ECE 526L Lab7

1. Create a Verilog model of a scalable multiplexer, mux.v. Use the following module structure:

```
module mux(A, B, SEL, OUT)
...
endmodule
```

a. The size of the data ports is determined by a parameter, SIZE. If the multiplexer is not scaled when instantiated, it defaults to one bit wide.
b. Use synthesizable, behavioral Verilog/SystemVerilog to write the multiplexer module. Note that there are several ways to code up the design, but you must not do anything that explicitly or implicitly includes any comparison to x.
c. Your design should function such that input A should be selected when sel = 0 and input B when sel = 1. When sel is unknown, your multiplexers should resolve any bits for which A and B are the same. Any bits in conflict should result in x outputs.

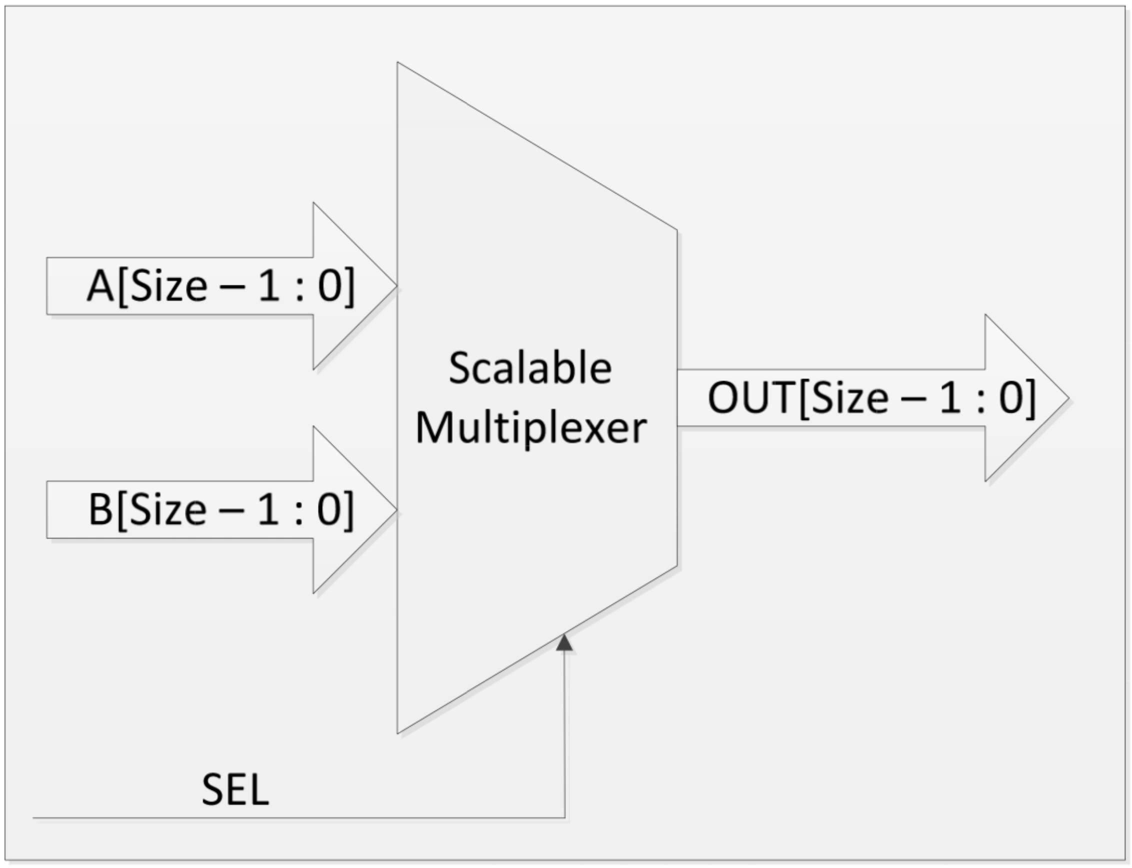
2. Create a Verilog testbench to verify that the scalable multiplexer works.

In the test module, create four instances of the scalable multiplexer. Use 1, 4, 5, 6 for different widths.

3. Use three different methods of parameter redefinition to change the widths of three of the instances. Leave the fourth instance with its default width of one bit. i.e. don’t attempt to set a width. Test each instance to show that all specifications are met. (Note: This is one testbench with four instances, not four testbenches with one instance each). Drive the largest multiplexer with as many bits as it needs for its A and B inputs. Use as many of the least significant bits of the A and B buses as needed to drive the smaller instances. The test vectors for the largest multiplexer should be sufficient to also test

the smaller multiplexers. Drive all of your instances in parallel, verifying all of them at the same time.

4. NOTE: when testing sel = 1'bx, you must apply A = B and A != B cases and at least one case where A and B share some common bits and some opposite bits. Every bit of every output must go to both zero and one at least once.


A[Size – 1 : 0]
B[Size – 1 : 0]
Scalable Multiplexer
OUT[Size – 1 : 0]
SEL