M A S S A C H U S E T T S I N S T I T U T E O F T E C H N O L O G Y
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.111 Introductory Digital Systems Laboratory
Fall 2016

Lecture PSet #3
Due: Tue, 09/20/2016

Problem 1

For each of the parts below write one or more statements of Verilog that implement the desired functionality. Your Verilog just has to produce the same values for its outputs – it doesn't have to replicate the schematic logic gate-for-gate (in fact, you should not use those "structural" constructs). Be sure to include the appropriate declarations for any wires or regs used in your code.

(A) A circuit that divides an 8-bit input operand by 16 and produces a 4-bit value.

(B) A circuit to compute the 9-bit sum of two 8-bit operands

(C) Implement a 7-bit priority encoder (a circuit which examines its seven inputs (I1, I2, I3, I4, I5, I6, I7) and outputs a 3-bit binary number indicating the highest-priority input which has a value of "1", where I7 has the highest priority and I1 the lowest) with "dataflow" (assign) constructs.

(D) Implement a 7-bit priority encoder with "sequential" (always) construct and "case" statement.

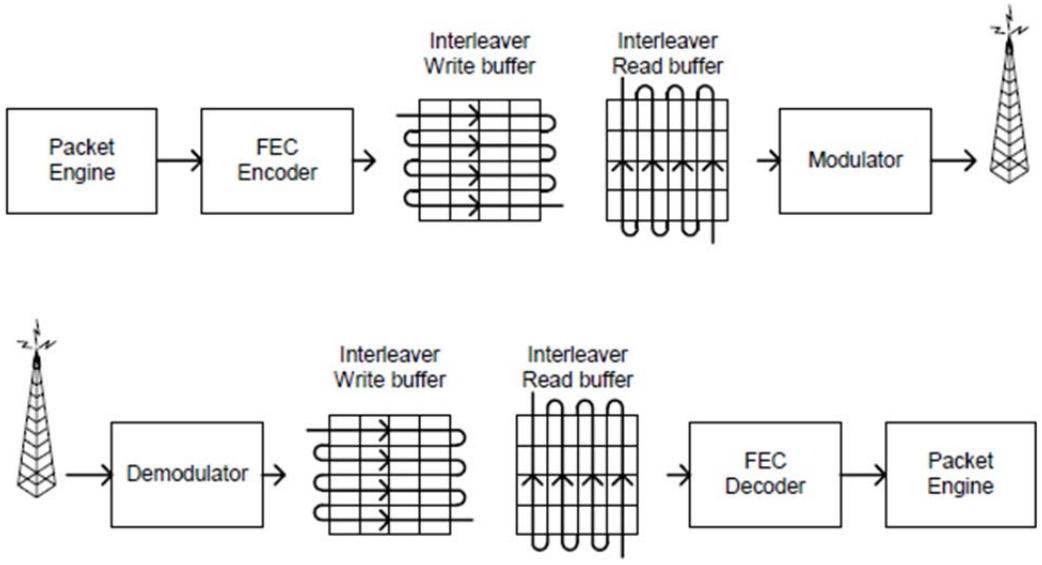
Problem 2. Using the Verilog parameterized module mechanism it's possible to write modules whose operations depends on parameters specified when the module is instantiated.

(A) Write a parameterized parity check module which takes as input a bus whose size is set by a parameter. The module has a single output which is 0 if the number of 1's in the input vector is even and 1 otherwise.

(B) How would one instantiate an instance of your module to compute parity check on the 16-bit data bus DATA[15:0]?

Problem 3. For data transmission, a forward error correcting (FEC) code such as a convolution code, will allow the receiver to correct erroneous bits when errors occur randomly in a coded sequence of, for example, 4 bytes. However, the bursty nature of noise will often wipe out large number of adjacent data bits - defeating the convolution code. A simple solution is to interleave the data bits so that adjacent data bits are spaced out in the transmitted sequence. Instead of sending all 8 bits of the byte 0, the low order

bit pair of bytes 3, 2, 1, and 0 (starting at the LSB end) are transmitted. This is implemented in many satellite communication systems[1].



(A) Implement a Verilog module that will interleave 4 bytes as shown.

```
module interveaver(
    input [7:0] byte0, // data = 4'h00
    input [7:0] byte1, // data = 4'h0E
    input [7:0] byte2, // data = 4'h8C
    input [7:0] byte3, // data = 4'h0C
    output [7:0] out0,
    output [7:0] out1,
    output [7:0] out2,
    output [7:0] out3
    );

    assign out0 =
    assign out1 =
    assign out2 =
    assign out3 =

endmodule
```

There are multiple implementations. To receive credit your interleaver must encode this input **[00 0E 8C 03]** to the following output **[C8 3C 00 20].** This will ensure compatibility with the deinterleaver. [This Verilog was actually used in a research project.]

(B) Write the Verilog for a deinterleaver. Any interesting observation?

[1] from http://www.ti.com/lit/an/swra113a/swra113a.pdf