto Experiments

- Want to determine best K for LUTs
- Bigger LUTs
  - handle complicated functions efficiently
  - less interconnect overhead
- Smaller LUTs
  - handle regular functions efficiently
  - interconnect allows exploitation of compute sturcture
- What's the typical complexity/structure?

Familiar Systematization

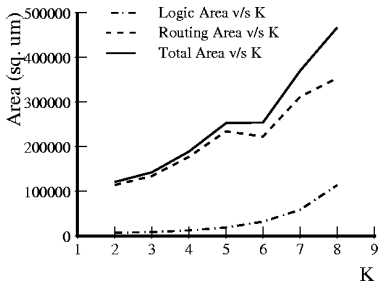
- Define a design/optimization space
  - pick key parameters
  - *e.g.* K = number of LUT inputs
- Build a cost model
- Map designs
- Look at resource costs at each point
- Compose:
  - Logical Resources⊕Resource Cost
- Look for best design points

Toronto LUT Size



- Map to K-LUT
  - use Chortle
- Route to determine wiring tracks
  - global route
  - different channel width W for each benchmark
- Area Model for K and W
  - A_{lut} exponential in K
  - Interconnect area based on switch count.

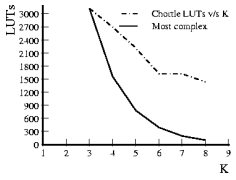
LUT Area vs. K

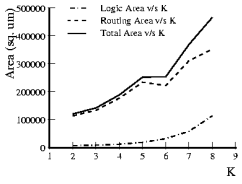


- Routing Area roughly linear in K ?

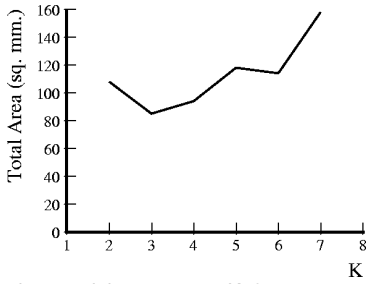
Mapped LUT Area

- Compose Mapped LUTs and Area Model





Mapped Area vs. LUT K

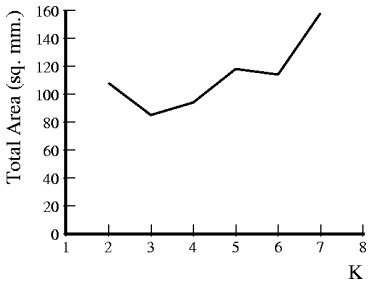


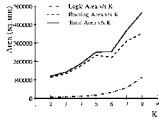
N.B. unusual case minimum area at K=3

Toronto Result

- Minimum LUT Area
  - at K=4
  - Important to note minimum on previous slides based on particular cost model
  - robust for different switch sizes
    - (wire widths)
    - [see graphs in paper]

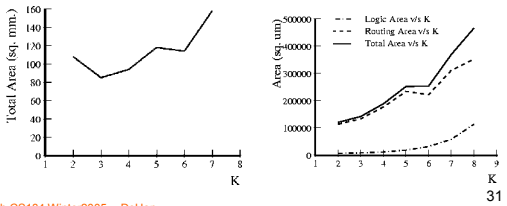
Implications





Implications

- Custom? / Gate Arrays?
- More restricted logic functions?



Relate to Sequential?

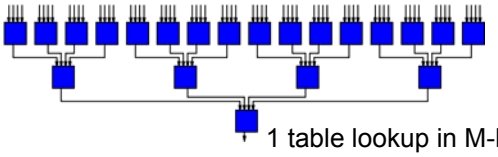
- How does this result relate to sequential execution case?
- Number of LUTs = Number of Cycles
- Interconnect Cost?
  - Naïve
  - structure in practice?
- Instruction Cost?

Delay

Back to Spatial

Delay?

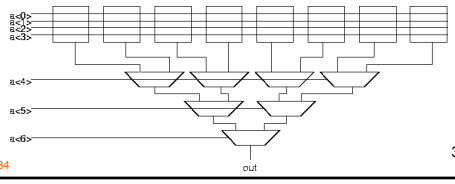
- Circuit Depth in LUTs?
- "Simple Function" → M-input AND



1 table lookup in M-LUT
$\log_k(M)$ lookups in K-LUT

Delay?

- M-input "Complex" function
  - 1 table lookup for M-LUT
  - Lower bound: $\lceil \log_k(2^{(M-k)}) \rceil + 1$
  - $\log_k(2^{(M-k)}) = (M-k)\log_k(2)$

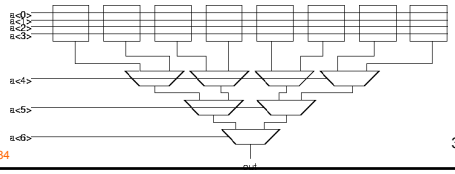


Some Math

- $Y=\log_k(2)$
- $k^Y = 2$
- $Y\log_2(k) = 1$
- $Y=1/\log_2(k)$
- $\log_k(2)=1/\log_2(k)$
- $(M-k)\log_k(2)$
- $(M-k)/\log_2(k)$

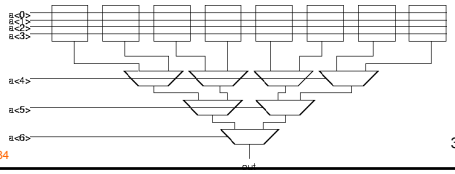
Delay?

- M-input "Complex" function
  - Lower bound: $\lceil \log_k(2^{(M-k)}) \rceil + 1$
  - $\log_k(2^{(M-k)}) = (M-k)\log_k(2)$
  - Lower Bound: $\lceil (M-k)/\log_2(k) \rceil + 1$



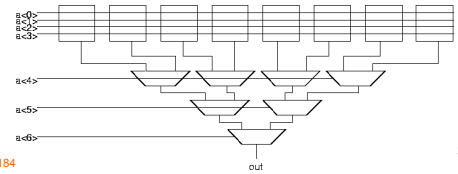
Delay?

- M-input "Complex" function
  - Upper Bound:
    - use each k-lut as a k- $\log_2(k)$ input mux
  - Upper Bound: $\lceil (M-k)/\log_2(k- \log_2(k)) \rceil + 1$



Delay?

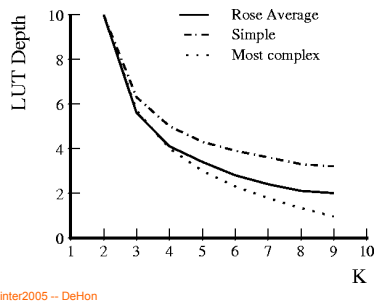
- M-input "Complex" function
  - 1 table lookup for M-LUT
  - between: $\lceil (M-k)/\log_2(k) \rceil + 1$
  - and $\lceil (M-k)/\log_2(k- \log_2(k)) \rceil + 1$



Delay

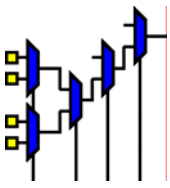
- **Simple**: log M
- **Complex**: linear in M
- Both scale as 1/log(k)

Circuit Depth vs. K



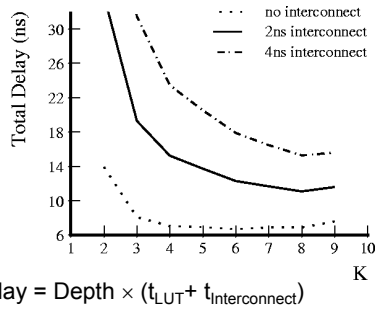
LUT Delay vs. K

- For small LUTs:
  - $t_{LUT} \approx c_0 + c_1 \times K$



- Large LUTs:
  - add length term
  - $c_2 \times \sqrt{2^K}$
- Plus Wire Delay
  - ~√area

Delay vs. K



Why not satisfied with this model?

Delay = Depth × (t_{LUT} + $t_{Interconnect}$)

Observation

- General interconnect is expensive
- "Larger" logic blocks
  - ➔ less interconnect crossing
  - ➔ lower interconnect delay
  - ➔ get larger
  - ➔ get slower
    - Happens faster than modeled here due to area
  - ➔ less area efficient
    - don't match structure in computation

Big Ideas [MSB Ideas]

- Memory most dense programmable structure for the **most complex** functions
- Memory inefficient (scales poorly) for structured compute tasks
- Most tasks have some structure
- Programmable interconnect allows us to exploit that structure

Big Ideas [MSB-1 Ideas]

- Area
  - LUT count decrease w/ K, but slower than exponential
  - LUT size increase w/ K
    - exponential LUT function
    - empirically linear routing area
  - Minimum area around K=4

Big Ideas [MSB-1 Ideas]

- Delay
  - LUT depth decreases with K
    - in practice closer to log(K)
  - Delay increases with K
    - small K linear + large fixed term
    - minimum around 5-6