ESE534: Computer Organization

Day 13: February 29, 2012
Compute 2:
Cascades, ALUs, PLAs

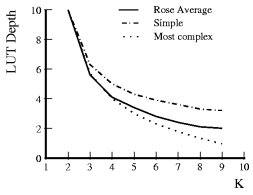
Last Time

- LUTs
  - area
  - structure
  - big LUTs vs. small LUTs with interconnect
  - design space
  - optimization

Today

- ALUs
- Cascades
- PLAs

Last Time



- Larger LUTs
  - Less interconnect delay

+ General: Larger compute blocks
  - Minimize interconnect crossings
- Large LUTs
  - Not efficient for typical logic structure

Preclass

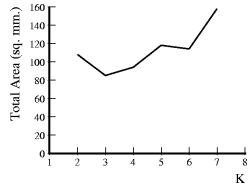
- How does addition delay compare between ALU-based and LUT-based architectures?
- What is the source of the advantage?

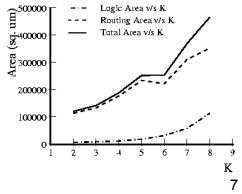
Preclass

- Advantages of ALU bitslice compute block over LUT?
- Disadvantages?

Implications from LUT-size study

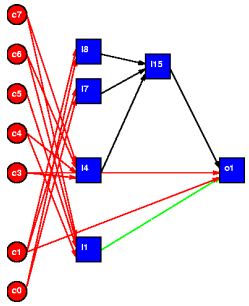
- ALU has more restricted logic functions
  - Will require more blocks





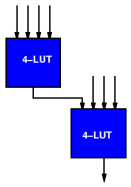
Structure in subgraphs

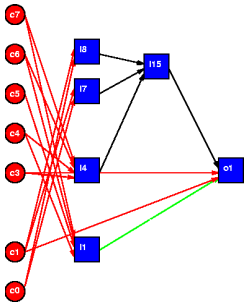
- Small LUTs capture structure
- What structure does a small-LUT-mapped netlist have?



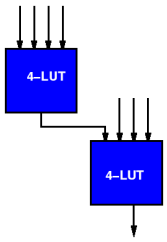
Structure

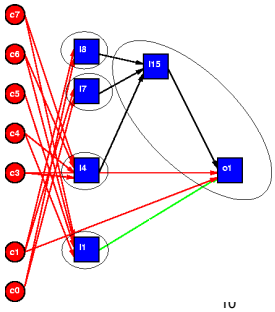
- LUT sequences ubiquitous





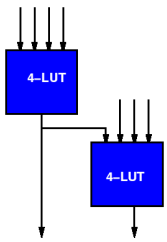
Hardwired Logic Blocks

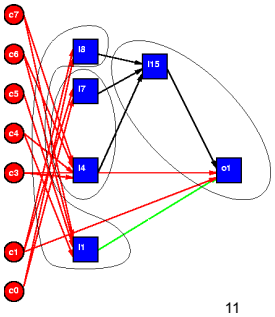




Single Output

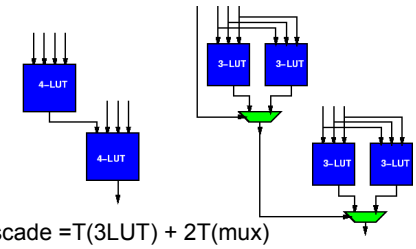
Hardwired Logic Blocks





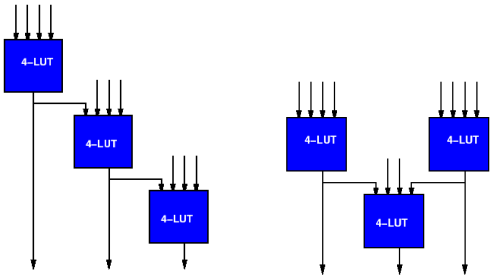
Two outputs

Delay Model



- Tcascade =T(3LUT) + 2T(mux)
- Don't pay
  - General interconnect
  - Full 4-LUT delay

Options



Chung & Rose Study

- Which most help:
  - Ripple adder?
  - Associative reduce tree?
  - Multiplier?

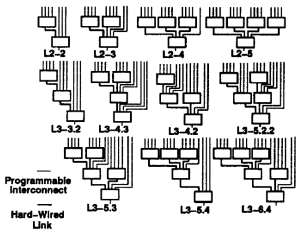


Figure 8: Delay Study HLB Topologies

[Chung & Rose, DAC '92]

Chung & Rose Study

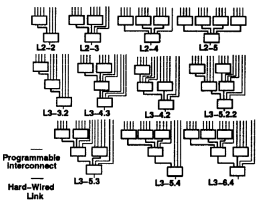


Figure 8: Delay Study HLB Topologies

Logic Block	$\overline{N_R}$	% decr in $\overline{N_R}$	$\overline{D_{tot}}$ (ns)	% decr in $\overline{D_{tot}}$
K4	5.4	0	30	0
L2-2	4.2	22	26	13
L2-3	3.4	37	22	27
L2-4	3.1	43	21	30
L2-5	3.0	44	21	30
L3-3.2	4.0	26	25	17
L3-4.2	3.0	44	21	30
L3-4.3	3.1	43	21	30
L3-5.2.2	3.1	43	21	30
L3-5.3	3.0	44	21	30
L3-5.4	2.9	46	20	33
L3-6.4	2.8	48	20	33

Table 3: Delay Performance of Different HLBs

[Chung & Rose, DAC '92]

Cascade LUT Mappings

Logic Block	$\overline{N_R}$	% decr in $\overline{N_R}$	$\overline{D_{tot}}$ (ns)	% decr in $\overline{D_{tot}}$
K4	5.4	0	30	0
L2-2	4.2	22	26	13
L2-3	3.4	37	22	27
L2-4	3.1	43	21	30
L2-5	3.0	44	21	30
L3-3.2	4.0	26	25	17
L3-4.2	3.0	44	21	30
L3-4.3	3.1	43	21	30
L3-5.2.2	3.1	43	21	30
L3-5.3	3.0	44	21	30
L3-5.4	2.9	46	20	33
L3-6.4	2.8	48	20	33

Table 3: Delay Performance of Different HLBs

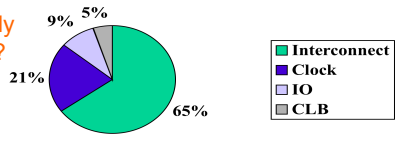
Bench Ckt	3-inp L2-3	X4000 CLB	4-inp L2-3	4-inp L3-4.2
9symml	40	36	26	19
alu2	77	71	48	37
alu4	126	122	82	61
apex7	44	38	27	20
b9	22	20	16	13
c1355	131	91	80	59
c8	21	17	15	11
cc	17	8	9	7
cm162a	8	5	5	4
comp	27	17	14	13
count	21	21	14	10
decod	10	10	8	8
mux	10	5	5	5
vda	96	97	70	52
x4ml	3	3	3	2
Tot HLBs	653	561	421	320
LUT Bits	15672	22440	20208	20480
Ratios	0.70	1	0.90	0.91
HLB pins	6530	6171	5473	5440
Ratios	1.06	1	0.89	0.88

Table 2: Area Measures of Different HLBs

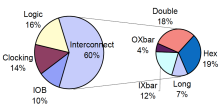
[Chung & Rose, DAC '92]

Energy Impact?

- What's the likely energy impact?

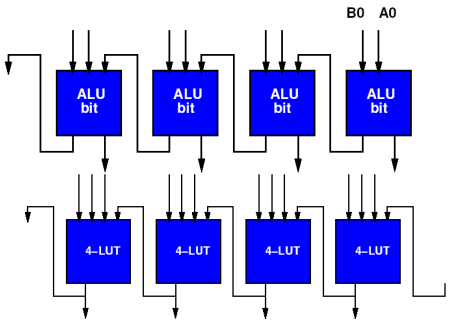


XC4003A data from Eric Kusse (UCB MS 1997)



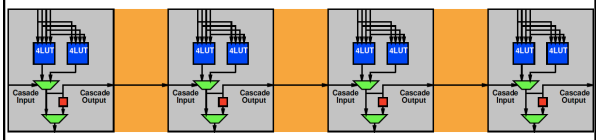
[Virtex II, Shang et al., FPGA 2002]

ALU vs. Cascaded LUT?

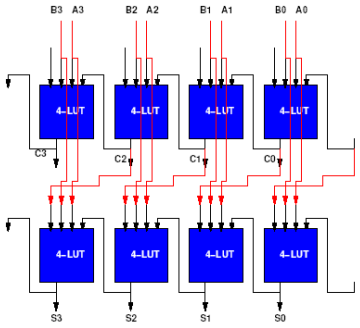


Datapath Cascade

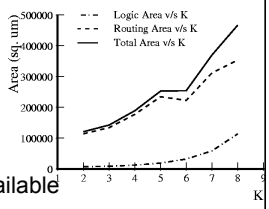
- ALU/LUT (datapath) Cascade
  - Long "serial" path w/out general interconnect
  - Pay only Tmux and nearest-neighbor interconnect



4-LUT Cascade ALU



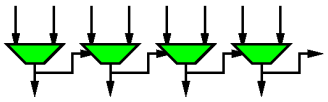
ALU vs. LUT ?



- ALU
  - Only subset of ops available
  - Denser coding for those ops
  - Smaller
  - …but interconnect area dominates
  - [Datapath width orthogonal to function]

Accelerating LUT Cascade?

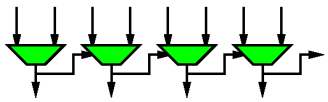
- Know can compute addition in O(log(N)) time
- Can we do better than N×Tmux for LUT cascade?
- Can we compute LUT cascade in O(log(N)) time?
- Can we compute mux cascade using parallel prefix?



- Can we make mux cascade associative?

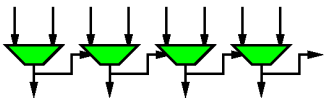
Parallel Prefix Mux cascade

- How can mux transform S→mux-out?
  - A=0, B=0 → mux-out=0
  - A=1, B=1 → mux-out=1
  - A=0, B=1 → mux-out=S
  - A=1, B=0 → mux-out=/S



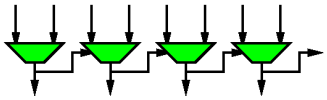
Parallel Prefix Mux cascade

- How can mux transform S→mux-out?
  - A=0, B=0 → mux-out=0 Stop= S
  - A=1, B=1 → mux-out=1 Generate= G
  - A=0, B=1 → mux-out=S Buffer = B
  - A=1, B=0 → mux-out=/S Invert = I



Parallel Prefix Mux cascade

- How can 2 muxes transform input?
- Can I compute 2-mux transforms from 1 mux transforms?



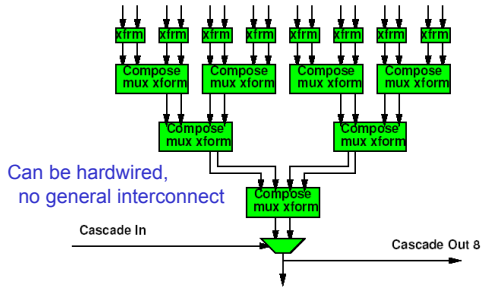
Two-mux transforms

- SS→S
- SG→G
- SB→S
- SI→G
- GS→S
- GG→G
- GB→G
- GI→S
- BS→S
- BG→G
- BB→B
- BI→I
- IS→S
- IG→G
- IB→I
- II→B

Generalizing mux-cascade

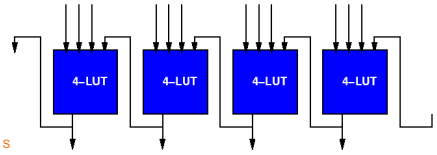
- How can N muxes transform the input?
- Is mux transform composition associative?

Parallel Prefix Mux-cascade



LUT Cascade Implications

- We can compute **any** LUT cascade in logarithmic time
  - Not just addition carry cascade
  - Can be different functions in each LUT
    - Not demand SIMD op



"ALU"s Unpacked

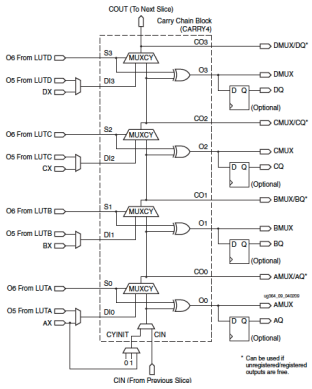
Traditional/Datapath ALUs combine 3 ideas

1. SIMD/Datapath Control
   - Architecture variable w
2. Long Cascade
   - Typically also w, but can shorter/longer
   - Amenable to parallel prefix implementation in O (log(w)) time w/ O(w) space
3. Restricted function
   - Reduces instruction bits
   - Reduces expressiveness

Commercial Devices

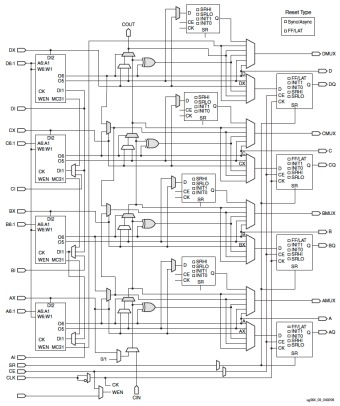
Virtex 6 & 7 Carry

- Prop=A xor B
- Otherwise
  - Generate
  - Squash
- Sum A xor B xor C



Virtex 6

- V7 similar



Programmable Array Logic (PLAs)

PLA

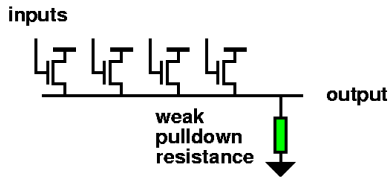
- Directly implement flat (two-level) logic
  - O=a*b*c*d + !a*b*!d + b*!c*d
- Exploit substrate properties allow wired-OR

Wired-**or**

- Connect series of inputs to wire
- Any of the inputs can drive the wire high

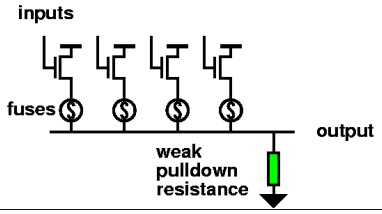
Wired-**or**

- Implementation with Transistors



Programmable Wired-**or**

- Use some memory function to programmable connect (disconnect) wires to OR
- Fuse:



Programmable Wired-**or**

- Gate-memory model

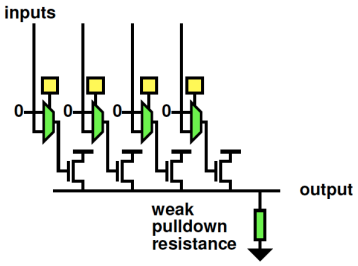
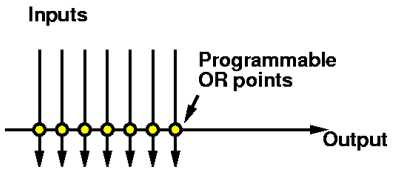
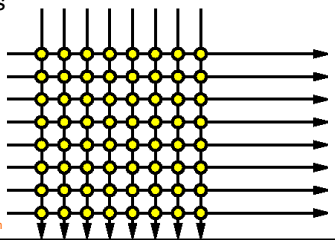


Diagram Wired-**or**



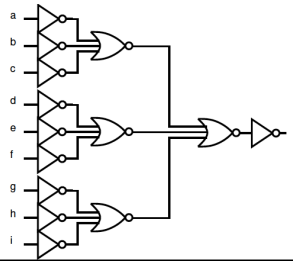
Wired-**or** array

- Build into array
  - Compute many different **or** functions from set of inputs



Preclass 6

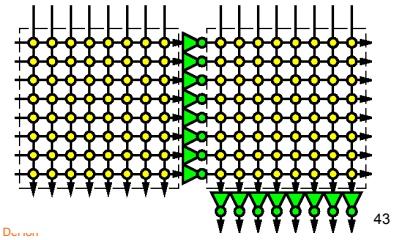
- What function (MSP) ?



Combined **or**-arrays to PLA

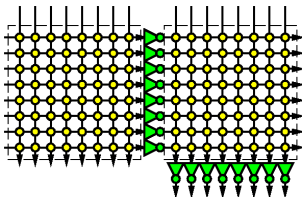
- Combine two or (**nor**) arrays to produce PLA (**and**-**or** array)

Programmable Logic Array



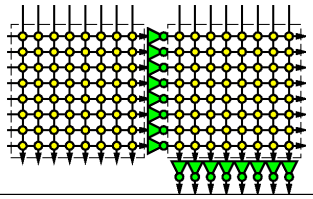
PLA

- Can implement each **and** on single line in first array
- Can implement each **or** on single line in second array



PLA

- Efficiency questions:
  - Each **and**/**or** is linear in total number of *potential* inputs (not actual)
  - How many product terms between arrays?



Preclass 5

- How many product terms in S[w] in MSP?

Carries

- C[1]=A[0]*B[0]
  - P[1]=1
- C[2]=A[1]*B[1]+C[1]*A[1]+C[1]*B[1]
  - P[2]=1+2*P[1]=3
- C[3]=A[2]*B[2]+C[2]*A[2]+C[2]*B[2]
  - P[3]=1+2*P[2]=7
- P[4]=15, P[5]=31, P[6]=63
- $P[w]=2^w-1$

Sum

- S[w]=A[w] xor B[w] xor C[w]
- S[w]=(A[w]*B[w]+/A[w]*/B[w])*C[w] +(/A[w]*B[w]+A[w]*/B[w])*/C[w]
- Product terms = 2*P[w]+2*Pterms(/C[w])

S[2] in MSP

a2*/a0*/b2*/b1+
/a2*/a0*b2*/b1+
a2*/a1*/a0*/b2+
/a2*/a1*/a0*b2+
a2*/b2*/b1*/b0+
/a2*b2*/b1*/b0+
a2*/a2*/b2*/b0+
/a2*/a1*b2*/b0+
/a2*a0*/b2*b1*b0+
/a2*a1*a0*/b2*b0+
a2*a0*b2*b1*b0+
a2*a1*a0*b2*b0+
a2*/a1*/b2*/b1+
/a2*/a1*b2*/b1+
/a2*a1*/b2*b1

PLA Product Terms

- Can be exponential in number of inputs
- *E.g.* n-input **xor** (parity function)
  - When flatten to two-level logic, requires exponential product terms
  - a*!b+!a*b
  - a*!b*!c+!a*b*!c+!a*!b*c+a*b*c
- …and additions, as we just saw…

PLAs

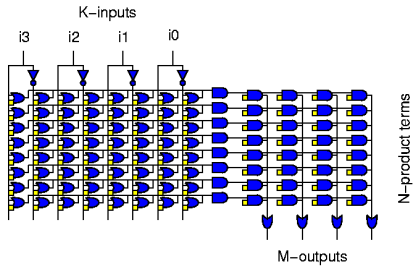
- Fast Implementations for large ANDs or ORs
- Number of P-terms **can be** exponential in number of input bits
  - most complicated functions
  - not exponential for many functions
- Can use arrays of small PLAs
  - to exploit structure
  - like we saw arrays of small memories last time

PLAs vs. LUTs?

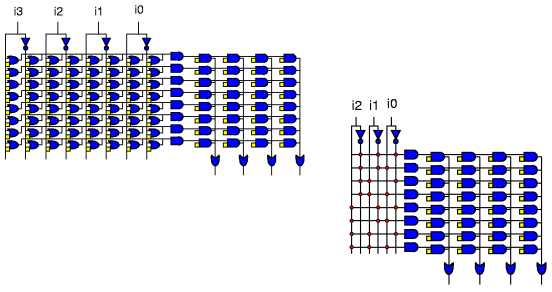
- Look at Inputs, Outputs, P-Terms
  - minimum area (one study, see paper)
  - K=10, N=12, M=3
- A(PLA 10,12,3) comparable to 4-LUT?
  - 80-130%?
  - 300% on ECC (structure LUT can exploit)
- Delay?
  - Claim 40% fewer logic levels (4-LUT)
    - (general interconnect crossings)
    - Not comparing to hardwired cascades

[Kouloheris & El Gamal/CICC'92]

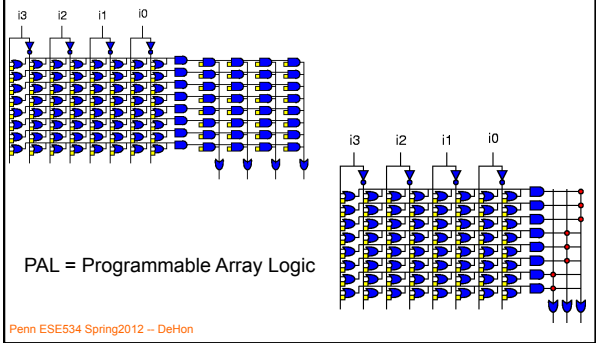
PLA



PLA and Memory

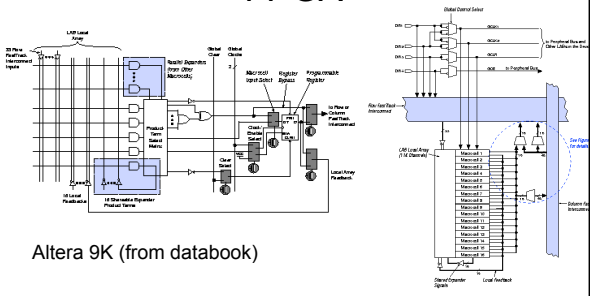


PLA and PAL



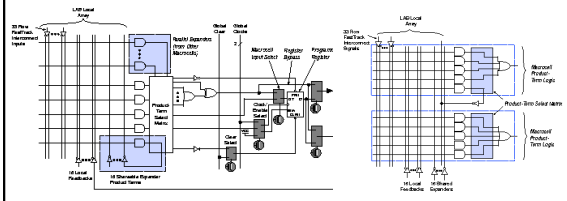
PAL = Programmable Array Logic

Conventional/Commercial FPGA



Altera 9K (from databook)

Conventional/Commercial FPGA



Altera 9K (from databook)

Like PAL

Admin

- HW6.1-2 due Friday
- Spring Break next week
- Reading for Monday after break on web
- HW6.3-4 on Wed. after break

Big Ideas [MSB Ideas]

- Programmable Interconnect allows us to exploit structure in logic
  - want to match to application structure
  - Prog. interconnect delay expensive
- Hardwired Cascades
  - key technique to reducing delay in programmables (both ALUs and LUTs)
- PLAs
  - canonical two level structure
  - hardwire portions to get Memories, PALs