CADENCE

Application Note

Using the Xilinx Interface 4.0 with XACT 5.x

The intent of this application note is to alert users to some of the issues they will encounter when trying to process their designs using the pre-5.0 Cadence interface (9304, 9401, 9402) with XACT 5.0. Full support of the new XACT 5.0 features will be available with the 9404 release.

The processing flow has changed

`See P2-36 to 2-38 in Vol. II of the XACT Reference Guide for details.`

Purpose

With the release of XACT 5.0 comes support for the Unified Libraries for Xilinx designs. The purpose of this application note is to describe the flow required to process a Concept or Composer schematic that is not built from Unified Library components.

Audience

The Audience for this application is designers using the Cadence/Xilinx Interface from 9403 or earlier who are using XACT 5.0 for design implementation.

Use this application note if:

- You are making changes in an old design drawn with pre-5.0 symbol bodies.
- You require XACT 5.0 to implement designs for new FPGA devices (XC4013, XC4000H, XC3000A and XC3000L).

Xilinx Core Tools enhancements

- Improved performance by PPR 5.0
  PPR 5.0 is providing solutions with improved performance in many designs.
- Incremental Design for XC4000
  A Guide option has been added to PPR to allow you to do incremental and iterative design of XC4000 parts. Note that only LCA files created by PPR 5.0 can be used as guides.
- Better error checking up front via XNFPREP. XNFPREP will catch many design errors before the routing stage.
- More Architectures supported by PPR
  PPR has been enhanced to support XC4013, XC4000H, XC3000A and XC3000L devices.
- RPMs Replace Hard Macros

  Hard Macros have been replaced by Relationally Placed Macros (RPMs) in XACT 5.0. RPMs are an improvement over the Hard Macros in that you can specify their location in relative terms, instead of explicit locations. This gives the software more flexibility in the placement of logic blocks. When translating pre-5.0 designs containing Xilinx library hard macros, XNFMERGE 5.0 will merge the XNF files of the corresponding RPMs from $XACT/data/hmlib to create a flattened XNF file, which can then be placed and routed.
- Improved Error Messages
  Clearer, more informative error messaging has been added to the Xilinx Core Tools in 5.0.
- More Architectures supported by X-BLOX
  Support for XC3000A, XC3000L, and XC3100A devices has been added to X-BLOX.
- Improved error checking in XBLOX
  Better error checking has been added to X-BLOX to catch design errors up front.

Schematic Entry & Netlisting

New Properties and Library Elements

XACT Performance has added new features for better control of timing constraints during design implementation. You can refer to a predefined group by specifying one of the corresponding keywords - FFS, PADS, LATCHES, or RAMS and use these as path endpoints.

- You can create arbitrary groups of FF, PAD, LATCH or RAM symbols by tagging them with TNM (pronounced tee-name) attributes.
- You can create groups that are combinations of existing groups using TIMEGRP symbols.

GXilinx

Since TNM attributes and TIMEGRP symbols are new with this release, you will need to make some changes to a number of GXilinx property files in order to make GXilinx aware of these elements before you can process them correctly in your netlist.

Supporting TNM

TNM is a property that attaches to flip-flops latches, pads and RAMs. GXilinx depends on two files to tell it how to process properties attached to a design. The file, `xilinx.prop`, contains a list of legal properties to be translated into XNF. The `xilinx.format` file specifies the format to be used by GXilinx when translating each property to an XNF file. Both files are located at `<cds_dir>/interface`. Editing these Xilinx-specific property files will allow GXilinx to recognize these properties when explicitly attached to individual FFS (flip-flops), LATCHES, PADS, and RAM modules.

Follow these step to include TNM in GXilinx:

1. **Edit xilinx.prop**

   Add the following line to the bottom of the file:

   ```
   TNM:    permit (body);
   ```

2. **Edit xilinx.format**

   Add the following line to the bottom of the file:

   ```
   TNM:    TNM=;
   ```

3. **Strip out the "=" prefix attached to each TNM property in the XNF file generated by GXilinx.**

This additional step is necessary because the "=" prefix indicates to the Xilinx tools that the property is a user property, and user properties are usually ignored.

Supporting TIMEGRP And TIMESPEC

TIMEGRP and TIMESPEC serves as a placeholder for timing specifications. To support these features, you need to create a library with these two primitives. This library can then be included in your global.cmd file whenever you are designing XC4000 orXC3000A/L devices.

Fortunately the TIMESPEC symbol already exists in the LCA4K library. This makes the task of creating the library much easier.

To create the new library, you will need to do the following:

1. **Login as cds_mgr**

   Login as cds_mgr and change directory into <cds_install>/lib.

2. **Create the library directory**

   From <cds_install>/lib, create the directory, "tspec"

3. **Create timespec.lib**

   In the *tspec* directory, create a file called `tspec.lib`. This will be the library map file.

   Contents of `tspec.lib`:

   ```
   FILE_TYPE = LOGIC_DIR;
   "TIMESPEC" 'timespec';
   "TIMEGRP"  'timegrp';
   END.
   ```

4. **Copy the TIMESPEC directory from lca4k**

   Copy the entire TIMESPEC directory from <cds_install>/lib/lca4k to tspec.

   From the new tspec directory, type:

   ```
   cp -r ../lca4k/timespec .
   ```

5. **Create the TIMEGRP symbol**

   Create the timegrp symbol by copying the `timespec` directory to the `tspec` directory and modifying the timegrp symbol body.

   ```
   cp -r timespec timegrp
   ```

   From Concept, modify the timegrp body file and change the name "TIMESPEC" to "TIMEGRP"

6. Update master.lib

Edit <cds_install>/lib/master.lib and include the following entry:

```
"tspec"'tspec/tspec.lib';
```

Using TIMESPEC or TNM On Macros

You can also use TIMESPEC and TNM on a macro to control its timing. To do this, you must use block mode methodology--that is, you must process the macro schematic separately to create an XNF for your macro, then use XNFMERGE to merge the XNF with the rest of your design. When your design is merged, TIMESPEC and TNM properties are propagated from your top schematic down to the predefined groups in the macro (FFS, LATCHES, RAMS, or PADS). Note that for TNM properties, the primitives underlying the macro must all be of the same type (FFS, LATCHES, RAMS, or PADS. The steps to follow are:

1. **Create your macro schematic**

   Draw the schematic for your macro design and do not include any TIMESPEC or TNM parameters in your schematic.

2. **Run GXilinx to create an XNF for the macro**

   Be sure to run GXilinx with the -m option so that the schematic is treated as a macro circuit. Use the following command line:

   ```
   gxilinx -m <macro_name>
   ```

3. **Attach the macro property on your macro symbol**

   To prevent GXilinx from processing the schematic underneath your macro symbol, attach this property to the macro symbol body:

   ```
   xilinx=macro \parameter
   ```

   Now you are ready to use the macro on your schematic.

4. **Add the macro to your design and attach TIMESPEC and TNM attributes to it.**

   Place the symbol body of your macro onto your schematic and attach TIMESPEC and TNM properties according to the Xilinx Reference Manual.

5. **Process your entire design through GXilinx**

   Before running GXilinx to process your complete design, you must add a directive to your compilier.cmd to tell it where to find expansion rules for Xilinx netlist processing.

   In your compiler.cmd file, add the following line:

```
exp_rules_file   "<cds_dir>/tools/interface/xilinx.exprule"
```

Note: Be sure to enclose the path in quotes.

FUNCTIONAL SIMULATION

Designs without X-BLOX or Carry Logic

Designs that consists entirely of schematics without X-BLOX or memory modules may be simulated as described in the documentation of the 4.0 design kit. An XNF netlist is generated from the design and is then translated into a Verilog netlist for unit-delay simulation. Rapidsim users can go directly into Rapidsim without any translation.

Figure 2-1 Concept Flow

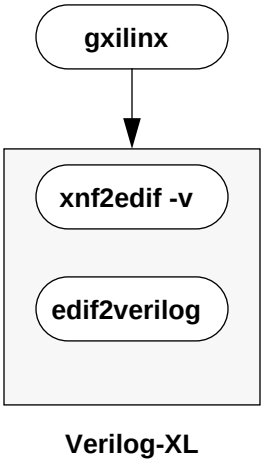
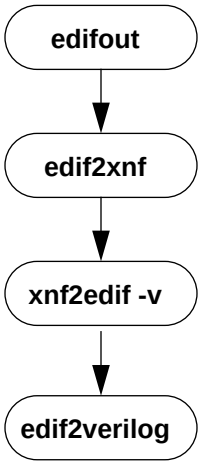


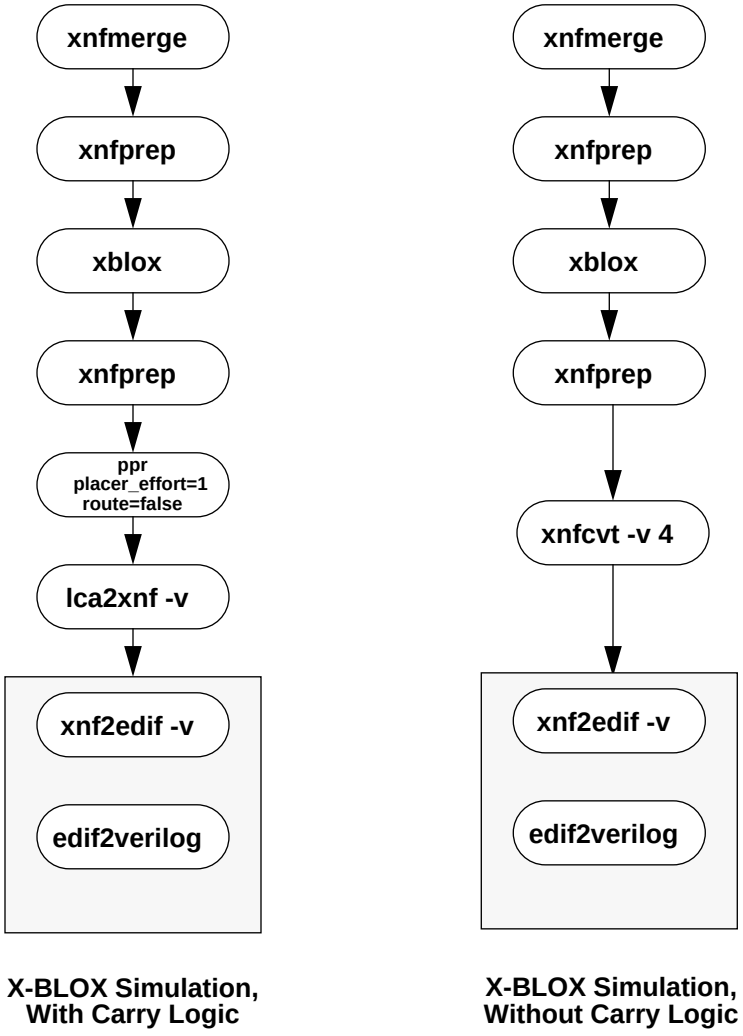
Figure 2-2 Composer Flow



Designs with XBLOX Modules

Designs with X-BLOX modules require additional processing steps before simulation. X-BLOX which do not expand out to use carry logic can be simulated without running PPR. If you use COUNTER, ADD_SUB or ACCUM modules, PPR must be run to convert the carry logic primitives into gates.

Figure 2-3 X-BLOX Simulation Flow



X-BLOX Simulation, With Carry Logic

X-BLOX Simulation, Without Carry Logic

Designs with User Created Hard Macros

Hard Macros have been replaced by Relationally Placed Macros (RPM) in XACT 5.0. For designs that currently include user-created hard macro files you must translate the hard macro files into RPM files. A utility called HM2RPM is provided to convert your hard macros into RPMs.

You must convert any user-created hard macros into RPMs with the HM2RPM program when you use these macro symbols in a design that will be processed by XACT 5.0. When you perform this conversion, you must specify whether your design uses elements from the Unified Libraries or from previous libraries. By default, HM2RPM outputs an XNF file that is compatible with schematics generated with pre-5.0, pre-Unified library elements.

Figure 2-4 Verilog Functional Simulation with User Created Hard Macros

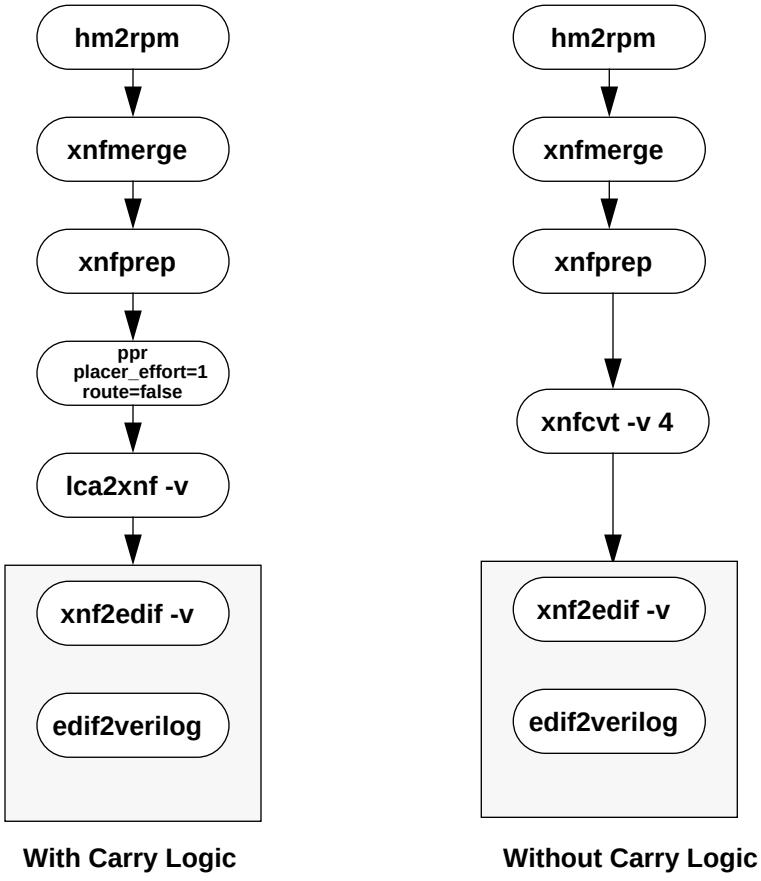
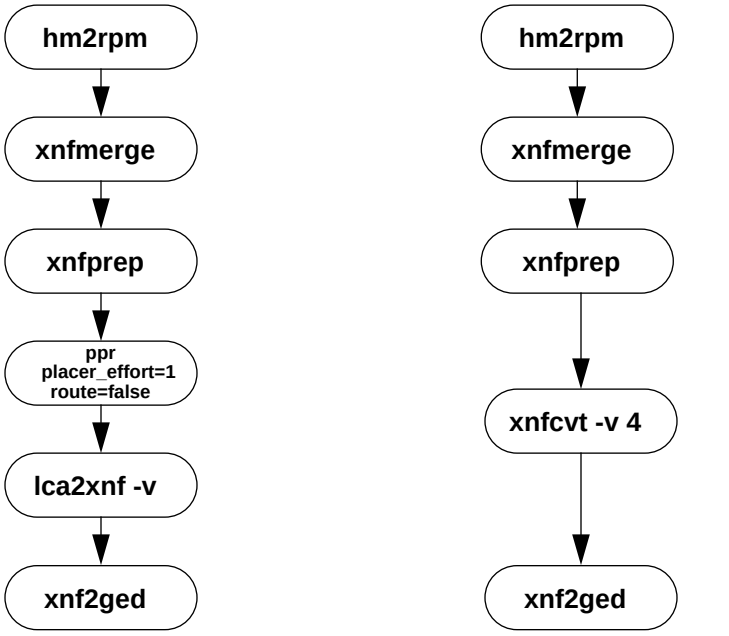


Figure 2-5 Rapidsim Functional Simulation with User Created Hard Macros



With Carry Logic **Without Carry Logic**

Processing the Design

Running Xmake

Most of the problems interfacing the pre-5.0 interface with XACT 5.0 stem from the change in the XNF netlist format and the addition of the new CY4 carry logic primitive.

- When implementing designs from GXilinx you must run XMAKE with the -l option. This will tell XMAKE to specify use of the old pre-Unified libraries when it calls X-BLOX, XABEL and MEMGEN. Failure to specify the -l option in designs with pre-5.0 X-BLOX, XABEL, or MEMGEN modules will result in error messages about using mixed libraries.
- If running MEMGEN manually, specify the `old_library` option. For example:

```
memgen <memory_name> old_library = true
```

- **DO NOT** mix version 4.0 XNF with 5.0 XNF.

Timing Verification

Getting An XNF For Timing Simulation

There are two programs that need to be run before starting simulation. They are LCA2XNF and XNFBA.

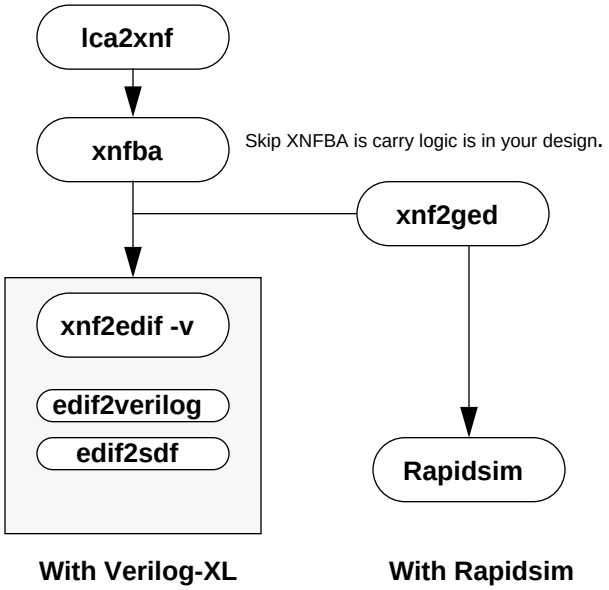
The LCA2XNF program converts an LCA file to an XNF file, which you can use for functional or timing simulation. If the LCA input file is placed and routed,, the resulting XNF file contains all worst-case block and net delays for use in timing simulation. In XC3000 designs, APR writes the delay information to the LCA file; in XC3000A and XC4000 designs, you must run XDelay -dw on the LCA file to annotate the delay information. You can then use the XNF file created by the LCA2XNF program to generate a simulation file for timing verification.

By default, LCA2XNF version 5.0 will write out a version 5 XNF netlist. Be sure to use the -v option in LCA2XNF to write out a version 4 XNF netlist for timing verification, as only version 4 XNF files can be read by XNF2GED and XNF2EDIF. Customers using TIMENET for Verilog simulation must modify the script to include the -v option where LCA2XNF is invoked.

The XNFBA program combines the pre-route XNF files and the post-route XNF file into a new file which has the original symbol and signal names with post-route delays. XNFBA will not annotate carry logic primitives into a version 4 XNF. Skip XNFBA if carry logic is present in the design.

Note: XNF2GED v4.4-p1 will not warn you about your input being a version 5.0 XNF netlist if you neglect to use the -v option. However, the translated netlist will not be correct.

Figure 2-6 Timing Simulation Flow



Things to look out for:

DESIGNS BASED ON OLD LIBRARIES

- Run XMAKE with the `-l` option for back compatibility with the old libraries. This option will ensure that MEMGEN, and X-BLOX will write their outputs using old libraries when processing your design.
- Do not mix XACT 5.0 library elements with pre-5.0 library elements, as this will lead to problems in backannotation and simulation.

XILINX LIBRARY HARD MACROS (Concept)

- Copy the XNF files for any hard macros in your design to your design directory from `$XACT/data/hmlib`

USER-CREATED HARD MACROS

- Run `hm2rpm macro_name dflt_logic=true` on each user hard macro in your design.
- Move all CST file constraints which apply to user hard macros onto your schematic. LOC constraints do not work in XACT 5.0 when specified from a constraint file.

INVALID CONDITIONS

As a result of improved error checking by XNFPREP, design errors which were ignored before will turn up as errors in XACT 5.0. Some of the more important changes you will need to make are:

- Remove invalid parameters, such as overly generalized constraints like LOC=CLB_R*C*.

  If you have this connection in your XC4000 design:

  ```
  ipad-->ibuf-->bufgp,
  ```

  you must constrain the IBUF, because PPR assumes that if you use an IBUF between the IPAD and BUFGP, you did not want to use a dedicated BUFGP input pin. Connect the IPAD directly to BUFGP if you want PPR to use a dedicated BUFGP input pin.

XC3000A/L & XC3100A Designs

- PPR is used to route XC3000A, XC3000L, and XC3100A designs. If you were using a .CST constraints file in the pre-5.0 software, the constraints would have been written in a different format from that

required by PPR. Run the CSTCVT utility on the CST file to convert it to the required PPR format.

3000A/3000L/3100A designs:

In XACT 5.0, PPR is used to route XC3000A, XC3000L, and XC3100A designs. If you were using a .CST constraints file in the pre-5.0 software, the constraints would have been written in a different format from that required by PPR. For XACT 5.0, these constraints must be converted to a format that PPR 5.0 will understand. Run the CSTCVT utility on the CST file to convert it to the required PPR format.