

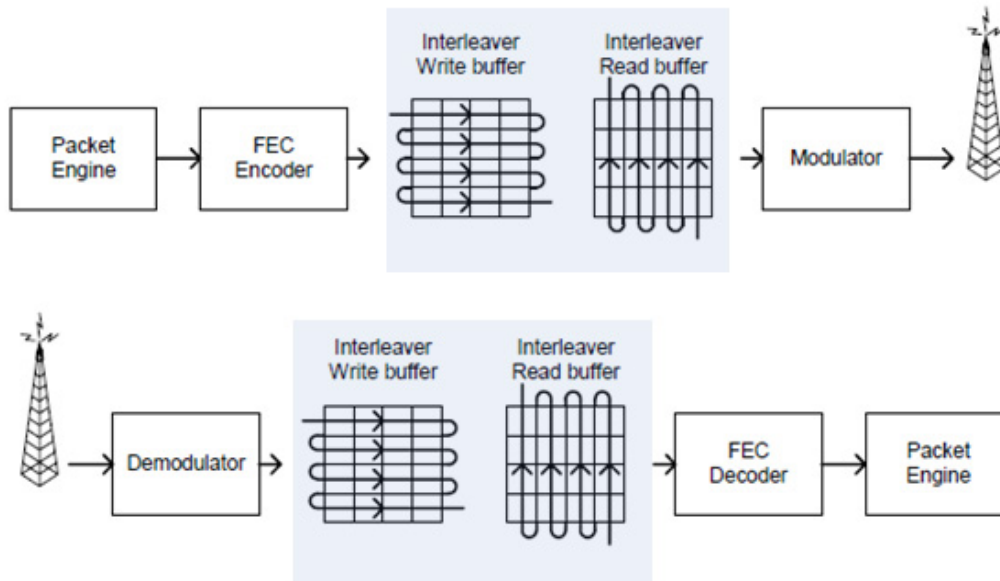
6.111 Introductory Digital Systems Laboratory

Fall 2017

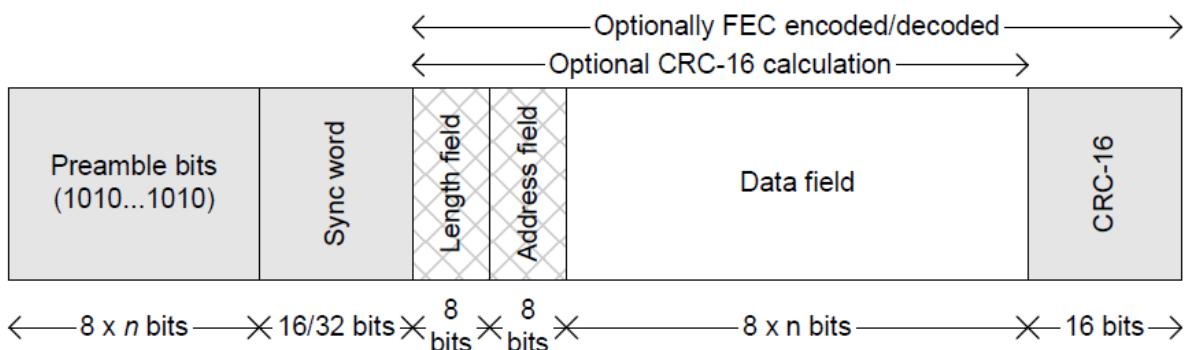
Lecture PSet #6 of 8

Due: *Wed 10/4/2017 upload by 23:59*

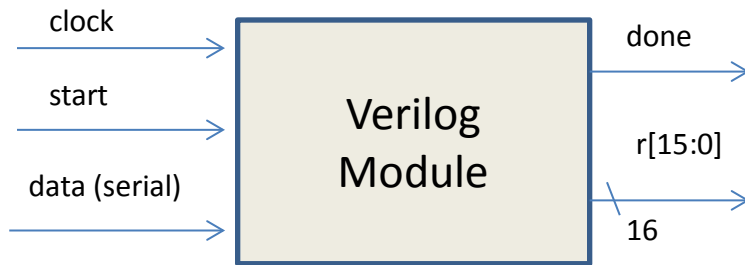
This LPset requires ISE. The LPset is based on an actual FPGA implementation for a low power wireless ECG monitor attached to a patient and transmitted to a receiver at a nurse base station. The Interleaver Module was part of an earlier lpset. This problem is focused part of the FEC (Forward Error Correction) encoder algorithm.



A typical transmitted data packet is show below.



data input 48'h03_01_02_03_30_3A



Getting started:

Step 1: Using ISE, create a new Verilog module with inputs and outputs as shown above.

Step 2: The Verilog module: when **start** is asserted (one clock pulse wide), reset your FSM; reset counters and other registers; and load any initial values. With each following clock pulse, begin the CRC calculation. Assert **done** when 48 bits are processed. The module should use only one clock domain always `@(posedge clock)`.

Step 3: Using ISE and the attached test bench (also posted on the course website) verify your design with a simulation using the process outlined in Lab 2 exercise 1(b). The test bench includes a 5ns clock. Note the syntax `[@posedge]` for a test bench is slightly different than a Verilog module. The input data is `48'h03_01_02_03_30_3A` and sent one bit at time. The first eight bits sent to the Verilog module are six zero followed by two ones corresponding to hex `[03]`. You may modify the test bench if needed for your implementation (generally not the case). In the actual FPGA ECG implementation, the data length is variable and processed one byte at a time.

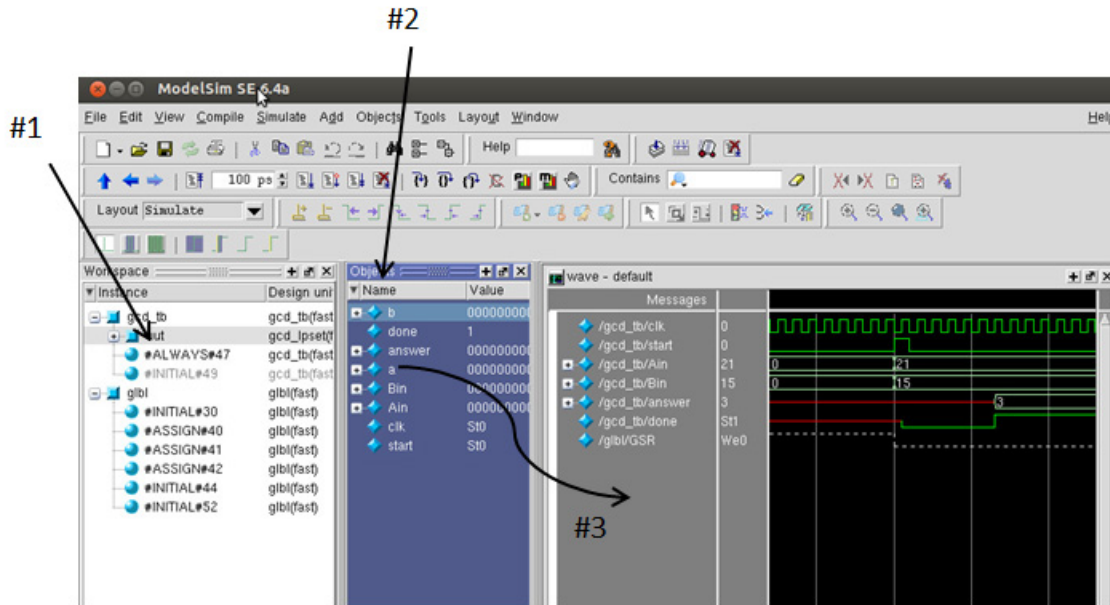
Step 4: Take a screen shot showing `r[15:0]` after 32 bits are shifted in and a screen shot showing `r[15:0]` when **done** is asserted. Use hex radix for `r[15:0]`. Include the Verilog and screen shots in one pdf file. Upload to the course website.

Lpset grading rubric

Grading	
1	Easy to read & formatted Verilog (See "Verilog Editors" tab for help.)
1	Correct use of blocking/non-blocking assignments
1	Comments in Verilog when needed
3	Functional Verilog & test bench
2	Screen Shot 1 <code>r[15:0]</code> after 32 bits are shifted in
2	Screen Shot 2 <code>r[15:0]</code> when done is asserted
10	Total Grade

In simulation, state values are unknown unless explicitly set. (Unknown values are shown in red during simulation. Outputs not defined are shown in blue.) For a simulation to run correctly, state variables must be initialized or set to some value at some point in the simulation. This can be accomplished by using a reset (recommended) or other input. For the CRC you can use the **start** pulse to initialize the CRC:

In simulation, by default, only inputs and outputs from the unit under test are displayed in the Wave window. It may be useful to display internal wires in your module that are not inputs nor outputs, for example, a bit counter. After running the initial simulation, to display the internal wires, click “ uut ” (unit under test) in the Workspace window (#1). This will display the internal signals in the Object window (#2). Drag the desired signals to the Wave window (#3).



To display the additional signals, rerun the simulation. In the Transcript window, type

```

restart -f // force a restart
run 2000ns // run simulation for 2000ns (longer if needed)

`timescale 1ns / 1ps
/////////////////////////////////////////////////////////////////
//
// lpset CRC test bench 9/29/2017
//
/////////////////////////////////////////////////////////////////

module crc_tf;

// Inputs
reg clock;
reg data_clk;
reg start;
reg data;

// Outputs
wire done;
wire [15:0] r;

// Instantiate the Unit Under Test (UUT)
lpset6 uut (
    .clock(clock),
    .start(start),
    .data(data),
    .done(done),
    .r(r)
);

```

```

// this is the input data
reg [47:0] input_data = 48'h03_01_02_03_30_3A;
integer i;          // required for "for" loop

initial begin      // system clock
    forever #5 clock = !clock;
end

initial begin      // data_clk, ensures setup time met
    #2
    forever #5 data_clk = !data_clk;
end

initial begin
    // Initialize Inputs
    clock = 0;
    data_clk = 0;
    start = 0;
    data = 0;

    // Wait 100 ns for global reset to finish
    #100;

    // Add stimulus here
    start=1;
    #10 start = 0;
    #5;
    //forever #5 data_clk = !data_clk;
    for (i=0; i<48; i=i+1)
    begin
        data = input_data[47];
        // at each clock, left shift the data
        // note syntax for test bench "for" loop - no "always"
        // note the blocking assignment (immediate)
        @(posedge data_clk) input_data = {input_data[46:0],1'b0};
    end

    $stop; // Pause simulation
end

endmodule

```

A link to the test bench is posted on the course website.

You can verify your design with different inputs by comparing your results with a CRC calculator website http://www.sunshine2k.de/coding/javascript/crc/crc_js.html.

Select CRC-16, use custom CRC parameterization:

Uncheck "Input reflected", uncheck "Result reflected"

Polynomial 0x8005

Initial value 0xFFFF

Final XOR value 0x0000

CRC input data (bytes) 0x03 0x01 0x02 0x03 0x30 0x3a