CPEN 211 Introduction to Microcomputers, 2018
Lab Proficiency Test #1

Question 1 [1.5 marks]: Create a file named "**q1.v**" and inside it write **synthesizable** Verilog that implements a combinational logic module matching the specification below. Your top-level module **must** be called "`detect_cover`" and be declared as:

```
module detect_cover(ain, bin, f);
  input [5:0] ain, bin;
  output [5:0] f;
```

Your q1.v file **must** include definitions for **any** modules instantiated inside `detect_cover` (even code provided in slides, labs or textbook). You should test your code and you can include testbench modules in q1.v, but testbench modules inside q1.v will be ignored by the autograder. **Submit your q1.v via "Lab Proficiency Test #1" on Canvas before 6:50 pm as the submission site closes at exactly 6:50 pm.** Do NOT "zip" your submission. The file you upload for this question must be called "q1.v". If you resubmit and Canvas renames the file to "q1-N.v" where N is an integer that is OK. Your q1.v will get zero if any of the following are true:

1. Your **last** "Lab Proficiency Test #1" attempt on Canvas does not include a file "q1.v",
2. Your q1.v file does not compile using ModelSim (e.g., due to syntax errors),
3. Your q1.v does not contain a module `detect_cover` with inputs/outputs as above,
4. Your `detect_cover` cannot be simulated in ModelSim, using only q1.v,
5. The Verilog used by your `detect_cover` module is not synthesizable by Quartus,
6. The Verilog used by your `detect_cover` module has inferred latches, or
7. Your `detect_cover` output "`f`" does not match the "Specification" given below.

Specification: Each bit of "`ain`", "`bin`" and "`f`" corresponds to one square in a 2×3 grid as shown in **Figure 1(a)**. A one in position *n* of "`ain`" indicates player A has placed a piece in the corresponding square. Similarly, a one in position *n* of "`bin`" indicates player B has placed a piece in the corresponding square. Player B is allowed to place pieces anywhere on the board *including* in the same squares occupied by Player A. The output "`f`" should equal "`ain`" when player B has placed a piece in every square that A has placed a piece in, and all zeros otherwise. More precisely, for all values of `i`, if, when `ain[i]` equals 1, it is also true that `bin[i]` equals 1, then "`f`" should equal "`ain`". If this condition is not met for any value of `i`, then "`f`" should be all zeros. E*xamples* of inputs `ain, bin` and corresponding `f` are shown in **Figure 1(b)-(f)** below. **Your solution must work correctly for *any* value inputs "`ain`" and "bin" (i.e., not only for these examples).**

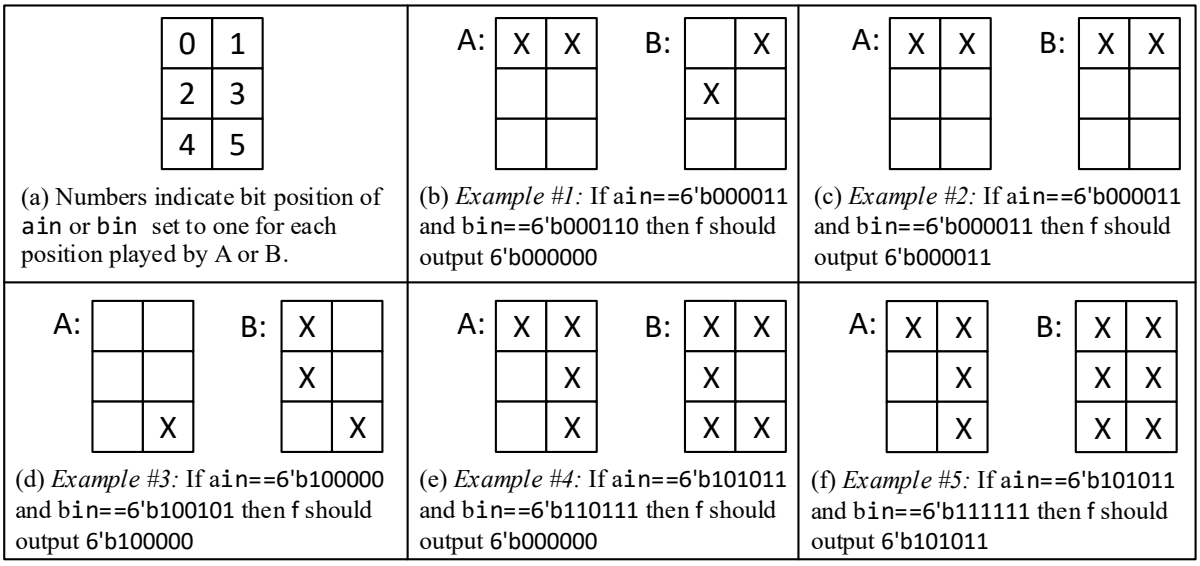

Figure 1