Getting Started with BORPH

1. Introduction
2. Setup
3. Creating Your Design
4. Generting BOF Files from Your Design
5. Running Your Design
6. Conclusion

Introduction

In this tutorial, you will create a simple Simulink design using both standard Xilinx system generator blockset, as well as library blocks specific to BEE2. At the end of this tutorial, you will have a BORPH executable file, a BOF file, and you will know how to interact with your running hardware design using BORPH.

Setup

Before you start, you have to make sure you have set up your `path` variable in Matlab correctly. Make sure it includes `<BEE_LIBRARY_PATH>/mlib/xps_library` and `<BEE_LIBRARY_PATH>/mlib/bee_library`, where `<BEE_LIBRARY_PATH>` is the directory where you have installed your matlab library scripts downloaded for BEE2.

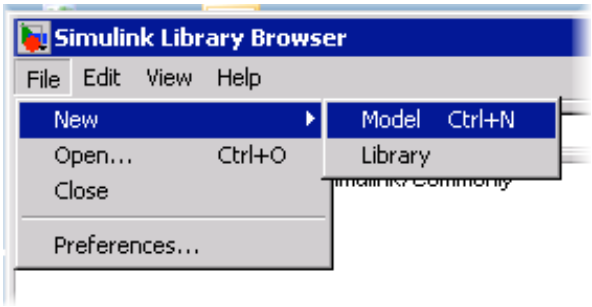
In our BWRC setup environment, note the following 3 things:

1. Set <BEE_LIBRARY_PATH> to `/tools/designs/BEE`
2. Most of the latest work are located in the `mlib_devel` directory instead of `mlib`
3. You can have these path setup automatically if you start Matlab using the Windows batch file (Matlab7sp2.bat) from `\\hitz\designs\BEE\mlib_devel`

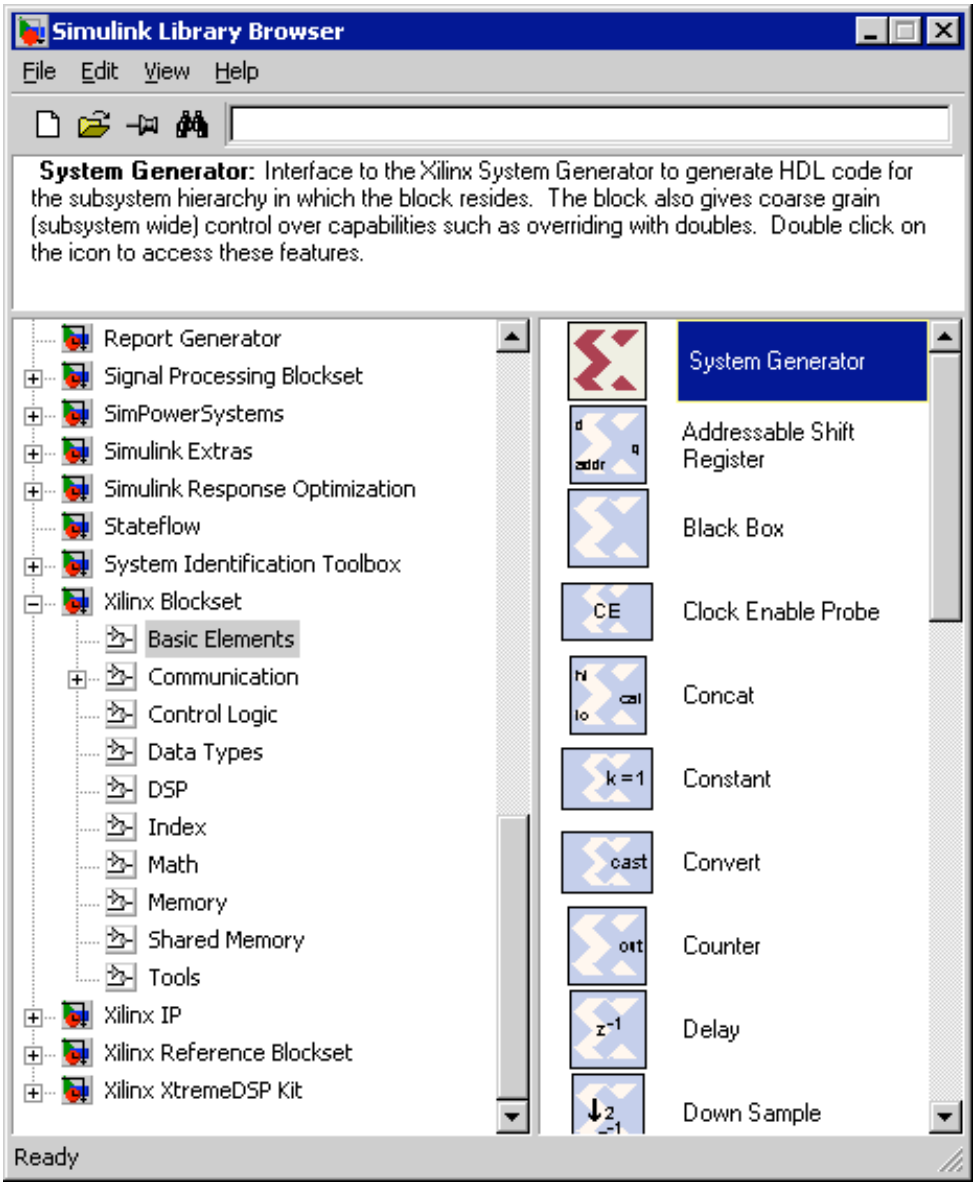
Once Matlab is started, you can start Simulink by typing `simulink` in the Matlab prompt. If your path is setup correctly, you should see among other libraries, the **BEE2 System Blockset** and **Xilinx Blockset**.

Creating Your Design

1. Create a new model:



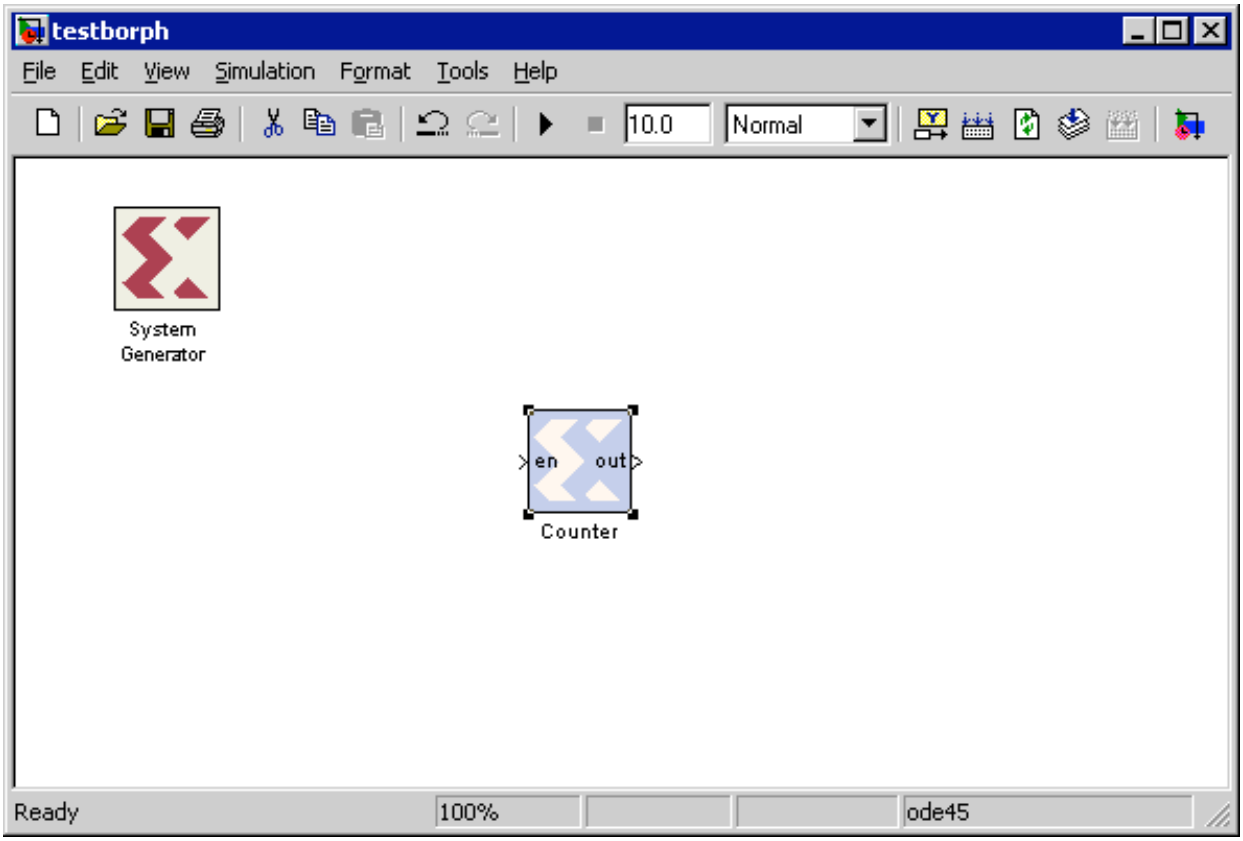
2. Select **Xilinx Blockset**:



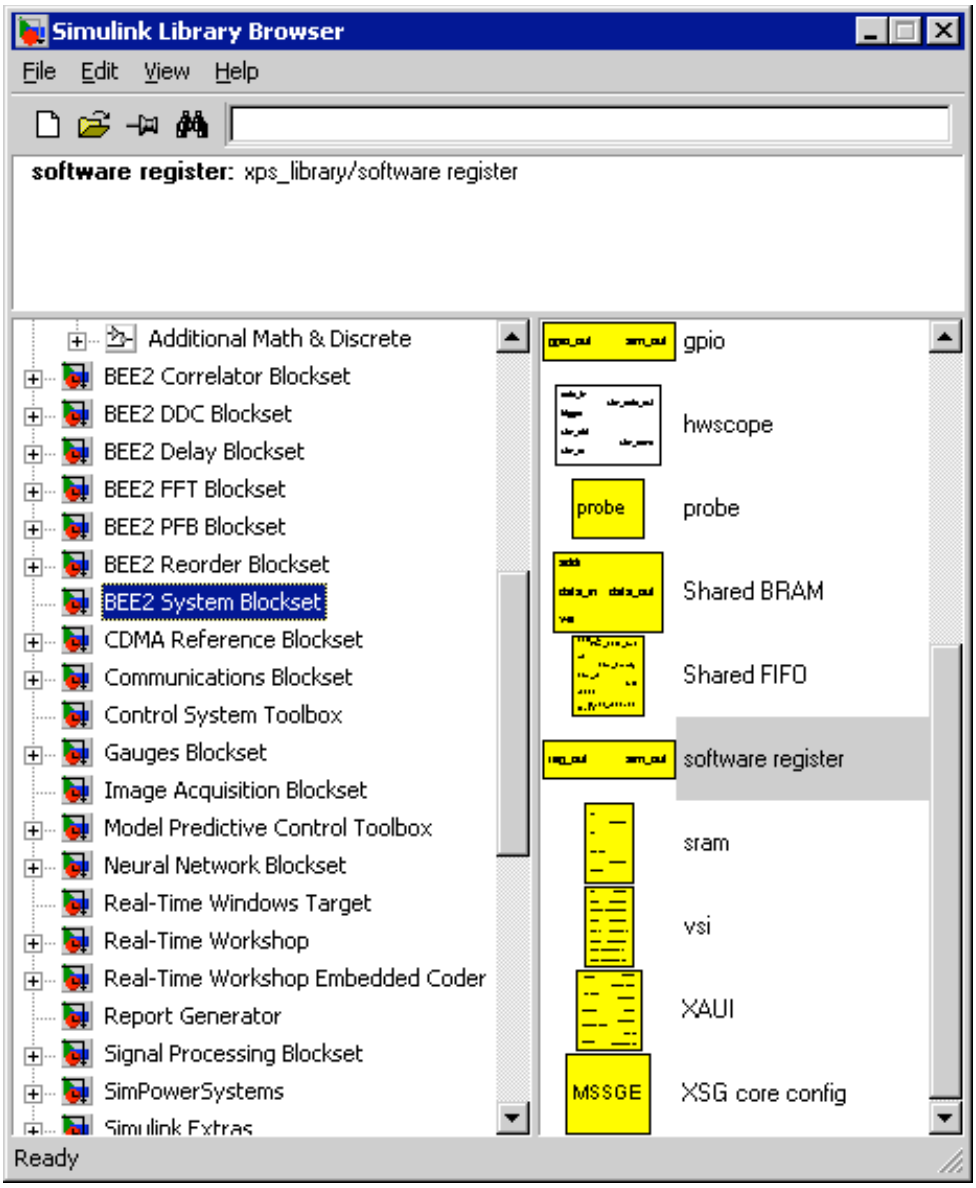
3. Add 1 instance of **Counter** and 1 instance of **System Generator**

4. Double-click on **Counter**
    i. Set **Number of Bits** to **32**
    ii. Check the box next to **Provide Enable Port**
    iii. Select **OK**

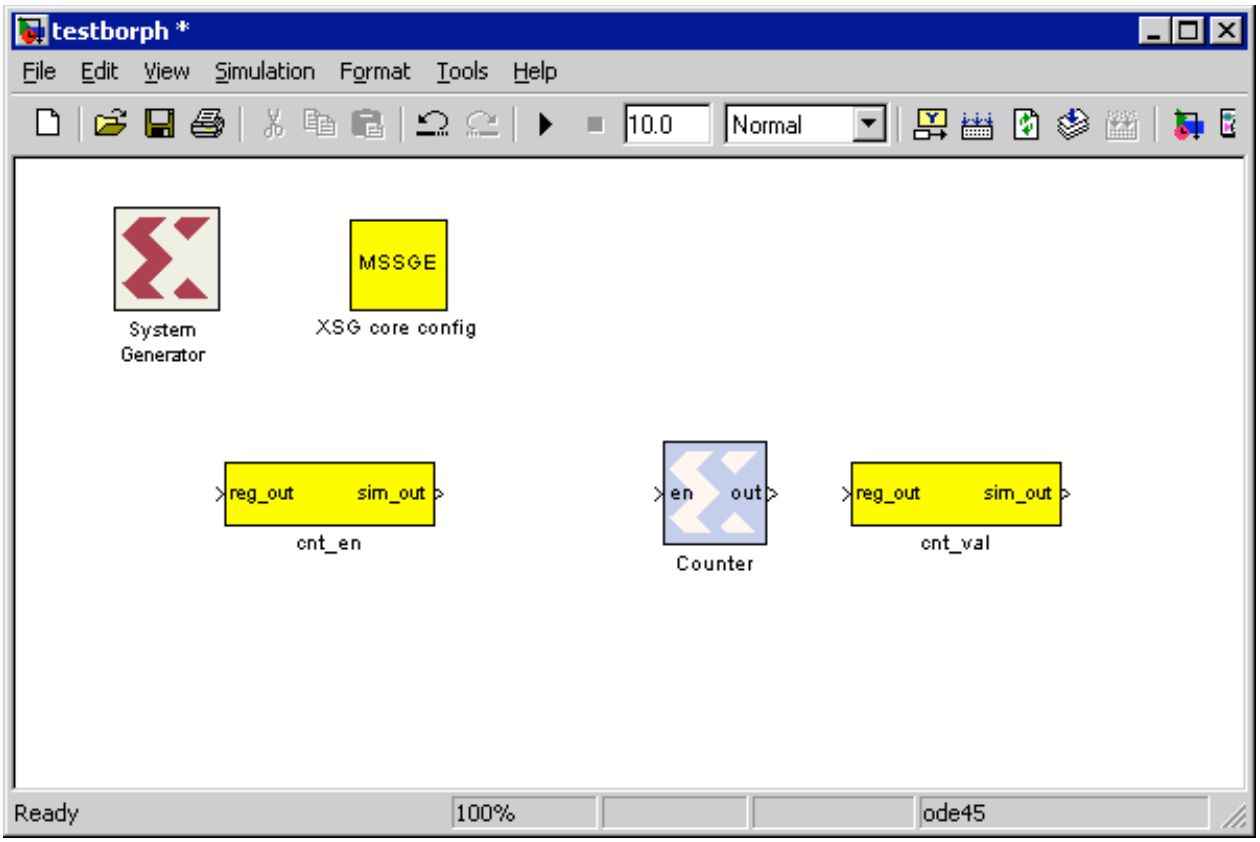
   You should have a design that looks like this:



5. It is a good time to save this new design. In this tutorial, we will call it "*testborph*".
6. Now, select **BEE2 System Blockset**:

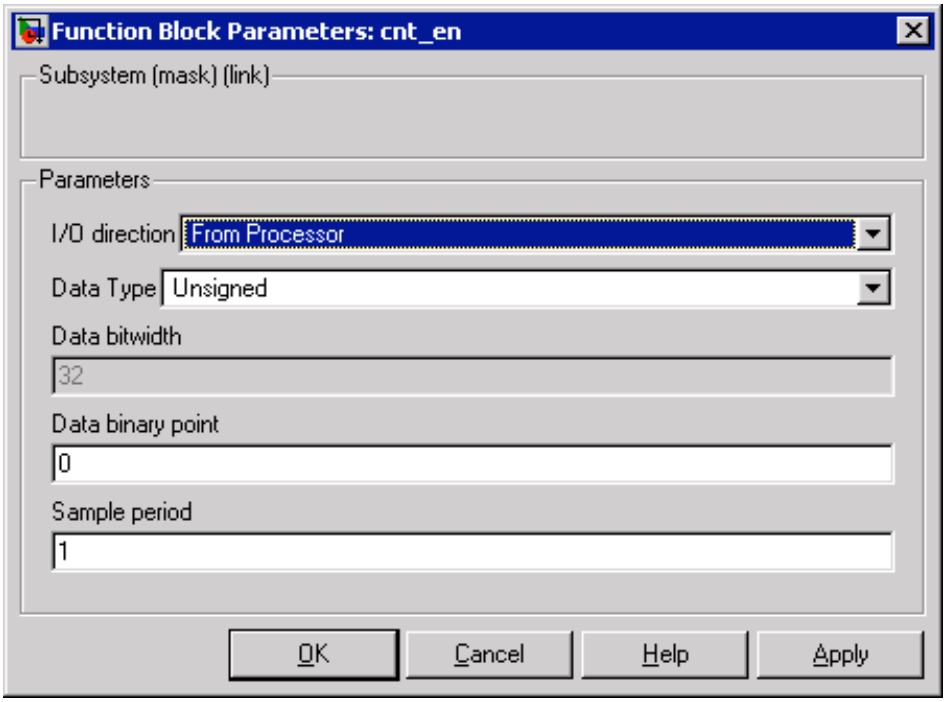


7. Add **1** instace of **XSG core config** and **2** instances of **software register**



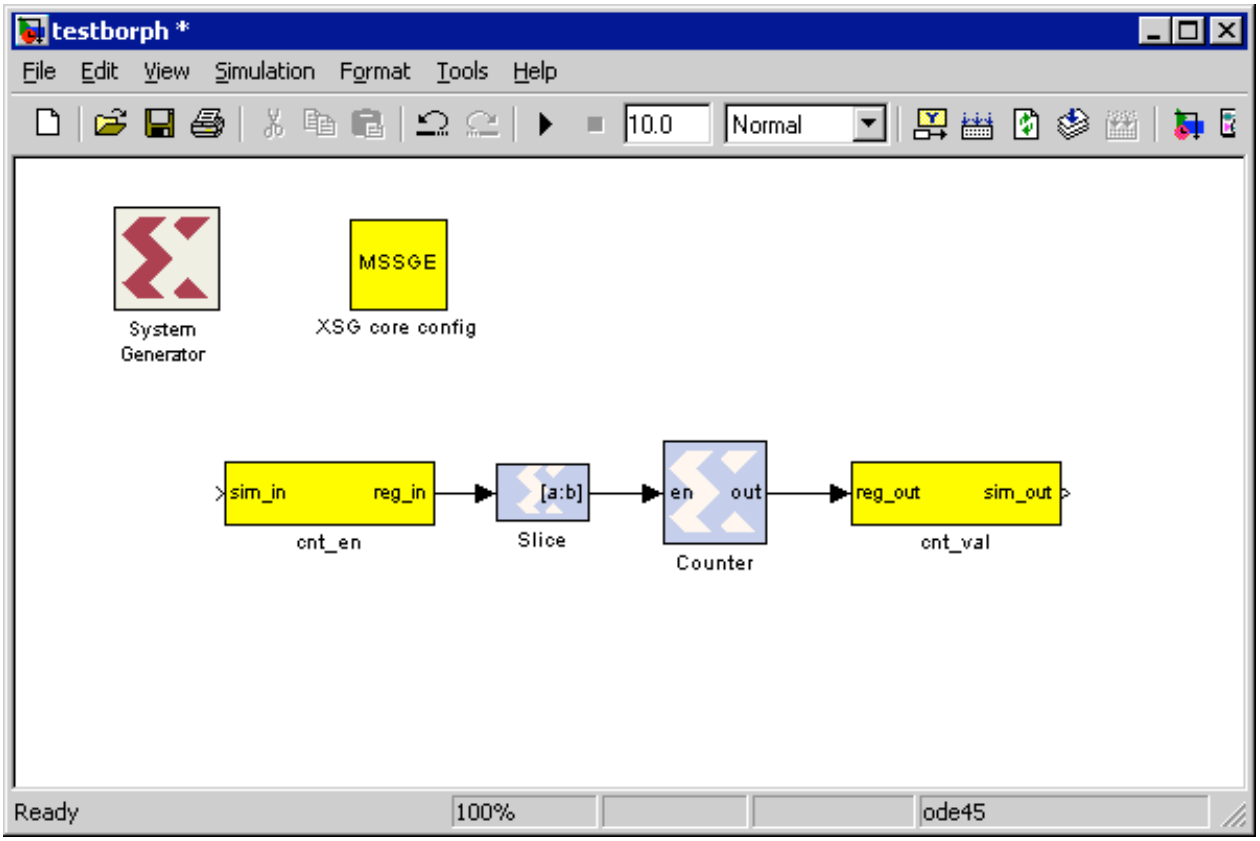
Name the instance of **sofware register** on the left of Counter as **cnt_en**, while naming the other instance as **cnt_val**.

8. Double-click on the software rigster named **cnt_en** and select **I/O direction** as **From Processor**.

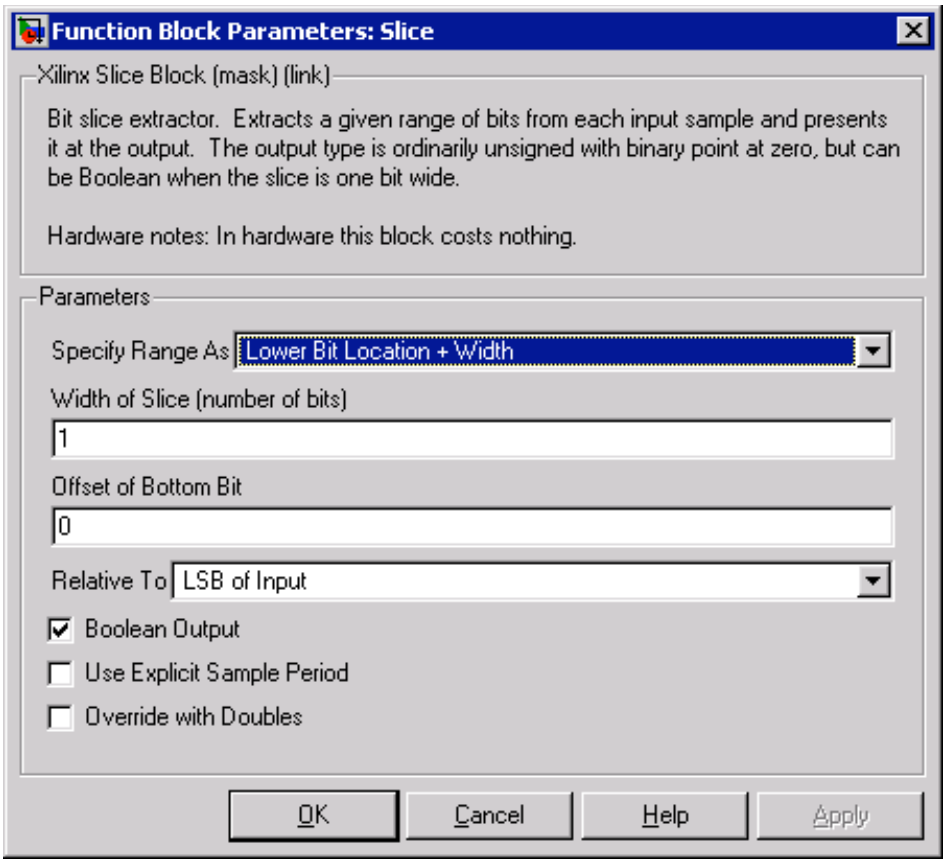


Note the field **Data bitwidth** is greyed out with a value **32**. It is because all software registers have a *fixed* data bitwidth of 32 bits.

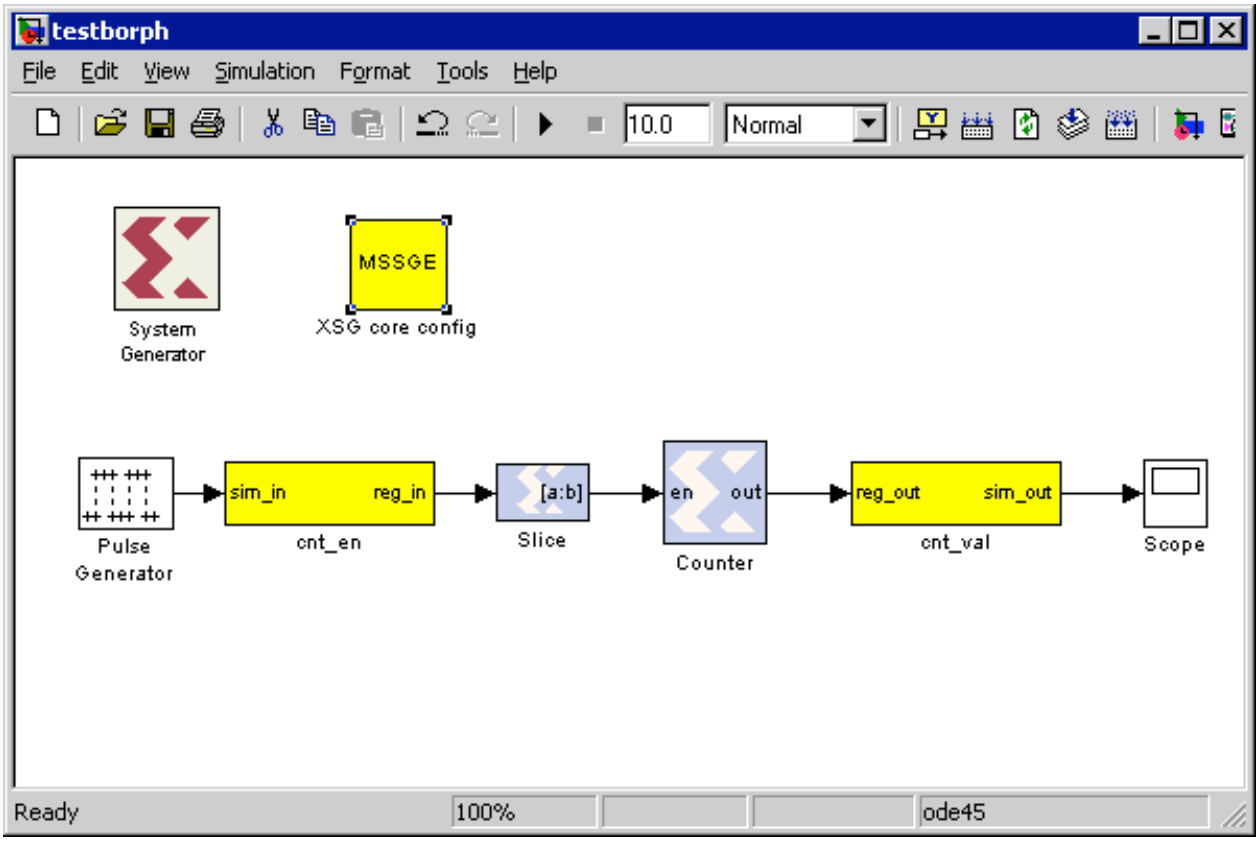
9. Since we want to control the **en** port of Counter, which takes a Boolean input, we need a way to extract 1 bit out of this 32-bits software register. That's why we need to add one more block, a **Slice** from the Xilinx Blockset. Add an instance of **Slice** from the Xilinx Blockset to our design, connecting it between the cnt_en software register and the counter.



10. Double-click on the newly added Slice block to set it's parameter:



11. In order to generate a BOF file from this Simulink design using our MSSGE flow, we will need to add one final block: a **XSG core cofing** block from the BEE2 System Blockset. Furthermore, for sake of simulating your design withing Simulink, as well as to make our MSSGE flow to run correctly, you will need to connect up the **sim_in**/**sim_out** port of any software registers. (If you don't intend to simulate your design in Simulink, simply connect a constant block to the input of any software register.) You now have a design that looks like this:

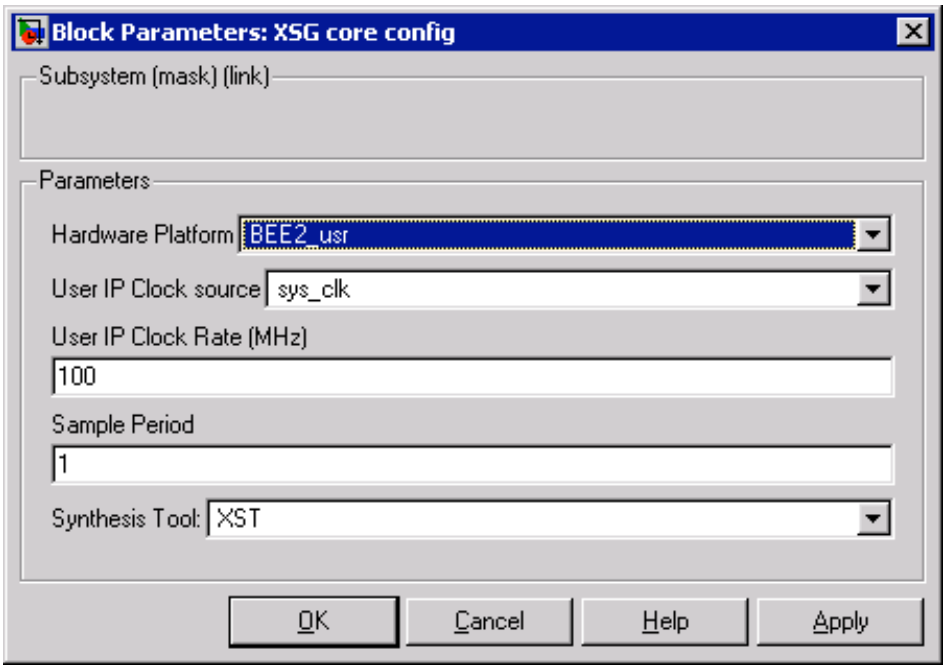


12. Save your work, you now have a fully functional design there is ready to be run on BEE2.

Generting BOF Files from Your Design

To generate a BOF file from your design, you must first specify the target platform and the synthesis tool to be used.

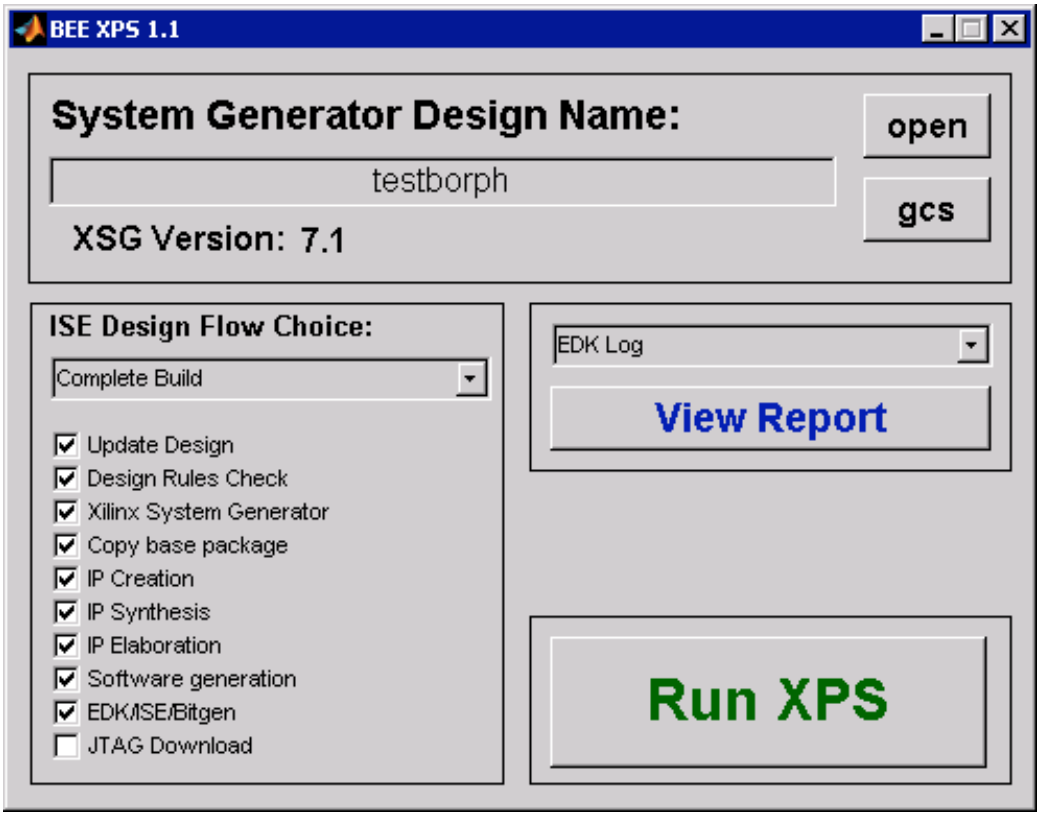
1. Double-click the **XSG core config** block. For a design to be run on a user FPGA on BEE2, select **BEE2_usr**. Selecting XST for synthesis tool is sufficient for most designs.



2. Once that's done, click OK and save your work.
3. Now, go to the Matlab prompt, and type `bee_xps`:

```
matlab>> bee_xps
```

4. You should see a new window pops up like this:



5. Make sure the name showing on the top of this new window corresponds to the design

you want to generate your BOF file. (If it is not, click anywhere on your design such that it is the highlighted window, then click **gcs**.)

6. Finally, click **Run XPS**

Once the MSSGE script has finished, it should have created a subdirectory with your design name in the directory where you save your design. For example, if our design is saved in a directory `FOO`, then if you list the content of the directory, you will see:

```
user@local$ ls -lF FOO
drwxrwxrwx  7 user grp  4096 May 15 12:12 testborph/
-rwxrwxrwx  1 user grp 36370 May 15 13:21 testborph.mdl
```

Inside the directory `testborph` is a directory called `bit_files`, which contains 1 `.bit` file and 5 BOF files. For this tutorial, focus only on the one with name `<design_name>_floating_<date_time>.bof`. This "floating" BOF file is what we will use now. Feel free to rename it to a shorter name if you want.

Running Your Design

Running your design is as simple as running a compiled software program from a compiler: All you need is to execute the floating BOF file within BORPH as if it is any other executable file in the system such as `firefox`.

The exact way of transfering a BOF file to a BEE2 system depends on your system setup.

For BWRC setup, the easiest way is to copy BOF files into BEE2 systems by using `scp` from **wright.eecs.berkeley.edu**:

```
user@wright$ scp testborph.bof user@beeopen:
```

In any case, once you have copied your BOF files, log on to your BEE2, and execute your BOF file like this (using **beeopen** as an example):

```
user@wright$ ssh user@beeopen

user@beeopen$ ls
testborph.bof
user@beeopen$ ./testborph.bof
```

Congratulations! You have just launched a *hardware process* in BORPH. If you have access to the physical BEE2 platform, you should see one of the user FPGA get configured automatically as a result of the BOF file execution. You can also see if your Simulink hardware is running by the command **ps**:

```
user@beeopen$ ps
  PID TTY          TIME CMD
26427 pts/6    00:00:00 bash
26488 pts/6    00:00:00 testborph.bof
26525 pts/6    00:00:00 ps
```

Once your hardware process is running, you can interact with it through the **ioreg** interface of BORPH. Without going into the detail, assume your hardware process is runnign with process ID (PID) of 26488 as shown above, issue the following commands:

```
user@beeopen$ cd /proc/26488/hw
user@beeopen$ ls
ioreg  ioreg_mode  region
user@beeopen$ echo 0 > ioreg_mode
```

By default, the values are accessed as binary values, and ioreg_mode is set to 1. By setting ioreg_mode to 0, hexadecimal I/O values will be used. Then, the following commands can be issued:

```
user@beeopen$ cd ioreg
user@beeopen$ ls
cnt_en cnt_val
user@beeopen$
```

You see, the two "software registers" you put in your design now shows up automatically on Linux side. You can check the current value of by:

```
user@beeopen$ cat cnt_val
00AC1000
user@beeopen$ cat cnt_val
00BD2590
user@beeopen$
```

According to our Simulink design, we can disable or enable the counter by setting `cnt_en` to 0 or 1 respectively:

```
user@beeopen$ echo 0 > cnt_en
user@beeopen$ cat cnt_val
0C1F8346
user@beeopen$ cat cnt_val
0C1F8346
user@beeopen$ echo 1 > cnt_en
user@beeopen$ cat cnt_val
20BF0A99
user@beeopen$ cat cnt_val
3AB012F0
user@beeopen$
```

Conclusion

In this tutorial, you have leaned how to create an executable BOF file from Simulink. You have also learned the basic of running a hardware design in BORPH and interating with it using BORPH's ioreg interface.

To learn more about BORPH and how to use a on-chip memory on user FPGA has a shared memory betwen software and hardware, please see another tutorial, BorphUsingIoreg.

BorphGettingStart (last edited 2006-11-13 23:15:31 by hchen05)