ESE534: Computer Organization

Day 13: March 17, 2014
Compute 1: LUTs

Previously

• Instruction Space Modeling
  – huge range of densities
  – huge range of efficiencies
  – large architecture space
  – modeling to understand design space

Today

• Look at Programmable Compute Blocks
• Specifically LUTs
• Introduce recurring theme (methodology):
  – 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
    – NAND
    – ALU
    – LUT

Lookup Table

• Load bits into table
  – $2^N$ bits to describe
  – $\rightarrow 2^{2N}$ different functions

• Table translation
  – performs logic transform
We could...

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

How big is a k-LUT?

- k-input, 1-output?
- k-input, m-output?

FPGA = Many small LUTs
Alternative to one big LUT

What’s best to use?

- Small LUTs
- Large Memories
- ...small LUTs or large LUTs
- Continuum question: how big should our memory blocks used to perform computation be?

Memories and 4-LUTs

- SRAM 32Kx8 F=1.2μm
  - 85MF\(^2\) (21ns latency)
- XC3042 F=1.2μm
  - 90MF\(^2\) (13ns delay per CLB)
  - 288 4-LUTs
Memory and 4-LUTs

- For "regular" functions?
  - 15-bit parity
    - entire 32Kx8 SRAM
    - How many 4-LUTs?
    - 5 4-LUTs
      - (2% of XC3042 \( \sim 3.2M \times 2^{\sim 1/50th Memory} \))

- 7b Add
  - entire 32Kx8 SRAM (largest will support)
  - 14 4-LUTs
    - (5% of XC3042, 8.8M \times 2^{\sim 1/20th Memory} )

Preclass: 16-bit Adder from Memory and 3-LUTs

- How many inputs? outputs?
- Area for single large LUT?
- How many 3-LUTs?
- Area per 3-LUT?
- LUT area to implement adder with 3-LUTs?
  - Not include interconnect
- Ratio?

LUT + Interconnect

- Interconnect allows us to exploit structure in computation
- Consider addition:
  - N-input add takes
    - 2N 3-LUTs
    - one N-output (2N)-LUT
  - \( N \times 2^{(2N)} \gg 2N \times 2^3 \)
  - \( N=16; 16 \times 2^{32} \gg 2 \times 16 \times 2^3 \)
  - \( 2^{36} \gg 2^8 \rightarrow \text{factor of } 2^{28} =256 \text{ Million} \)

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-1) for K-lut

How Much?

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

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
  - \(\diamond\) SRAM 32Kx8 \(\lambda=0.6\mu m\)
    - 170M\(\lambda^2\) (21ns latency)
    - 8\(\times2^{11}\) =16K 4-LUTs
  - \(\diamond\) XC3042 \(\lambda=0.6\mu m\)
    - 180M\(\lambda^2\) (13ns delay per CLB)
    - 288 4-LUTs
- Memory is 50+x denser than FPGA and faster
Different Instance of a Familiar Concept

- The most general functions are huge
- Applications exhibit **structure**
  - Typical functions not so complex
- Exploit structure to optimize “common” case

LUT Count vs. base LUT size

- Complex: $2^{M-K}$

LUT vs. K

- DES MCNC Benchmark
  - moderately irregular

Gross Scaling Trend

- Simple: $1/K$
  - Complex: $2^K$

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 structure
- What’s the typical complexity/structure?

Standard Systematization

1. Define a design/optimization space
   - pick key parameters
   - e.g. K = number of LUT inputs
2. Build a cost model
3. Map designs
4. Look at resource costs at each point
5. Compose:
   - Logical Resources $\oplus$ Resource Cost
6. 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
  - \( A_{\text{lut}} \) exponential in K
  - Interconnect area based on switch count

LUT Area vs. K

- Routing Area roughly linear in K?

LUT Area vs. K

Interconnect ~ 20x logic

Mapped LUT Area

- Compose Mapped LUTs and Area Model

Total Area = \#k-LUTs \times \text{Area/k-LUT}

N.B. unusual case minimum area at K=3
Area vs. K (different tools)

Toronto Result

- Minimum LUT Area
  - at K=4
  - robust for different switch sizes
    - (wire widths)
    - [see graphs in paper]

Implications

Can we make more general conclusions?
- More restricted logic functions than LUTs?

Implications (Deep)

In the range the minimizes area:
- LUT area negligible compared to interconnect
- Anything less flexible than LUT will require more interconnect

Delay?

- Circuit Depth in LUTs?
- Lower bound?
  - (M-input fun using K-LUTs)
- “Simple Function” → M-input AND
  - 1 table lookup in M-LUT
  - \( \log_k(M) \) lookups in K-LUT
Delay?

- M-input "Complex" function
  - Upper Bound:
    - use each k-lut as a k \cdot \log_2(k) input mux
  - Upper Bound: \left\lceil \frac{(M-k)}{\log_2(k- \log_2(k))} \right\rceil + 1

Will not cover in class, here if want to see additional details.

Some Math

- Y = \log_b(2)
- k^Y = 2
- Y \log_b(k) = 1
- Y = 1/\log_b(k)
- \log_b(2) = 1/\log_b(k)

Will not cover in class, here if want to see additional details.

Delay?

- M-input "Complex" function
  - 1 table lookup for M-LUT
  - Lower Upper bound: \left\lceil \log_b(2^{(M-k)}) \right\rceil + 1
  - \log_b(2^{(M-k)}) = (M-k)\log_b(2)
  - Lower Upper Bound: \left\lceil \frac{(M-k)}{\log_2(k)} \right\rceil + 1

Will not cover in class, here if want to see additional details.

Delay?

- Simple: \log M
- Complex: linear in M
- Both scale with k as 1/\log(k)
**Circuit Depth vs. K**

- How LUT delay scale with $k$ for small LUTs?
  - $t_{\text{LUT}} = c_0 + c_1 \times K$

- Large LUTs:
  - add length term
  - $c_2 \times \sqrt{2^k}$

- Plus Wire Delay
  - $\sim \sqrt{\text{area}}$

**LUT Delay vs. K**

- Delay vs. K (different tools)

- Delay vs. K (proper critical path interconnect)

**Energy**

- General interconnect is expensive
- “Larger” logic blocks
  - fewer interconnect crossings
  - reduces interconnect delay
  - get larger
  - less area efficient
    - don’t match structure in computation
    - get slower
  - Happens faster than modeled here due to area
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 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

Admin
- HW4, HW5.1 graded
- Reading
  - Today’s: classic paper…**definitely read**
  - Wed. → on canvas
- Office hours Tuesday
- HW5.2 due on Wednesday