Project Proposal

2359A

September 2021

1 Introduction (Project description)

In my project, I will develop a tool for easy device building from modules in Bluespec. To do this I will need to create a library that can use the Bluetcl tool to extract data from .bo package files. My project will use existing .bo (and .ba) package files which represent modules in their pre-elaboration form (.ba are after elaboration but they behave in a comparable manner they contain more information but because they are translated to hardware description language, they can't capture concepts like polymorphism). Bluetcl is a tool made in Tcl language which is a bit obscure and by creating this library I will also create a python wrapper for Bluetcl which could be useful for further automation of the generation of Bluetcl files.

After extraction, I will have an internal representation of different modules and their interfaces, this will be later used to synthesize top-level files, thanks to typing information I will be able to make sure to synthesize only connections between modules that are possible (at least according to typing information).

One of the larger difficulties of this project will be the integration of shared busses, they allow for the connection of multiple slaves and masters in some orderly fashion, they come with extra complexities coming from the fact that some devices have different widths of connectors, but you might still want to connect them together via bus and module converting widths.

After all of this, we can tackle the main goal of this project which is creating an interface for synthesizing those top-level files. There are currently alternatives like .qsys files but they can easily run in tens of megabytes, and while they store a lot of other information that my format will not, it doesn't change the fact that for humans such bodies of text are effectively not readable nor writable. I plan to create a simpler format that is more human-oriented. Ofc this file format won't be read by any other software but because those synthesized top-level files will be written in Bluespec they can comply with Verilog (thanks to Bluespec compiler).

If all of this goes smoothly, I should have quite a bit of time left over for an optional goal, If I will be taking on QSys text file format I might as well go after QSys itself this means creating GUI for adding and connecting components. Everyone who did practicals or otherwise had a chance to work in QSys knows that the development of UI of this program stopped somewhere in the mid-2000s. It's ugly and difficult to navigate, with small targets for mice (some are few pixels wide). Since this is an optional goal, I plan here to focus on developing something quickly, so I plan to use established game engines like Unity3d or Unreal Engine to do this UI as I have already worked with them, and they allow for compilation for multiple platforms.

Evaluation will be done in form of a qualitative comparison between my solutions, QSys and QSys Pro. The criteria I will evaluate it on are ease of modification and readability. For more quantitative data I will use the number of lines and tokens used. I will prepare one or more tasks like Given two connected FIFOs, how much change is needed to connect 3rd FIFO and then solve them using all 3 solutions then explain differences. My

project will be successful if created tools will allow for solving those tasks more easily while still producing valid Bluespec that behaves in a reasonably similar manner (function/device comparison is exceedingly difficult and beyond scope of this project, if logic produces the same behavior, it's fine, we don't care about exact timings etc.)

I plan to use an iterative software development strategy where every 2 weeks or more often I will push out incremental changes or new features in accordance with the timeline.

2 Timeline

Whenever I mention here word package, I mean work package lasting 2 weeks. Start 15.10.2021

- 29.10.2021 - Understanding Bluespec and Bluetcl, during those 2 weeks I will read up on Bluespec and how it fits into the System Verilog ecosystem. At the end of those two weeks, my goal is to produce a document explaining the core of Bluespec typing, and the toolchain I will use to get typing information, and where this information will be fed back. This will be an exercise for me and a good introduction to what's going on for people unfamiliar with Bluespec reading this.
- 12.11.2021 - Busses, during this fortnight, make some reading on busses what they require and work on how to integrate them into my project. Goal produces a short document detailing how busses behave, how to represent them, what information is needed to synthesize a bus, and possibly some explanations of unforeseen difficulties. Thanks to doing this reading-up work early I will be able to use an iterative/spiral development strategy.
- 26.11.2021 - Extracting type information using Bluetcl, the goal here is to produce code that is capable of calling Bluetcl with parameters that allow for extracting data from .bo packages. Here time will be also spent creating an internal representation that allows for a large amount of flexibility. (Interfaces in Bluespec are a bit complicated, they have multiple functions, and those functions can have multiple ports for ready / data signals.) The goal of this package is a bit vague, but let's say that I will extract interface types, and interface functions, with parsed types.
- 10.12.2021 - Continuation of above. The goal here is that parsing is fully functional, and it creates an internal representation of modules, interfaces, functions, ports, types it can find in the package.
- 24.12.2021 Synthesizing Bluespec. Given my internal representation, I will synthesize a Bluespec that represents a top-level file, with modules used and connections between them. Goal has functionality in code that allows synthesizing: Creation of module, a connection between two modules, possibly some nonstandard connection that requires more fiddling (adding automatic widening/shortening in connections between busses and slave/masters). Plus, progress report.
- 07.01.2022 Creating text file interface, Goal use some existing file format like Yaml / XML / JSON to store and read how synthesized top-level file needs to look like. Extra time here will be spent to make sure this file format is both human-readable and human writable, as this will be the main way of interfacing with my system.
- 21.01.2022 - Evaluation Part 1, creation of files in other editors and write up on differences between them and my file format.
- OP1 (Optional goal) GUI / Interactive text, while editing text is fine, and can be done even from terminal it would be nice to have some way of creating a top-level file that uses typing information to guide the process. Goal: Create a GUI/Interactive text support.

- 04.02.2022 - (Optional goal evaluation) Here I will evaluate my GUI vs QSys, in terms of ease of use and flexibility.
- 18.02.2022 - Writing dissertation
- 04.03.2022 - Writing dissertation, final work package.

3 Determining special resources and checking their availability.

No special resources are needed.

4 Securing the services of a suitable Supervisor.

Jonathan Woodruff agreed to supervise my project.