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

ECE 120: Introduction to Computing

Bit-Sliced Comparator

How Do You Compare Unsigned Numbers?

Let's develop a bit-sliced design to **compare two unsigned numbers**.

Which 8-bit unsigned number is bigger?

0 1 1 0 1 0 0 0
0 1 0 1 0 1 1 1

How did you know?
Did you start on the left or the right?

Humans Go from Left to Right

Usually, humans start on the left. Why?

As soon as we notice a difference, we're done!

humans compare this way →

0 1 1 $a_4a_3a_2a_1a_0$
0 1 0 $b_4b_3b_2b_1b_0$

Bit-sliced hardware cannot stop in the middle.

The information flows from one end to the other.

Output wires produce the answer (in bits).

Our Design Compares from Right to Left

Our comparator design will start on the right.

humans compare this way →

$a_7a_6a_5a_4a_3a_2a_1a_0$
$b_7b_6b_5b_4b_3b_2b_1b_0$

← our design will compare this way

From least significant to most significant bit.

Three Possible Answers for Comparison of A and B

When comparing two numbers, **A** and **B**, we have **three possible outcomes**:

A < B

A = B

A > B

To decide the answer for **N+1** bits, we need:

- the answer for **N** (less significant) bits,
- one bit of **A**, and
- one bit of **B**.

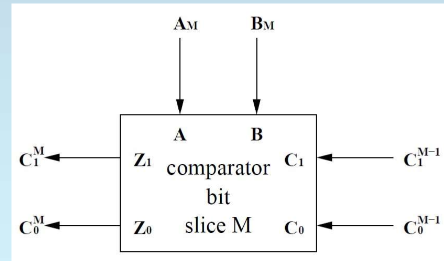
An Abstract Model of the Comparator Bit Slice

A question for you:

How many bits must pass between slices?

Two!

This figure shows an abstract model of our bit slice.



We Need a Representation for Answers

Another question for you:

How do we represent the three possible answers?

Any way we want!

Our choice of representation will affect the amount of logic we need.

Here's a good one…

C_1	C_0	meaning
0	0	`A = B`
0	1	`A < B`
1	0	`A > B`
1	1	not used

A Single Bit Requires Two Minterms on A, B

Let's start by solving a single bit.

In this case, there are no less significant bits.

So we consider only **A** and **B**.

Fill in the meanings, then the bits.

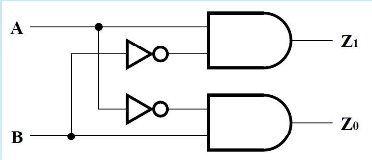
Note that Z_1 and Z_0 are minterms.

A	B	Z_1	Z_0	meaning
0	0	0	0	`A = B`
0	1	0	1	`A < B`
1	0	1	0	`A > B`
1	1	0	0	`A = B`

Comparing Two Bits is Fairly Easy

An implementation for a single bit appears below.

This structure forms the core of our bit slice, since it compares one bit of **A** with one bit of **B**.



When A and B are Equal, Pass Along the Answer

Now for the full problem.

We'll start with the case of **A = 0** and **B = 0**.

A	B	C_1	C_0	meaning	Z_1	Z_0	meaning
0	0	0	0	A = B	0	0	A = B
0	0	0	1	A < B	0	1	A < B
0	0	1	0	A > B	1	0	A > B
0	0	1	1	???	x	x	don't care

When A and B are Equal, Pass Along the Answer

Is there any difference when **A = 1** and **B = 1**?

No, outputs are the same as the last case.

A	B	C_1	C_0	meaning	Z_1	Z_0	meaning
1	1	0	0	A = B	0	0	A = B
1	1	0	1	A < B	0	1	A < B
1	1	1	0	A > B	1	0	A > B
1	1	1	1	???	x	x	don't care

When A and B Differ, Override the Previous Answer

What about case of **A = 0** and **B = 1**?

Always output A < B (for valid inputs).

A	B	C_1	C_0	meaning	Z_1	Z_0	meaning
0	1	0	0	A = B	0	1	A < B
0	1	0	1	A < B	0	1	A < B
0	1	1	0	A > B	0	1	A < B
0	1	1	1	???	x	x	don't care

When A and B Differ, Override the Previous Answer

And the case of **A = 1** and **B = 0**?

Always output A > B (for valid inputs).

A	B	C_1	C_0	Meaning	Z_1	Z_0	meaning
1	0	0	0	A = B	1	0	A > B
1	0	0	1	A < B	1	0	A > B
1	0	1	0	A > B	1	0	A > B
1	0	1	1	???	x	x	don't care

Z_1 is a Majority Function

Let's use a K-map to solve Z_1.

What are the loops?

AB'

AC_1

$B'C_1$

So

$Z_1 = AB' + AC_1 + B'C_1$

Z_1: C_1C_0 \ AB	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	x	x	x	x
10	1	0	1	1

Z_0 is Also a Majority Function

Now let's use a K-map to solve Z_0.

What are the loops?

$A'B$

$A'C_0$

BC_0

So

$Z_0 = A'B + A'C_0 + BC_0$

Z_0: C_1C_0 \ AB	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	x	x	x	x
10	0	1	0	0

Full Implementation as SOP Expressions

$Z_1 = AB' + AC_1 + B'C_1$

$Z_0 = A'B + A'C_0 + BC_0$

Notice the symmetry.

And the single-bit core.