HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
www.hunteng.co.uk
www.hunt-dsp.com

Converting Hardware Interface Layer (HIL) v1.x Projects to v2.0

v 1.0 R.Williams 11-12-02

For modules in the HERON-FPGA and HERON-IO families, HUNT ENGINEERING provide a comprehensive VHDL support package. The VHDL package consists of a "top level", with corresponding user constraints file, VHDL sources and simulation files for the Hardware Interface Layer, and User VHDL files as part of many examples.

The Hardware Interface Layer correctly interfaces with the Module hardware, while the top level (top.vhd) defines all inputs and outputs from the FPGA on your module.

With Version 2.0 of the Hardware Interface Layer, HUNT ENGINEERING introduced two new interface functions that provide full access to multiple HERON input FIFOs and multiple HERON output FIFOs.

This document discusses the required steps for converting an ISE project that uses Version 1 of the Hardware Interface Layer to a project that uses Version 2.0. This conversion is suitable for those users that wish to make use of the new multiple FIFO access functionality provided in the new HIL version.

For users of the old version, v1 Hardware Interface Layer, it is not completely necessary to convert your projects to use the new version. Where you have an existing project that works correctly with the single FIFO interface components there is no need to make this conversion.

History

Rev 1.0	First written

Converting an Existing v1.x HIL Project

This tutorial assumes that you are converting a v1.x HIL project to use the new v2.0 Hardware Interface Layer. It is assumed that both the old project that you are converting, and the new project you are converting to will be used under the same version of ISE.

Although there are several different versions of the Xilinx ISE development tools available, it is not the purpose of this document to discuss modifying your project to work with a different version of ISE.

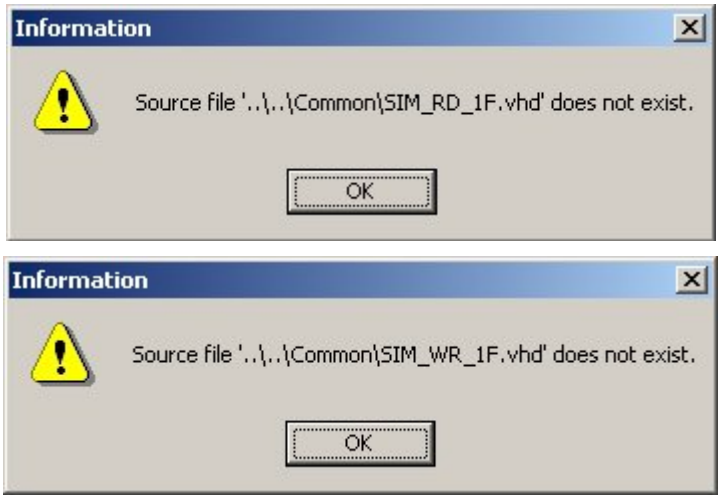
If you intend to change the version of ISE with which your project works, please do this before starting the v1.x HIL to v2.0 HIL conversion. There are several documents provided on the HUNT ENGINEERING CD that describe the process of moving between ISE versions, should you wish to do this.

Converting the Example1 Project

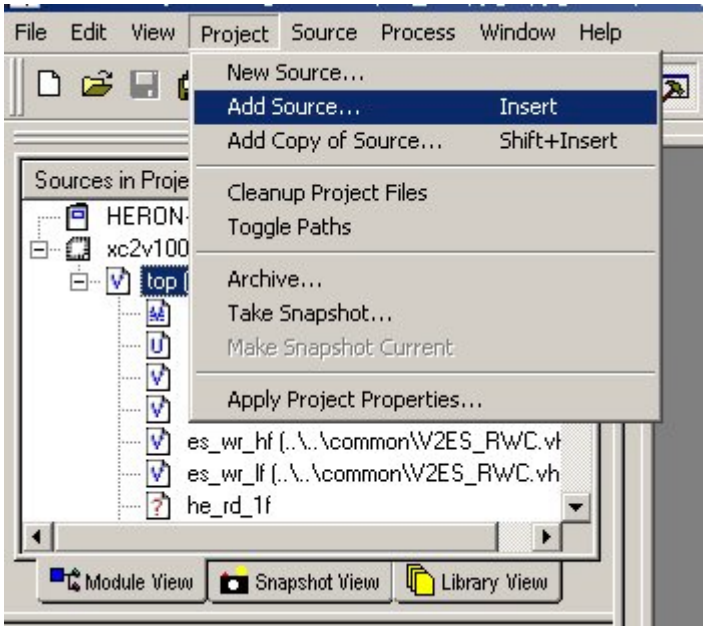
For this tutorial we will illustrate the steps in converting an Example1 project for the HERON-FPGA3. For all other board types and examples these key steps are the same.

1. Create a new working directory. Copy your current **/Common** and **/Example1** directories into this new directory. By creating a new directory structure the conversion process will not affect any other projects that currently use the v1.x /Common directory installed on your machine.
2. Delete all of the files in the new **/Common** directory.
3. Copy the entire contents of the HERON-FPGA3 **/Common** directory from your HUNT ENGINEERING CD into the directory emptied in the previous step. Please note, this must be done with a version of the HUNT ENGINEERING CD that includes the new HIL v2.0 support.
4. Open the ISE Project Navigator. Select 'File → Open Project'. To load the Example1 project, go to the directory **dir\example1\ISE** (where **dir** is the new working directory you created in step 1). Open the Example1 project file.
5. When the project is first opened with the new /Common directory ISE will report four missing files, as follows:

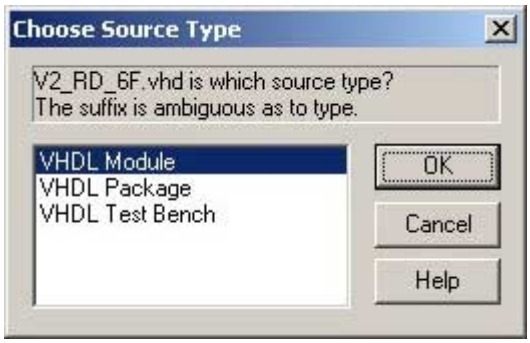
For each window click 'ok' to dismiss the window.



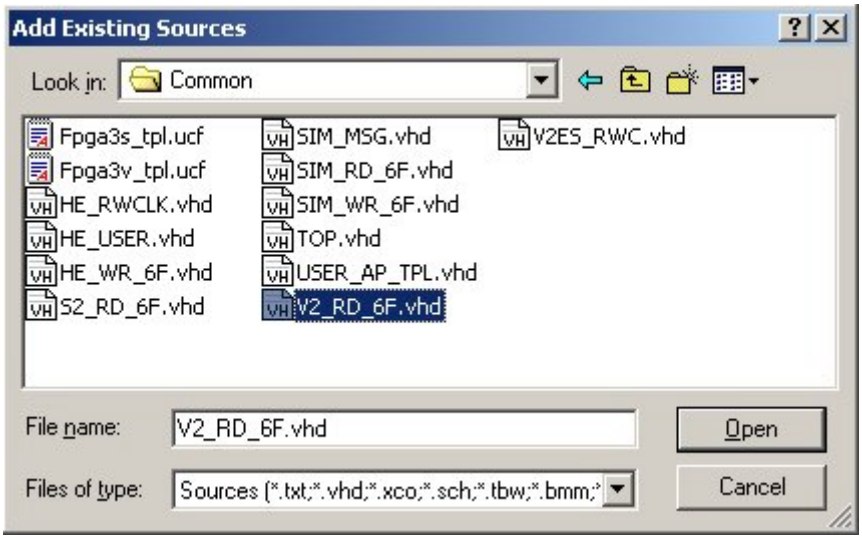
6. When the project has finished opening use Windows Explorer or an equivalent utility to delete all files with the extension '`.jhd`' in the **`/Example1/ISE`** directory.
7. The next step is to add the new HIL components into the project. To do this click on the menu item 'Project-> Add Source'. There are two files that must be added to the project… one for the six FIFO read interface, and one for the six FIFO write interface.



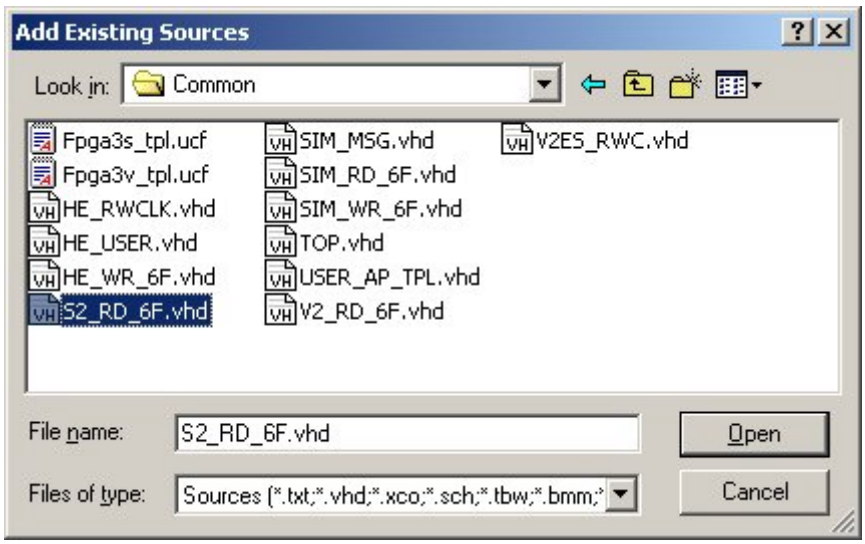
Please note for each file added to the project, you will be asked what the source type of that file is. In each case for the FIFO interface components, ensure that you select VHDL Module for the source type.



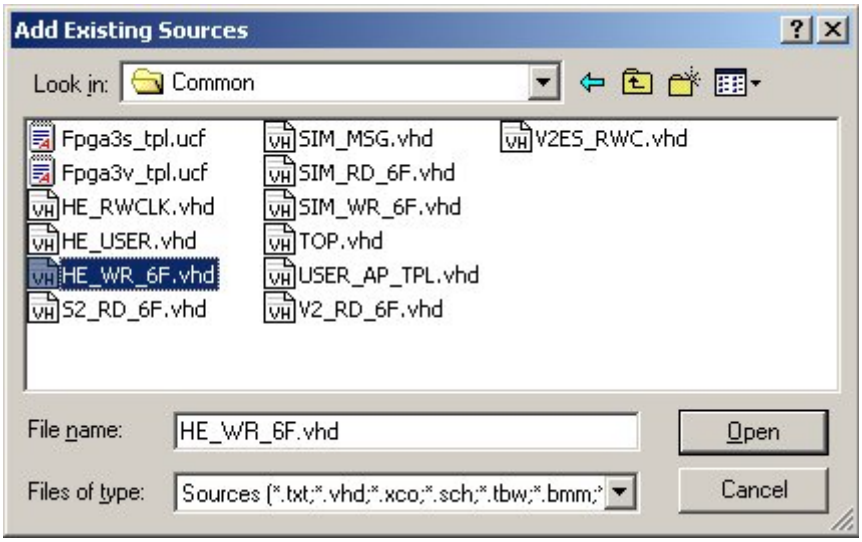
The files that are to be added to the project are found in the new /Common directory. If the Example1 project opened was for a Virtex-II module, then for a Six FIFO Read Interface you will need to add the file 'V2_RD_6F.VHD', or for a Spartan-II project, you should instead add the file 'S2_RD_6F.VHD'. For the Six FIFO Write Interface there is simply one file HE_WR_6F.



Six FIFO Read Interface for Virtex-II projects.



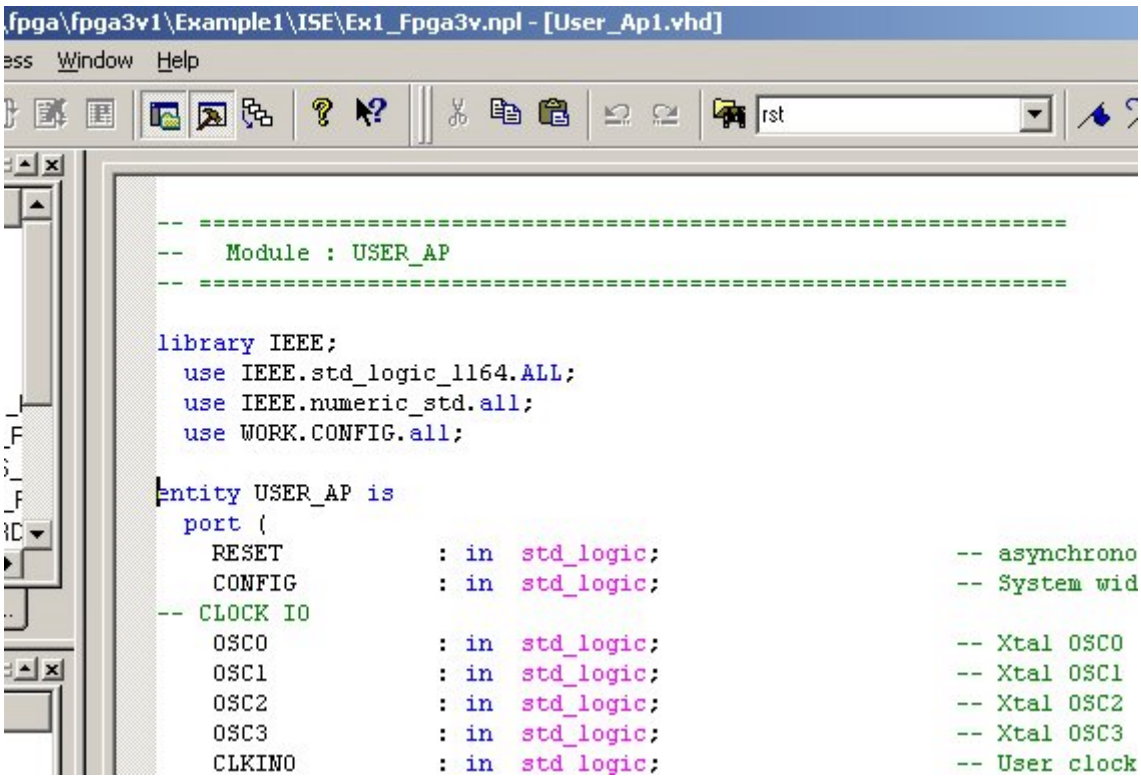
Six FIFO Read Interface for Spartan-II projects.



Six FIFO Write Interface component.

8. Once you have added the read-interface file and the write-interface file, open the USER_AP template file in ISE. This file is named 'user_ap_tpl.vhd' and is located in the new /Common directory. Navigate to this file using File->Open and then browsing to the correct directory. Also open the USER_AP module for the example by double-clicking on the correct source module in the 'Sources in Project' window.

   Copy the entire 'ENTITY' declaration at the top of the user-ap template, and paste it over the existing ENTITY declaration in the 'user-ap' file of the example.



9. Now that the new entity declaration is in place, close the USER_AP_TPL file. By changing the entity declaration we have changed the signal names for the HERON FIFO Read Interface and the HERON FIFO Write Interface. The Version 1 Hardware Interface Layer declarations are:

```
-- FIFO IN / HE_RD_1F interface
INFIFO_D      : in  std_logic_vector(31 downto 0);
INFIFO_EMPTY  : in  std_logic;
INFIFO_SEL    : out std_logic_vector(5 downto 0);
INFIFO_READ   : out std_logic;
-- FIFO OUT / HE_WR_1F interface
OUTFIFO_D     : out std_logic_vector(31 downto 0);
OUTFIFO_FULL  : in  std_logic;
OUTFIFO_SEL   : out std_logic_vector(5 downto 0);
OUTFIFO_WRITE : out std_logic;
```

These declarations are now replaced in Version 2 of the Hardware Interface Layer as follows:

```
-- FIFO IN / HE_RD_6F interface
INFIFO_READ_REQ : out std_logic_vector(5 downto 0);
INFIFO_DVALID   : in  std_logic_vector(5 downto 0);
INFIFO_SINGLE   : in  std_logic_vector(5 downto 0);
INFIFO_BURST    : in  std_logic_vector(5 downto 0);
INFIFO0_D       : in  std_logic_vector(31 downto 0);
```

```
INFIFO1_D       : in  std_logic_vector(31 downto 0);
INFIFO2_D       : in  std_logic_vector(31 downto 0);
INFIFO3_D       : in  std_logic_vector(31 downto 0);
INFIFO4_D       : in  std_logic_vector(31 downto 0);
INFIFO5_D       : in  std_logic_vector(31 downto 0);
-- FIFO OUT / HE_WR_6F interface
OUTFIFO_READY   : in  std_logic_vector(5 downto 0);
OUTFIFO_WRITE   : out std_logic_vector(5 downto 0);
OUTFIFO_D       : out std_logic_vector(31 downto 0);
```

With this change to the signal names we must change the signal names that appear in the architecture body.

The HIL V1 Example1 project transfers data from one single HERON input FIFO to a single HERON output FIFO. For the new project, we are using a Hardware Interface Layer that allows us to use all six input FIFOs and all six output FIFOs concurrently. For the purpose of this document we will choose FIFO 0 to use in place of the single FIFO connection made in the example that we started with. If you wish to find out how the Six FIFO Write and Six FIFO Read interfaces can be used to transfer data to and from many FIFOs at the same time, please refer to the examples provided on the HUNT ENGINEERING CD that demonstrate multiple FIFO access.

So to change the example to work with the new V2 Hardware Interface Layer, the following changes should be made below the **begin** statement in the architecture of the USER_AP module. Please note, this only requires changing signal names in your design, not the names of component pins.

Replace signal `INFIFO_D` with `INFIFO0_D`

Replace signal `INFIFO_EMPTY` with `INFIFO_BURST(0)`

Replace signal `INFIFO_READ` with `INFIFO_READ_REQ(0)`

Remove signal `INFIFO_SEL`

Remove signal `OUTFIFO_SEL`

Replace signal `OUTFIFO_FULL` with `OUTFIFO_READY(0)`

Replace signal `OUTFIFO_WRITE` with `OUTFIFO_WRITE(0)`

(Note: `OUTFIFO_D` stays as `OUTFIFO_D`)

If you have made these changes correctly to the original project, you should then have code that is similar to the following VHDL code for the FIFO connections:

```
-------------------------------------------
-- FIFOs
SRC_FCLK_RD <= '0';   -- clock source for READ FIFO
SRC_FCLK_WR <= '0';   -- clock source for WRITE FIFO
SRC_FCLK_G  <= MCLK;  -- global clock source for both FIFOs
OUTFIFO_D <= INFIFO0_D;
iTRANS : TRANSFER
    port map (
      RST            => RESET,
```

```
CLK             => FCLK_G,
INFIFO_EMPTY    => INFIFO_BURST(0),
OUTFIFO_FULL    => OUTFIFO_READY(0),
INFIFO_READ     => INFIFO_READ_REQ(0),
OUTFIFO_WRITE   => OUTFIFO_WRITE(0) );
```

10. At this point we have a project that should synthesize. Check this by highlighting the file 'top.vhd' in the Sources view and double-clicking on Synthesize in the Process view. The design should synthesize without error. If there are any errors reported when you synthesize, re-check all of the previous steps.
11. At this point we have a project that is correct as far as syntax is concerned, but we need to make one more important set of changes before the project is safe and functionally correct.

    With the introduction of the new Six FIFO Read and Write interface components it is now possible to access multiple FIFOs at the same time both for reading and writing. Along with the increase in available FIFOs there has been a change to the way the interface signals behave.

    For example, in the case of the V1 HERON FIFO Read Interface, the `INFIFO_READ` signal would be used to read a word of data when set high, as long as the FIFO was not empty (`INFIFO_EMPTY` set low). For V2 of the Hardware Interface Layer, we now have six read-request signals that can be set to 'request' data, when available, from each input FIFO. This new signal does not indicate the actual transfer of data.

    When data is available to be transferred from one of the HERON Input FIFOs, the corresponding data valid signal will becoming asserted (`INFIFO_DVALID`(x) signal) when data is valid.

    Considering this the following changes are necessary for our new example project. The following code will transfer data, when available from HERON input FIFO 0 to HERON output FIFO 0. This code must be written in place of the existing code that uses the module 'transfer'. That is, the instanciation of TRANSFER must be removed from the source.

```
-- Transfer data from FIFO 0 in to FIFO 0 out
OUTFIFO_D <= INFIFO0_D;

-- Request data when the output interface is ready
INFIFO_READ_REQ(0) <= OUTFIFO_READY(0);
-- Tie unused requests inactive (low)
INFIFO_READ_REQ(5 downto 1) <= (others=>'0');
-- Transfer data when it becomes available
OUTFIFO_WRITE(0) <= INFIFO_DVALID(0);
-- Tie unused write signals inactive (low)
OUTFIFO_WRITE(5 downto 1) <= (others=>'0');
```

Conclusion

When converting a V1 HIL project to V2, there a few key steps to follow. The first step is to replace the /Common directory for the project you are converting.

Then the new modules for the Read Interface and Write Interface must be added to the project.

Following this, change the User_Ap entity declaration to match that in the User-Ap template file in the new /Common directory.

With the new entity in place, change the use of all FIFO control and data signals to match those used by the new Hardware Interface Layer. Remember, this involves both changing the signal names that appear in the architecture body **and** changing how those signals are used.

If you want to find out more information about using the FIFO interfaces in the new Hardware Interface Layer, you can read the document 'Accessing Multiple FIFOs in your FPGA Design', which can be found on the HUNT ENGINEERING CD. There are also several examples provided that demonstrate using these FIFOs for reading and writing multiple FIFOs concurrently.

The new Hardware Interface Layer, version 2.0 is only different to that of version 1 by the use of new HERON Read FIFO and HERON Write FIFO interfaces. All other interfaces, such as the HSB message interface and the A/D and D/A interfaces available for HERON-IO modules have remained the same between versions.

Please note, in working through this tutorial you will have taken an old (v1.0 HIL compatible) project and modified it to use the new interfaces. This was demonstrated by converting Example1. However, the new project that is created will not match the Example1 that is provided on the HUNT ENGINEERING CD. If you wish to start from Example1 for the development of a new project, it is recommended that you start from the appropriate Example1 provided on the CD.