VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh
Department of Electronics and Communication Engineering
IIIT-Delhi

Tutorial 3
Lecture - 14
High-level synthesis using Bambu

Hello everyone. My name is Jasmine Kaur. I am a Ph.D. student at IIIT Delhi, and I will be your TA for the course VLSI Design Flow: RTL to GDS. This tutorial focuses on high-level synthesis. High-level synthesis is a design methodology used in digital hardware design. So, this refers to the process of converting a high-level behavioral description of hardware functionality into an optimized hardware implementation.

So, it allows the designer to express the hardware system using high-level languages such as C, C++, or system C and generates an output in the form of RTL description, normally expressed using hardware description languages such as VHDL or Verilog. For this, you can use any HLS tool. I will be using the high-level synthesis tool Bambu, which is an open-source tool. So, let us see what are the steps to install Bambu.

To install Bambu, first of all, you will need to update the package index files on the system. So, we can do that using

```
sudo apt-get update
```

command. So, it is updating the system files. Now, we need to install the dependencies for Bambu.

So, we will use this command:

```
sudo apt-get install -y --no-install-recommends build-essential ca-certificates gcc-multilib git iverilog, verilator wget
```

and it is installing all these dependencies. So, we need these dependencies to run the Bambu tool. So, it is almost done.

So, the installation is complete. We need to download the app image for Bambu. So, using

```
wget https://release.bambuhls.eu/appimage/bambu-0.9.7.AppImage
```

command (using this link from the Bambu website), we can download the app image. So, this downloading will take a few minutes. Now, the app image has been downloaded.

Now, using

```
chmod +x bambu-0.9.7.AppImage
```

command, we need to make this app image executable, and then we need to install libfuse2 library, which is required for running app images. So, here we are installing libfuse2:

```
sudo add-apt-repository universe
sudo apt install libfuse2
```

Once it is done, we can check if the Bambu is installed correctly using this command. So, we can see it is working. So, let us look at an example to see how it works.

```
long func(int,int,int,int);
main()
{
int j; int k; int c; int d;
int res = func(j,k, c, d);
return 0;
}
long func(int j,int k, int c, int d)
{
int i=0;
if(c > 2){  i = j - k; }
else if (d < 5) { i = j + k; }
else { i= 12; }
return i;
}
```

So, this is a C file in which what we are doing is if c variable is greater than 2 (c>2), then it will subtract the two inputs (i = j - k;) else, if d<5, it will add (i = j + k;), and in all other cases it will take a value of 12 (i = 12;). So, let us see how to run it. We have to run it from the home directory. So, for high-level synthesis, the command looks like this. So, we give the command:

```
./bambu-0.9.7.AppImage <path-to-c-file> --top-fname=<accelerator-function-to-be-implemented-in-hardware>
```

where the path to the C file provides the C file input that needs to be synthesized, and the --top-fname option is used to tell which function we want to implement in hardware. So, in our case, it is func. After running this command, func.v file has been created, an RTL description of the C file we gave in. So, let us see what are the contents of this file.

So, this is how we have different modules here. Conditional expression, greater than expression, less than equal to expression, minus expression, plus expression, data path, controller function, etc. So, you can analyze the different modules in the .v file created, and we will give you a tutorial sheet for this tutorial that you can refer to. Thank you.