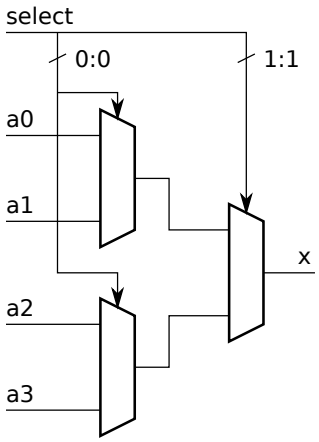
LSU EE 4755 **Homework 1** Solution **Due: 8 September 2017**

Start working on the solutions to the problems below on paper, but complete them using the computers in the lab. For instructions visit `https://www.ece.lsu.edu/koppel/v/proc.html`. *For the complete Verilog for this assignment without visiting the lab visit* `https://www.ece.lsu.edu/koppel/v/2017/hw01.v.html`.

Problem 1: Appearing below, and in `hw01.v`, is a Verilog description of a 2-input multiplexer, `mux2`, and a partially completed description of a 4-input mux, `mux4`, along with a diagram showing how a four-input mux can be made using three two-input multiplexers. Complete `mux4` as described in the diagram.

It is important that `mux4` instantiate three `mux2` modules. Other correct 4-input multiplexer implementations will not receive credit. Also, don't forget to set the parameters correctly when instantiating modules.




```
module mux2
  #( int w = 16 )
   ( output uwire [w-1:0] x,
     input uwire s,
     input uwire [w-1:0] a, b );

   assign  x = s == 0 ? a : b;

endmodule

module mux4
  #( int w = 6 )
   ( output uwire [w-1:0] x,
     input uwire [1:0] s,
     input uwire [w-1:0] a[3:0] );

   /// SOLUTION
   //
   //  Notice that wires and modules are named based upon the select
   //  bits for which they connect to the output.
   //
   uwire [w-1:0] x0x, x1x;

   mux2 #(w) m0x(x0x, s[0], a[0], a[1]);
   mux2 #(w) m1x(x1x, s[0], a[2], a[3]);

   mux2 #(w) mxx(x, s[1], x0x, x1x);

endmodule
```

Problem 2: Appearing below is a `mux8` module. Complete `mux8` so that it implements an 8-input multiplexer using two `mux4` modules and one `mux2` module. Notice that the data input to `mux8` is an 8-element array of w-bit quantities. To see how to extract a subrange of an array (called a *part select* in Verilog) see the `testbench` module. Solve this problem by generalizing the technique appearing in the previous problem.

Credit will only be given for `mux8` modules that *instantiate* two `mux4` modules and a `mux2` module. Yes, `assign x = a[s];` is correct and the best way to do it in other situations, but the goal here is to learn about instantiation.

```
module mux8
  #( int w = 5 )
   ( output uwire [w-1:0] x,
     input uwire [2:0] s,
     input uwire [w-1:0] a[7:0] );

   /// SOLUTION
   uwire [w-1:0] x0xx, x1xx;

   mux4 #(w) m0xx(x0xx, s[1:0], a[3:0]);
   mux4 #(w) m1xx(x1xx, s[1:0], a[7:4]);

   mux2 #(w) m(x, s[2], x0xx, x1xx);

endmodule
```

Appearing below is the start of the testbench code. To see the complete testbench and other modules follow `https://www.ece.lsu.edu/koppel/v/2017/hw01.v.html`.

```
module testbench();

   localparam int w = 10;
   localparam int n_in_max = 8;
   localparam int n_mut = 3;

   uwire [w-1:0] x[n_mut];
   logic [2:0]  s;
   logic [w-1:0] a[n_in_max-1:0];

   mux2 #(w) mm2(x[0], s[0], a[0], a[1]);
   mux4 #(w) mm4(x[1], s[1:0], a[3:0]);
   mux8 #(w) mm8(x[2], s[2:0], a[7:0]);

   initial begin

      automatic int n_test = 0;
      automatic int n_err = 0;
```