Please compile your solutions into a single PDF file for submission.

Problem 1

Design a SystemVerilog module called `seqDetect` to recognize an N-bit sequence of bits. The input bit ‘`a`’ is valid on the positive clock edge. The output `valid` should be asserted for one clock cycle if the last N ‘`a`’ bits match the `seq` input. Following a reset, the module should not assert valid until at least N clock cycle (ie. at least N bits have been input).

```
module seqDetect #(parameter N=6)(output logic valid,
                    input logic a, input logic [N-1:0] seq,
                    input logic clk, reset_n);
```

Write a testbench that tests the module as a 3-bit sequence detector (N=3). In addition, synthesize your design in Quartus. Your solution should include your code (both the module and testbench), the simulation waveform, and the RTL netlist from Quartus.

Problem 2

Design a SystemVerilog module called `vendingMachine` that keeps track of the amount of money put into a vending machine and asserts the `valid` output for one cycle once $1.00 or more has been received (no change will be given for more than $1.00). Following the assertion of the `valid` output, the module should begin counting money again for the next purchase. The inputs `nickel`, `dime` and `quarter` will be high for one clock cycle when a coin of that denomination is put in the machine. It is possible that multiple coins are detected simultaneously (ie. both nickel and quarter might be high for the same clock cycle).

```
module vendingMachine (output logic valid,
                      input logic nickel, dime, quarter,
                      input logic clk, reset_n);
```

Write a testbench that tests the module. In addition, synthesize your design in Quartus. Your solution should include your code (both the module and testbench), the simulation waveform, and the RTL netlist from Quartus.