LogiCORE™ FlexRay v1.1

Getting Started Guide

UG339 May 17, 2007

XILINX®

Revision History

The following table shows the revision history for this guide.

	Version	Revision
05/17/07	1.1	Initial Xilinx release.

09/29/09 - This is the final publication. No content was changed.

Table of Contents

About This Guide

The *LogiCORE™ FlexRay v1.1 Getting Started Guide* provides information about generating a FlexRay core, customizing and simulating the core with the provided example design, and running the design files through implementation using Xilinx tools.

Guide Contents

The following chapters are included in this guide:

- Chapter 1, "Introduction" describes the core and related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- Chapter 2, "Licensing the Core" provides information license types, and how to license the core.
- Chapter 3, "Quick Start Example Design" provides instructions to quickly generate the FlexRay core and run the example design through implementation and simulation.
- Chapter 4, "Detailed Example Design" provides detailed information about the example design, including a description of files and the directory structure generated by the CORE Generator. It also includes a definition of the purpose and content of the provided scripts and example HDL wrappers, as well as information about the operation of the demonstration test bench.

Additional Resources

For additional information, visit www.xilinx.com/support. The table below defines some of the resources you can access from the Xilinx website or go to using the provided URLs.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging www.xilinx.com/support/techsup/tutorials/index.htm
Answer Browser	Database of Xilinx solution records www.xilinx.com/xlnx/xil_ans_browser.jsp
Application Notes	Descriptions of device-specific design techniques and approaches www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes

Resource	Description/URL
Data Sheets	Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging www.xilinx.com/xlnx/xweb/xil_publications_index.jsp
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues www.xilinx.com/support/troubleshoot/psolvers.htm
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment www.xilinx.com/xlnx/xil_tt_home.jsp

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands you enter in a syntactical statement	**`ngdbuild`** `design_name`
Italic font	Variables in a syntax statement for which you must supply values	See the *Development System Reference Guide* for more information.
	References to other manuals	See the *User Guide* for details.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Dark Shading	Items that are not supported or reserved	This feature is not supported
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[option_name] design_name`
Braces { }	A list of items from which you must choose one or more	**`lowpwr ={on|off}`**
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**

Convention	Meaning or Use	Example

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis `...`	Omitted repetitive material	**`allow block`** `block_name loc1 loc2 ... locn;`
Notations	The prefix '`0x`' or the suffix '`h`' indicate hexadecimal notation	A read of address `0x00112975` returned `45524943`h.
	An '`_n`' means the signal is active low	`usr_teof_n` is active low.

Online Document

The following linking conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See "Additional Resources" for details. See "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

Chapter 1

Introduction

The FlexRay core interface is fully verified with OPB for automotive applications. This chapter contains information about recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.

System Requirements

Windows

- Windows® 2000 Professional (Service Pack 2-4)
- Windows XP Professional (Service Pack 1)

Solaris/Linux

- Sun Solaris™ 8/9
- Red Hat™ Enterprise Linux 3 (32-bit and 64-bit)

Software

- ISE™ 9.1i with applicable Service Pack

Check the release notes for the required Service Pack; ISE Service Packs can be downloaded from www.xilinx.com/xlnx/xil_sw_updates_home.jsp?update=sp.

About the Core

The FlexRay core is a Xilinx CORE Generator™ IP core, included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-FLEXRAY.

For information about system requirements, installation, and licensing options, see Chapter 2, "Licensing the Core."

Recommended Design Experience

Although the FlexRay core is a fully verified solution, the challenge associated with implementing a complete FlexRay design varies depending on the application requirements. For best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation software and User Constraints Files (UCF) is recommended.

Contact your local Xilinx representative for assistance in evaluating your specific requirements.

Additional Core Resources

For detailed information and updates related to the FlexRay core, see the following documents, located on the Xilinx FlexRay product page: www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=?key=DO-DI-FLEXRAY

- *FlexRay Data Sheet*
- *FlexRay Release Notes*

Technical Support

For technical support, visit www.support.xilinx.com/. Questions are routed to a team of engineers who have FlexRay expertise.

Xilinx will provide technical support for use of this product as described in the *FlexRay Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the FlexRay core and its related documentation.

FlexRay Core

For comments or suggestions about the core, please submit a WebCase from support.xilinx.com/. Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

Document

For comments or suggestions about this document, please submit a WebCase from support.xilinx.com/. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Chapter 2

Licensing the Core

This chapter provides instructions for installing and obtaining a license for the FlexRay core, which you must do before using it in your designs. The FlexRay core is provided under the terms of the Xilinx LogiCORE Site License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium. Purchase of the core entitles you to technical support and access to updates for a period of one year.

Before you Begin

This chapter assumes you have installed the core using either the CORE Generator IP Software Update installer or by performing a manual installation after downloading the core from the web. For information about installing the core, see the FlexRay product page.

License Options

The FlexRay core provides three licensing options. After installing the core, choose a license option.

Simulation Only Evaluation

The Simulation Only Evaluation license is provided with the FlexRay core through the CORE Generator. This license lets you generate a simulation model and simulate it using the provided demonstration test bench. (Functional simulation is supported by a dynamically generated gate-level netlist).

The Simulation Only Evaluation license lets you assess the core functionality with either the provided example design or alongside your own design and demonstrates the core simulation.

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place and route the design, evaluate timing, and perform back-annotated gate-level simulation of the core using the provided demonstration test bench.

In addition, the license lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before *timing out* (ceasing to function) at which time it can be reactivated by reconfiguring the device.

You can obtain the Full System Hardware Evaluation license in one of the following ways, depending on the core:

- By registering on the Xilinx IP Evaluation page and filling out a form to request an automatically generated evaluation license
- By contacting your local Xilinx FAE to request a Full System Hardware Evaluation license key

Click Evaluate on the core's product page for information about how to obtain a Full-System Hardware Evaluation license.

Full

The Full license is provided when you purchase the core and enables full access to all core functionality both in simulation and in hardware, including:

- Gate-level functional simulation support
- Back annotated gate-level simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time-outs

Obtaining Your License

Obtaining a Full System Hardware Evaluation License

To obtain a Full System Hardware Evaluation license, do the following:

- Navigate to the FlexRay product page: www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-FLEXRAY.
- Click Evaluate; then click Full System Hardware Evaluation
- Follow the instructions to download the CORE Generator files (delivered as an IP Update) and satisfy any additional requirements associated with the license.

Obtaining a Full License

To obtain a Full license, you must purchase the core. After purchase, you will receive a letter containing a serial number, which is used to register for access to the *lounge*, a secured area of the FlexRay product page.

- From the product page, click Register to register and request access to the lounge.
- Xilinx will review your access request and typically grants access to the lounge in 48 hours. (Contact Xilinx Customer Service if you need faster turnaround.)
- After you receive confirmation of lounge access, click Access Lounge on the FlexRay product page and log in.
- Follow the instructions in the lounge to fill out the license request form; then click Submit to automatically generate the license. An e-mail containing the license and installation instructions will be sent to you immediately.

Installing Your License File

After selecting a license option, you will receive an email containing instructions for installing your license. In addition, the email provides information about advanced licensing options and technical support.

Chapter 3

Quick Start Example Design

The instructions in this chapter show you how to quickly generate a FlexRay core, run the design through implementation using the Xilinx tools, and simulate the example design by using the provided demonstration test bench. For detailed information about the example design, see Chapter 4, "Detailed Example Design."

Overview

The FlexRay example design consists of the following:

- A demonstration test bench, to provide stimulus to the local FlexRay core.
- Two instances of a FlexRay core.

The FlexRay example design has been tested with Xilinx ISE 9.1i, ModelSim SE 6.1e and Cadence™ IUS v5.5.

Generating the Core

To generate a FlexRay core using the CORE Generator:

1. Start the CORE Generator.

   For help starting and using the CORE Generator tool, see the Xilinx CORE Generator Guide, available from the ISE documentation web page.
2. Choose File > New Project.
3. Type a directory name. In this example, the directory name *coregen* is used.
4. Set the following project options:
   - Part Options
     - From Target Architecture, select the desired family. For a list of supported families, see the *FlexRay Data Sheet*.

     Note: If an unsupported silicon family is selected, the FlexRay core does not appear in the taxonomy tree.
   - Generation Options
     - For Design Entry, select either VHDL or Verilog.
     - For Vendor, select Synplicity or Other (for XST).
5. After creating the project, locate the FlexRay core in the taxonomy tree under Automotive & Industrial > Automotive > FLEXRAY.

6. Double-click the core name to display the main FlexRay screen (Figure 3-1).

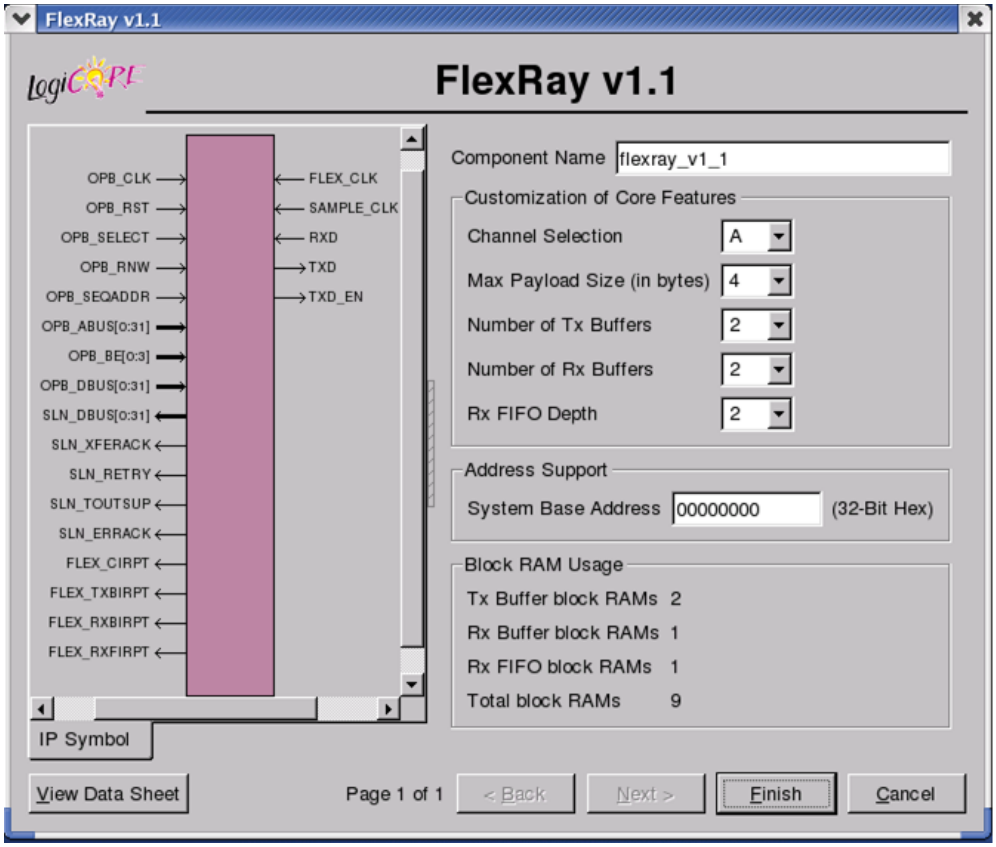

Figure 3-1: **FlexRay Main Screen**

7. In the Component Name field, enter a name for the core instance. For the example design, the name *flexray_v1_1* is used.
8. After selecting the desired parameters from the FlexRay screen, click Generate.

   Note that the test bench and example design for this core have been designed to support a minimum configuration option.

   The core and its supporting files, including the example design, are generated in the project directory. For detailed information about the example design files and directories see "Directory and File Contents."

Implementing the Example Design

After generating a core with a Full System Hardware Evaluation or Full license, the netlists and the example design can be processed with the Xilinx implementation tools. The generated output files include scripts to assist the user in running the Xilinx software.

To implement the FlexRay example design, open a command prompt or terminal window and type the following commands:

Windows

```
ms-dos> cd <coregen>\<flexray_v1_1>\implement
ms-dos> implement.bat
```

UNIX

```
unix-shell% cd <coregen>/<flexray_v1_1>/implement
unix-shell% ./implement.sh
```

These commands execute a script that synthesizes, builds, maps, and place-and-routes the example design. The script then generates a post-par simulation model for use in timing simulation, and the resulting files are placed in the results directory.

Simulating the Example Design

The FlexRay core provides a quick way to simulate and observe the behavior of the core utilizing the provided example design. There are two types of simulation: functional and timing. The provided simulation models will be in either VHDL or Verilog depending on the CORE Generator Design Entry project option selected. The Custom Output Products option is not supported for this core.

Setting up for Simulation

The Xilinx UniSim and SimPrim libraries must be mapped to the simulator. If the UniSim or SimPrim libraries are not set for your environment, see Answer Record 15338 on www.xilinx.com/support for assistance compiling Xilinx simulation models. Note that simulation scripts are provided for MTI and IUS.

Functional Simulation

Instructions for running a functional simulation of the FlexRay core using either VHDL or Verilog are provided below. Functional simulation models are provided when the core is generated. Note that implementing the core before simulating the functional models is not required.

To run a VHDL or Verilog functional simulation of the example design:

1. Set the current directory to:

   ```
   <flexray_v1_1>/simulation/functional/
   ```

2. Launch the simulation script:

   ```
   modelsim: vsim -do simulate_mti.do
   ncsim (ms-dos>): simulate_ncsim.bat
   ncsim (unix-shell%): ./simulate_ncsim.sh
   ```

The simulation script compiles the functional simulation models and the demonstration test bench, adds relevant signals to the wave window, and runs the simulation. To observe the operation of the core, inspect the simulation transcript and the waveform.

Timing Simulation

Timing simulation is only supported for Full System Hardware Evaluation and Full license types. Instructions for running a timing simulation of the FlexRay core using either VHDL or Verilog are provided. A timing simulation model is generated when the core is run through the Xilinx tools using the implement script. Note that the core must be implemented before attempting to run timing simulation.

To run a VHDL or Verilog functional simulation of the example design:

1. Set the current directory to:

```
<flexray_v1_1>/simulation/timing/
```

2. Launch the simulation script:

```
modelsim: vsim -do simulate_mti.do
ncsim (ms-dos>): simulate_ncsim.bat
ncsim (unix-shell%): ./simulate_ncsim.sh
```

The simulation script compiles the timing simulation model and the demonstration test bench, adds relevant signals to the wave window, and runs the simulation. To observe the operation of the core, inspect the simulation transcript and the waveform.

Chapter 4

Detailed Example Design

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx CORE Generator, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

- **<project directory>**
  Top-level project directory; name is user-defined.
  - <project directory>/<component name>
    Core release notes file.
    - <component name>/doc
      Product documentation.
    - <component name>example design
      Verilog and VHDL design files.
    - <component name>/implement
      Implementation script files.
      - implement/results
        Contains implement script results.
    - <component name>/simulation
      Simulation scripts.
      - simulation/functional
        Functional simulation files.
      - simulation/timing
        Timing simulation files.
    - <component name>/sw
      Software files.
      - sw/device_driver_v1_1/build
        Files for compiling low-level driver data.
      - sw/device_driver_v1_1/data
        Data files for automatic generation of parameter specific files.
      - sw/device_driver_v1_1/doc
        Documentation generated within Xilinx Platform Studio.
      - sw/device_driver_v1_1/examples
        Application examples that use the low-level driver files.

sw/device_driver_v1_1/src
Low-level driver source C files.

Directory and File Contents

The FlexRay core directories and their associated files are defined in the following sections.

<project directory>

The project directory contains all CORE Generator project files.

Table 4-1: **Project Directory**

Name	Description
<project_dir>	
<component_name>.ngc	Top-level netlist.
<component_name>.v[hd]	Verilog or VHDL simulation model.
<component_name>.xco	CORE Generator project-specific option file; can be used as an input to the CORE Generator.
<component_name>_flist.txt	List of files delivered with the core.
<component_name>_xmdf.tcl	Project Navigator integration file for the core
<component_name>.{veo|vho}	VHDL or Verilog instantiation template.

Back to Top

<project directory>/<component name>

The <component name> directory contains the release notes file provided with the core, which may include last-minute changes and updates.

Table 4-2: **Component Name Directory**

Name	Description
<project_dir>/<component_name>	
flexray_release_notes.txt	FlexRay release notes file.

Back to Top

<component name>/doc

The doc directory contains the PDF documentation provided with the core.

Table 4-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
flexray_ds554.pdf	*FlexRay v1.1 Data Sheet*

Table 4-3: **Doc Directory *(Continued)***

Name	Description
flexray_gsg339.pdf	*FlexRay v1.1 Getting Started Guide*
flexray_ug338.pdf	*FlexRay v1.1 User Guide*

Back to Top

<component name>example design

The example design directory contains the example design files provided with the core.

Table 4-4: **Example Design Directory**

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_top.v[hd]	Verilog /VHDL top-level instantiation of the core and clocking circuitry.
<component_name>.v	Verilog instantiation (generated only when Verilog design flow is used).
<component_name>_top.ucf	FlexRay example design UCF.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 4-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
implement.{sh|bat}	Implementation scripts for the top-level example design for Windows and UNIX platforms.
xst.prj xst.scr	XST Project file for the example design. Lists all the source files needs to be synthesized. Available only when the CORE Generator Vendor Project option is set to “Others.”

Back to Top

implement/results

The results directory is created by the implement script, after which the implement script results are placed in the results directory.

Table 4-6: **Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
Implement script result files.	

Back to Top

<component name>/simulation

This directory contains the simulation scripts that are provided with the core.

Table 4-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
flexray_v1_1_tb.v[hd]	Verilog or VHDL test bench. It instantiates the example design and the core and loads the example design with transactions.

Back to Top

simulation/functional

The functional directory contains functional simulation scripts provided with the core.

Table 4-8: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_mti.do	The do file used by Modelsim to compile the test bench and simulate the core.
simulate_ncsim.{sh|bat}	IUS functional simulation script for Windows and UNIX.
wave.do	A macro file to open a wave window in ModelSim and add key signals to it.
wave.sv	A macro file to open a wave window in IUS and add key signals to it.

Back to Top

simulation/timing

This directory is generated for Full System Hardware Evaluation and Full License Types.

Table 4-9: **Timing Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	ModelSim timing simulation script.
simulate_ncsim.{sh|bat}	IUS timing simulation script for Windows and UNIX.
wave.do	A macro file to open a wave window in ModelSim and add key signals to it.
wave.sv	A macro file to open a wave window in IUS and add key signals to it.

Back to Top

<component name>/sw

sw/device_driver_v1_1/build

This directory contains the files for compiling the low-level driver data that is provided with the core.

Table 4-10: **Driver Build Directory**

Name	Description
<project_dir>/<component_name>/sw/device_driver_v1_1/build	
Makefile	Makefile to compile the drivers

Back to Top

sw/device_driver_v1_1/data

This directory contains the data files for automatic generation of parameter specific files when integrated into Xilinx Platform Studio. These are not currently used.

Table 4-11: **Driver Data Directory**

Name	Description
<project_dir>/<component_name>/sw/device_driver_v1_1/data	
flexray_v2_1_0.mdd	Current MDD file, including the version of the tools interface.
flexray_v2_1_0.tcl	Provides design rule checks within Xilinx Platform Studio.

Back to Top

sw/device_driver_v1_1/doc

Contains placeholders for the documentation generated within Xilinx Platform Studio.

Table 4-12: **Driver Doc Directory**

Name	Description
<project_dir>/<component_name>/sw/device_driver_v1_1/doc	
flexray_v1_1.file	Currently empty

Back to Top

sw/device_driver_v1_1/examples

This directory contains application examples that use the low-level driver files and the FlexRay core.

Table 4-13: **Driver Examples Directory**

Name	Description
<project_dir>/<component_name>/sw/device_driver_v1_1/examples	
xflexray_intr_example.c	Contains an example design application for testing the FlexRay core in interrupt mode.
xflexray_polled_example.c	Contains an example design application for testing the FlexRay core in polled mode.

Back to Top

sw/device_driver_v1_1/src

This directory contains the low-level driver source C files.

Table 4-14: **Driver Source Directory**

Name	Description
<project_dir>/<component_name>/sw/device_driver_v1_1/src	
README.txt	Describes all files in the driver source directory.
xflexray.c	Contains the bare minimum driver functions to initialize and run the FlexRay device.
xflexray.h	Main header file for the Xflexray driver. The file header contains complete details of the device driver.
xflexray_config.c	Contains the functions needed to configure the FlexRay device for various configurations, features, and options.
xflexray_g.c	Contains a structure of all the configuration values selected during the hardware design for the FlexRay device.

Table 4-14: **Driver Source Directory**

Name	Description
xflexray_intr.c	Contains the interrupt related functions.
xflexray_l.c	Contains all the constant definitions and bare minimum functions for register read/write access.
xflexray_selftest.c	Contains the basic test functions for the sanity check of the FlexRay device.
xflexray_sinit.c	Contains the function for static initialization of the FlexRay device/driver.

Back to Top

Implementation Scripts

Note: Included with the Full and Full System Hardware Evaluation licenses only.

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. It is located at:

UNIX

```
<project_dir>/<component_name>/implement/implement.sh
```

Windows

```
<project_dir>/<component_name>/implement/implement.bat
```

The implement script performs the following steps:

1. Synthesizes the HDL example design files using XST.
2. Runs Ngdbuild to consolidate the core netlist and the example design netlist into the NGD file containing the entire design.
3. Maps the design to the target technology.
4. Place-and-routes the design on the target device.
5. Performs static timing analysis on the routed design using Timing Analyzer (TRCE).
6. Generates a bitstream.
7. Enables Netgen to run on the routed design to generate a VHDL or Verilog netlist (as appropriate for the Design Entry project setting) and timing information in the form of SDF files.

The Xilinx tool flow generates several output and report files. These are saved in the following directory which is created by the implement script:

```
<project_dir>/<component_name>/implement/results
```

Simulation Scripts

Functional Simulation

The test scripts are a ModelSim or IUS macro that automate the simulation of the test bench. They are available from the following location:

```
<project_dir>/<component_name>/simulation/functional
```

The test script performs the following tasks:

- Compiles the structural UNISIM simulation mode.
- Compiles HDL Example Design source code.
- Compiles the demonstration test bench.
- Starts a simulation of the test bench.
- Opens a Wave window and adds signals of interest (wave_mti.do/wave_ncsim.sv).
- Runs the simulation to completion.

Timing Simulation

Note: Timing Simulation test scripts are available with the Full and Full System Hardware Evaluation licenses only.

The test scripts are a ModelSim or IUS macro that automates the simulation of the test bench. They are located in:

```
<project_dir>/<component_name>/simulation/timing/
```

The test script performs the following tasks:

- Compiles the SimPrim-based gate-level netlist simulation model.
- Compiles the demonstration test bench.
- Starts a simulation of the test bench.
- Opens a Wave window and adds signals of interest (wave_mti.do/wave_ncsim.sv).
- Runs the simulation to completion.

Example Design

Top Level Example Design

The following files describe the top-level example design for the FlexRay core.

VHDL

```
project_dir>/<component_name>/example_design/<component_name>flexray_
v1_top.vhd
```

Verilog

```
project_dir>/<component_name>/example_design/<component_name>flexray_
v1_1_top.v
```

Figure 4-1 is a block diagram of the example design. The example design adds input and output buffers and DCM to the FlexRay core. This allows the entire design to be

synthesized and implemented in a target device to provide post place-and-route gate-level simulation.

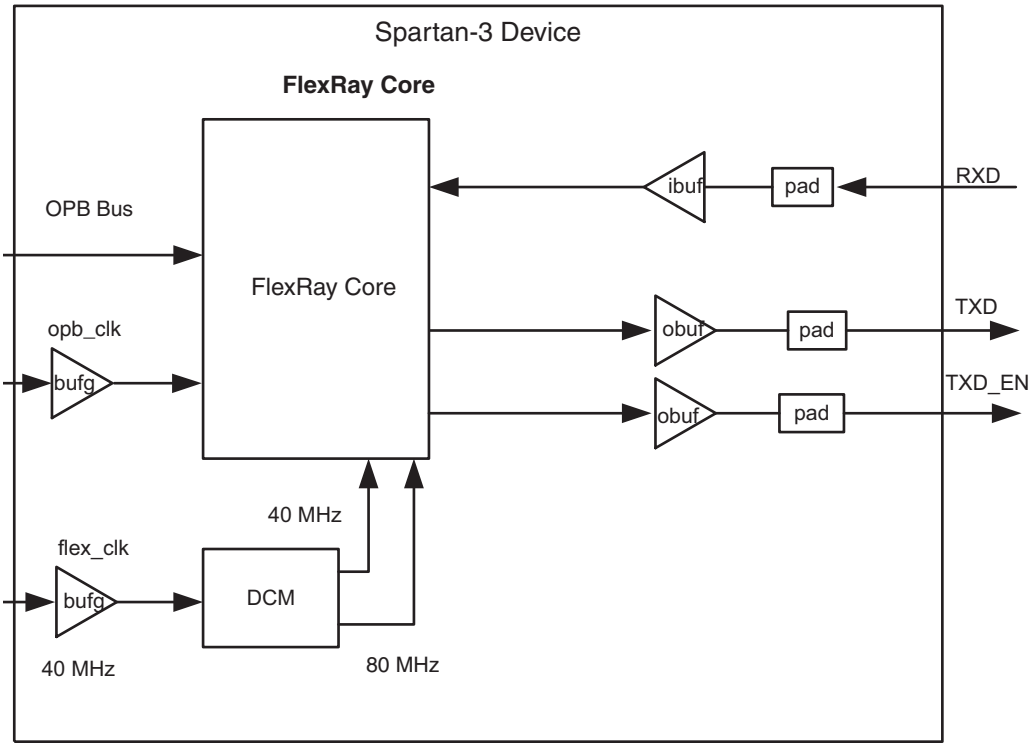

Figure 4-1: **FlexRay Example Design**

Block Level

The following files describe the block-level design for the FlexRay core:

VHDL

```
project_dir>/<component_name>/example_design/<component_name>.vhd
```

Verilog

```
project_dir>/<component_name>/example_design/<component_name>.v
```

The HDL Block level design contains the following:

- An instance of the FlexRay core
- IOB instantiation
- DCM instantiation

Demonstration Test Bench

Figure 4-2 shows the FlexRay example design test bench.

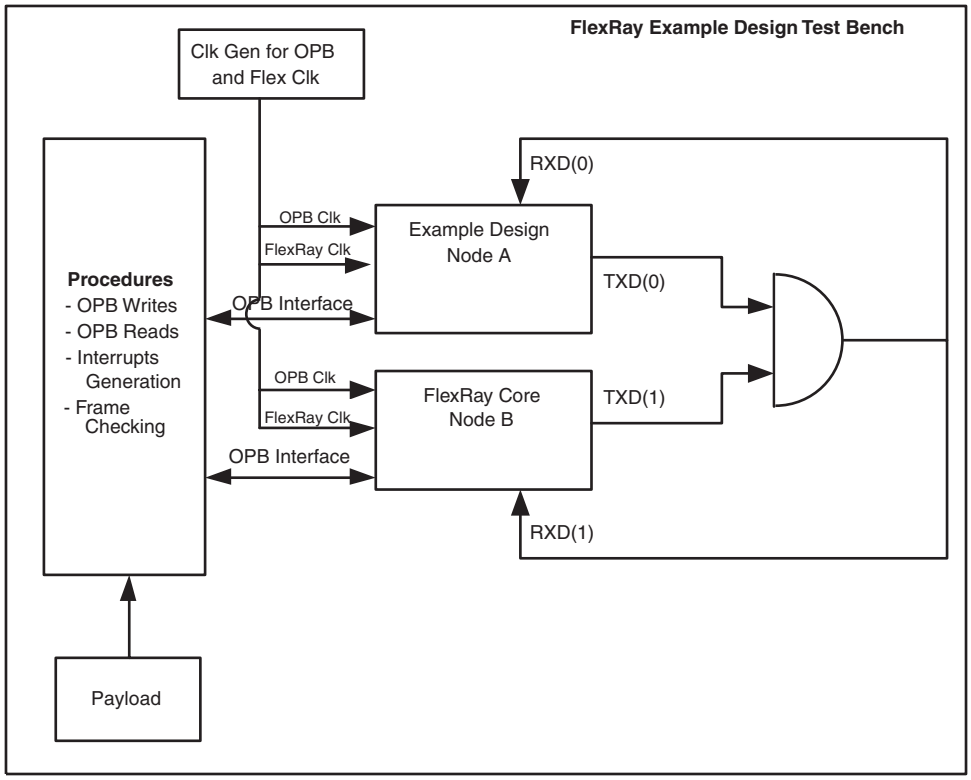

Figure 4-2: **FlexRay Example Design Demonstration Test Bench**

The following files describe the demonstration test bench.

VHDL

```
project_dir>/<component_name>/simulation/flexray_v1_1_tb_pkg_declare.vhd
project_dir>/<component_name>/simulation/flexray_v1_1_tb_pkg_define.vhd
project_dir>/<component_name>/simulation/flexray_v1_1_tb_pkg.vhd
project_dir>/<component_name>/simulation/flexray_v1_1_tb.vhd
```

Verilog

```
project_dir>/<component_name>/simulation/flexray_v1_1_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core.

The demonstration test bench performs the following tasks:

- Generates input clock signals.
- Applies a reset to the example design.
- Instantiate two nodes of FlexRay core.
- Transmits the frames from both the nodes.
- Receive the frames on both the modes.
- OPB_WRITE procedure writes the frames on OPB bus.

- OPB_READ procedure reads the data available on OPB bus.
- CHECK_FRAME procedure check the receive frame and compare with the expected result. If any error occurs, it report the message.

Customizing the Test Bench

The core should be verified in a *complete* system when connected to the FlexRay bus. However, when starting, follow the guidelines below to modify the demonstration test bench provided with the core:

- It is possible to change the contents of the configuration register. For information about using the configuration register, see the *FlexRay v1.1 User Guide*.