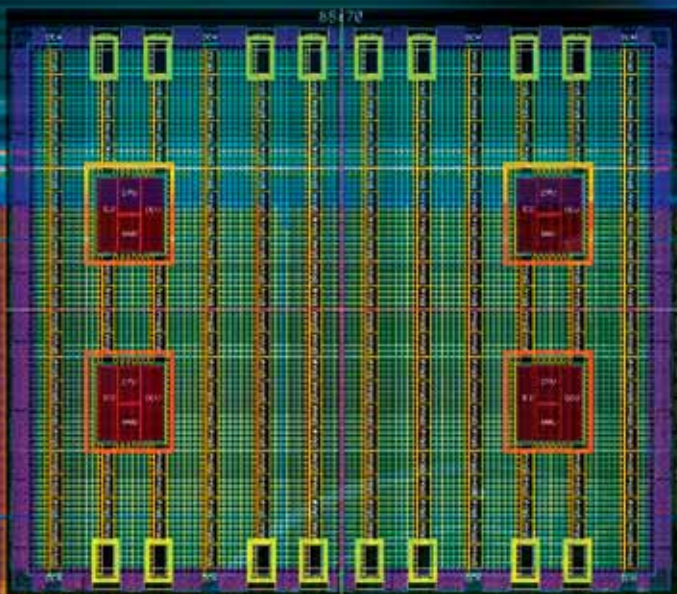


# Generating Efficient Board Support Packages

by Rick Molerer  
Manager of Software IP  
Xilinx, Inc.  
[rick.molerer@xilinx.com](mailto:rick.molerer@xilinx.com)

The Xilinx Platform Studio toolset enables quick and easy BSP generation for Virtex FPGAs with immersed PowerPC processors.

Milan Saini  
Technical Marketing Manager  
Xilinx, Inc.  
[milan.saini@xilinx.com](mailto:milan.saini@xilinx.com)



Platform FPGAs with embedded processors offer you unprecedented levels of flexibility, integration, and performance. It is now possible to develop extremely sophisticated and highly customized embedded systems inside a single programmable logic device.

With silicon capabilities advancing, the challenge centers on keeping design methods efficient and productive. In embedded systems development, one of the key activities is the development of the board support package (BSP). The BSP allows an embedded software application to successfully initialize and communicate with the hardware resources connected to the processor. Typical BSP components include boot code, device driver code, and initialization code.

Creating a BSP can be a lengthy and tedious process that must be incurred every time the microprocessor complex (processor plus associated peripherals) changes. With FPGAs, fast design iterations combined with the inherent flexibility of the platform can make the task of managing the BSP even more challenging (Figure 1). This situation clearly underscores the need for and importance of providing an efficient process for managing BSPs.

In this article, we'll describe an innovative solution from Xilinx that simplifies the creation and management of RTOS BSPs. We chose the WindRiver VxWorks flow to illustrate the concept; however, the underlying technology is generic and equally applicable to all other OS solutions that support Xilinx® processors.

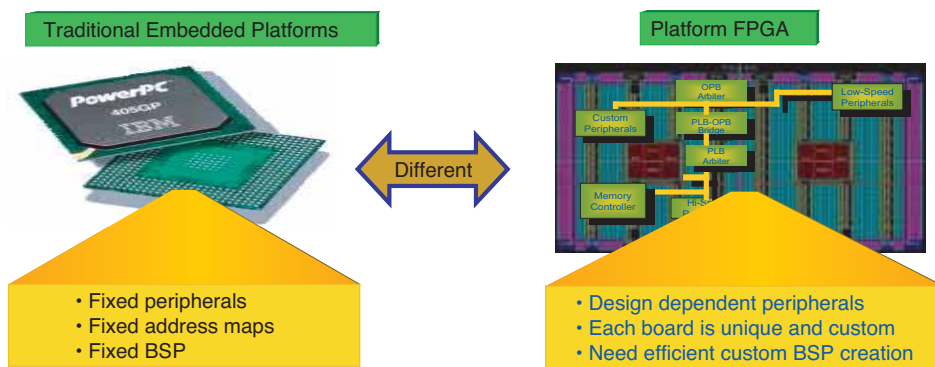


Figure 1 – Platform FPGA flexibility requires the software BSP generation process to be efficient.

### Xilinx Design Flow and Software BSP Generation

Designing for a Xilinx processor involves a hardware platform assembly flow and an embedded software development flow. Both of these flows are managed within the Xilinx Platform Studio (XPS) tool, which is part of the Xilinx Embedded Development Kit (EDK).

You would typically begin a design by assembling and configuring the processor and its connected components in XPS. Once the hardware platform has been defined, you can then configure the software parameters of the system. A key feature of Platform Studio is its ability to produce a BSP that is customized based on your selection and configuration of processor, peripherals, and embedded OS. As the system evolves through iterative changes to the hardware design, the BSP evolves with the platform.

An automatically generated BSP enables embedded system designers to:

- Automatically create a BSP that completely matches the hardware design
- Eliminate BSP design bugs by using pre-certified components
- Increase designer productivity by jump-starting application software development

### Creating BSPs for WindRiver VxWorks

Platform Studio can generate a customized Tornado 2.0.x (VxWorks 5.4) or Tornado 2.2.x (VxWorks 5.5) BSP for the PowerPC™ 405 processor and its periph-

erals in Xilinx Virtex™-II Pro and Virtex-4 FPGAs. The generated BSP contains all of the necessary support software for a system, including boot code, device drivers, and VxWorks initialization.

Once a hardware system with the PowerPC 405 processor is defined in Platform Studio, you need only follow these three steps to generate a BSP for VxWorks:

1. Use the Software Settings dialog box (see Figure 2) to select the OS you plan to use for the system. Platform Studio users can select vxworks5\_4 or vxworks5\_5 as their target operating system.
2. Once you have selected the OS, you can go to the Library/OS Parameters tab, as shown in Figure 3, to tailor the Tornado BSP to the custom hardware. You have the option of selecting any UART device in the system as the standard I/O device (stdin and stdout). This results in the device being used as the VxWorks console device.

You can also choose which peripherals are connected peripherals, or which devices will be tightly integrated into the VxWorks OS. For example, the Xilinx 10/100 Ethernet MAC can be integrated into the VxWorks Enhanced Network Driver (END) interface. Alternately, the Ethernet device need not be connected to the END interface and can instead be accessed directly from the VxWorks application.

3. Generate the Tornado BSP by selecting the Tools > Generate Libraries and BSP menu option. The resulting BSP resembles a traditional Tornado BSP and is located in the Platform Studio project directory under `ppc405_0/bsp_ppc405_0` (see Figure 4).

Note that `ppc405_0` refers to the instance name of the PowerPC 405 processor in the hardware design. Platform Studio users can specify a different instance name, in which case the subdirectory names for the BSP will match the processor instance name.

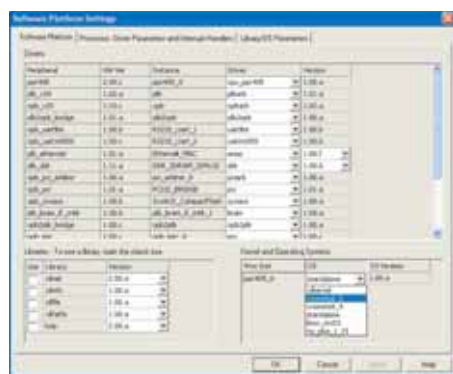


Figure 2 – Setting the embedded OS selection

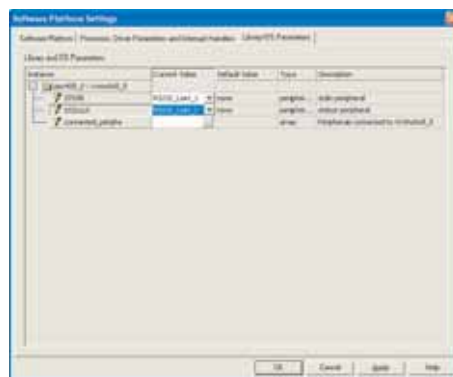


Figure 3 – Configuring the OS-specific parameters

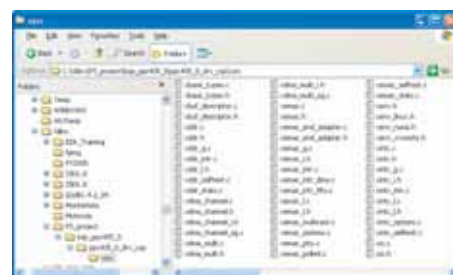


Figure 4 – Generated BSP directory structure

The Tornado BSP is completely self-contained and transportable to other directory locations, such as the standard Tornado installation directory for BSPs at `target/config`.

### Customized BSP Details

The XPS-generated BSP for VxWorks resembles most other Tornado BSPs except for the placement of Xilinx device driver code. Off-the-shelf device driver code distributed with Tornado typically resides in the `target/src/drv` directory in the Tornado distribution directory. Device driver code for a BSP that is automatically generated by Platform Studio resides in the BSP directory itself.

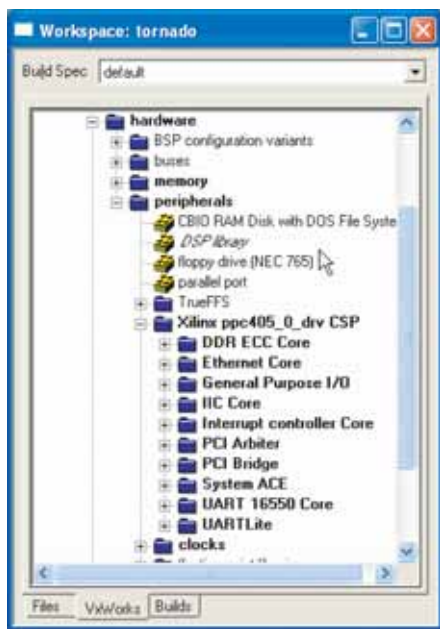


Figure 5 – Tornado 2.x Project: VxWorks tab

This minor deviation is due to the dynamic nature of FPGA-based embedded systems. Because an FPGA-based embedded system can be reprogrammed with new or changed IP, the device driver configuration can change, calling for a more dynamic placement of device driver source files. The directory tree for the automatically generated BSP is shown in Figure 4. The Xilinx device drivers are placed in the `ppc405_0_drv_csp/xsrc` subdirectory of the BSP.

## The Tornado BSP created by Platform Studio has a makefile that you can modify at the command line if you would rather use the `diab` compiler instead of the `gnu` compiler.

Xilinx device drivers are implemented in C and are distributed among several source files, unlike traditional VxWorks drivers, which typically consist of single C header and implementation files. In addition, there is an OS-independent implementation and an optional OS-dependent implementation for device drivers.

The OS-independent part of the driver is designed for use with any OS or any processor. It provides an application program interface (API) that abstracts the func-

named `ppc405_0_drv_<driver_version>.c` in the BSP directory. This file includes the driver source files (\*.c) for the given device and is automatically compiled by the BSP makefile.

This process is analogous to how VxWorks' `sysLib.c` includes source for Wind River-supplied drivers. The reason why Xilinx driver files are not simply included in `sysLib.c` like the rest of the drivers is because of namespace conflicts and maintainability issues. If all Xilinx driver files are part of a single compilation unit, static functions and data are no longer private. This places restrictions on the device drivers and would negate their OS independence.

### Integration with the Tornado IDE

The automatically generated BSP is integrated into the Tornado IDE (Project Facility). The BSP is compilable from the command line using the Tornado make tools or from the Tornado Project. Once the BSP is generated, you can simply type `make vxWorks` from the command line to compile a bootable RAM image. This assumes that the Tornado environment has been previously set up, which you can do through the command line using the `host/x86-win32/bin/torVars.bat` script (on a Windows platform). If you are using the Tornado Project facility, you can create a project based on the newly generated BSP, then use the build environment provided through the IDE to compile the BSP.

In Tornado 2.2.x, the `diab` compiler is supported in addition to the `gnu` compiler. The Tornado BSP created by Platform Studio has a makefile that you can modify at the command line if you would rather use the `diab` compiler instead of the `gnu` compiler. Look for the make variable named `TOOLS` and set the value to “`diab`” instead of “`gnu`.” If using the Tornado Project facility, you can select the desired compiler when the project is first created.

The file `50ppc405_0.cdf` resides in the BSP directory and is tailored during creation of

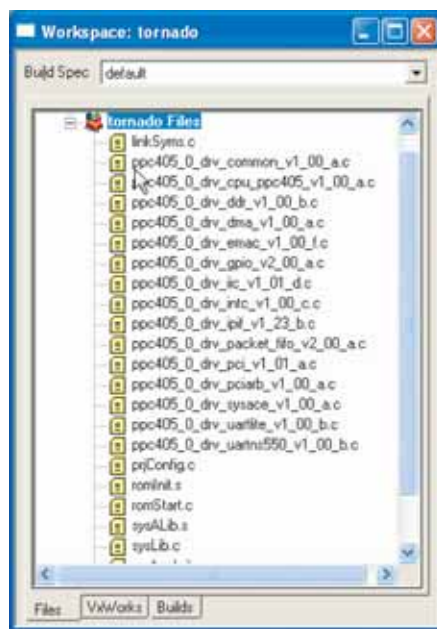


Figure 6 – Tornado 2.x Project: Files tab

tionality of the underlying hardware. The OS-dependent part of the driver adapts the driver for use with an OS such as VxWorks. Examples are Serial IO drivers for serial ports and END drivers for Ethernet controllers. Only drivers that can be tightly integrated into a standard OS interface require an OS-dependent driver.

Xilinx driver source files are included in the build of a VxWorks image in the same way that other BSP files are included in the build. For every driver, a file exists

the BSP. This file integrates the device drivers into the Tornado IDE menu system. The drivers are hooked into the BSP at the Hardware > Peripherals subfolder. Below this are individual device driver folders. Figure 5 shows a menu with Xilinx device drivers.

The Files tab of the Tornado Project Facility will also show the number of files used to integrate the Xilinx device drivers into the Tornado build process. These files are automatically created by Platform Studio and you need only be aware that the files exist. Figure 6 shows an example of the driver build files.


Some of the commonly used devices are tightly integrated with the OS, while other devices are accessible from the application by directly using the device drivers. The device drivers that have been tightly integrated into VxWorks include:

- 10/100 Ethernet MAC
- 10/100 Ethernet Lite MAC
- 1 Gigabit Ethernet MAC
- 16550/16450 UART
- UART Lite
- Interrupt Controller
- System ACE™ technology
- PCI

All other devices and associated device drivers are not tightly integrated into a VxWorks interface; instead, they are loosely integrated. Access to these devices is available by directly accessing the associated device drivers from the user application.

## Conclusion

With the popularity and usage of embedded processor-based FPGAs continuing to grow, tool solutions that effectively synchronize and tie the hardware and software flows together are key to helping designer productivity keep pace with advances in silicon.

Xilinx users have been very positive about Platform Studio and its integration with VxWorks 5.4 and 5.5. Xilinx fully intends to continue its development support for the Wind River flow that will soon include support for VxWorks 6.0 and Workbench IDE. 

## Microprocessor Library Definition (MLD)

The technology that enables dynamic and custom BSP generation is based on a Xilinx proprietary format known as Microprocessor Library Definition (MLD). This format provides third-party vendors with a plug-in interface to Xilinx Platform Studio to enable custom library and OS-specific BSP generation (see Figure 7). The MLD interface is typically written by third-party companies for their specific flows. It enables the following add-on functionality:

- Enables custom design rule checks
- Provides the ability to customize device drivers for the target OS environment
- Provides the ability to custom-produce the BSP in a format and folder structure tailored to the OS tool chain
- Provides the ability to customize an OS/kernel based on the hardware system under consideration

The MLD interface is an ASCII-based open and published standard. Each RTOS flow will have its own set of unique MLD files. An MLD file set comprises the following two files:

- A data definition (.mld) file. This file defines the library or operating system through a set of parameters set by the Platform Studio. The values of these parameters are stored in an internal Platform Studio database and intended for use by the script file during the output generation.
- A .tcl script file. This is the file that is called by XPS to create the custom BSP. The file contains a set of procedures that have access to the complete design database and hence can write a custom output format based on the requirements of the flow.

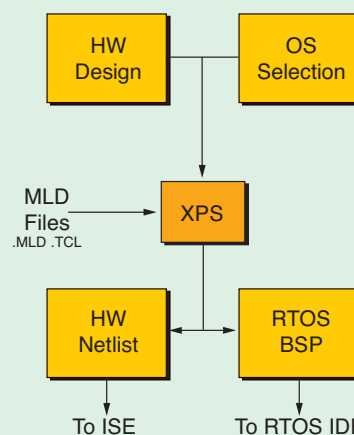


Figure 7 – Structure of an MLD flow

The MLD syntax is described in detail in the EDK documentation (see “Platform Specification Format Reference Manual” at [www.xilinx.com/ise/embedded/psf\\_rm.pdf](http://www.xilinx.com/ise/embedded/psf_rm.pdf)). You can also find MLD examples in the EDK installation directory under [sw/lib/bsp](http://sw/lib/bsp).

Once MLD files for a specific RTOS flow have been created, they need to be installed in a specific path for Xilinx Platform Studio to pick up on its next invocation. The specific RTOS menu selection now becomes active in the XPS dialog box (Project > SW Platform Settings > Software Platform > OS).

Currently, the following partners’ MLD files are available for use within XPS:

- Wind River (VxWorks 5.4, 5.5) (included in Xilinx Platform Studio)
- MontaVista (Linux) (included in Xilinx Platform Studio)
- Mentor Accelerated Technologies (Nucleus) (download from [www.xilinx.com/ise/embedded/mld/](http://www.xilinx.com/ise/embedded/mld/))
- GreenHills Software (Integrity) (download from [www.xilinx.com/ise/embedded/mld/](http://www.xilinx.com/ise/embedded/mld/))
- Micrium (µC/OS-II) (download from [www.xilinx.com/ise/embedded/mld/](http://www.xilinx.com/ise/embedded/mld/))
- µCLinux (download from [www.xilinx.com/ise/embedded/mld/](http://www.xilinx.com/ise/embedded/mld/)).