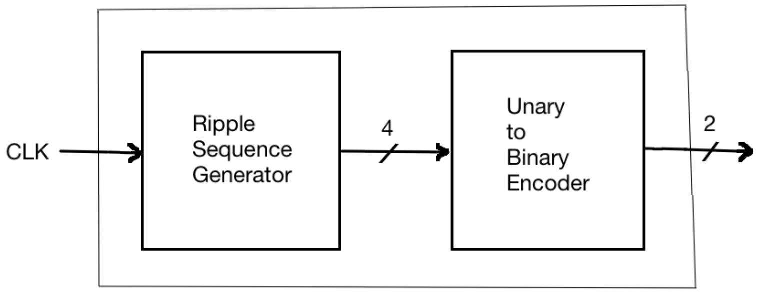
Sequential Logic: Sequence Generator

The use of memory constitutes a sequential-logic circuit. The memory in Verilog programming is the use of `reg` to declare bit storage which can contain a *state* of the electronics, and can be used as part of the input to generate the next state in time driven by clock cycles. Example of sequential circuits are counter, sequence generator, and sequence recognizer.

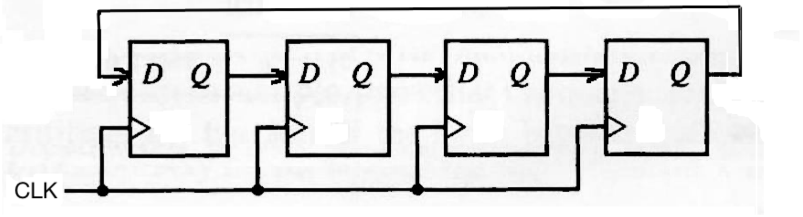
In this assignment we will program a sequential circuit in Verilog code as a sequence generator that generates, instead of binary counts, the **letters in your full name (given name and family name combined)**, character by character. Display the sequence for at least **two cycles** (run the demo to match how it runs with your code).

The ripple counter is an example that illustrate the sequential nature of the circuit: the value 1 is passed among several bit registers, and the collective output is a *unary* pattern which is then *encoded* into the binary format by way of an *encoder* (which does the inverse function of a decoder). See the diagram below.

Ripple Counter



Ripple Sequence Generator



Suppose the output of the ripple generator is an arrray of Q. Then, the truth table of encoding is:

Q0	Q1	Q2	Q3	C1	C0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

```
C1 = Q2 or Q3
C0 = Q1 or Q3
```

There are other counters can be used like InverseCounter or Modulo4Counter. Here we used ripple counter.
The supposed *Runtime Output*.
The pseudo code of *counter.v*.

See in the demo part of code how the clock signal can be given correctly. In the ripple counter, all counting registers are initially set zero except the first is set 1 to be passed to the next register upon new clock cycles. Your programs should generate your full name.

Must use arrays at all places that are possible, and **never use "if-else" statements** anywhere in your program. Submit your completed source file **name.v** only into folder **2ndPrgAssig** that is under your named folder on host Voyager.