**Guide**:

* There should be a signal that holds the current hardware state
* A state transition function operates on the current state provided by the signal
  + This function should not use an applicative.
  + This function is appended with “Proc” as the current
* The Proc function must take a state and as argument
  + Usually, if there are any inputs, then the Proc function must take a PIn
    - A PIn will be defined as containing inputs needed to create the next state, unless it makes sense to include a value as external to the Pin
      * Typically other arguments are there only if a value needs a history to be computed
        + e.g. Is this a rising edge?
        + Functions that require a history need to use a Signal Applicative OR store the necessary previous values in the state
  + State will contain both the output and internal values
    - A function can be made to retrieve only the desired outputs

**Notes**:

* Avoid using a Signal Applicative if possible
  + We can always turn values into signals, it’s majorly unadvisable to remove the signal applicative from a value in CLaSH/Haskell.
* “topEntity” is where CLaSH begins its HDL generation
  + Only functions that are encapsulated within topEntity get generated into a HDL
  + Most tests will be outside of this encapsulation
    - This means you can revert to using more traditional Haskell
    - Things built using Template Haskell also do not need to adhere to CLaSH rules, as they are run before CLaSH begins HDL generation

**GIT:**

* All code pushed to master needs
  + To have been looked at by another person
  + The ability to compile
* You may place broken code into a dev branch

**Things to fix/Add:**