CS294-7: Reconfigurable Computing

Day 21: Tuesday, April 8

Compute Blocks

Reducing Latency

- Admin
- ALU bits
- Cascades
- Hardwired LUTs

Source for Xilinx and Altera component diagrams is their respective databooks.

Admin

- NEXT Thursday (4/17) $\rightarrow$ FCCM
  (No class)

- Reading Comments

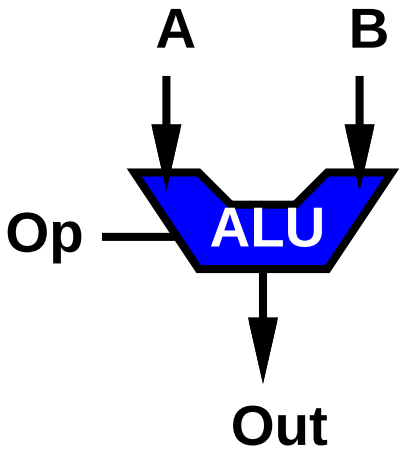
ALU Bits

Traditional microprocessors, DSPs, etc. don't use LUTs (or PLAs, or ULMs, or NAND gates).

Instead use a w-bit "Arithmetic and Logic Unit" (ALU).

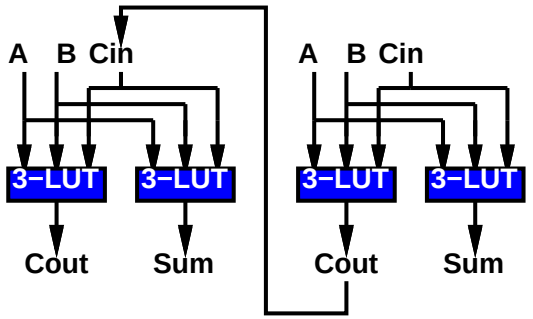
AND2
OR2
XOR2

ADD
SUB
COMPARE



ALU Bit

AND2 OR2 XOR2	**2-LUT**
ADD SUB COMPARE	**2 3-LUTs**



Carry connection hardwired in ALU $\rightarrow$ fast

(no switches, no stubs, short wires $\rightarrow$ minimal RC delay)

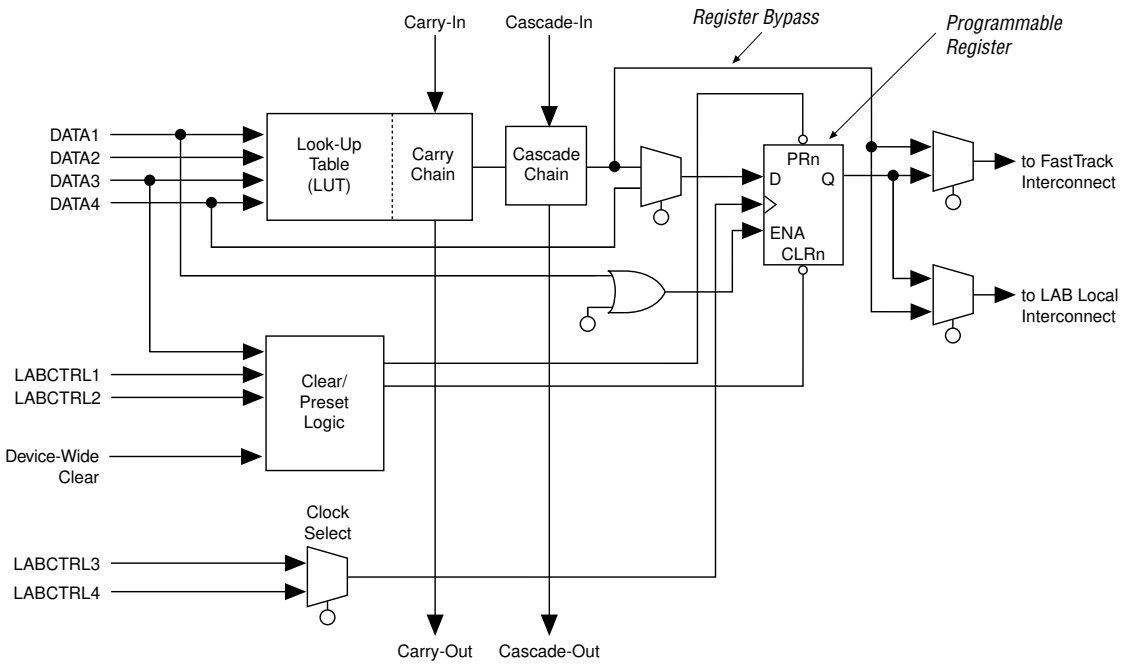
LUT Add Delay

1.5-2 ns/LUT	logic delay (day 18, Altera)
0.5 ns	Single channel delay (day 14 model)
2.5 ns	per bit (Also Altera in LAB)

Compare: $\lambda = 0.6\mu$

32b ALU (minimal area)	16 ns
32b ALU (cycle optimized)	6 ns
32b 4-LUT ripple cascade	80 ns

Carry Support (Altera)



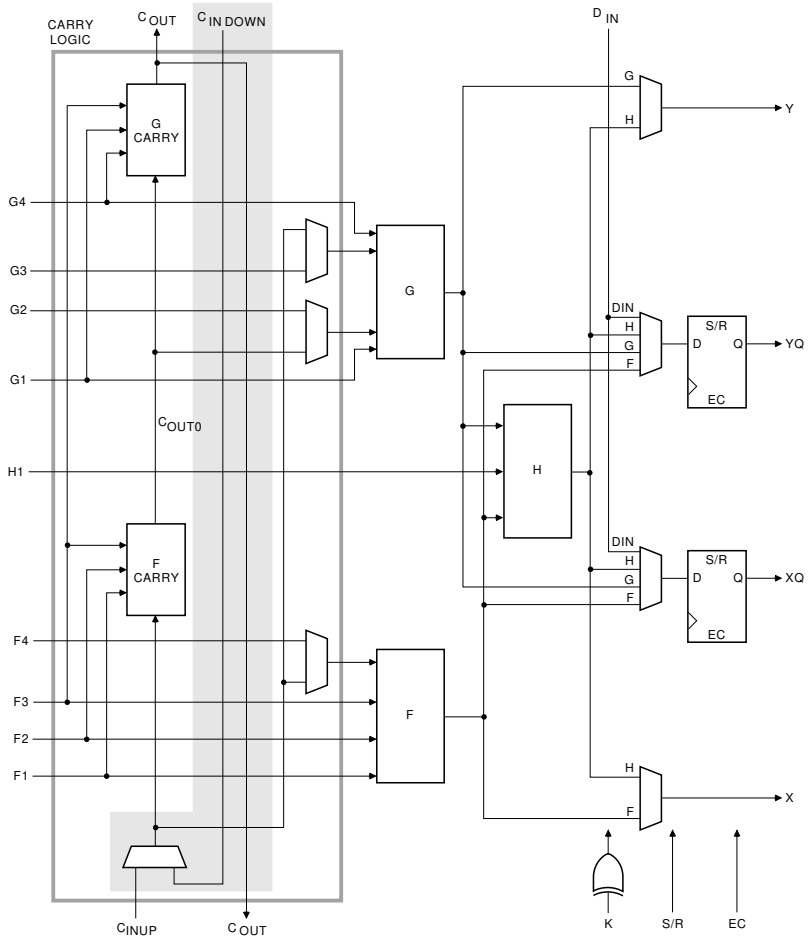
Carry Support (Altera)

Delay	Operation
2.5 ns	LUT combinational (in LAB)
0.5 ns	data to carry
0.5 ns	Cin$\rightarrow$Cout (1 bit)
0.3 ns	LAB carry
3.0 ns	column routing (same column)

16b add:

Carry	delay	Area
no	43 ns	32 LEs
	8 ns (16 ns ?)	16 LEs

Carry Support (Xilinx)



Carry Support (Xilinx)

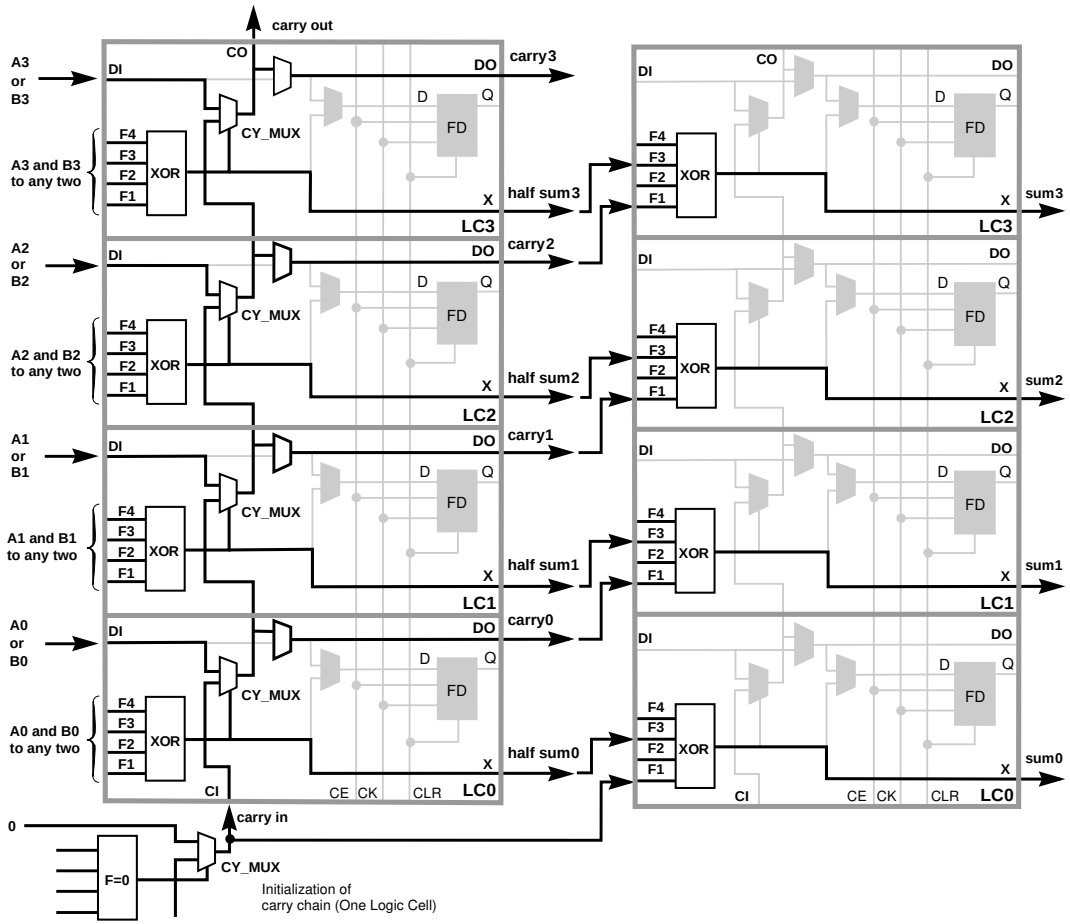
Xilinx 4K (-4):

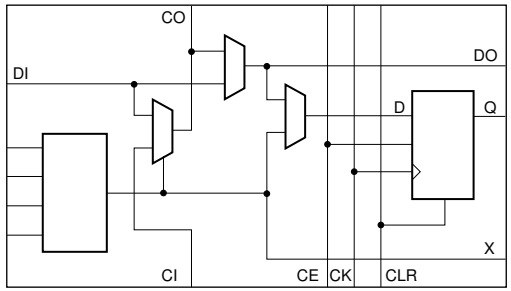
Delay	Operation
4 ns	LUT combinational
5.5 ns	Data in to carry
1.5 ns	Cin$\rightarrow$Cout (2 bits)

16b add:

Carry	delay	Area
no	64 ns	16 CLBs
	20.5 ns	9 CLBs

Carry Support (Xilinx)





Carry Support (Xilinx)

Xilinx 5K (-4):

Delay	Operation
3.8 ns	LUT combinational
2.9 ns	data to carry
0.5 ns	Cin$\rightarrow$Cout (1 bit)
3.8 ns	(final sum from carry)

16b add:

Carry	delay	Area
no	61 ns	33 LUTs (8.25 CLBs)
	16 ns (17 ns)	33 LUTs (8.25 CLBs)

LUT Carry Summary

- Common operation
- Dedicated path $\rightarrow$ speed benefit
- Simple combining logic $\rightarrow$ area benefit

Other opportunities?

Cascades

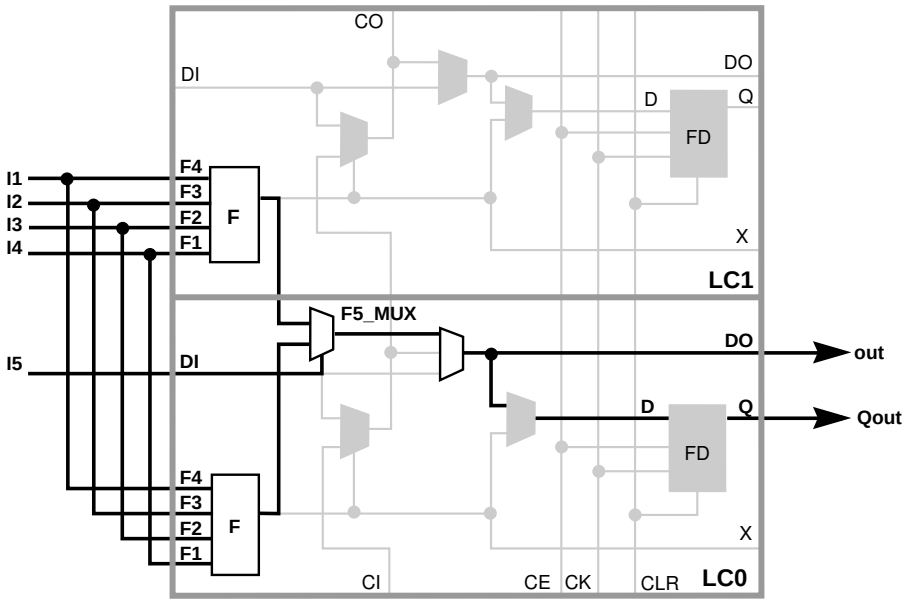
Large fanin operations (Reductions)

- decoding
- matching
- completion detection
- many-to-one reductions for control

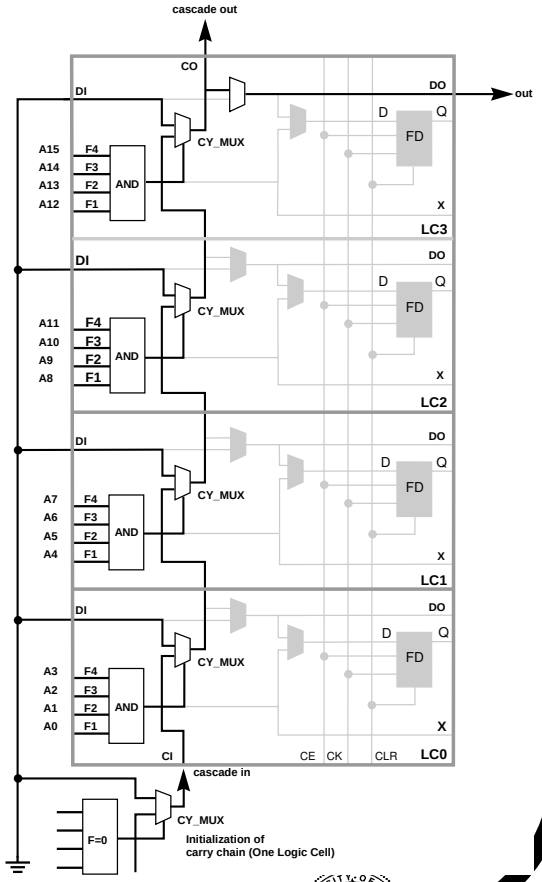
Combining logic is simple (little area)
Make use of dedicated paths

Xilink 5K Cascade

5-LUT	Cascade
3.8ns 4-LUT	3.8ns base
5.0ns 5-LUT	+0.5ns cascade stage



5-Input Function

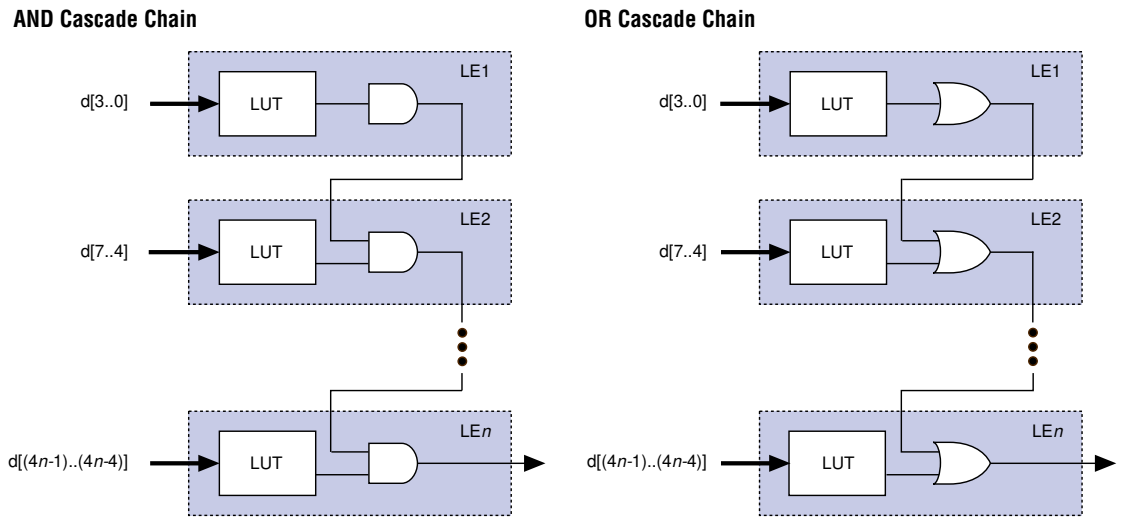


Altera 8K Cascade

Cascade

2.5ns base
+0.7ns element

$$t_{casc} = \textbf{3–4}\ t_{local}$$



Hardwired Logic Blocks

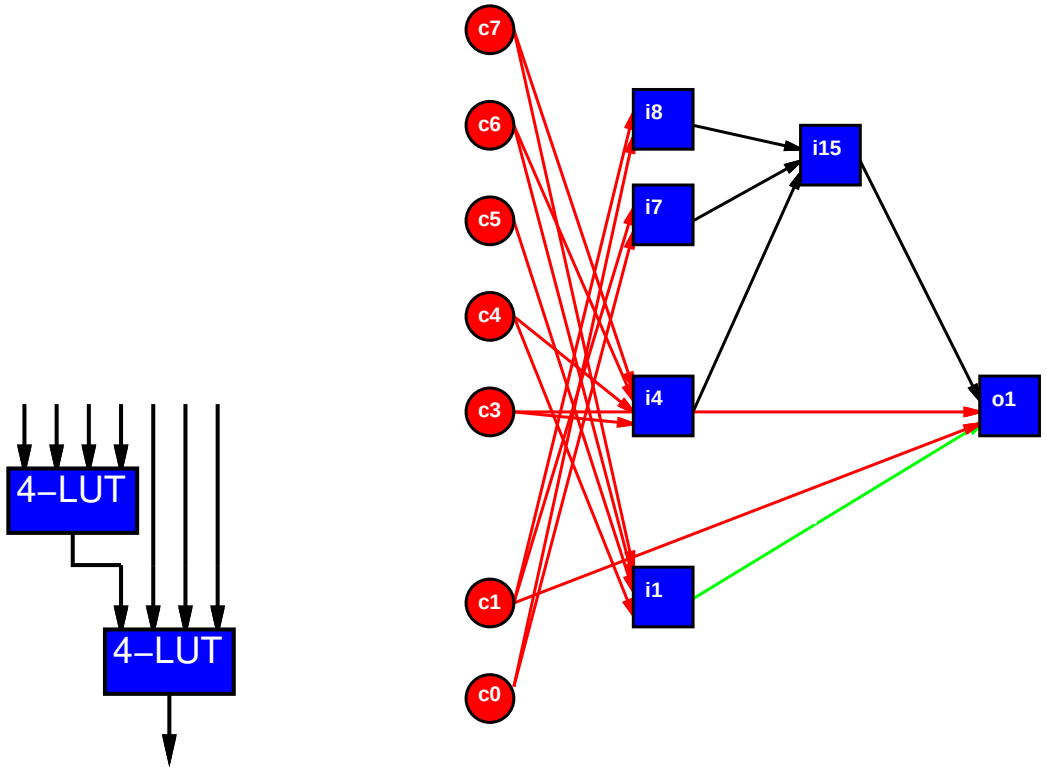
Continuing the theme that "general" interconnect is slow...
...can we reduce it in a more general way?

Have seen that mapping to larger blocks
may reduce times through general interconnect
$\rightarrow$ reduced delay

- larger LUTs
- PLAs
- dedicated cascades

Hardwired Logic Blocks

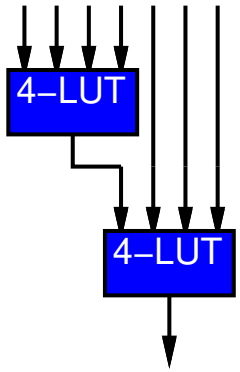
LUT sequences are ubiquitous subgraphs...

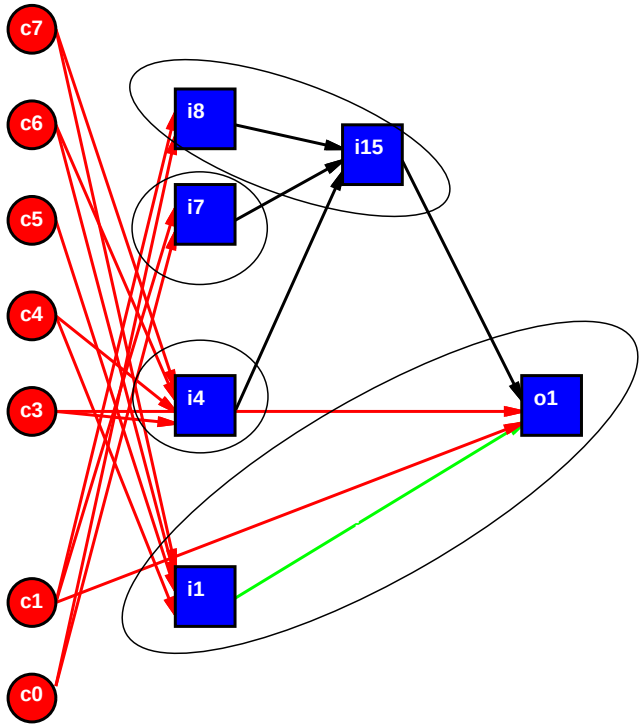


...can think hardwired LUT sequence as an alternate (*c.f.* larger k) way of building a "larger" fixed LUT block

Hardwired Logic Blocks

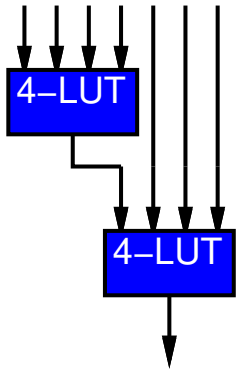
Cover (one out)

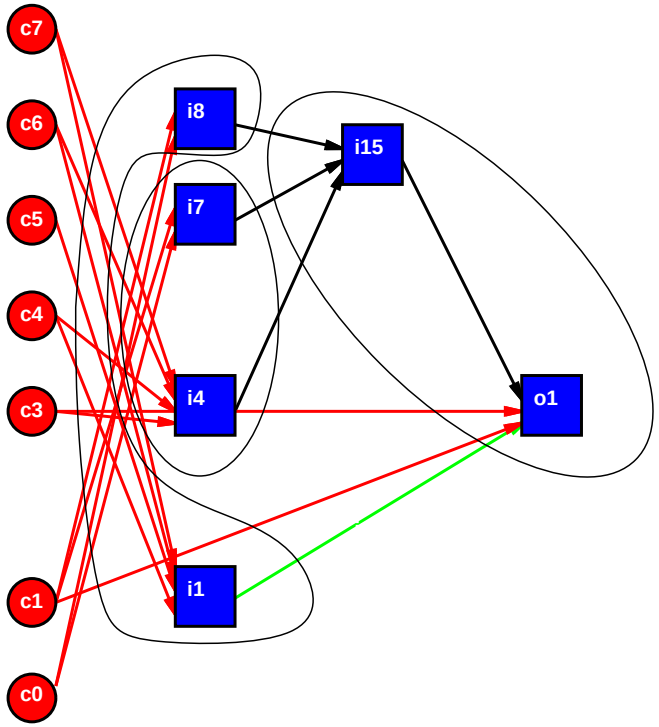




Hardwired Logic Blocks

Cover (separate outputs)





Hardwired Logic Blocks

- HLB selection (Chung/FPL'91 Fig. 3)
- Path decrease + area increase (Chung/FPL'91 Table 1+2)
- Area optimized (Chung/FPL'91 Table 3)
- Delay Mapped (Seo Table 2.1)
- Area Mapped (Seo 2.2)
- LEGO HLB (Seo Figure 3.11)
- HLB Delay (Fig 3.25 + Table 3.2)

Summary

- Several approaches to reducing design latency
  - fast carry
  - cascade
  - hardwired connections
- All exploit "hardwired" connections to avoid programmable interconnect delay