ESE680-002 (ESE534): Computer Organization

Day 12: February 21, 2007
Compute 2:
Cascades, ALUs, PLAs

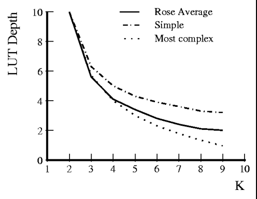
Last Time

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

Today

- Cascades
- ALUs
- PLAs

Last Time



- Larger LUTs
  - Less interconnect delay

+ General: Larger compute blocks
  - Minimize interconnect crossings

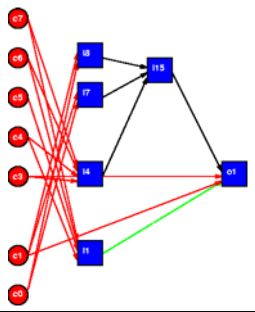
- Large LUTs
  - Not efficient for typical logic structure

Different Structure

- How can we have "larger" compute nodes (less general interconnect) without paying huge area penalty of large LUTs?

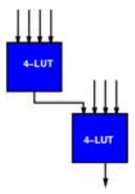
Structure in subgraphs

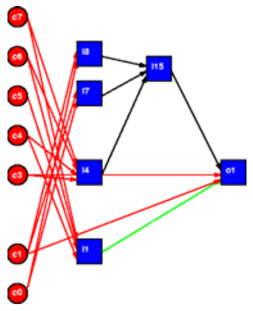
- Small LUTs capture structure
- What structure does a small-LUT-mapped netlist have?



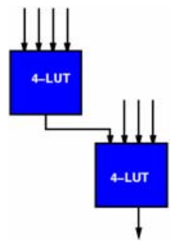
Structure

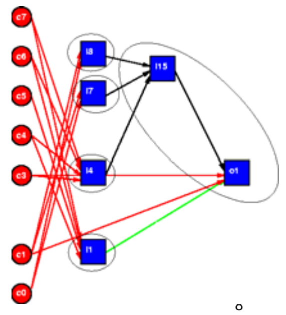
- LUT sequences ubiquitous





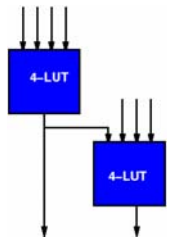
Hardwired Logic Blocks

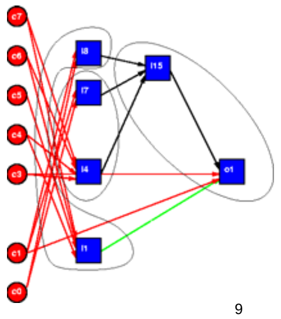




Single Output

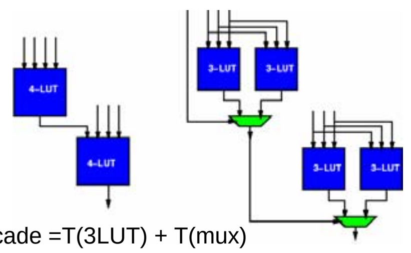
Hardwired Logic Blocks





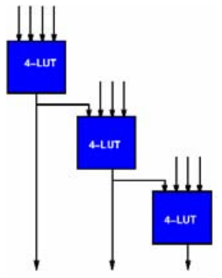
Two outputs

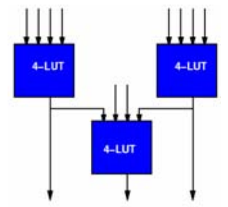
Delay Model



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

Options





Chung & Rose Study

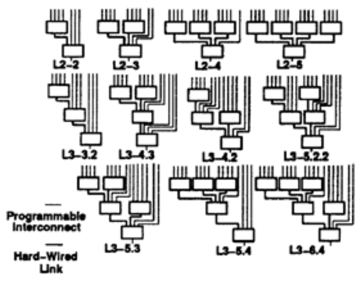
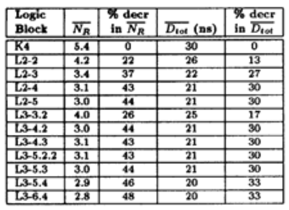


Figure 8: Delay Study HLB Topologies

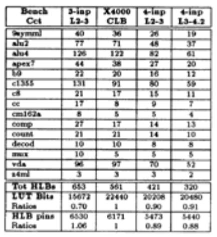
[Chung & Rose, DAC '92]

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

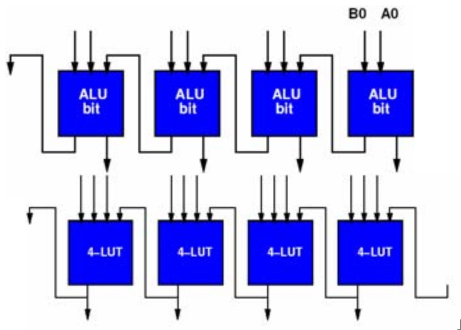


Bench Cct	3-inp L2-3	X4000 CLB	4-inp L2-3	4-inp L3-4.2
9symml	40	36	26	19
alu2	77	71	48	37
alu4	126	122	82	61
apex7	44	38	27	20
b9	22	20	16	12
c1355	131	91	80	59
c8	21	17	15	11
cc	17	8	9	7
cm162a	8	5	5	4
comp	27	17	14	13
count	21	21	14	10
decod	10	10	8	8
mux	10	5	5	5
vda	96	97	70	52
z4ml	3	3	3	2
Tot 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]

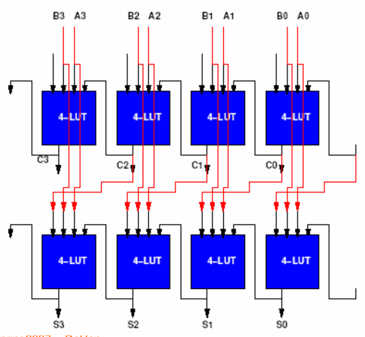
ALU vs. Cascaded LUT?



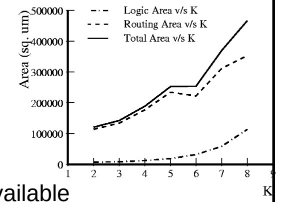
Datapath Cascade

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

4-LUT Cascade ALU



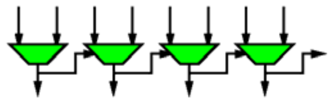
ALU vs. LUT ?



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

Parallel Prefix LUT Cascade?

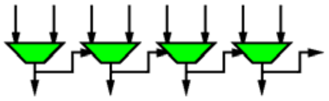
- Can we do better than N×Tmux?
- Can we compute LUT cascade in O(log(N)) time?
- Can we compute mux cascade using parallel prefix?



- Can we make mux cascade associative?

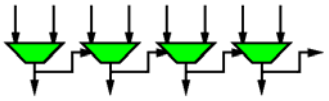
Parallel Prefix Mux cascade

- 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



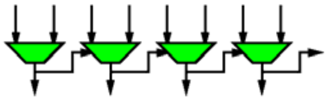
Parallel Prefix Mux cascade

- 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?



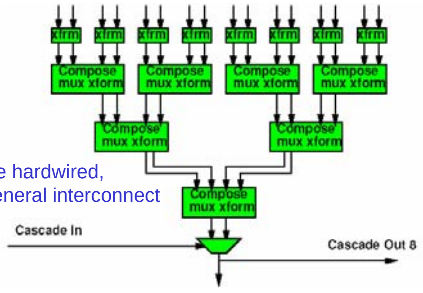
Two-mux transforms

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

Generalizing mux-cascade

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

Parallel Prefix Mux-cascade



Can be hardwired, no general interconnect

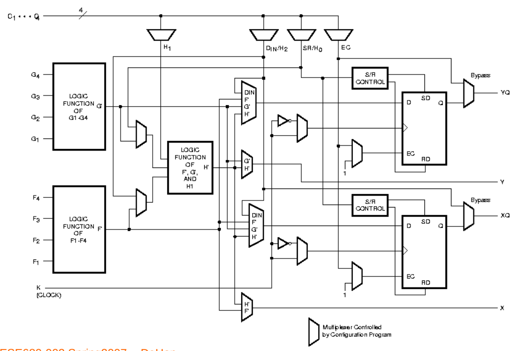
"ALU"s Unpacked

Traditional/Datapath ALUs

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

Xilinx XC4000 CLB



LOGIC FUNCTION OF G1-G4
LOGIC FUNCTION OF F', G', AND H1
LOGIC FUNCTION OF F1-F4
S/R CONTROL
Bypass
YQ
Y
XQ
X
K (CLOCK)
Multiplexer Controlled by Configuration Program

Xilinx Virtex-II

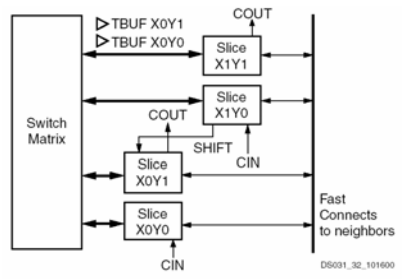


Figure 14: **Virtex-II CLB Element**

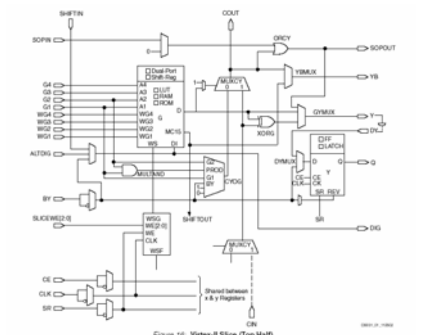


Figure 16: **Virtex-II Slice (Top Half)**

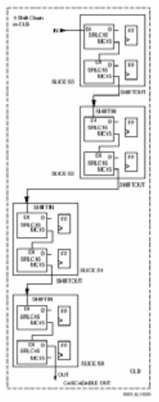


Figure 22: Cascadable Shift Register

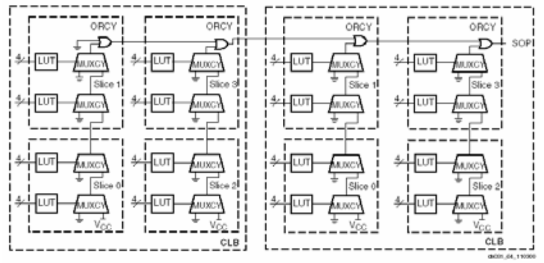
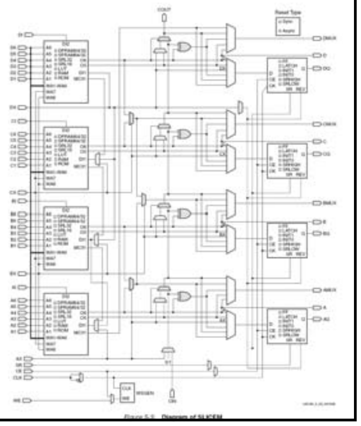


Figure 25: **Horizontal Cascade Chain**

Virtex-5



Altera Stratix

Figure 5. Stratix LE

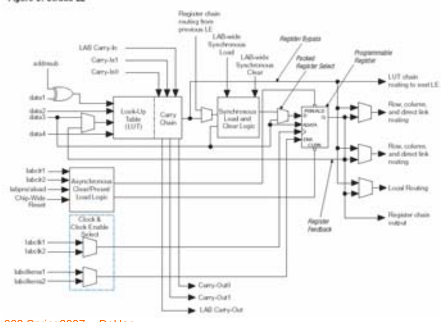

Register chain routing from previous LE
LAB Carry-In
Carry-In1
Carry-In0
addnsub
data1
data2
data3
data4
Look-Up Table (LUT)
Carry Chain
Synchronous Load and Clear Logic
LUT chain routing to next LE
Row, column, and direct link routing
Local Routing
Register chain output
Asynchronous Clear/Preset/Load Logic
Clock & Clock Enable Select
Carry-Out0
Carry-Out1
LAB Carry-Out

Figure 7. LE in Dynamic Arithmetic Mode

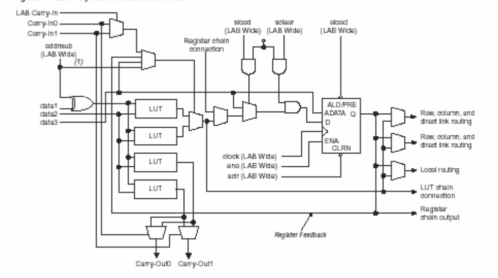
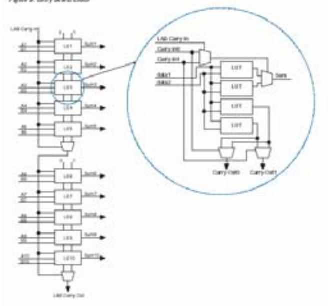


Figure 8. Carry Select Chain



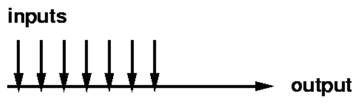
Programmable Array Logic (PLAs)

PLA

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

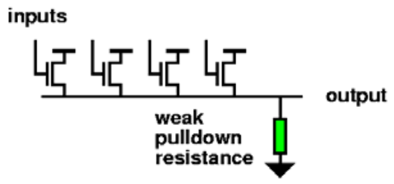
Wired-**or**

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



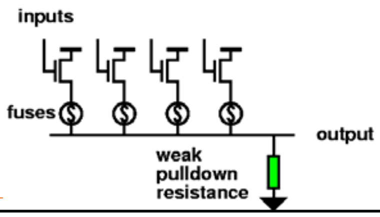
Wired-**or**

- Implementation with Transistors



Programmable Wired-**or**

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



Programmable Wired-**or**

- Gate-memory model

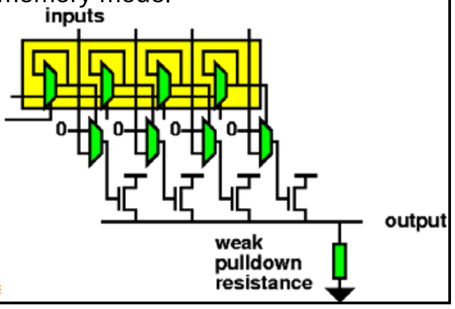
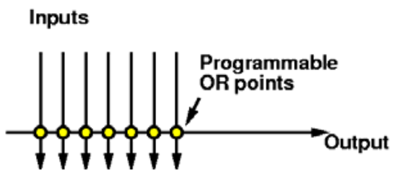
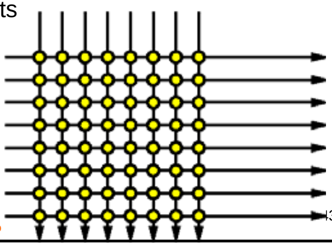


Diagram Wired-**or**



Wired-**or** array

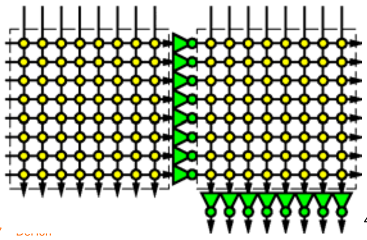
- Build into array
  - Compute many different **or** functions from set of inputs



Combined **or**-arrays to PLA

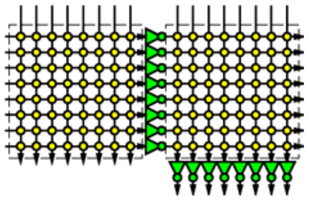
- Combine two or (**nor**) arrays to produce PLA (**and-or** array)

Programmable Logic Array



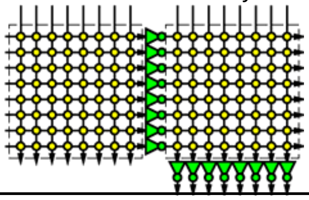
PLA

- 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?



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 shows up in important functions
  - Like addition…

PLAs

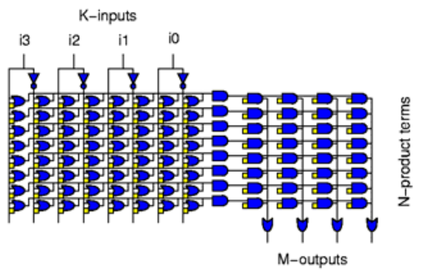
- Fast Implementations for large ANDs or ORs
- Number of P-terms **can be** exponential in number of input bits
  - most complicated functions
  - not exponential for many functions
- Can use arrays of small PLAs
  - to exploit structure
  - like we saw arrays of small memories last time

PLAs vs. LUTs?

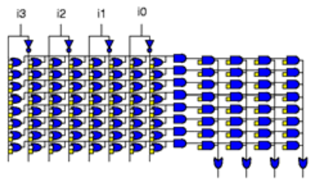
- Look at Inputs, Outputs, P-Terms
  - minimum area (one study, see paper)
  - K=10, N=12, M=3
- A(PLA 10,12,3) comparable to 4-LUT?
  - 80-130%?
  - 300% on ECC (structure LUT can exploit)
- Delay?
  - Claim 40% fewer logic levels (4-LUT)
    - (general interconnect crossings)

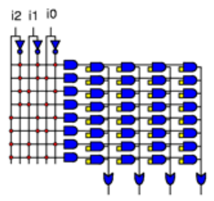
[Kouloheris & El Gamal/CICC'92]

PLA

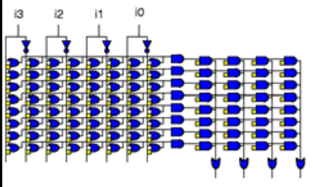


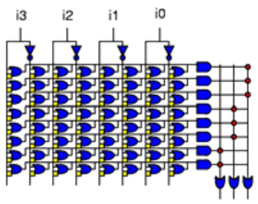
PLA and Memory





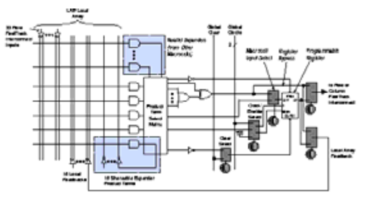
PLA and PAL

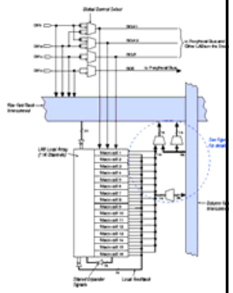




PAL = Programmable Array Logic

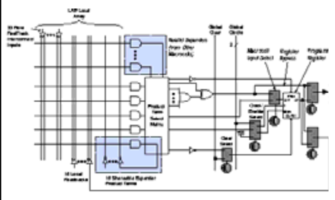
Conventional/Commercial FPGA

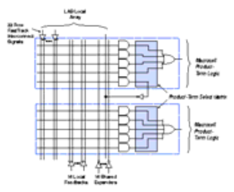




Altera 9K (from databook)

Conventional/Commercial FPGA





Altera 9K (from databook)

Like PAL

Big Ideas [MSB Ideas]

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

Big Ideas [MSB-1 Ideas]

- Better structure match with hardwired LUT cascades