XMAS DESIGN TOOL

This project is a design tool for NoC system (Network on Chip). It uses the language Intel detailed in papers concerning xmas

To generate the PDF version of this document use the command:

pandoc -f markdown -o readme.pdf readme.md

Directories (Contents)

- minutes The minutes, only the TeX sources.
- preparation Files used during preparation plus plan
- xmas-info Information on xmas (pdf)
- design The directory containing design documents
- src The source tree containing all sub projects.

License

Use of development environment

We currently use QtCreator because of its ease of project management (qmake). QtCreator also uses cmake if instructed to.

Use of Git

Configuring Git

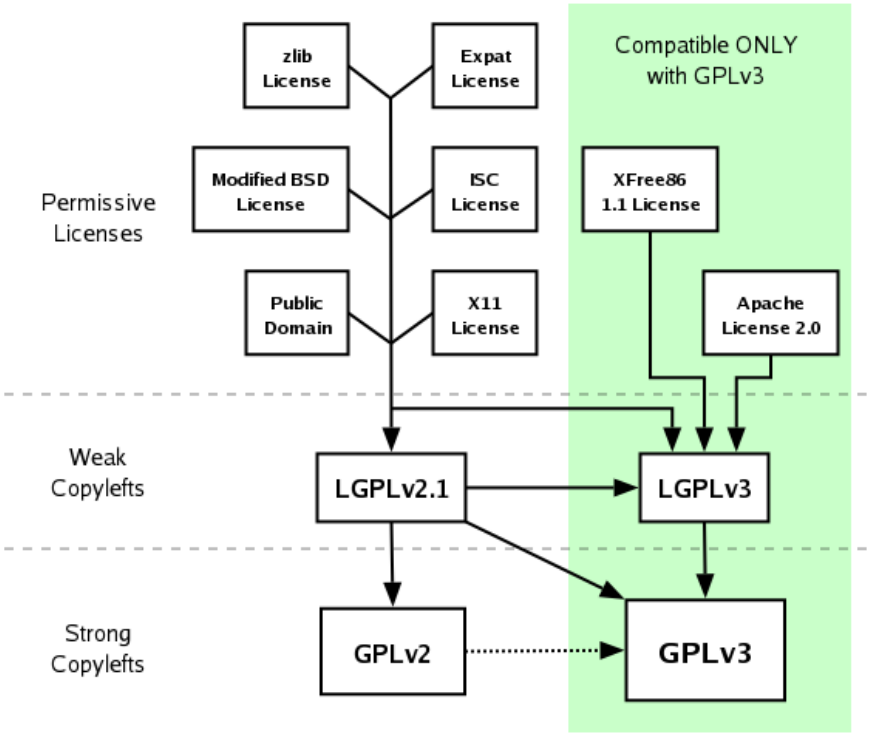

Figure 1: GPL dependencies in short

We are developing in a heterogeneous environment with both Unix and Windows machines. Therefore it is imperative to have all files in the repository in the canonical LF-only format. Therefore, Windows users must configure their editor to create the unix LF-only files in their sources. Also, Windows users should run:

```
$ git config --global core.autocrlf true
```

to automatically get CRLF line endings which are suitable for the native tools, and Unix users should use

```
$ git config --global core.autocrlf input
```

(this is a safety measure for the case where files with CRLF line endings get into the file system – this can happen when archives are unpacked, attachments saved, etc.).

To be able to create commits which can be pushed to the server, you need to set up your committer information correctly:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "me@example.com"
```

Please do not use nicknames or pseudonyms instead of the real name unless you have really good reasons.

We use git for communication with the following use cases

1. New repo Clone the github directory using git as described below. *nix users from a bash shell, ms windows from git bash.

   machine/xmas> git clone https://github.com/gbonnema/xmas

   If you can become collaborator, you should be able to push your updates to the same URL.

   If you cannot, then you could fork the repo and do a pull request.

Remark: We work with git flow, which is a specific approach to branching. The following description does not presume git flow. If you want to know how git flow works google "git flow". You should find at least the article of Vincent Driessen at http://nvie.com/posts/a-successful-git-branching-model/ A fine intro to git flow is: http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/ A cheat sheet is: http://danielkummer.github.io/git-flow-cheatsheet/

2. Creating a new branch. You create a new branch "plan" with the command

- git checkout -b plan

do the alterations and commits until the modifications are finished then push the commits to remote

- git push –set-upstream origin plan

If you forget the option "–set-upstream" git will issue an error because it does not recognize the branch yet. Once the branch is created on remote with "–set-upstream", you can do a "git push".

3. Working from a branch. While working from a branch, sometimes master is updated and if you want to be current with master, you need to merge the changes in master into your branch.

   As long as the branch is only local you can do a rebase (see several tutorials for how to do this). If you already published the branch, then it is better to merge master into the branch. That way you will not deviate too much from master.

   You switch to a branch with "git checkout branchname" and back to master with "git checkout master". Make sure you only switch with a clean working directory i.e. after a commit and with no changes unstaged.

4. Collaborator Clone the repo, use it as you normally would with change-commit cycles and when you want to publish do a git push. If you set up git remote properly, then the github repo should be updated.

   If you want to update your working directory with any changes from github, make sure you committed and then do a git pull. Again, remember to set up git remote.

   Sometimes a "git push" will result in an error, because your working directory is not updated with the latest changes from remote. Make sure to do a "git pull", resolve any conflicts, do a commit and a push to get your committed changes to remote.

5. Any github user If you are using the system and want to update it, you can clone the repo locally, make the changes with the usual change-commit cycles. When done (including documentation and tests) you can do a pull request.

Documentation

We use two document formats: LaTeX and markdown.

Project documentation

We use LaTeX for most of the project documents like the plan, a tutorial or a guide. Any kind of document that needs to look good on paper and is more than just a screenful.

System documentation

We use markdown with pandoc format for most of the system documentation

Development process

We use Disciplined Agile Delivery (DAD) for our process and use a cloud based agile management tool to visualise our work and to keep track of the progress.

Compile and install

- README-LINUX.md The instructions to develop from one of the linux platforms

Dependencies

This project depends on the following libraries:

Qt5.4 We specifically use Qml 2

Remark that we use as much from the Qt libraries as is possible.

Tips

- Set the warnings in GCC to Wall, and use the extra warnings in Settings ... Compiler ...
- When using Code::Blocks be sure to increment the option "number of processes" from Settings ... Compiler .... and the tab build options (usually not visible directly, you have to scroll toward it using the arrow).
- Switch the hmtl logging for building on (Settings ... Compiler ... build option tab).

- Use the tool configuration option (Tools ... Configure Tools ...). It starts a dialog where you can add any tool you have on your system and use standard macros to denote directory name, filename etcetera. As an example we use "git diff HEAD ${ACTIVE_EDITOR_FILENAME}" with the standard option "Launch tool in a new console windows and wait for a keypress when done".

This will show the exact output including colouring that we would get using the console.

I also use "gitk" without parameters and with the option "Launch tool visible (without output redirection)". This will start the graphical output for gitk.