**Authors: Colin Fritz and Devon Stedronsky**

*Branching Strategy for Version Control:*

Note: Version control shall be performed on a Git platform of a team’s choosing (e.g. Bitbucket, GitHub, etc.) and as such, this branching strategy shall apply for Git protocol structure.

* Branching strategy will follow git-flow principles as outlined here <https://nvie.com/posts/a-successful-git-branching-model/>
* Remote repo serves as the ground truth for the current state of the project
* Admin(s) chosen will resolve merge conflicts and approve pushes to the remote repo to organize the growing commit history
* Source code (design and sim) will be tracked in the remote repo only to avoid huge numbers of merge conflicts that occur when tracking the Vivado project files.
* The remote repository will include a documentation folder containing functional descriptions, specifications, standards, requirements, test cases, and simulation results.
  + Team members are expected to update the project schedule within the documentation folder at each deadline.
* The main, stable version of the program shall be represented by the master branch.
  + Only admin(s) shall push changes to the master branch.
  + Only fully verified versions of the project shall be pushed to the master branch.
  + Commits to the master branch must be tagged with a version number
  + Only a merge target for release and hotfix branches.
* Feature branches are for continuous development of the development version of the project.
  + These branches are used for development of assigned modules and features.
  + May not be completely functional or able to compile at any given time.
  + Each team member is expected to retain local copies of changed files to ensure work is not lost due to merging conflicts.
  + Only merge target for feature branches is the develop branch
  + All merges into develop branch will be via merge commits
  + Feature branches must be deleted immediately after merge commit into the develop branch, create new branch to restart work off the develop branch
* The develop branch will be the merge target for any feature branches, hotfixes, or finished release branches.
  + Multiple commits with less than 20 lines of edited source code shall be rebased to a single commit before merging with the develop branch.
    - This will reduce clutter and make the change log more readable.
* Release Branches will be created from the develop branch when it is believed a fully verified and functioning version of the project exists.
  + Finish by merge commit with the master and develop branches.
  + Each release branch is intended to have a new version number