LSU EE 4755 **Homework 4** **Due: 4 November 2022**

For instructions visit `https://www.ece.lsu.edu/koppel/v/proc.html`. *For the complete Verilog for this assignment without visiting the lab follow* `https://www.ece.lsu.edu/koppel/v/2022/hw04.v.html`.

Problem 0: Following instructions at `https://www.ece.lsu.edu/koppel/v/proc.html`, set up your class account, copy the assignment, and run the Verilog simulator and synthesis program on the unmodified homework file, `hw04.v`. Do this early enough so that minor problems (*e.g.*, password doesn't work) are minor problems.

Helpful Past Homework Assignments
For those who would like to see a fairly simple sequential circuit, and one that counts characters, see 2017 Homework 4, maxrun.

Problem 1: Module `word_count` has three inputs, an 8-bit `char` input, and 1-bit inputs `clk` and `reset`. At each positive edge of `clk` a new ASCII character will be available at input `char`. The characters might be from a text file, a keyboard, or some other source of English text. Based on the word rules given below these characters form words, and the module is to count the words and provide other information.

Module `word_count` has three parameters, `wl`, `wn`, and `n_avg_of`. The module has six outputs. Output `len_word`, which is `wl` bits, is the length so far of the current word, or the length of the most recent word. Output `n_words`, which is `wn` bits, is the number of complete words counted since the last reset.

Output `len_avg`, which is also `wl` bits, is the average length of the `n_avg_of` most recent completed words with the fractional part truncated. **If fewer than** `n_avg_of` words have ended since the last reset then `len_avg` should be zero. For example, if `n_avg_of=4` and the lengths of the four most recent words are 8, 4, 12, and 15 then `len_avg` should be set to $\lfloor(8+4+12+15)/4\rfloor = \lfloor 39/4\rfloor = \lfloor 9.75\rfloor = 9$. If there is a reset and then only three words have ended, `len_avg` should be `0`.

Output `word_start` should be set to `1` iff the current character starts a word. Output `word_part` should be set to `1` if the current character is part of a word based on the word rules described further below. (If `word_start` is `1` then `word_part` is `1`.) Output `word_ended` is 1 if the character in the previous cycle was the last character of a word.

For an example of how these output should be set examine the testbench output below, collected for the text "A or bee":

```
       W-M    I  Text---->!         SPE L  N A {D}
Trace  2-5    0 "          A"    ___ SP_ 1  0 0 {1}
Trace  2-5    1 "         A "    sp_ __E 1  1 0 {0}
Trace  2-5    2 "        A o"    __e SP_ 1  1 0 {1}
Trace  2-5    3 "       A or"    sp_ _P_ 2  1 0 {1}
Trace  2-5    4 "      A or "    _p_ __E 2  2 1 {0}
Trace  2-5    5 "     A or b"    __e SP_ 1  2 1 {1}
Trace  2-5    6 "    A or be"    sp_ _P_ 2  2 1 {1}
Trace  2-5    7 "  A or bee"     _p_ _P_ 3  2 1 {1}
Trace  2-5    8 " A or bee "     _p_ __E 3  3 2 {0}
Trace  2-5    9 "A or bee 2"     __e ___ 3  3 2 {0}
Trace  2-5   10 " or bee 2n"     ___ ___ 3  3 2 {1}
```

Each line shows the output at one cycle, the `I` column shows an index (which is something like a cycle number). The `W` column shows the value of `n_avg_of` and the `M` column shows the maximum possible word length. The last column, `{D}`, is for debugging, see the discussion further below.

The most-recent ten characters are shown under the `Text` heading, in the first line (index 0), `A` is the most recent character. There will be an `R` to the right of the text in a cycle when `reset` is `1`.

The `L` column shows the length of the word so far, or the length of the most recent word. The `N` column shows the number of words (incremented when the word ends), and the `A` column shows a running average of the last word lengths, the last 2, in this case. The column headed `SPE` shows the state of the outputs of `word_start`, `word_part`, and `word_ended` outputs. An upper case letter shows the state after the positive edge of the clock (which is the one that is needed). To help with debugging, the lower case letters show the state just before the positive edge.

Note: `word_part` should only be `1` if `char` is a word-part char and a word has already started. Notice that at index 10 the arriving character is an `n`, which is a word-part character. But because it was not preceded by a non-word-part character a word does not start at index 10 (nor 9).

Notice that `L` is updated as each character arrives, while `N` and `A` only update when the word ends.

The testbench will trace the first few lines, and then only show trace lines when there are errors (along with a few trace lines preceding the error). For lines with an error the correct output is also shown:

```
       W-M    I  Text---->!         SPE L  N A
Trace  2-5    5 "    I II I"    __e SP_ 1  2 1
Trace  2-5    6 "   I II II"    sp_ _P_ 2  2 1
Trace  2-5    7 "  I II III"    _p_ _P_ 3  2 1
Trace  2-5    8 " I II III "    _p_ __E 3  3 3 <- Error  Correct -> __E 3  3 2
       W-M    I  Text---->!         SPE L  N A                      SPE L  N A
```

In the example above, the running average, `A`, is wrong. The module output is 3 but the testbench expects a 2.

Reset Behavior

If input `reset` is `1` on a positive edge then `len_word`, `num_words`, and `len_avg` should all be set to zero and input `char` should be considered a non-word character (regardless of its value). The trace below shows an example of reset behavior. The reset occurs at index 6. Because of when the reset occurs `bee`, rather than being a three-letter word is considered a one-letter word, the last `e`. Notice also that the average length (column `A`) does not show a value until two complete words arrive.

```
       W-M    I  Text---->!         SPE L  N A {D}
Trace  2-5    3 "      A or"    sp_ _P_ 2  1 0 {1}
Trace  2-5    4 "     A or "    _p_ __E 2  2 1 {0}
Trace  2-5    5 "    A or b"    __e SP_ 1  2 1 {1}
Trace  2-5    6 "   A or be"  R sp_ ___ 0  0 0 {1}
Trace  2-5    7 "  A or bee"    ___ SP_ 1  0 0 {1}
Trace  2-5    8 " A or bee "    sp_ __E 1  1 0 {0}
Trace  2-5    9 "A or bee k"    __e SP_ 1  1 0 {1}
Trace  2-5   10 " or bee kn"    sp_ _P_ 2  1 0 {1}
Trace  2-5   11 "or bee kno"    _p_ _P_ 3  1 0 {1}
Trace  2-5   12 "r bee knot"    _p_ _P_ 4  1 0 {1}
Trace  2-5   13 " bee knot "    _p_ __E 4  2 2 {0}
Trace  2-5   14 "bee knot  "    __e ___ 4  2 2 {0}
```

Testbench Information

The testbench will instantiate and test `word_count` at three different sizes, varying both the value of `n_avg_of` and the maximum word size. The values of `n_avg_of` will be 2, 1, and 9. To change these sizes search for `pset` in `hw04.v`. Several items in the testbench can be changed to facilitate debugging and familiarization. Search for `HW04` and read the comments for more info. The testbench will start streaming characters from the string `test_one`, and after that will construct a stream of random characters. Feel free to change `test_one` to facilitate debugging.

The testbench shows the first few errors encountered, and then silently tallies errors. After each instantiation is tested a summary of errors is shown:

```
Trace  9-7   10 " or bee   "    ___ ___ 3  3 0 {0}
Trace  9-7   11 "or bee    "    ___ ___ 3  3 0 {0}
Done with n_avg_of=9, max wd len=7. Errors: st 0, pa 0, en 0, nc 0, nw 0, av 0
```

The line starting `Done` shows a tally of errors by type after the word `Errors`. Six types of errors are tallied (all have zero errors in the output above). They are `st`, the `word_start` output, `pa`, the `word_part` output, `en`, the `word_ended` output, `nc`, the `len_word` output, `nw`, the `num_words` output, and `av`, the `len_avg` output. Remember that the line describes one instantiation, so there should be three lines printed.

The trace can be helpful for looking at values of objects in your module (not just inputs and outputs). As an example, the trace shows the value of object `char_az`, but feel free to change that or add others. To do so search for `wd_cnt.char_az`. It appears as an argument to `$sformatf` which prepares part of the trace text. Here `wd_cnt` is the instance name that the testbench uses for `word_count`. Change or add arguments to `$sformatf` to examine additional objects in your module. Be sure to change the format string to match the arguments. The end of the format string, the part in curly braces, handles the last argument `wd_cnt.char_az`.

The value of `wc` will always be chosen so that output `len_chars` never overflows. It is unlikely but not impossible that the number of words is too large for `wn`.

Word Rules

A character is an 8-bit quantity. A character is called a *word-start character* if it an ASCII alphabetic character (upper or lower case). In `word_char` net `char_wd_start` is set to one if the `char` input is a word-start character. A character is called a *word-part character* if it an ASCII alphabetic character (upper or lower case), a digit, or an underscore character. The `word_count` module net `char_wd_part` is set to one if the `char` input is a word-part character. Note that all word-start characters are word-part characters.

A word starts when the current character is a word-start character and the previous character was not a word-part character or if the module was reset in the previous cycle. A word ends when an arriving character is not a word-part character.

The length of a word is the number of characters. The output `len_word` should only be zero after a reset and until the next word starts.

Design Requirements and Goals

As always, avoid costly designs. Pay particular attention to the logic computing `len_avg`. Do not use `n_avg_of-1` adders to compute this. And definitely don't use `n_avg_of` division units.

The design can use procedural code, but it must be synthesizable. Use command `genus -files syn.tcl` to synthesis. Timing and area (cost) reports will be placed in a file named `syn-report.log`.