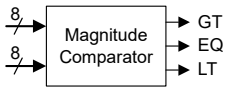
Comparator

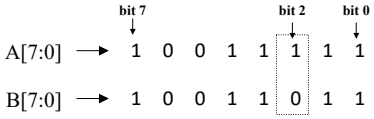
The design of a binary magnitude comparator

A magnitude comparator is a device that receives two N-bit inputs and asserts one of three possible outputs depending on whether one input is greater than, less than, or equal to the other (simpler comparators, called equality comparators, provide a single output that is asserted whenever the two inputs are equal). Comparators are readily described in behavioral Verilog, but they are somewhat more difficult to design using structural or schematic methods. In fact, comparator design is an excellent vehicle to showcase the power of behavioral design, and the relative tedium of structural design.



Block definition of 8-bit magnitude comparator.

A structural comparator design is best attacked using the bit-slice method. Consider an 8-bit magnitude comparator circuit that creates the GT, LT, and EQ output signals for two 8-bit operands. In the figure below, if A=159 and B=155 are presented to the comparator, then the GT output should be asserted, and the LT and EQ outputs should be de-asserted. The operand bits are equal in all slices except the bit 2 slice. Somehow, the inequality in the bit 2 slice must influence the overall circuit outputs, forcing GT to a '1' and LT and EQ to a '0'. Any bit pair could show an inequality, and any bit-slice module design must work in any bit position.



Compare two binary numbers (A = 159, B = 155)

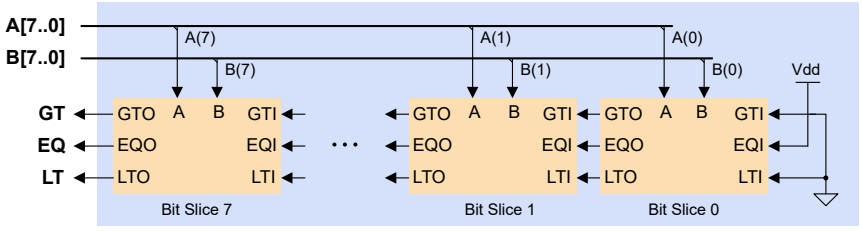
Clearly, a bit-slice design cannot work in isolation, using only the two data bits as inputs. A bit-slice design must take into account information generated from neighboring bit-slices. Specifically, each comparator bit-slice must receive not only the two operand input bits, but also the GT, LT, and EQ outputs of its less-significant bit neighbor. In the present example, the bit 3 slice in isolation would assert the EQ output, but the inequality in the bit 2 slice should force the bit 3 slice to assert GT and de-assert both EQ and LT. In fact, the outputs from any stage where the operand bits are equal depend on the inputs arising from the neighboring stage.

A bit-slice magnitude comparator circuit must have five inputs and three outputs as shown in the truth table. As with any combinational design, the truth table completely specifies the required comparator bit-slice behavior. Normally, a truth table for a five-input function would require 32 rows. The 8-row truth table in is adequate because certain input combinations are not possible (i.e., the inputs from the neighboring slice are mutually exclusive), and others are immaterial (i.e., if the current operand inputs show A > B, the neighboring slice inputs do not matter). You are encouraged to examine the truth table in detail, and convince yourself that you agree with the information it contains.

The truth table can be used to find a minimal bit-slice comparator circuit using pencil-and-paper methods or computer-based methods. Either way, a bit-slice circuit with the block diagram shown in the figure below can be designed. Once designed, a bit-slice circuit can be used in an N-bit comparator as shown in below figure. Note that for the N-bit comparator, no neighbor bit-slice exists for the least-significant bits; those non-existent bits are assumed to be equal. Note also that the overall comparator output arises from the outputs from the most-significant bit pair. In the exercises and lab project that accompany this module, you are asked to design a comparator bit-slice design as well as an 8-bit comparator circuit.

Operand inputs		Inputs from neighboring slices			Bit-slice outputs		
An	Bn	GTI	LTI	EQI	GTO	LTO	EQO
0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	0	0	1
0	1	ϕ	ϕ	ϕ	0	1	0
1	0	ϕ	ϕ	ϕ	1	0	0
1	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	0	0	1	0	0	1

Truth table for bit-sliced magnitude comparator



Block diagram of 8-bit magnitude comparator using bit-sliced magnitude comparator.

Implement Comparator Structurally

Bit-Sliced Comparator Module

Create a Verilog module for a 4-bit bit-sliced magnitude comparator according to the truth table presented below.

Operand inputs		Inputs from neighboring slices			Bit-slice outputs		
An	Bn	GTI	LTI	EQI	GTO	LTO	EQO
0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	0	0	1
0	1	ϕ	ϕ	ϕ	0	1	0
1	0	ϕ	ϕ	ϕ	1	0	0
1	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	0	0	1	0	0	1

Truth Table for a Bit-Sliced Magnitude Comparator

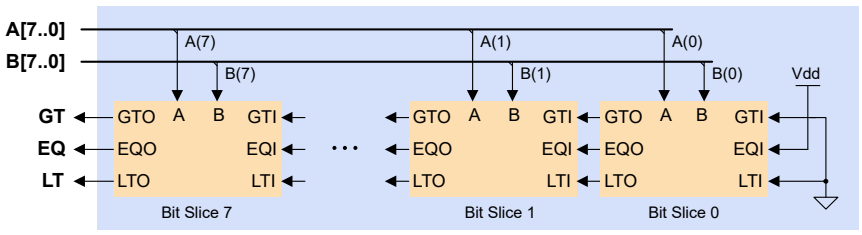
```
module cmp_bitslice(
    input A,
    input B,
    input LT_I,
    input EQ_I,
    input GT_I,
    output LT_O,
    output EQ_O,
    output GT_O
    );

assign GT_O = ( A & ~B ) | (~(A ^ B) & GT_I);
assign EQ_O = EQ_I & (( A & B ) | (~A & ~B));
assign LT_O = ( B & ~A ) | (~(A ^ B) & LT_I);

endmodule
```

Structurally Describe 4-bit Comparator with Bit Sliced Module

Connect the bit-sliced magnitude comparator according to the block diagram shown below.



Block diagram of an 8-bit magnitude comparator using a bit-sliced magnitude comparator

```
module cmp(
    input [3:0] A,
    input [3:0] B,
    output LT_O,
    output EQ_O,
    output GT_O
    );

wire [3:0] GT_int;
wire [3:0] EQ_int;
wire [3:0] LT_int;

cmp_bitslice slice_0 (
    .A(A[0]),
    .B(B[0]),
    .LT_I(1'b0),
    .EQ_I(1'b1),
    .GT_I(1'b0),
    .LT_O(LT_int[0]),
    .EQ_O(EQ_int[0]),
    .GT_O(GT_int[0])
    );

cmp_bitslice slice_1 (
    .A(A[1]),
    .B(B[1]),
    .LT_I(LT_int[0]),
    .EQ_I(EQ_int[0]),
    .GT_I(GT_int[0]),
    .LT_O(LT_int[1]),
    .EQ_O(EQ_int[1]),
    .GT_O(GT_int[1])
    );

cmp_bitslice slice_2 (
    .A(A[2]),
    .B(B[2]),
    .LT_I(LT_int[1]),
    .EQ_I(EQ_int[1]),
    .GT_I(GT_int[1]),
    .LT_O(LT_int[2]),
    .EQ_O(EQ_int[2]),
    .GT_O(GT_int[2])
    );

cmp_bitslice slice_3 (
    .A(A[3]),
    .B(B[3]),
    .LT_I(LT_int[2]),
    .EQ_I(EQ_int[2]),
    .GT_I(GT_int[2]),
    .LT_O(LT_int[3]),
    .EQ_O(EQ_int[3]),
    .GT_O(GT_int[3])
    );

assign LT_O = LT_int[3];
assign EQ_O = EQ_int[3];
assign GT_O = GT_int[3];

endmodule
```