Lecture 14 In-Class Worksheet

1 Learning Objectives

This worksheet is based on Lumetta course notes section 2.4. After completing this lesson, you will know how to:

- [S14-1] Build bit-sliced comparators for unsigned and two's complement numbers.

2 Comparators

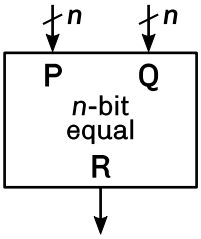
A comparator is a logic circuit that takes two values and returns a result that says something about their relationship. The specifics depend on what kinds of comparison we are talking about. In this worksheet, we will build the following comparators:

- Equality (Sec. 3),
- Greater-Than (Sec. 10), and
- Three-Way (Sec. 7).

We will use bit-sliced design, so that it can be extended to any number of bits.

3 Equality Comparator

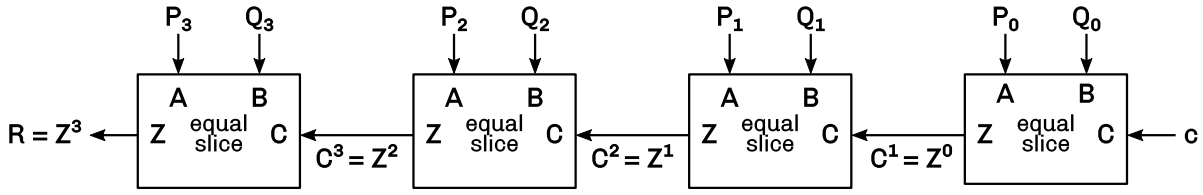
The simplest kind of comparator tells us if two numbers are are equal. Let P and Q be the sequences of bits we want to compare. Equality is the same for unsigned and two's complement integers: we just want to know if for every i, bit i of P is equal to bit i of Q, that is, $P_i = Q_i$.



R	Meaning
0	$P \neq Q$
1	$P = Q$

A diagram symbol for our n-bit equality comparator is shown on the right. (This is not a standardized symbol, just one we'll use here.) The box tells us that the inputs are called P and Q and that they each consist of n bits. The output is called R and consists of 1 bit. The meaning of this output is explained in the table below the comparator. The box on the right is an *abstraction:* it tells us what the comparator does, but not *how* it does it. Now, let's build what is inside the box using bit-sliced design.

This diagram below shows us one way to implement what is inside the box for $n = 4$ using bit-sliced design.



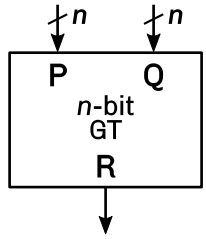
Our n-bit comparator is made up of n single-bit slices. We pass each slice one bit from P and one bit from Q. We also connect the output of one slice to the input of another, from least-significant to most-significant bit. Inside the bit slice, we've named the inputs A, B, and C, and the output Z. The output of a slice, called Z inside the slice, becomes the input C of the next slice. The output of the slice operating on the most significant bits of inputs to the comparator becomes the output of the comparator itself. In the diagram, we've used superscripts to index the variables representing the information passed between slices. The input to the first slice (operating on the least-significant bit) is a constant c^0 that we will need to pick.

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

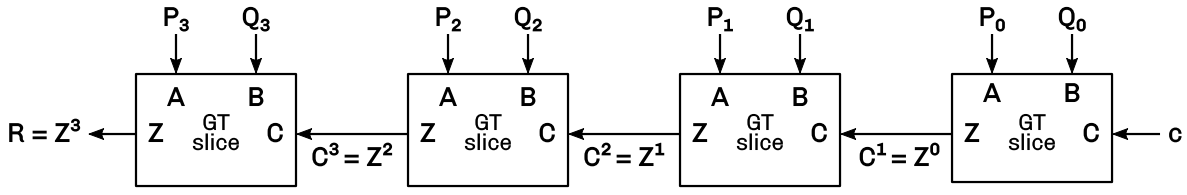
Now let's design what goes inside the slice. Inside, each slice only "sees" the slice input variables A, B, and C, from which it must produce the output Z. To understand how to compute Z, let's look at the last slice, which operates on the most significant bits of the comparator inputs P_3 and Q_3. We want the output Z to be 0 if $P \neq Q$ and 1 if $P = Q$. But this slice only sees $A = P_3$ and $B = Q_3$—how can it know about the rest of P and Q? It will need to get that information from its other input, C. We would like C to tell us if $P_2 = Q_2$, $P_1 = Q_1$ and $P_0 = Q_0$, or not. This is the same as asking if the 3-bit values $P_2P_1P_0$ and $Q_2Q_1Q_0$ are equal, and this is exactly what the output of the previous slice tells us. Thus, $C = 1$ if all of the bits so far have been equal and 0 otherwise. Given this information, and the value of P_3 and Q_3, we can write down the truth table of Z, shown on the right. This gives $Z = \bar{A}\bar{B}C + ABC$. We have one thing left to do, and that is to define the initial value c. From the truth table and the formula for Z we derived above, we can see that if input $C = 0$ then the output $Z = 0$, regardless of A and B. This is clearly not what we want. If $C = 1$, then Z is just the result of comparing A and B. So setting $c = 1$ will give us the behavior we want.

4 Unsigned Greater-Than

Let's build a comparator that tells us if $P > Q$ or not. The box interface for this comparator is shown on the right. The box diagram of the 4-bit comparator is shown below.



R	Meaning
0	$P \leq Q$
1	$P > Q$

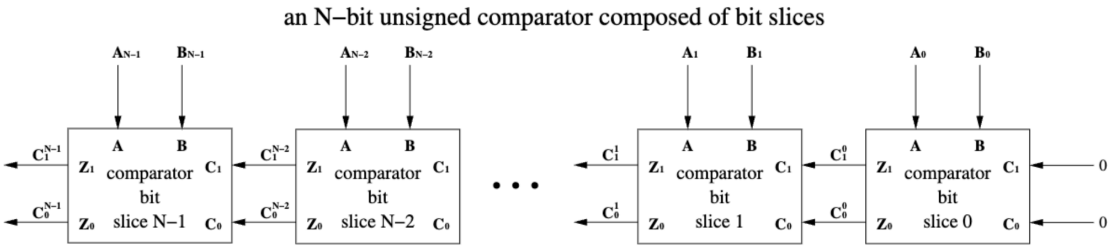


Q1. Derive Z for the Greater-Than comparator.

Q2. Determine the initial value c for the Greater-Than comparator shown above.

5 Right-to-Left Unsigned Three-Way Comparator

Let's build a three-way comparator (like the one described in the Lumetta course notes Section 2.4) that works right to left:



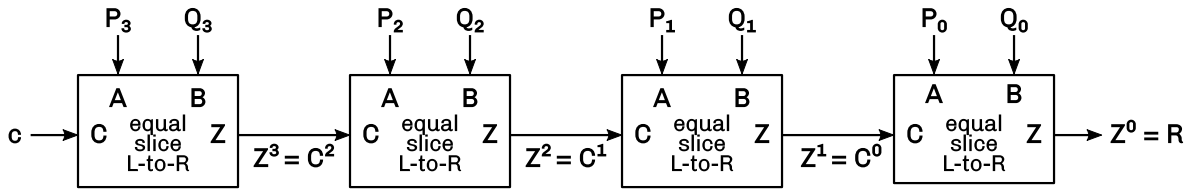
Q3. Derive the truth table for Z_1 and Z_0 in the Right-to-Left three-way Comparator shown above. The meaning of the outputs R_0 and R_1 is reproduced below.

R_1	R_0	Meaning
0	0	$P = Q$
0	1	$P < Q$
1	0	$P > Q$
1	1	Unused

Q4. Derive the expressions for Z_1 and Z_0 and the constant values c_1 and c_0.

6 Left-to-Right Equality Comparator

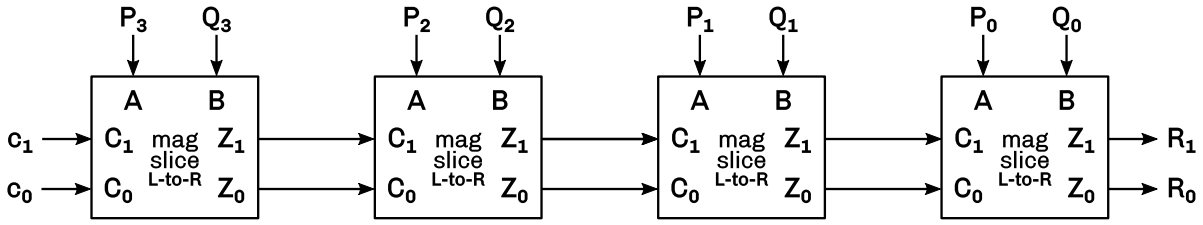
A bit sliced design does not always have to pass information right-to-left. For example, the equality comparator can also be designed to pass information from the most significant bit to the least significant bit:



In this case, the logic is exactly the same: $Z = \bar{A}\bar{B}C + ABC$ and $c = 1$.

7 Left-to-Right Unsigned Three-Way Comparator

We can also build a three-way comparator (like the one described in the Lumetta course notes Section 2.4) that works left to to right:



Q5. Derive the truth table for Z_1 and Z_0 in the Left-to-Right three-way Comparator shown above. The meaning of the outputs R_0 and R_1 is reproduced below.

R_1	R_0	Meaning
0	0	$P = Q$
0	1	$P < Q$
1	0	$P > Q$
1	1	Unused

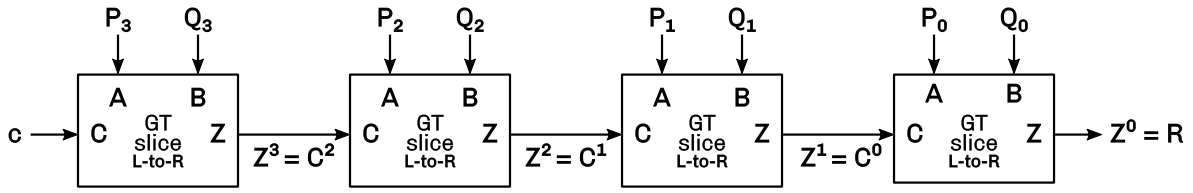
Q6. Derive the expressions for Z_1 and Z_0 and the constant values c_1 and c_0.

Q7. Compare the area heuristic of the left-to-right three-way comparator found above and the design given in the Lumetta course notes Section 2.4.4.

The area heuristic for the left-to-right three-way comparator is 6 (4 literals, 2 gates) for both Z_1 and Z_0. The Lumetta right-to-left comparator has area heuristic 10 (6 literals, 4 gates) for both Z_1 and Z_0 in the unoptimized design and 12 total for both Z_1 and Z_0 in the optimized design.

8 Left-to-Right Unsigned Greater-Than

Let's build a left-to-right version of our Greater-Than comparator. Here is the box diagram:



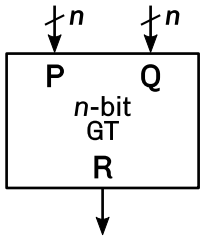
Q8. Derive the truth table for Z in the left-to-right Greater-Than comparator bit slice.

9 Two's Complement Greater-Than

Recall that the two's complement three-way comparator described in the Lumetta course notes was the same as the unsigned three-way comparator, *except* that the sign bit inputs were switched: the B input was P_{n-1} (A_{N-1} in the Lumetta course notes), and the A input was Q_{n-1} (B_{N-1} in the Lumetta course notes).

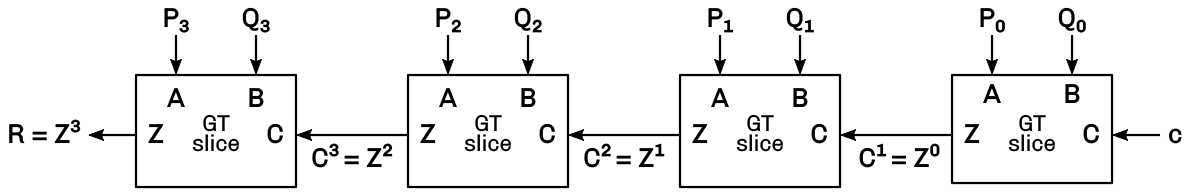
10 Two's Complement Greater-Than Comparator using Unsigned Greater-Than of Sec. 4

Let's build a comparator that tells us if $P > Q$ or not. Where, P and Q are n-bit two's complement representations. The box interface for this comparator is shown on the right.



R	Meaning
0	$P \leq Q$
1	$P > Q$

The box diagram of the 4-bit comparator is shown below.



Q9. Show that the same trick (switching the A and B inputs) also works for the Greater-Than comparator.