University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Analyzing and Optimizing the Bit-Sliced Comparator

Area Heuristic for One Comparator Bit Slice is 20

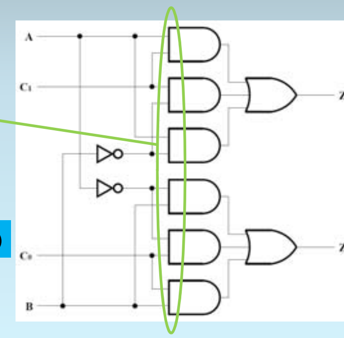
Let's analyze area and delay.

How many literals? 12

How many operators?

8 (6 AND, 2 OR)

So **area is 20**.



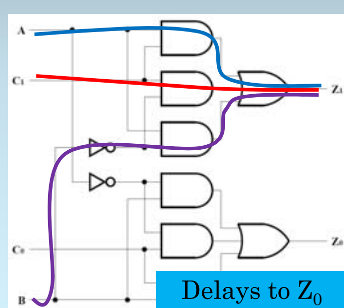
How Many Gate Delays to Z_1?

A to Z_1:
2 gate delays

C_1 to Z_1:
2 gate delays

C_0 to Z_1:
not relevant

B to Z_1:
2 gate delays (ignoring NOT)

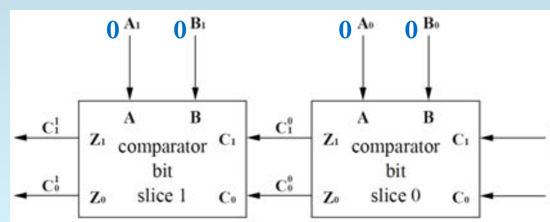


Delays to Z_0 are symmetric.

Extending from One Bit Slice to N Bit Slices

What happens in an **N-bit** design?

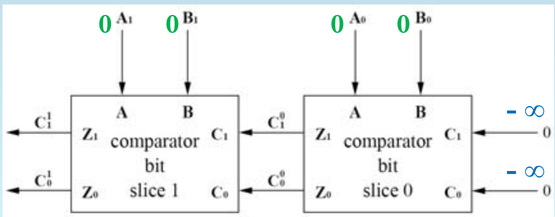
Say that **A and B are available at time 0**.



Constant Inputs are Available Arbitrarily Early

What about the 0s on the right?

Available "forever" … (**time -∞**).



Use Bit Slice Timing to Calculate Times Between Slices

Now we must

- use the delays that we found for one bit slice
- to calculate times for inter-slice **C** values.

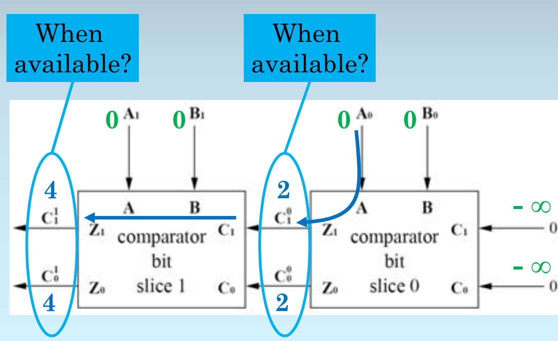
Recall that

- all **A** and **B** bits are available at **time 0**,
- so the **C to Z delays are the most important**.

We found

- **C_1 to Z_1: 2 gate delays**
- **C_0 to Z_0: 2 gate delays**

Calculate the Time at Which C^M Becomes Available



A More Detailed Version of Our Calculations

Grey is "not relevant," and green is maximum (time at which Z_i is available).

(bit slice 0)	A	B	C_1	C_0
input available at	0	0	-∞	-∞
delay from input to Z_1	+2	+2	+2	
Z_1 not available until	2	2	-∞	
delay from input to Z_0	+2	+2		+2
Z_0 not available until	2	2		-∞

A More Detailed Version of Our Calculations

Grey is "not relevant," and green is maximum (time at which $\mathbf{Z_i}$ is available).

(bit slice 1)	**A**	**B**	**C_1**	**C_0**
input available at	0	0	2	2
delay from input to Z_1	+2	+2	+2	
Z_1 not available until	2	2	4	
delay from input to Z_0	+2	+2		+2
Z_0 not available until	2	2		4

Generalize the Result to an N-Bit Comparator

$\mathbf{C_1^0}$ and $\mathbf{C_0^0}$ are available at time 2 (2 gate delays).*

$\mathbf{C_1^1}$ and $\mathbf{C_0^1}$ are available at time 4.

When are $\mathbf{C_1^{N-1}}$ and $\mathbf{C_0^{N-1}}$ available (these are the answer for an **N-bit** comparator)?

N-bit answer is available at time 2N.

*In the notes, the inverters are counted, so paths from A and B are slightly longer, and all timings are increased by 1.

We May be Able to Improve Our Comparator Design

Can we do better?

(You should ask: better in what sense?)

Can we reduce delay?
- **Unlikely** with a bit-sliced design.
- Not easy to implement most functions with one gate.

Can we reduce area?
- **Maybe** …
- Let's do some algebra.

Use Algebra to Find Common Subexpressions (A'B, AB')

Start with $\mathbf{Z_1 = AB' + AC_1 + B'C_1}$

then use distributivity to pull out $\mathbf{C_1}$:

$$\mathbf{Z_1 = AB' + (A + B')C_1}$$

and rewrite the **(A + B')** factor as a NAND:

$$\mathbf{Z_1 = AB' + (A'B)'C_1}$$

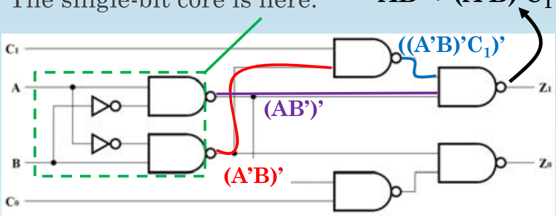
Similarly, $\mathbf{Z_0 = A'B + (AB')'C_0}$

Notice that we now reuse **AB'** and **A'B**.

The New Implementation Uses Fewer Gates

The diagram below shows the new equations using NAND gates. $Z_1 = [\ (AB')'\ ((A'B)'C_1)'\ ]'$

The single-bit core is here. $= AB' + (A'B)'C_1$

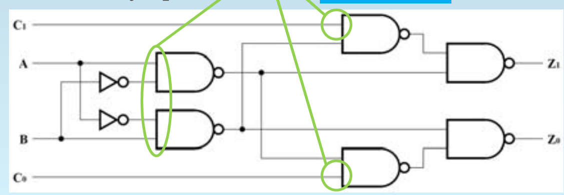


Area Heuristic for the New Design is 12

Let's analyze area for the new design.

How many literals? 6

How many operators? 6 (NAND)



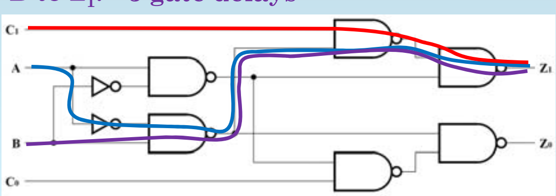
Delay Analysis for the New Design

A to Z_1: 3 gate delays (ignoring NOT)

C_1 to Z_1: 2 gate delays

B to Z_1: 3 gate delays

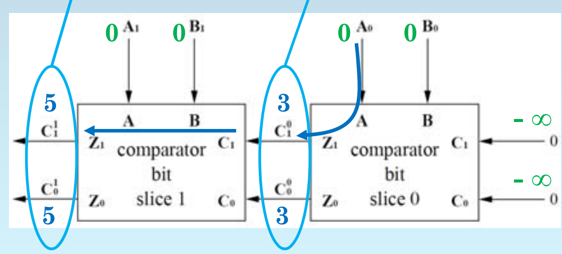
50% slower?!



Calculate the Time at Which C^M Becomes Available

When available?

When available?



A More Detailed Version of Our Calculations

Grey is "not relevant," and green is maximum (time at which $\mathbf{Z_i}$ is available).

(bit slice 0)	A	B	C_1	C_0
input available at	0	0	-∞	-∞
delay from input to Z_1	+3	+3	+2	
Z_1 not available until	3	3	-∞	
delay from input to Z_0	+3	+3		+2
Z_0 not available until	3	3		-∞

A More Detailed Version of Our Calculations

Grey is "not relevant," and green is maximum (time at which $\mathbf{Z_i}$ is available).

(bit slice 1)	A	B	C_1	C_0
input available at	0	0	3	3
delay from input to Z_1	+3	+3	+2	
Z_1 not available until	3	3	5	
delay from input to Z_0	+3	+3		+2
Z_0 not available until	3	3		5

The Slice-to-Slice Paths are the Important Ones

$\mathbf{C_1^0}$ and $\mathbf{C_0^0}$ are available at time 3 (2 gate delays).*

$\mathbf{C_1^1}$ and $\mathbf{C_0^1}$ are available at time 5.

When are $\mathbf{C_1^{N-1}}$ and $\mathbf{C_0^{N-1}}$ available (these are the answer for an **N-bit** comparator)?

N-bit answer is available at time 2N+1.

*In the notes, the inverters are counted, so paths from A and B are slightly longer, and all timings are increased by 1.

Overall: Much Better Area for Slightly More Delay

So the new design

- **reduces area by about 40%** (area 12N compared to area 20N).
- **increases delay by 1** (2N+1 gate delays compared to 2N gate delays).

Can We Do Even Better?

Yes, but it's not as easy.

For example, we can design a slice
- that compares multiple bits of **A** and **B**.
- See Notes 2.4.6 for an example.

We can also solve the full **N-bit** problem.

In other words, trade **more human work and complexity for better area and delay**.