On-line turn-in. Turn in a brief report AND the Verilog file for your design (not your test fixture or synopsys files) using wolfware. These will be checked using Code Comparison tools. If your code is substantially similar to someone else’s you will both receive an academic violation for cheating.

Question 1

Note, tutorial 3 gives another example of a complex design and how it is executed. You do NOT have to complete tutorial 3 though. You do NOT have to complete the SPEF or SAIF post-synthesis flows.

You are to design a module that accumulates statistics on an incoming data stream consisting of two individual bytes. The I/O to the module are as follows:

```verilog
input clock;     // Clock
input reset;     // synchronous reset - active low
input clear;     // Clears statistics when high (synchronous)
input [7:0] DataIn1;     // Input Data 1
input [7:0] DataIn2;     // Input Data 2

// all outputs are registered
output [7:0] EvenParity; // # of data with Even parity
output [7:0] GreyCode;   // # of data with pattern 10101010 or 01010101
output overflow;       // =1 if any of the counters above overflow, stays high until clear goes high
```

Thus, an example of this module running might behave....

| clock | _|_|_|_|_|_|_......_|_|_|_|
| clear | _|_|_|_|_|_|_|
| DataIn1 | 02  AA  AA | ...... | 02 |
| DataIn2 | 03  03  55 | ...... | 03 |
| EvenParity | 00  01  03  05 | ...... | FF  00 |
| GreyCode | 00  00  01  03 | ...... |
| Overflow | 0 | ...... | 0  1 |
Design, verify, synthesize a module that meets these specifications.

Please turn in the following:
- A drawing of your design.
- A fully commented Verilog listing of your design and test fixture. (Make sure you verify the functionality of overflow.)
- Final report_timing summary (from Synopsys).
- Final area (from Synopsys) - use report_area;

[50 points]