DataLab. Internship Program (2023 Summer)

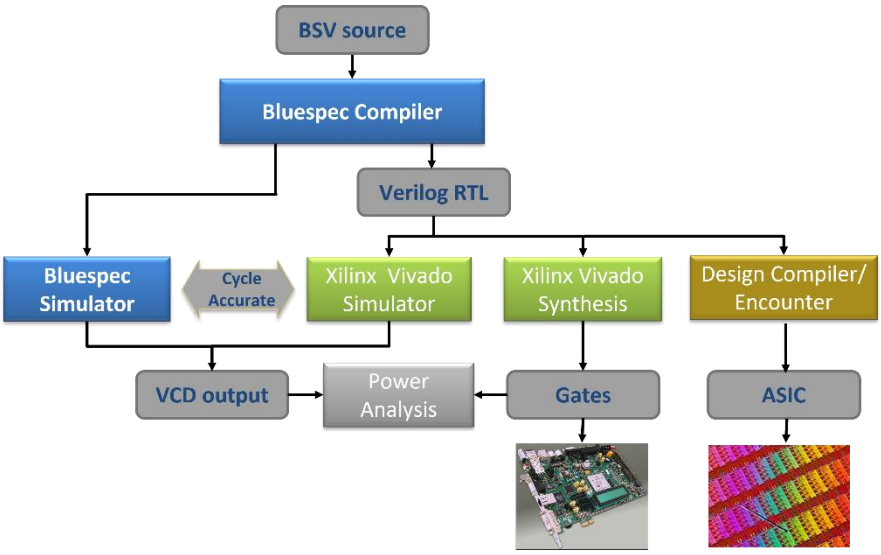
Week 3

Tutor: 배은태

High-Level HDL (HLHDL)

Verilog, VHDL 등의 Hardware Description Language(HDL)를 이용하면 RTL 수준의 하드웨어 디자인을 기술하고, 이는 다시 논리 합성을 거쳐 FPGA나 ASIC에 구현될 수 있습니다. 소프트웨어와 마찬가지로 하드웨어 역시 설계가 복잡하고 규모가 커질수록 코드 역시 길어지고 유지보수 비용도 증가합니다. 더 높은 생산성과 직관성을 갖는 고수준의 프로그래밍 언어가 등장했듯이 HDL 세계에서도 더 편리하고 생산성 높은 고수준 언어, 즉 HLHDL(High-Level HDL)이 등장했습니다.

HLHDL에는 대표적으로 Chisel과 Bluespec이 있습니다. Chisel은 함수형 언어인 Scala를 기반으로 하고, 오픈소스 ISA인 RISC-V의 오픈소스 구현체 중 하나인 RocketChip이 이 Chisel로 작성된 것으로 유명합니다. 한편 Bluespec은 Haskell과 유사한 문법을 가지는 BH와 Verilog와 가까운 문법 형태를 가지는 BSV(Bluespec System Verilog)라는 두 가지 스타일의 언어로 구성되어 있습니다. 둘은 interchangeable하다는 특징이 있습니다. BSV는 rule과 method, interface 등 Bluespec만의 고수준 문법을 제공하면서도 Verilog와 비슷하거나 호환되는 문법이 많기 때문에 Verilog에 익숙하면 친숙하게 다가갈 수 있는 언어입니다. 그 외에도 많은 고수준 HDL이 존재하며, HLHDL은 컴파일러가 Verilog 등의 HDL 코드를 생성한다는 점이 특징이 있습니다. 생성된 RTL 코드를 FPGA나 ASIC 등의 하드웨어에 구현하려면 다시 Vivado 등의 합성 툴을 거쳐야 합니다. BSV를 이용한 개발 플로우는 다음 그림에서 잘 요약하고 있습니다. Bluespec toolchain에는 컴파일러인 bsc뿐 아니라 bluesim이라는 별도의 시뮬레이터도 포함되어 있습니다. Icarus Verilog 등의 시뮬레이터가 Verilog 코드를 컴파일하여 실행파일을 생성하듯이 bsc는 BSV 소스파일을 컴파일하여 실행파일을 생성할 수 있습니다.



그 외에도 C/C++ 문법을 빌린 HLS(High-Level Synthesis)라는 것도 있습니다. Xilinx에서는 이 HLS를 이용하여 하드웨어 커널을 개발할 수 있는 Vitis HLS라는 툴을 제공하고 있습니다. 각각의 장단점이 존재하기 때문에 필요에 맞게 선택하면 되겠습니다.

Bluespec 컴파일러 설치하기

Bluespec 컴파일러(bsc)의 소스코드는 다음 링크에서 얻을 수 있습니다. bsc를 사용하기 위해서는 소스코드를 직접 빌드해서 설치해야 합니다. 설치 방법은 INSTALL.md 문서를 참고해주세요.

https://github.com/B-Lang-org/bsc

1. Clone the repository

$ git clone https://github.com/B-Lang-org/bsc.git

$ cd bsc

$ git submodule update --init --recursive

2. Install dependencies

```
$ apt-get install ghc

$ apt-get install \
    libghc-regex-compat-dev \
    libghc-syb-dev \
    libghc-old-time-dev \
    libghc-split-dev

$ apt-get install tcl-dev

$ apt-get install pkg-config

$ apt-get install \
    autoconf \
    gperf

$ apt-get install flex bison

$ apt-get install iverilog
```

3. For building install-doc (optional)

레퍼런스나 bsc 매뉴얼 등 LaTex로 작성된 문서를 직접 빌드할 때 필요한 패키지들을 설치합니다. 문서들은 다음 repo에서 PDF 파일로 얻을 수 있기 때문에 이 단계는 생략 가능합니다.

https://github.com/B-Lang-org/Documentation

```
$ apt-get install \
    texlive-latex-base \
    texlive-latex-recommended \
    texlive-latex-extra \
    texlive-font-utils \
```

```
    texlive-fonts-extra
$ apt-get install asciidoctor
```

빌드가 제대로 되지 않을 수도 있는데, 문서 빌드 명령인 `make install-doc`로 빌드가 제대로 되는지 확인해보고 필요한 패키지는 추가로 설치하시면 됩니다.

4. install summary

빌드 결과물은 기본적으로 "inst"라는 이름의 디렉토리에 생성됩니다. make를 실행할 때 변수 "PREFIX"를 지정하면 inst 대신 다른 경로를 사용할 수 있습니다. make 다음에는 다음 target들이 있으니 필요에 맞게 선택하시면 됩니다. release(=install-doc+install-src)와 install-doc의 경우에는 3번에서 설명했던 LaTex로 작성된 문서가 빌드되기 때문에 3번 단계를 생략하셨으면 make install-src를 선택하시면 됩니다.

```
euntae471@EUNTAE-X1:~/project/bsc$ make help
This Makefile will create an installation of the Bluespec Compiler tools,
in a directory named "inst".  This directory can be moved anywhere, but
the contents should remain in the same relative locations.  Intermediate
files are stored in a directory named "build".  The "clean" target will
delete the "build" directory; the "full_clean" target will delete both
the "build" and "inst" directories.

    make  release      Build a release dir with the tools and docs

    make  install-src  Build and install just the tools
    make  install-doc  Build and install just the documentation

    make  check-smoke  Run a quick smoke test
    make  check-suite  Run the test suite (this will take time!)

    make  clean        Remove intermediate build-files unnecessary for execution
    make  full_clean   Restore to pristine state (pre-building anything)
```

```
$ make install-src
```

참고로 GHCJOBS를 이용하여 스레드 수를 지정할 수도 있습니다.

```
$ make GHCJOBS=12 install-src
```

코어 수의 경우 다음 명령어를 통해 확인할 수 있습니다.

```
$ cat /proc/cpuinfo | grep processor
```

5. 원하는 경로에 설치하기

빌드 결과물(기본적으로 inst라는 디렉토리 밑에 생성된다고 했죠?)을 "/opt/tools/bsc"라는 경로 밑으로 옮기려면 다음 명령어를 실행하시면 됩니다. 첫째 줄은 bsc의 버전을 BSC_VERSION이라는 변수에 얻어오는 명령입니다. /opt/tools/bsc 밑에 bsc의 버전별로 디렉토리를 두는 방식입니다.

```
$ BSC_VERSION=$(echo 'puts [lindex [Bluetcl::version] 0]' | inst/bin/bluetcl)
$ mkdir -p /opt/tools/bsc
$ mv inst /opt/tools/bsc/bsc-${BSC_VERSION}
$ cd /opt/tools/bsc
```

```
$ ln -s bsc-${BSC_VERSION} latest
```

참고로 버전 정보를 제대로 얻어왔는지는 echo 명령어로 쉽게 확인할 수 있습니다.

```
$ echo $BSC_VERSION
```

bsc를 환경 변수 PATH에 추가하려면 다음 명령을 실행합니다.

```
$ export PATH=/opt/tools/bsc/latest/bin:$PATH
```

홈 디렉토리에 있는 .bashrc에 이 명령을 추가하면 터미널을 새로 킬 때마다 재실행할 필요가 없어집니다.

설치가 마무리됐으면 터미널에 bsc를 입력해봅시다.

```
euntae471@EUNTAE-X1:~/project/bsc$ bsc
Usage:
  bsc -help                                to get help
  bsc [flags] file.bsv                     to partially compile a Bluespec file
  bsc [flags] -verilog -g mod file.bsv     to compile a module to Verilog
  bsc [flags] -verilog -g mod -u file.bsv  to recursively compile modules to Verilog
  bsc [flags] -verilog -e topmodule        to link Verilog into a simulation model
  bsc [flags] -sim -g mod file.bsv         to compile to a Bluesim object
  bsc [flags] -sim -g mod -u file.bsv      to recursively compile to Bluesim objects
  bsc [flags] -sim -e topmodule            to link objects into a Bluesim binary
  bsc [flags] -systemc -e topmodule        to link objects into a SystemC model
euntae471@EUNTAE-X1:~/project/bsc$ bsc -v
Bluespec Compiler, version 2022.01-29-gc526ff54 (build c526ff54)
This is free software; for source code and copying conditions, see
https://github.com/B-Lang-org/bsc

Invoking command line:
bsc -v
```