Moving interpinic into CLM

For a number of reasons, we are considering moving interpinic into CLM.

Here are some reasons:

1. Running interpinic can be a difficult part of setting up a new CLM production run
2. Maintaining all of the initial conditions files is somewhat painful; e.g., currently you need to run interpinic to regenerate all out-of-the-box initial conditions files whenever changing CLM's memory layout
3. With CESM moving to online regridding, one problem is what to do with CLM's initial conditions when you don't know the landmask ahead of time
4. This could be leveraged by dynamic landunits, when a new landunit comes into existence.
    a. This is one of a number of proposed ways to initialize a new landunit. Some of the other approaches may be simpler and more scientifically justifiable, but different people have different opinions on this. This means that an interpinic-based initialization may never be used, but if interpinic was being moved into the CLM code anyway, it's possible that people would want to leverage it for dynamic landunits too.
5. interpinic is in need of an overhaul anyway, so this is a good time to think about totally reworking how we handle interpinic

Here is a proposed algorithm for parallelizing interpinic - in particular, the time-consuming piece that finds nearest neighbors:

```
For each point on the destination grid (all tasks work on the same destination point at the same time):
  - Task responsible for this destination point broadcasts its info (lat/lon and type) to all other tasks
  - All tasks loop through their source points, finding closest point of the correct type
  - Use a parallel reduce to find the absolute closest point (across all tasks), and which task owns that point
```

If we plan to use this for the initialization of new grid cells with dynamic landunits (which requires running this interpinic algorithm at runtime rather than just at initialization), note that we might need more generality than is needed if it is only used at initialization. In particular, there may be a larger set of variables that need to be interpolated at runtime compared to the set that needs to be interpolated at initialization - in particular, because I could imagine there being some variables that are computed based on other variables in initialization. e.g., variables A and B may be on the restart file, and then variable C is computed based on the values of A and B in some initialization code; in this case variable C would also need to be included when doing interpinic at runtime. But I don't know if this is really the case for any variables.

There are a number of bug reports that should be addressed by this work - search bugzilla for 'interpinic'.