LSU EE 4755 Homework 4 Due: 1 November 2017

For instructions visit `http://www.ece.lsu.edu/koppel/v/proc.html`. *For the complete Verilog for this assignment without visiting the lab visit* `http://www.ece.lsu.edu/koppel/v/2017/hw04.v.html`.

Problem 1: A *run* of characters in a sequence is a set of consecutive characters that are the same, the length of a run is the number of times the character is repeated in the run. For example, the string `aabbbcaaaa` contains four runs: a run of length 2 for character `a`, a run of length 3 for character `b`, a run of length 1 for character `c`, and a run of length 4 for character `a`. (Note that `a` has two runs.)

Module `maxrun`, when completed, will keep track of the maximum-length run in a sequence of characters appearing at its `in_char` input. In this assignment *character* refers to a c-bit value. The testbench (including the excerpt below) shows character values as two-digit hexadecimal numbers. For example, at cycle 2 the table shows `c=8d` for `in_char`, meaning that the character value is $8d_{16} = 141_{10}$. The fact that the character can be represented using two hexadecimal digits or three decimal digits does not change the fact that it is one character.

At the positive edge of input `clk`, `maxrun` will compare the character at `in_char` to the character seen at the previous positive edge. If they are the same it will increment a *current run counter*, if the characters are different it will set the current run counter to 1. If `reset` is `1` at the clock positive edge then the current run counter should be set to 1 (which is the same as setting it to 0 and then incrementing it). A second *max run counter* is also set to zero on the `reset` signal at the positive clock edge. If the current run counter is greater than the max run counter then the max run counter is set to the current run counter, and the character appearing in that run is remembered and used used for output `mr_char`.

If input `mr` is 1, then output `len` should be set to the value of the max run counter, otherwise it should be set to the value of the current run counter. This should be done asynchronously (`len` should be updated whenever `mr` changes, not just at a positive clock edge).

For example, look at the output of the testbench below. Column `R` shows the value of the reset signal and column `in_char` shows the input character, both appearing before and "during" the positive clock edge. The remaining columns show the value of the current run counter, max run counter, and the `mr_char` output after the positive clock edge. At cycle 1 the input character, `00`, has a run of 2. At cycle 6 character `8d` has reached a run of 5, etc. The testbench shows an `ok` if the output is correct, otherwise it shows what the correct output should be.

```
Cycle R in_char current-run          max-run             mr_char
----- - -----   -------------       -------------       ----------
    0 r c=00    cr_len   1 ok       mr_len   1 ok        mr_c 00 ok
    1   c=00    cr_len   2 ok       mr_len   2 ok        mr_c 00 ok
    2   c=8d    cr_len   1 ok       mr_len   2 ok        mr_c 00 ok
    3   c=8d    cr_len   2 ok       mr_len   2 ok        mr_c 00 ok
    4   c=8d    cr_len   3 ok       mr_len   3 ok        mr_c 8d ok
    5   c=8d    cr_len   4 ok       mr_len   4 ok        mr_c 8d ok
    6   c=8d    cr_len   5 ok       mr_len   5 ok        mr_c 8d ok
    7   c=c5    cr_len   1 ok       mr_len   5 ok        mr_c 8d ok
    8   c=77    cr_len   1 ok       mr_len   5 ok        mr_c 8d ok
    9   c=f2    cr_len   1 ok       mr_len   5 ok        mr_c 8d ok
   10 r c=f2    cr_len   1 ok       mr_len   1 ok        mr_c f2 ok
   11   c=f2    cr_len   2 ok       mr_len   2 ok        mr_c f2 ok
   12   c=f2    cr_len   3 ok       mr_len   3 ok        mr_c f2 ok
```

```
13   c=f2    cr_len   4 ok        mr_len   4 ok        mr_c f2 ok
14 r c=f2    cr_len   1 ok        mr_len   1 ok        mr_c f2 ok
15   c=9d    cr_len   1 ok        mr_len   1 ok        mr_c f2 ok
16   c=0d    cr_len   1 ok        mr_len   1 ok        mr_c f2 ok
17   c=d5    cr_len   1 ok        mr_len   1 ok        mr_c f2 ok
```

Complete the `maxrun` module so that it passes the testbench and is synthesizable. Please pay attention to the parameters, which indicate the size of a character and the number of bits in the counters. Use the parameters, not their default values.

Use `simvision` for debugging, which is explained in the course procedures page.

Problem 2: Run the synthesis script, using command `rc -files syn.tcl`. If it runs correctly, a file `spew-file.log` will be created which contains a timing report for a design. On paper or in comments in the submission file, indicate where the critical path is in your design.

Provide suggestions on making it faster, or explain what you actually did for a high clock frequency.