Report 3

Gregor Robertson

27 Oct 2016

Progress This Week

As I mentioned in my last report, this week I continued to look over different tutorials and decided upon a few ways that it might be possible for me to progress through to the next step.

One way that it would be possible for me to implement this would be to edit each stage of the compiler to handle a pragma which we define ourselves, similar to how [1] describes doing it in the second section. This has the advantage that it would allow more control and have the ability to provide more information about each stage. However, this also has the major disadvantage that it would involve editing the files of the Clang compiler, meaning that the changes would need to be reimplemented each time a new version of the compiler was installed. Even with a manual to tell someone how to implement the changes, or even a script to automatically place them in, this would be far from ideal.

Another possibility would be to make use of the __attribute__((annotation(""))) method with some predefined format. This has the advantage of meaning that the implementation would not need to touch Clang and instead would be limited to only running as an LLVM pass. At the moment, I do not know how to read the annotations uring a pass, but [4] provides some hints and I am sure that it would not take a long time to find a way to do this.

The third possibility is that the user has to have a file in the same directory (or possibly specify a file in the command line args) which would follow a standard format (say JSON) and have a special extension (say .fpga) to help identify the file (a special name would also work). I believe this could work as any tutorial I have seen, as well as the skeleton of an LLVM pass in [3], show that the code for a pass is just standard C++ code, meaning that I can not see why it would not work any other way. This has the same advantage as the version above, as well as having the benefit of separating the concerns.

A possible fourth method would be to combine the second and third methods together, using some form of preprocessing to create the extra file which would then be used later on. This would require some form of preprocessor and a new pass in LLVM, but has the advantage over the first method of being more modular.

From what I can tell, I will be using C++ as that is how you can interact with LLVM. To interact with Clang it is possible to use other languages, and the

method to use the other languages is more stable, but it is also not as powerful as it does not allow full access to the AST for Clang.

I have also installed all of the packages that I should need, based on [2], in order to progress.

Aims For Next Week

I think that seeing as me and Dr Syed Waqar Nabi are having a meeting tomorrow, that it would be best to wait until I discuss with him what my next move should be. At the moment I would say that the best thing would be, more or less, to write the LLVM pass required to add the metadata to each instruction as a proof of concept. I also think that it might be worth while cleaning up the reports that I have written so far and creating a more useful collection of links as my current method is not very descriptive of what is in them (appologies for that).

Questions

- Dr Vanderbauwhede, is there any method in the ones that I listed above that you would prefer?
- Would either of you happen to know anyone who owns a copy of "LLVM Cookbook by Mayur Pandey and Suyog Sarda"? If so, would it be possible for me to borrow it?
- Is it okay for me to put this project on github as a non-private repo? I can place it as private should you wish, I am mearly curious.

References

[1] Serge Guelton. *Implementing a Custom Directive Handler in Clang*. 2016. URL: `http://blog.quarkslab.com/implementing-a-custom-directive-handler-in-clang.html`.

[2] LLVM Project. *Getting Started with the LLVM System. Example with Clang*. URL: `http://llvm.org/docs/GettingStarted.html#example-with-clang`.

[3] sampsyo. *llvm-pass-skeleton*. URL: `https://github.com/sampsyo/llvm-pass-skeleton`.

[4] user2022455. *llvm get annotations*. URL: `http://stackoverflow.com/questions/15114284/llvm-get-annotations`.