

Using the Xillinx MicroBlaze Processor

© 2012 ... Impulse Accelerated Technologies, Inc.

Table of Contents

Part I Quick Start Tutorials

| 1 | Tutorial 1: Hello World on the MicroBlaze platform | 5 |
|---|--|----|
| | Loading the Hello World Application | 5 |
| | Compiling the Application for Simulation | 7 |
| | Hello World Source Code | 9 |
| | Building the Application for the Target Platform | 11 |
| | Exporting Files from CoDeveloper | 13 |
| | Creating a Platform Using Xilinx System Builder | 14 |
| | Importing the Generated Hardware to the Xilinx Tools | 22 |
| | Importing the Application Software to the Xilinx Tools | 26 |
| | Building and Downloading the Application | |
| 2 | Tutorial 2: Complex FIR Filter for Xilnx Design Suite 13.4 (or | |
| | higher) | 34 |
| | Loading the Complex FIR Application | 35 |
| | Understanding the Complex FIR Application | 36 |
| | Compiling the Application for Simulation | 37 |
| | Building the Application for the Target Platform | 39 |
| | Exporting Files from CoDeveloper | 41 |
| | Creating a Platform Using Xilinx Tools | |
| | Configuring the New Platform | 50 |
| | Importing the Generated Hardware | 55 |
| | Generating the FPGA Bitmap | 57 |
| | Importing the Application Software | 58 |
| | Running the Application | 64 |
| | | |

3

1 Quick Start Tutorials



Overview

Impulse C can be used to generate hardware modules that are directly connected to an embedded processor (such as the Xilinx MicroBlaze) or to other hardware elements that may have been described using other design tools or techniques. As you have seen in earlier tutorials (found in the CoDeveloper primary Help information), the Impulse C programming model emphasizes the use of data streams, signals, and shared memories for process-to-process communication. These interfaces can be used to connect Impulse C processes to a wide variety of hardware devices and processors.

For embedded processors such as MicroBlaze, there are multiple possible ways to provide communication between a software application running on the processor, and a hardware accelerator running in the FPGA fabric. These include (among others):

- Using the PLB (Processor Local Bus) to create an Impulse C peripheral on a shared bus
- Using the FSL (Fast Simplex Link) to create a high-speed data stream
- Using shared memory

The following tutorials focus on data streaming applications, using both PLB and FSL. Shared memory examples are provided in your CoDeveloper product installation.

Using the Xilinx MicroBlaze Processor

This following tutorials will lead you step-by-step through the compilation, execution and RTL generation of your first Impulse C applications on the Xilinx MicroBlaze platform.

The tutorials that follow assume that you have previously gone through at least one of the tutorials included in your standard CoDeveloper installation. It is also assumed that you are somewhat familiar with the Xilinx ISE and Platform Studio (EDK) tools.

Note: These Tutorials uses Xilinx Design Suite version 13.4 or later.

Tutorial 1: Hello World on the MicroBlaze platform

Using the "Generic" Xilinx FPGA Platforms

If you are not using an embedded MicroBlaze processor or the Xilinx Platform Studio (EDK) tools, then you will use CoDeveloper to generate logic (in the form of HDL output files) that is connected to other hardware modules or to external device pins, typically through the use of named ports. (See Tutorial 4 in the CoDeveloper primary Help file for more information about **co_port** types.)

The following section describes how HDL files generated by CoDeveloper can be easily imported into a Xilinx ISE project for synthesis. (These steps are similar for other synthesis and simulation tools, including the Synplicity Synplify Pro tools, Mentor's Modelsim, and others.)

Exporting Files to Xilinx ISE

See Also

1.1 Tutorial 1: Hello World on the MicroBlaze platform



Overview

This tutorial will demonstrate how to generate hardware and related software interfaces in the form of RTL (Register Transfer Logic) HDL descriptions appropriate for use with the Xilinx Platform Studio[™] (Embedded Development Kit) software. This tutorial assumes Platform Studio version 13.4; other versions of the Xilinx software may require changes to some of the steps presented.

The sample project will implement a trivial "Hello World" application. This sample project includes a software process (running on the MicroBlaze processor) and a hardware process (running in the FPGA) that communicate via a single stream over the PLB bus.

Note: The FSL bus may also be used for increased I/O performance, but if FSL is used, some steps will differ from those described in this tutorial. <u>Tutorial 2</u> describes the use of FSL.

The purpose of this tutorial is to take you through the entire process of generating hardware and software interfaces and importing the relevant files to the Xilinx environment. The tutorial will also describe how to create the platform and downloadable FPGA bitmap using the Xilinx tools.

This tutorial will require approximately 90 minutes to complete (including software run times).

Steps

Loading the Hello World Application Compiling the Application for Simulation Building the Application for the Target Platform Hello World Source Code Exporting Files from CoDeveloper Importing the Generated Hardware to the Xilinx Tools Importing the Application Software to the Xilinx Tools

Note

This tutorial assumes you have purchased or are evaluating the CoDeveloper Platform Support Package for Xilinx Microblaze, and that you have installed and have valid licenses for the Xilinx ISE and Platform Studio (EDK) products.

See Also

Hello World Source Code

1.1.1 Loading the Hello World Application

Hello World Tutorial for MicroBlaze, Step 1

To begin, start the CoDeveloper Application Manager by selecting Application Manager from the Start - > Programs -> Impulse Accelerated Technologies -> CoDeveloper program group.

Note: this tutorial assumes that you have already read and understand the basic "Hello World" tutorial presented in the CoDeveloper User's Guide.

The first step is to copy the Impulse HelloWorld example to a working directory. Locate the HelloWorld example in the Impulse tree, typically on your C:\ disk drive:



Copy these files to a working directory so that if a mistake is made the original files are not altered, in this case a working directory is made in the C:\sandbox\HelloWord folder. Be certain to copy all of the files.

Launch the Impulse CoDeveloper software by double-clicking the HelloWorld.icProj file or from the Start menu with:

Start -> Impulse Accelerated Technologies -> CoDeveloper X.xx -> CoDeveloper Application Manager. You should have a display very similar to this one:

| | | -1-1 |
|---|--|-------|
| | cation Manager Universal Edition - [HelloWorld] | _ 🗆 : |
| Eile Edit View Project Too | | |
| : 🚰 😅 🖬 📔 🕮 🎽 🔝 🗍 | ※■ 85 つ / 1 / 1 / 1 4 公 20 頁 : 曲 22 ▶ 曲 12 ▶ 曲 12 12 ◆ 頁 | |
| roject Explorer | * × | |
| 급 금 (글린 Application HelloWorld | | |
| Source Files | | |
| sw C HelloWorld_sw.c | | |
| hw C HelloWorld_hw.c sw C HelloWorld.c | | |
| | | |
| Project Files Document Files | | |
| Readme.htm | | |
| Other Files | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| ld | | ņ |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| 🖞 Build 🖼 Find in Files 🖸 Syste | m | |
| ady | | NUM |

The HelloWorld project is now loaded and it is time to compile it for simulation.

See Also

Compiling the Application for Simulation

1.1.2 Compiling the Application for Simulation

Hello World Tutorial for MicroBlaze, Step 2

Before compiling the HelloWorld application to the target MicroBlaze platform, let's first take a moment to understand its basic operation and the contents of its primary source files.

The specific process that we will be compiling to hardware is represented in HelloWorld_hw.c by the following function:

void say_hello(co_stream hello_out, co_parameter arg)

This hardware process simply outputs data to an output stream (**hello_out**), based on the value of the compile-time parameter **arg**.

This hardware process is accompanied in the HelloWorld_hw.c file by a configuration function

(config_hello) that specifies the connection between the hello_out output stream and the corresponding input stream for the software process hear_hello.

The process **hear_hello** (which is declared as an **extern** function at the start of HelloWorld_hw.c) is defined in the source file HelloWorld_sw.c. This process will, when compiled, be loaded onto the MicroBlaze processor and will communicate with the FPGA hardware via an automatically-generated hardware/software interface. In this case the software process simply reads data from the stream associated with the hardware process. (In a more typical Impulse C application there are multiple input/ output streams associated with a given hardware process.)

The remaining source file, HelloWorld.c, includes a main function for the application. This main function makes reference to the lower-level configuration file through the Impulse C standard function **co_initialize()**, which has been declared as an **extern** function and may be found in HelloWorld_hw.c.

Note: the organization of source files shown in this example is not required, but is recommended in order to simplify the compilation process. It is a good idea to keep the **main()** function of your application in one file (as shown), and to place hardware and software processes in different files. Doing so will avoid potential problems with cross-compiler incompatibilities between hardware and software processes as well as simplifying the compilation and linking of complete applications for desktop simulation.

Simulating the Hello World Application

To compile and simulate the application for the purpose of functional verification:

 Select Project -> Build Software Simulation Executable (or click the Build Software Simulation Executable button) to build the HelloWorld.exe executable. A command window will open, displaying the compile and link messages as shown below:

| Building target build exe'in file Makefile | |
|--|----------------|
| C:/mpules/CoDeveloper3/MinGW/bin/coc"; 9:"C:/mpules/CoDeveloper3/binGW/bin/coc"; 9:"C:/mpules/CoDeveloper3/binGW/bin/coc"; 9:"C:/mpules/CoDeveloper3/binGW/bin/coc"; 9:"C:/mpules/CoDeveloper3/binGW/bin/coc"; 9:"C:/mpules/CoDeveloper3/binGW/bin/coc"; 9: "C:/mpules/CoDeveloper3/binGW/bin/coc"; 9: "C:/mpules/CoDeveloper3/bin/cod"; 9: "C:/mpules/CoDeveloper3/bin/cod; 9: "C:/m | _hw.c _sw.c |
| 🕮 Build 🖼 Find in Files 🗔 System | |

 You now have a Windows executable representing the HelloWorld application implemented as a desktop (console) software application. Run this executable by selecting Project -> Launch Simulation Executable. A command window will open and the simulation executable will run as shown below:

| "C:\sandbox\MBHe | lloWorld\HelloWorld.ex | e" |
|------------------|------------------------|----|
| CPU listening fo | | |
| FPGA hardware sa | | |
| Press any key to | continue | |
| | | |
| | | |
| | | |
| | | · |

Verify that the simulation produces the following output:

```
CPU listening for hello...
FPGA hardware says: 0
FPGA hardware says: 1
FPGA hardware says: 2
FPGA hardware says: 3
FPGA hardware says: 4
FPGA hardware says: 5
FPGA hardware says: 6
FPGA hardware says: 7
FPGA hardware says: 8
FPGA hardware says: 9
```

Note that, although the application display messages indicating that the FPGA hardware has generated the indicated values, in fact the values are being generated by a hardware process which, for simulation purposes, has been compiled as a software process on your host development system (Windows). The next steps will show how to take this same Impulse C application and compile to actual hardware.

See Also

Building the Application for the Target Platform HelloWorld Source Code

1.1.3 Hello World Source Code

HelloWorld_sw.c

```
// Copyright(c) 2003-2010 Impulse Accelerated Technologies, Inc.
// All Rights Reserved.
11
// HelloWorld_sw.c: Process to be executed on the target CPU.
11
#include "co.h"
#include "cosim_log.h"
#include <stdio.h>
#ifdef IMPULSE_C_TARGET
#define printf xil_printf
#endif
                                    // NOTE: The only interface objects you
void hear_hello(co_stream hello_in)
can use for arguments
                                    // are ImpulseC objects, such as
{
co_streams. However, software
   int i;
                                    // only functions can communicate with
global objects.
   int hi = 0;
   IF_SIM(cosim_logwindow log = cosim_logwindow_create("hear_hello");)
   co_stream_open(hello_in, O_RDONLY, INT_TYPE(32));
```

9

```
printf("CPU listening for hello...\n\r");
for ( i = 0; i < 10; i++ ) {
    co_stream_read(hello_in, &hi, sizeof(int));
    printf("FPGA hardware says: %d\n\r", hi);
}
co_stream_close(hello_in);
}
```

HelloWorld_hw.c

```
// Copyright(c) 2003-2010 Impulse Accelerated Technologies, Inc.
// All Rights Reserved.
//
// HelloWorld hw.c: Hardware processes and configuration code.
//
#ifdef WIN32
#include <windows.h>
#endif
#include <stdio.h>
#include "co.h"
#define MONITOR
#ifdef MONITOR
#include "cosim log.h"
#endif
// Software process
extern void hear hello(co stream hello in);
                                              // Software processes that communicate with the
hardware must be
                                              // forward declared as extern.
void say hello(co stream hello out, co parameter arg)
#ifdef MONITOR
  IF_SIM(cosim_logwindow log_say_hello;)
#endif
  co int32 count = 0;
  co_int32 iterations = (co_int32) arg;
#ifdef MONITOR
  IF_SIM(log_say_hello = cosim_logwindow_create("say_hello");)
#endif
  co_stream_open(hello_out, O_WRONLY, INT_TYPE(32));
  for ( count = 0; count < iterations; count++ ) {
     co stream write(hello out, &count, sizeof(co int32));
  }
  co_stream_close(hello_out);
```

```
}
void config hello(void * arg)
  co_process say_hello_proc, hear_hello_proc;
  co stream hello stream = co stream create("hello stream", INT TYPE(32), 8);
#ifdef MONITOR
  IF_SIM(cosim_logwindow_init();)
#endif
  say hello proc = co process create("say hello", (co function)say hello,
                                         // Number of arguments declared in the function interface.
                          2.
                          hello_stream, // Each argument must be listed individually and in the correct
order.
                          10);
                                         // This is how to pass a COMPILE-TIME value to a function.
  hear_hello_proc = co_process_create("hear_hello", (co_function)hear_hello,
                           1,
                           hello_stream);
  co_process_config(say_hello_proc, co_loc, "PE0"); // The "PE0" must be defined in the Platform
Support Package.
}
co architecture co initialize(int arg)
  // The "xilinx mb opb" must be defined in the Platform Support Package.
  return co_architecture_create("hello", "xilinx_mb_opb", config_hello, (void *) arg);
}
```

```
See Also
```

Building the Application for the Target Platform

1.1.4 Building the Application for the Target Platform

Hello World Tutorial for MicroBlaze, Step 3

Specifying the Platform Support Package

The next step, prior to compiling and generating the HDL and related output files, is to select a platform target. Which target you select has a number of implications, including:

- The output file format (e.g. VHDL, Verilog or other intermediate language)
- The primitive components that will be referenced in the output (e.g. memory cores or buffer types)
- The types of optimizations performed during compilation
- · The software library components that will be generated

To specify a platform target, open the Generate Options dialog as shown below using Project -> Options -> Generate.

| latform Support Package: | on |
|--|--|
| Xilinx MicroBlaze PLB (VHDL) | - |
| Hardware Optimization and Generation | |
| Enable constant propagation | Generate dual clocks |
| Scalarize array variables | Generate active-low reset |
| Relocate loop invariant expressions | Use std_logic types for VHDL interfaces |
| | Do not include co_ports in bus interface |
| Additional optimizer options: | Additional library options: |
| | |
| Floating Point Options | Output Directories Hardware build directory |
| | |
| Include floating point library | Hardware build directory: |
| Include floating point library | Hardware build directory: hw Software build directory: sw |
| Include floating point library Include co_meth library Enable floating point optimization Allow double precision types and operato | Hardware build directory: hw Software build directory: sw Hardware export directory: |
| Include floating point library Include co_math library Enable floating point optimization | Hardware build directory: mw Software build directory: gw Hardware expect directory: |

Specify *Xilinx MicroBlaze PLB (VHDL)* as shown. Also specify "hw" and "sw" for the hardware and software directories as shown, and specify "EDK" for the hardware export and EDK/SDK/HelloWorld for the software export directories. Click OK to save the options and exit the dialog.

Note: The Xilinx MicroBlaze Platform Support Package includes three options for the on-chip bus: FSL, PLB and OPB. Which bus you choose is a system-level decision; in general the FSL bus will provide higher hardware/software data throughput but will be less able to handle external peripherals. Consult your Xilinx MicroBlaze documentation for additional details. Note that OPB is not used in the latest Xilinx products.

Generate HDL for the Hardware Process

To generate hardware in the form of HDL files, and to generate the associated software interfaces and library files, select Generate HDL from the Project menu, or click on the Generate HDL button. A series of processing steps will run in a command window at the bottom of the main window.



Toolbar Build Buttons

When processing is complete you will have a number of resulting files created in the hw and sw subdirectories of your project directory. Take a moment to review these generated files. They include:

Hardware directory ("hw")

• Generated VHDL source files (HelloWorld_comp.vhd and HelloWorld_top.vhd) representing the hardware process and the generated hardware stream interface.

- A lib subdirectory containing required VHDL library elements.
- A pcore subdirectory containing generated files required by the Xilinx EDK tools.

Software directory ("sw")

- C source files extracted from the project that are required for compilation to the embedded processor (in this case HelloWorld_sw.c and HelloWorld.c).
- A generated C file (co_init.c) representing the hardware initialization function. This file will also be compiled to the embedded processor.
- A **driver** subdirectory containing additional software libraries to be compiled as part of the embedded software application. These libraries implement the software side of the hardware/ software interface.

If you are an experienced Xilinx EDK user you may copy these files manually to your EDK project area and, if needed, modify them to suit your needs. In the next step, however, we will show how to use the hardware and software export features of CoDeveloper to move these files into your Xilinx EDK project automatically.

See Also

Exporting Files from CoDeveloper

1.1.5 Exporting Files from CoDeveloper

Hello World Tutorial for MicroBlaze, Step 4

From the CoDeveloper Application Manager. Recall that in Step 3 you specified the directory "EDK" as the export target for hardware and EDK/SDK/HelloWorld for software:

| | on |
|--|--|
| tform Support Package: | - |
| linx MicroBlaze PLB (VHDL) | 1 |
| Hardware Optimization and Generation | Generate dual clocks |
| Scalarize array variables | Generate active-low reset |
| Relocate loop invariant expressions | Use std_logic types for VHDL interfaces |
| | Do not include co ports in bus interface |
| Additional and an entropy of the second | Additional library options: |
| Additional optimizer options: | |
| | |
| | |
| loating Point Options | Output Directories |
| Roating Point Options | Hardware build directory: |
| | |
| Include floating point library | Hardware build directory: hw Software build directory: |
| Include floating point library Include co_math library | Hardware build directory: Inw Software build directory: Isw |
| Include floating point library Include co_math library Enable floating point optimization | Hardware build directory: hw Software build directory: w Hardware export directory: |
| Include floating point library Include co_math library Enable floating point optimization Allow double precision types and operato | Hardware build directory: hw Software build directory: w Hardware export directory: EDK |
| Include floating point library Include co_math library Enable floating point optimization Alow double-precision types and operator Use higher latency, faster clock operator | Hardware build directory: hw Software build directory: w Hardware export directory: |

These export directories specify where the generated hardware and software processes are to be copied when the Export Software and Export Hardware features of CoDeveloper are invoked. Within these target directories (in this case "EDK"), the specific destination (which may be a subdirectory under "EDK") for each file previously generated is determined from the Platform Support Package architecture library files. It is therefore important that the correct Platform Support Package (in this case Xilinx MicroBlaze OPB) is selected prior to starting the export process.

To export the files from the build directories (in this case "hw" and "sw") to the export directories (in this case the "EDK" directory), select Project->Export Generated Hardware (HDL) and Project->Export Generated Software, or select the Export Generated Hardware and Export Generated Software buttons from the toolbar.



Note: you must select BOTH Export Software and Export Hardware before going onto the next step.

You have now exported all necessary files from CoDeveloper to the Xilinx tools environment.

See Also

Creating a Platform Using Xilinx System Builder

1.1.6 Creating a Platform Using Xilinx System Builder

Hello World Tutorial for MicroBlaze, Step 5

As you saw in the previous step, CoDeveloper creates a number of hardware and software-related

output files that must all be used to create a complete hardware/software application on the target platform (in this case a Xilinx FPGA with an embedded MicroBlaze processor). You can, if you wish, copy these files manually and integrate them into your existing Xilinx projects. Usually, you will use the export features of CoDeveloper (prior step) to integrate the files into the Xilinx tools semi-automatically. This section will walk you through the process, using a new EDK System Builder (Platform Studio) project as an example.

Note: you must have the Xilinx ISE and EDK software (version 13.4 or later) installed in order to proceed with this and subsequent steps. This tutorial demonstrates the steps need for EDK version 13.4. Other versions may require minor changes.

When exporting the project the directories shown below were created in the Impulse project directory. The organization of this directory tree is what the Xilinx tools expect to find and the Impulse generated files are already in the correct locations.



Directory structure for EDK/SDK projects

Creating a New Xilinx Platform Studio Project

Now we'll move into the Xilinx tool environment. Begin by launching Xilinx Platform Studio (from the Windows Start->Xilinx ISE DesignSuite 13.4-EDK->Xilinx Platform Studio and creating a new project by slecting the4 Create New Project Using Base System Builder option.



After clicking the BSB option the Create New XPS Project Using BSB Wizard will appear.

| | lbox\HelloWorld\EDK\syster | n.xmp | | Browse |
|--|--|---|-----------------------|------------------------------|
| elect an Interconne | ect Type | | | |
| C AXI System | | | | |
| current and f | erface standard recently ad future versions of Xilinx IP a ce Guide on xilinx.com. | | | |
| | | | | |
| PLB System | | | | |
| PLB is the leg Spartan6 and new designs | gacy bus standard used by d Virtex6. PLB IP will not s that may migrate to future face Simplifications docume | upport newer FPGA e FPGA families. Deta | families, so is not r | ecommend for |
| PLB is the leg Spartan6 and new designs PLBv46 Inter | Virtex6. PLB IP will not s that may migrate to future | upport newer FPGA e FPGA families. Det ent on xilinx.com. | families, so is not r | ecommend for |
| PLB is the leg Spartan6 and new designs PLBv46 Inter | d Virtex6. PLB IP will not s that may migrate to future face Simplifications docume | upport newer FPGA e FPGA families. Det ent on xilinx.com. | families, so is not r | ecommend for |
| PLB is the leg Spartan6 and new designs PLBv46 Inter elect Existing .bsb S | d Virtex6. PLB IP will not s that may migrate to future face Simplifications docume | upport newer FPGA e FPGA families. Det ent on xilinx.com. | families, so is not r | ecommend for found in the |

Browse to the directory where you want the project file stored, this will be the EDK directory you chose above. Then choose if you will be using the newer AXI interface standard or the legacy PLB standard supported up to and including Xilinx Spartan and Virtex 6 families. In this example we will be using the legacy PLB flow. If you have an earlier .bsb (settings file) from a previous design to reuse then select it using the "SelectExisting" control. For the "Set Project Perhiperal ..." control, navigate to your project directory that CONTAINS the EDK directory created above.

NOTE: The Peripheral Repository Search Path must be the directory that is TWO levels above the pcores directory. In our example above, the full path to pcores is:

C:\sandbox\HelloWorld\EDK\pcores

Therefore, we selected C:\sandbox\HelloWorld for the repository search path.

Then click "OK" and in the next wizard page select "I would like to create a new design" and click "Next".

Choose your development board from the dropdown boxes. This example will use the following board (you should choose the reference board you have available for this step):

Board Vendor: Xilinx Board Name: Virtex 6 ML605 Evaluation Platform Board Revision: D

| 😵 Base System B Welcome | CANCEL STREET, STRE | Board | System | Processor | Peripheral | Cache | <u>?</u> × Summary |
|----------------------------|---|----------------|-------------------|---------------------|---|-----------|-----------------------|
| Board Selection | | | -, | | | | |
| Select a target dev | velopment l | board. | | | | | |
| Board | | | | | | | |
| I would like to a | create a sys | tem for the fo | llowing developme | ent board | | | |
| Board Vendor | Xilinx | | | | | | - |
| Board Name | Virtex 6 M | 1L605 Evaluati | on Platform | | | | - |
| Board Revision | D | | | | | | - |
| C I would like to a | create a sys | tem for a cust | om board | | | | |
| - Board Information | | | | | | | |
| Architecture virtex6 | | Device | 16 | Package + ff1156 | Sp) | eed Grade | |
| Use Stepping | | - ACOVIAZA | <i></i> | | | | |
| Reset Polarity Act | ive Hiah | | | | | | |
| Related Information | | | | | | | |
| Vendor's Website | | | | | | | |
| Vendor's Contact In | nformation | | | | | | |
| Third Party Board D | Definition Fil | es Download V | Vebsite | | | | |
| | | | | | The ML605 board utilize | | |
| | | | | | R3 SDRAM SODIMM me System ACE CF contro | | |
| | | | | | the on-board high spe C) connector. Due to A | | |
| | | | | | s an option from ML605 | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| More Info | | | | | < Back | Next > | Cancel |

Click Next to continue with the System Builder Wizard. In the next wizard page, make sure that you select a single processor system.



Click Next to continue with the System Builder Wizard.

With this Wizard page you select the Processor Type, Clock Frequency and Local Memory allocation. If you will be doing floating point operations in software then you may also enable the Floating Point Unit.

| 😵 Base System Builder | | | | | | ? × |
|--|--------------|-------------|-----------|------------|--------|---------|
| Welcome | Board | System | Processor | Peripheral | Cache | Summary |
| Processor Configuration Configure the processor(s). | | | | | | |
| Reference Clock Frequency Processor 1 Configuration Processor Type | <u>.</u> | | | | | MHz |
| System Clock Frequency | 100.00 | | | | | ▼ MHz |
| Local Memory | 8 KB | | | | | - |
| Debug Interface | On-Chip HW D | ebug Module | | | | * |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| More Info | | | | < Back | Next > | Cancel |

Note: the Base System Builder options that follow may be different depending on the development board you are using.

The wizard will present a peripheral selection page similar to this one:



Instruction and Data Cache

The configuration of instruction and data cache is optional and configurable. We are not using cache in this example so you may skip the Wizard page. If you choose to use either or both caches the simply select the appropriate check box and choose the available displayed memory options, these come from the configuration of your evaluation board or your custom design.

Base System Summary Page

| 狻 Base System Builder | | | | | | ?) |
|---|---|---|--|------------|--------|--------|
| Welcome | Board | System | Processor | Peripheral | Cache | Summa |
| Summary | | | | | | |
| Below is the summary of | the <mark>system you</mark> are | creating. | | | | |
| ystem Summary | | | | | | |
| Core Name | Instance Name | Base Address | High Address | | | |
| ☐ Processor 1 | | 0x88000000 0x81440000 0x81420000 0x81420000 0x81400000 0x8400000 0x8400000 0x8400000 0x00000000 0x00000000 | 0x8FFFFFF 0x8104FFF 0x8100FFFF 0x8142FFFF 0x8140FFFF 0x8400FFFF 0x8340FFFF 0x00001FFF 0x00001FFF | | | |
| ile Location - Overall - C:\sandbox\HelloV - C:\sandbox\HelloV - C:\sandbox\HelloV - C:\sandbox\HelloV - C:\sandbox\HelloV - C:\sandbox\HelloV | Vorld\EDK\system.m Vorld\EDK\data\syst Vorld\EDK\etc\fast_u Vorld\EDK\etc\dowr | hs em.ucf runtime.opt lload.cmd | | | | |
| Save Base System Build C:\sandbox\HelloWorld\\ More Info | | ile | | < Back | Finish | Cancel |

Click the "Finish" button to save the base system and exit the Wizard.

The next steps will demonstrate how to configure the MicroBlaze processor and create the necessary I/ O interfaces for our sample application.

See Also

Importing the Generated Hardware to the Xilinx Tools

1.1.7 Importing the Generated Hardware to the Xilinx Tools

Hello World Tutorial for MicroBlaze, Step 7

After the Base System Builder Wizard finishes you will be taken to the Xilinx Platform Studio System Assembly View.

| Navigator 🗙 | IP Catalog ↔ □ ₽ × | | Bus Interfac | es Ports | Addresses | | 8 |
|------------------|------------------------------|--------------|--|-----------------|------------------|--------------------------------|----|
| | 2000 30 30 30 30 | | Name | Bus Name | IP Type | IP Version | |
| Design Flow | Description IP Version IP Ty | | dimb | | 📩 Imb v10 | 2.00.b | |
| | EDK In | | ilmb | | 📩 lmb_v10 | 2.00.b | |
| | Analog | | mb_plb | | 👷 plb_v46 | 1.05.a | |
| 2 | Bus and | | . E microblaze_ | 0 | 📩 microblaze | 8.20.b | |
| Run DRCs | E Clock, R | | Imb_bram | | 📩 bram_block | | |
| | 🕀 Commu | | • dimb_cntir | | 🙀 Imb_bram | | |
| plement Flow | E Commu | | 🗄 ilmb_cntlr | | 🙀 Imb_bram | | |
| ipienient now | DMA an | | E FLASH | | 🙀 xps_mch | | |
| 4 . | ⊕ Debug ⊕ FPGA R | s | DDR3_SDR. | | 📌 mpmc | 6.05.a | |
| | General | 1 • • | mdm_0 | | 🚖 mdm | 2.00.b | |
| enerate Netlist | E IO Mod | • | Ethernet | | 👷 xps_ether | | |
| | E Interpr | ÷ | DIP_Switch | 8 | 👷 xps_gpio | 2.00.a | |
| 1 | Memory | 1 | E LEDs_8Bit | | 👷 xps_gpio | 2.00.a | |
| 1010 | E PCI | 1 9 | E LEDs_Positi | | 👷 xps_gpio | 2.00.a | |
| | 🕀 Periphe | P | Push_Butt. IIC DVI | 2 | 🚖 xps_gpio | 2.00.a 2.03.a | |
| erate BitStream | Processor | 2 | IIC_EEPROI | 4 | 👷 xps_iic | 2.03.a | |
| | Utility | | . IIC FMC LP | | xps_ic | 2.03.a | |
| SDK | Verificat Project Loc | | | C. | xps_ic | 2.03.a | |
| \checkmark | ⊡ USER | | E SYSACE CO. | | xps_sysace | | |
| Export Design | Project Peri | | B RS232 Uar. | | xps_uartite | | |
| | | | - clock gene | | t clock gen | | |
| imulation Flow | | | FLASH CE | | 📩 util vector | | |
| | | | proc sys r. | | proc sys r | | |
| 6 | | | | | | | |
| Jan. | | | | | | | |
| nerate HDL Files | | | l <u> </u> | | | | |
| | | | | T | | | |
| - | Search IP Catalog: Clear | Master Slave | amaster/Slave 🚩 | license (aval) | r Connected OUnc | duction 🕄 Beta 🚟 Development | |
| J.L. | Search IP Catalog: | A Superseded | Discontinued | License (eval) | CLocal APre Pro | auction was beca paceveropment | |
| unch Simulator | 😵 Project 🎯 IP Catalog | | hary 🔣 🍪 Si | stom Assombly | View 🖸 🔗 Gran | hical Design View | |
| | - Hojece - P Catalog | Design Summ | | vacent Assembly | | | |
| | Errors | | | | | | ⇔⊑ |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Next, add the module representing the HelloWorld hardware process to your development system. Select the IP Catalog tab. Notice the USER icon.

| Project Loc |
|--------------|
| 😟 USER |
| Project Peri |
| |

Select the + symbol to open the USER modules. Notice that theplb_helloworld object.

| Project Loc | | |
|--------------|--------|----------------|
| | 1.00.a | plb helloworld |
| Project Peri | | pib_nellowond |

Right-click the plb_helloworld IP to display its context menu and select Add IP (as shonw):

| E USER | | |
|----------------|-------------|-----------|
| 🗣 plp- | <u>b100</u> | plb_hello |
| Project Periph | Add IP | |
| | View MPD | |

Now find the plb_helloworld... object in the system assembly view. Move the mouse pointer into the Bus Connections pane (to the left of the Bus Interfaces) and when the pointer is on the same line as the plb_hello_0 instance the open slave connection to the PLB will display (as shown);

| | | | 0.00.0 |
|----------|---------------------------|-------------|--------|
| | DDR2_SDR | 👷 mpmc | 6.05.a |
| | . <u>.</u> <u>.</u> mdm_0 | 📩 mdm | 2.00.b |
| <u>ک</u> | plb_hello_0 | 🔫 plb_hello | 1.00.a |
| 5 | Ethernet | 🚖 xps_ether | 4.00.a |
| 5 | DIP_Switch | 📩 xps_gpio | 2.00.a |

Note that the circle (indicating a bus slave) for the plb_hello_0 instance is empty, click the empty circle to implement the connection to the PLB.

| □ ⊕ imb_cntlr | 👷 lmb_bram | 3.00.b |
|-----------------|---------------|--------|
| DDR2_SDR | 📩 mpmc | 6.05.a |
| ti⊕ mdm_0 | 🚖 mdm | 2.00.b |
| b plb_hello_0 | 😌 plb_hello | 1.00.a |
| SPLB mb_plb | | |
| Ethernet | 📩 拉 xps ether | 4.00.a |

Specify the Addresses

Now you will need to set the addresses for each of the Peripherals specified for the platform. This can be done simply by choosing the Addresses tab resulting in a window that is similar to this:

| nstance | Base Name | Base Address | High Address | Size | | Bus Interface(s) | Bus Name | Lock |
|----------------------------|-------------|--------------|--------------|------|---|------------------|----------|------|
| microblaze_0's Address Map | | | | | | | | |
| dlmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | - | SLMB | dlmb | |
| ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | - | SLMB | ilmb | |
| Ethernet_MAC | C_BASEADDR | 0x81000000 | 0x8100FFFF | 64K | - | SPLB | mb_plb | |
| Push_Buttons_5Bit | C_BASEADDR | 0x81400000 | 0x8140FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_Positions | C_BASEADDR | 0x81420000 | 0x8142FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_8Bit | C_BASEADDR | 0x81440000 | 0x8144FFFF | 64K | - | SPLB | mb_plb | |
| DIP_Switches_8Bit | C_BASEADDR | 0x81460000 | 0x8146FFFF | 64K | - | SPLB | mb_plb | |
| - IIC_SFP | C_BASEADDR | 0x81600000 | 0x8160FFFF | 64K | - | SPLB | mb_plb | |
| IIC_FMC_LPC | C_BASEADDR | 0x81620000 | 0x8162FFFF | 64K | - | SPLB | mb_plb | |
| - IIC_EEPROM | C BASEADDR | 0x81640000 | 0x8164FFFF | 64K | - | SPLB | mb plb | |
| - IIC DVI | C BASEADDR | 0x81660000 | 0x8166FFFF | 64K | - | SPLB | mb plb | |
| | C_BASEADDR | 0x83600000 | 0x8360FFFF | 64K | • | SPLB | mb_plb | |
| RS232 Uart 1 | C BASEADDR | 0x84000000 | 0x8400FFFF | 64K | - | SPLB | mb plb | |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | - | SPLB | mb_plb | |
| FLASH | C_MEM0_BASE | 0x8C000000 | 0x8DFFFFFF | 32M | - | SPLB | mb_plb | |
| DDR3_SDRAM | C_MPMC_BASE | 0x90000000 | 0x9FFFFFFF | 256M | - | SPLB0 | mb_plb | |
| Unmapped Addresses | | | | | | | | |

Expand the "Unmapped Address" line by clicking the "+" icon to locate the plb_helloworld_arch_0 object instance.

| Instance | Base Name | Base Address | High Address | Size | Bus Interface(s) | Bus Name | Lock |
|-------------------------------|-------------|--------------|--------------|------|--------------------------|----------|------|
| in microblaze_0's Address Map | | | | | | | 1993 |
| dlmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | ✓ SLMB | dlmb | |
| ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | SLMB | ilmb | |
| Ethernet_MAC | C_BASEADDR | 0x81000000 | 0x8100FFFF | 64K | SPLB | mb_plb | |
| Push_Buttons_5Bit | C_BASEADDR | 0x81400000 | 0x8140FFFF | 64K | SPLB | mb_plb | |
| LEDs_Positions | C_BASEADDR | 0x81420000 | 0x8142FFFF | 64K | SPLB | mb_plb | |
| LEDs_8Bit | C_BASEADDR | 0x81440000 | 0x8144FFFF | 64K | SPLB | mb_plb | |
| - DIP_Switches_8Bit | C_BASEADDR | 0x81460000 | 0x8146FFFF | 64K | SPLB | mb_plb | |
| - IIC_SFP | C_BASEADDR | 0x81600000 | 0x8160FFFF | 64K | SPLB | mb_plb | |
| IIC_FMC_LPC | C_BASEADDR | 0x81620000 | 0x8162FFFF | 64K | SPLB | mb_plb | |
| - IIC_EEPROM | C_BASEADDR | 0x81640000 | 0x8164FFFF | 64K | SPLB | mb_plb | |
| IIC_DVI | C_BASEADDR | 0x81660000 | 0x8166FFFF | 64K | SPLB | mb_plb | |
| — SysACE_CompactFlash | C_BASEADDR | 0x83600000 | 0x8360FFFF | 64K | SPLB | mb_plb | |
| RS232_Uart_1 | C_BASEADDR | 0x84000000 | 0x8400FFFF | 64K | SPLB | mb_plb | |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | SPLB | mb_plb | |
| FLASH | C_MEM0_BASE | 0x8C000000 | 0x8DFFFFFF | 32M | SPLB | mb_plb | |
| DDR3 SDRAM | C MPMC BASE | 0x90000000 | 0x9FFFFFFF | 256M | ▼ SPLB0 | mb plb | |
| Unmapped Addresses | | | | | | | |
| plb helloworld arch 0 | C BASEADDR | | | U | SPLB | mb_plb | |

| Instance | Base Name | Base Address | High Address | Size | Bus Interface(s) | Bus Name | Lock |
|---|-------------|--------------|--------------|------|--------------------------|----------|------|
| 🖻 microblaze_0's Address Map | | | | | | | |
| dimb_cntir | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | ✓ SLMB | dimb | |
| - ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | ✓ SLMB | ilmb | |
| Ethernet_MAC | C_BASEADDR | 0x81000000 | 0x8100FFFF | 64K | ▼ SPLB | mb_plb | |
| Push_Buttons_5Bit | C_BASEADDR | 0x81400000 | 0x8140FFFF | 64K | ✓ SPLB | mb_plb | |
| LEDs_Positions | C_BASEADDR | 0x81420000 | 0x8142FFFF | 64K | ▼ SPLB | mb_plb | |
| - LEDs_8Bit | C_BASEADDR | 0x81440000 | 0x8144FFFF | 64K | ▼ SPLB | mb_plb | |
| DIP_Switches_8Bit | C_BASEADDR | 0x81460000 | 0x8146FFFF | 64K | ✓ SPLB | mb_plb | |
| - IIC_SFP | C_BASEADDR | 0x81600000 | 0x8160FFFF | 64K | ✓ SPLB | mb_plb | |
| IIC_FMC_LPC | C_BASEADDR | 0x81620000 | 0x8162FFFF | 64K | ✓ SPLB | mb_plb | |
| - IIC_EEPROM | C_BASEADDR | 0x81640000 | 0x8164FFFF | 64K | SPLB | mb_plb | |
| IIC_DVI | C_BASEADDR | 0x81660000 | 0x8166FFFF | 64K | SPLB | mb_plb | |
| SysACE_CompactFlash | C_BASEADDR | 0x83600000 | 0x8360FFFF | 64K | SPLB | mb_plb | |
| - RS232_Uart_1 | C_BASEADDR | 0x84000000 | 0x8400FFFF | 64K | SPLB | mb_plb | |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | SPLB | mb_plb | |
| FLASH | C_MEM0_BASE | 0x8C000000 | 0x8DFFFFFF | 32M | SPLB | mb_plb | |
| DDR3_SDRAM | C_MPMC_BASE | 0x90000000 | 0x9FFFFFFF | 256M | ▼ SPLB0 | mb_plb | |
| Unmapped Addresses | | | | | | | |
| plb_helloworld_arch_0 | C_BASEADDR | | | U | SPLB | mb_plb | |
| | | | | U | | | |
| | | | | 16 | 1 | | |
| | | | | 32 | | | |
| | | | | 64 | | | |
| | | | | 128 | | | |
| | | | | 256 | | | |
| | | | | 512 | | | |
| | | | | 1K | | | |
| | | | | 2K | | | |
| | | | | 4K | | | |
| | | | | 8K | - | | |

Now open the dropdown for the plb_helloworld_arch_o line to open the memory assignment popup.

Address space is not expensive, so even though only a few addresses are needed for the two streams used in this tutorial project, we're selecting 64. We can now see that the plb_helloworld_arch_0 instance is properly mapped into the MicroBlaze address space.

| Instance | Base Name | Base Address | High Address | Size | | Bus Interface(s) | Bus Name | Lock |
|------------------------------|-------------|--------------|--------------|------|---|------------------|----------|------|
| 🖻 microblaze_0's Address Map | | | | | | | | |
| dlmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | - | SLMB | dlmb | |
| ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | - | SLMB | ilmb | |
| plb_helloworld_arch_0 | C_BASEADDR | 0x00000000 | 0x0000003F | 64 | - | SPLB | mb_plb | |
| Ethernet_MAC | C_BASEADDR | 0x81000000 | 0x8100FFFF | 64K | - | SPLB | mb_plb | Ō |
| Push_Buttons_5Bit | C_BASEADDR | 0x81400000 | 0x8140FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_Positions | C_BASEADDR | 0x81420000 | 0x8142FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_8Bit | C_BASEADDR | 0x81440000 | 0x8144FFFF | 64K | - | SPLB | mb_plb | |
| DIP Switches 8Bit | C BASEADDR | 0x81460000 | 0x8146FFFF | 64K | - | SPLB | mb plb | Ā |
| IIC SFP | C BASEADDR | 0x81600000 | 0x8160FFFF | 64K | - | SPLB | mb plb | Ē |
| IIC FMC LPC | C BASEADDR | 0x81620000 | 0x8162FFFF | 64K | - | SPLB | mb plb | |
| - IIC EEPROM | C BASEADDR | 0x81640000 | 0x8164FFFF | 64K | - | SPLB | mb plb | Ē |
| - IIC DVI | C BASEADDR | 0x81660000 | 0x8166FFFF | 64K | - | SPLB | mb plb | Ē |
| SysACE CompactFlash | C BASEADDR | 0x83600000 | 0x8360FFFF | 64K | - | SPLB | mb plb | |
| RS232 Uart 1 | C BASEADDR | 0x84000000 | 0x8400FFFF | 64K | - | SPLB | mb plb | Ē |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | - | SPLB | mb_plb | |
| FLASH | C MEMO BASE | 0x8C000000 | 0x8DFFFFFF | 32M | - | SPLB | mb plb | |
| DDR3 SDRAM | C MPMC BASE | | 0x9FFFFFFF | 256M | - | SPLB0 | mb plb | |

However, if you look carefully you will find that several addresses overlap. To correct this problem click the button (circled above) to remap the memory space. If your software will use fixed memory access based on this mapping it would be prudent to select the "Lock" box for those memories you want fixed. Subsequent remap activities will not change them.

NOTE: Please remember that the MicroBlaze is a Harvard Architecture processor. It has separate data and instruction address spaces. It is normal and acceptable for the dlmb and ilmb controllers to have overlapping addresses spaces, they are two different physical memories.

Generate Netlist and BitStream

It is now time to generate a netlist, BitStream, and export the design to the Xilinx SDK. The easiest way to do this is to open the Navigator toolbar (View->Toolbars->Navigator) and then sequentially click Generate Netlist, Generate BitStream, and Export Design.



Generate Netlist and Generate BitStream are complex operations that take significant wall clock time. Depending on the size of the design (the example is very small) and your systems memory and processor, even trivial designs will take at least 30 minutes and large complex designs may take in excess of eight hours. Should you find errors, the most likely cause will be overlapping address spaces that have not been corrected.

When you click Export Design, the design will be pushed out to the correct locations and the SDK tool will be opened to allow building the software side of our project.

See Also

Importing the Application Software to the Xilinx Tools

1.1.8 Importing the Application Software to the Xilinx Tools

Hello World Tutorial for MicroBlaze, Step 8

After clicking the Export Design button the popup (shown) is displayed giving you the option to perform an Export only or to Export files and start the Software Development Kit (SDK), for this tutoria please click "Export & Launch SDK".



After the SDK tool starts, the following popup will be displayed. The export tool will have correctly filled in the project workspace path data. We do not recommend setting this as the "default" since it is not likely to be the default for other projects.

| | × |
|---|--------|
| | |
| • | Browse |
| | Cancel |
| | ▼ ■ |

Once the SDK window is open you can close the EDK window as it will not be needed further for this tutorial. Your screen should be similar to the one shown:



Configure SDK

It is first necessary to configure the SDK environment to reference the EDK project directory as a "repository" in order to reference the FPGA logic driver by choosing the "Repositories" menu item from the "Xilinx Tools" menu. Click the "New" button next to the "Local Repositories" then navigate to the project's EDK directory and select it. The Preferences dialog should be similar to that shown here:



Click "Apply" and then "OK" to continue.

Create the HelloWorld Project

Choose File -> New -> Xilinx C Project from the File menu. Then select Empty Application and set the Project name: to HelloWorld_project. The window will appear similar to what is shown here:

| New Project | | _ 🗆 🗙 |
|---|---|--------|
| lew Xilinx C Project Create a managed make application project. | Choose from one of the sample applications. | G |
| Project name: HelloWorld_project | | |
| ✓ Use default location | | |
| Location: C:\sandbox\HelloWorld\EDK\SDK | \HelloWorld_project | Browse |
| Choose file system: default 💌 | | |
| Target Hardware | | |
| Hardware Platform: EDK_hw_platform | | - |
| Processor: microblaze_0 | | ~ |
| Dhrystone Empty Application Hello World WJP Echo Server Memory Tests Peripheral Tests SREC Bootloader Xikernel POSIX Threads Demo Zyng FSBL | Description A blank C project. | × |
| ? | < Back Next > Finish | Cancel |

Click Next to continue.

On the next dialog window set the Project name: to HelloWorld_project_bsp_0; the window will now appear similar to what is shown here:

| New Project | | |
|---|--|---------|
| New Xilinx C Project Create a managed make application project. Choo | se from one of the sample applications. | G |
| Create a new Board Support Package project | | |
| The template provided by application 'Empty | Application' will be used to configure the p | roject. |
| Project name: HelloWorld_project_bsp_0 | | |
| Use default location | | |
| Location: C:\sandbox\HelioWorld\EDK\SDK\H | felloWorld_project_bsp_0 | Browse |
| Choose file system: default 💌 | | |
| Available Board Support Packages: No Board Support Packages found | | |
| ? | < Back Next > Finish | Cancel |

Click **Finish** to continue.

The **Project Explorer** will now include the two new projects, HelloWorld_project and HelloWorld_project_bap_0. Notice that as soon as the files were read into the project that the entire project was built. This is the default behavior of SDK. If you followed this tutorial accurately there should have been no errors requiring edits.

It is now necessary to increase the Stack and Heap memory sizes using the linker script. From the Project Explorer open the lscript.ld file, you will find it in the HelloWorld_project src directory. Change the Stack and Heap sizes to 0x4000 as shown here:

| | lifferent sections of an executable are placed ry regions, and change the assignment of se | | |
|---|---|------------|---------|
| Name | Base Address | Size | Add Mem |
| ilmb_cntlr_dlmb_cntlr | 0x0000050 | 0x00001FB0 | |
| DDR2_SDRAM_MPMC_BASEADDR | 0xC8000000 | 0x08000000 | |
| Stack and Heap Sizes Stack Size 0x40000 Heap Size 0x40000 Section to Memory Region Mapping | · · · · · · · · · · · · · · · · · · · | | |

Either click the file save icon or enter <ctrl>s to save the modified linker script.

Now the software files exported from CoDeveloper must be added to the project. We will do this by linking to their location as follows:

Right click the **src** directory in the HelloWorld_project and select **New -> Folder**. Click the **Advanced** button and select **Link to alternate location (Linked Folder)**. Use the browse button to locate the HelloWorld\EDK\SDK\HelloWorld directory. The New Folder dialog should look similar to that shown here:

| 🐵 New Folder | <u>_ 🗆 ×</u> |
|--|---------------------------|
| Folder Create a new folder resource. | |
| Enter or select the parent folder: | |
| HelloWorld_project/src | |
| | |
| Folder name: HelloWorld << Advanced C >>> Use default location C >>> Folder is not located in the fill | e system (Virtual Folder) |
| C:\sandbox\HelloWorld\EDK\SDK C:\sandbox\HelloWorld\EDK\SDK Choose file system: default | |
| Resource Filters | Finish Cancel |

When you click Finish to continue the SDK will automatically rebuild the entire project and prepare the HelloWorld_project.elf file. In the next step, the BitStream and ELF files will be downloaded into the FPGA.

See Also

Building and Downloading the Application

1.1.9 Building and Downloading the Application

Setting up the Hardware

Configure your hardware for downloading a BitStream by following the detailed instructions provided by the target board vendor.

Downloading the FPGA BitStream

From the menu bar, select **Xilinx Tools -> Program FPGA**. The Program FPGA dialog will pop-up (as shown), click the **Program** button to complete the process.

| Program Fl Specify the | | ream and the ELF files that reside in BRAM memory | |
|----------------------------|--|---|---------|
| Hardware Co Hardware Sp | - | uration cation: C:\sandbox\HelloWorld\EDK\SDK\EDK_hw_platform\syster | n.xml |
| Bitstream: | stream: C:\sandbox\HelloWorld\EDK\SDK\EDK_hw_platform\system.bit | | Browse. |
| BMM File: | C:\s | andbox\HelloWorld\EDK\SDK\EDK_hw_platform\system_bd.bmm | Browse. |
| Software Co | nfigu | uration | |
| Processor | | ELF File to Initialize in Block RAM | |
| microblaze_ | 0 | bootloop | |
| microblaze_ | .0 | | |
| ~ | | Program | Cancel |

Downloading and Running the Software Binary

To download and run the software binary, in the **Project Explorer** view, right-click the **HelloWorld_project** project and select **Run As -> Launch on Hardware**. This will cause **SDK** to download the software binary and start the application

This tutorial is completed.

1.2 Tutorial 2: Complex FIR Filter for Xilnx Design Suite 13.4 (or higher)



Overview

This detailed tutorial will demonstrate how to use **Impulse C** to create, compile and optimize a digital signal processing (**DSP**) example for the **MicroBlaze** platform. We will also show how to make use of the **Fast Simplex Link (FSL)** bus provided in the **MicroBlaze** platform.

The goal of this application will be to compile the algorithm (a **Complex FIR Filter** function) on hardware on the FPGA. The **MicroBlaze** will be used to run test code (producer and consumer processes) that will pass text data into the algorithm and accept the results.

This example makes use of the Xilinx ML605 Evaluation Board. The board features is a Virtex-6 FPGA with a MicroBlaze soft processor. This tutorial also assumes you are using the Xilinx Design Suite 13.4 (or later) development tools.

This tutorial will require approximately two hours to complete, including software run times.

Note: this tutorial is based on a sample DSP application developed by Bruce Karsten of Xilinx, Inc. A more complete description of the algorithm can be found in the **Impulse C User Guide**. This tutorial assumes that you have are familiar with the basic steps involved in using the **Xilinx EDK** tools. For brevity this tutorial will omit some EDK details that are covered in introductory EDK and Impulse C tutorials.

Note also that most of the detailed steps in this tutorial only need to be performed once, during the initial creation of your **MicroBlaze** application. Subsequent changes to the application do not require repeating these steps.

Steps

Loading the Complex FIR Application Understanding the Complex FIR Application Compiling the Application for Simulation Building the Application for the Target Platform Creating the Platform Using the Xilinx Tools Configuring the New Platform Exporting Files from CoDeveloper Importing the Generated Hardware Generating the FPGA Bitmap Importing the Application Software Running the Application

See Also

Tutorial 1: Hello World on the MicroBlaze platform

1.2.1 Loading the Complex FIR Application

Complex FIR Filter Tutorial for MicroBlaze, Step 1

To begin, start the **CoDeveloper Application Manager** by selecting from the **Windows Start** -> **Programs** -> **Impulse Accelerated Technologies** -> **CoDeveloper Application Manager** program group.

Note: this tutorial assumes that you have read and understand the **Complex FIR Filter** example and tutorial presented in the main **CoDeveloper** help file.

Open the Xilinx MicroBlaze ComplexFIR sample project by selecting Open Project from the File menu, or by clicking the Open Project toolbar button. Navigate to the .\Examples\Embedded\ComplexFIR_MicroBlaze\ directory within your CoDeveloper installation. (You may wish to copy this example to an alternate directory before beginning.) The project file is also available online at http://impulsec.com/ReadyToRun/. The screen shots in this tutorial show the working directory being c:\sandbox\CompFIRmb. Opening the project will result in the display of a window similar to the following:



Files included in the Complex FIR Filter project include:

Source files **ComplexFilter.c**, **Filter_hw.c** and **Filter_sw.c** - These source files represent the complete application, including the **main()** function, consumer and producer software processes and a single hardware process.

Header files ComplexFilter.h and Filter.h - function prototypes and definitions.

See Also

Understanding the Complex FIR Application

1.2.2 Understanding the Complex FIR Application

Complex FIR Filter Tutorial for MicroBlaze, Step 2

Before compiling the **Complex FIR Filter** application to hardware, let's first take a moment to understand its basic operation and the contents of the its primary source files, and in particular **Filter_hw.c**.

The specific process that we will be compiling to hardware is represented by the following function (located in **Filter_hw.c**):

void complex_fir(co_stream filter_in, co_stream filter_out)

This function reads two types of data:

- Filter coefficients used in the Complex FIR Filter convolution algorithm.
- An incoming data stream

The results of the convolution are written by the process to the stream filter_out.

The **complex_fir** function begins by reading the coefficients from the **filter_in** stream and storing the resulting data into a local array (**coef_mem**). The function then reads and begins processing the data, one at a time. Results are written to the output stream **filter_out**.

The repetitive operations described in the complex_fir function are complex convolution algorithm.

The complete test application includes test routines (including **main**) that run on the **MicroBlaze** processor, generating test data and verifying the results against the legacy C algorithm from which **complex_fir** was adapted.

The configuration that ties these modules together appears toward the end of the **Filter_hw.c** file, and reads as follows:

```
void config_filt (void *arg) {
    int i;
    co_stream to_filt, from_filt;
    co_process cpu_proc, filter_proc;
    to_filt = co_stream_create ("to_filt", INT_TYPE(32), 4);
    from_filt = co_stream_create ("from_filt", INT_TYPE(32), 4);
    cpu_proc = co_process_create ("cpu_proc", (co_function))
    call_accelerator, 2, to_filt, from_filt);
```
```
filter_proc = co_process_create ("filter_proc", (co_function)
complex_fir, 2, to_filt, from_filt);
    co_process_config (filter_proc, co_loc, "PEO");
}
```

As in the **Hello World** example (described in the main **CoDeveloper** help file), this configuration function describes the connectivity between instances of each previously defined process.

Only one process in this example (filter_proc) will be mapped onto hardware and compiled by the Impulse C compiler. This process (filter_proc) is flagged as a hardware process through the use of the co_process_config function, which appears here at the last statement in the configuration function. Co_process_config instructs the compiler to generate hardware for complex_fir (or more accurately, the instance of complex_fir that has been declared here as filter_proc).

NOTE: In many cases the name of the hardware process and the name of the instance of the hardware process are the same. This tutorial points out that the instance name and the definition name need not be the same. This ability makes it possible to replicate a hardware module multiple times in a single application.

The **ComplexFilter.c** generates a set of **Complex FIR Filter** coefficients and also a group of input data being processed.

The **Filter_sw.c** will run in the **MicroBlaze** embedded processor, controlling the stream flow and printing results.

See Also

Compiling the Application for Simulation

1.2.3 Compiling the Application for Simulation

Complex FIR Filter Tutorial for MicroBlaze, Step 3

Simulation allows you to verify the correct operation and functional behavior of your algorithm before attempting to generate hardware for the FPGA. When using Impulse C, simulation simply refers to the process of compiling your C code to the desktop (host) development system using a standard C compiler, in this case the **GCC** compiler included with the Impulse **CoDeveloper** tools.

To compile and simulate the application for the purpose of functional verification:

 Select Project -> Build Software Simulation Executable (or click the Build Software Simulation Executable button) to build the FIR_Accelerator.exe executable. A command window will open, displaying the compile and link messages as shown below:

| Impulse CoDeveloper Application Manager Universal Edition - [FIR_Accelerate Build Simulation Executable Build Simulation | × |
|--|----|
| File Fair Alew Buolect Toole Helb | |
| थि ☞ ┛ ≌ ≌ ₽ ४ в в ヽ < // ४/ м % ๗ , : ::::::::::::::::::::::::::::::: | |
| oject Explorer 🛛 🗴 🗙 | |
| | |
| - 2 Application FIR_Accelerator | |
| E G Source Files | |
| -hy (G) Filter Juw c | |
| sy C Filter_sw.c | |
| General Header Files Sw In ComplexFilter.h | |
| | |
| Project Files | |
| E - The adment files | |
| | |
| | |
| | |
| | |
| | |
| | |
| Build Build Messages at | ١x |
| IC./Impulse/CoDeveloper3/MinGW/include" o ComplexFiture - complex new - | |
| "C://mpulse/CoDeveloper3/MinGW/bin/gcc" g "-IC:\/mpulse\CoDeveloper3\\nclude" "-IC://mpulse/CoDeveloper3/StageMaster/include" -DWIN32 "- IC://mpulse/CoDeveloper3/MinGW/include" -o Filter_hw.o -c Filter_hw.c | |
| "C://mpulse/CoDeveloper3/MinGW/bin/gcc" g "-IC:\/mpulse\CoDeveloper3\\nclude" "-IC://mpulse/CoDeveloper3/StageMaster/include" -DWIN32 "- IIC://mpulse/CoDeveloper3/MinGW/include" -o Filter sw.o -c Filter sw.c | |
| "C://mpulse/CoDeveloper3/MinGW/bin/gcc" g ComplexFilter.o Filter_hw.o Filter_sw.o -LC://mpulse/CoDeveloper3/StageMaster/lib "C:\lmpulse \CoDeveloper3\Libraries/ImpulseC.lib" -o FIR Accelerator.exe | |
| | |
| ====================================== | Ţ |
| 🖾 Build 🖼 Find in Files 🖸 System | - |
| | 1 |

 You now have a Windows executable representing the Complex FIR Filter application implemented as a desktop (console) software application. Run this executable by selecting Project -> Launch Simulation Executable. A command window will open and the simulation executable will run as shown below:

| C:\Windows\system32\cmd.exe | - 🗆 X |
|---|----------|
| | |
| "C:\sandbox\CompFIRmb\FIR_Accelerator.exe" | |
| Begin Filtering a Slot | |
| Complete Filtering a Slot | |
| | |
| Begin Filtering a Slot Complete Filtering a Slot COMPLETE APPLICATION | |
| Complete Filtering a Slot | |
| COMPLETE APPLICATION | |
| Press Enter to continue | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | _ |
| | |

Verify that the simulation produces the output shown. Note that although the messages indicate that the **ComplexFIR** algorithm is running on the FPGA, the application (represented by hardware and software processes) is actually running entirely in software as a compiled, native Windows executable. The

messages you will see have been generated as a result of instrumenting the application with simple printf statements such as the following:

```
#if defined(MICROBLAZE)
    xil_printf ("COMPLETE APPLICATION\r\n");
    return 0;
#else
    printf ("COMPLETE APPLICATION\r\n");
    printf ("Press Enter to continue...\r\n");
    c = getc(stdin);
#endif
```

Notice in the above C source code that **#ifdef** statements have been used to allow the software side of the application to be compiled either for the embedded **MicroBlaze** processor, or to the host development system for simulation purposes.

See Also

Building the Application for the Target Platform

1.2.4 Building the Application for the Target Platform

Complex FIR Filter Tutorial for MicroBlaze, Step 4

The next step in the tutorial is to create FPGA hardware and related files from the C code found in the **Filter_hw.c** source file. This requires that we select a platform target, specify any needed options, and initiate the hardware compilation process.

Specifying the Platform Support Package

To specify a platform target, open the Generate tab of the Options dialog as shown below:

| Platform Support Package: | |
|--|--|
| Xilinx MicroBlaze FSL (VHDL) | |
| Hardware Optimization and Generation | |
| Enable constant propagation | Generate dual clocks |
| Scalarize array variables | Generate active-low reset |
| Relocate loop invariant expressions | Use std_logic types for VHDL interfaces |
| | Do not include co_ports in bus interface |
| | A 1 July 119 |
| Additional optimizer options: | Additional library options: |
| Additional optimizer options: | |
| | |
| Floating Point Options | Output Directories |
| Roating Point Options | Output Directories Hardware build directory: |
| Floating Point Options | Output Directories Hardware build directory: /hw |
| Roating Point Options | Output Directories Hardware build directory: /hw Software build directory: |
| Floating Point Options | Output Directories Hardware build directory: hw Software build directory: sw |
| | Output Directories Hardware build directory: hw Software build directory: sw Hardware export directory: |
| Roating Point Options Include floating point library Include co_meth library Enable floating point optimization Allow double-precision types and operators | Output Directories Hardware build directory: hw Software build directory: sw |

Specify Xilinx MicroBlaze FSL (VHDL). Also specify hw and sw for the hardware and software directories as shown, and specify EDK for the hardware and software export directories. Also ensure that the Generate dual clocks option is checked.

Click **OK** to save the options and exit the dialog.

Generate HDL for the Hardware Process

To generate hardware in the form of HDL files, and to generate the associated software interfaces and library files, select **Generate HDL** from the **Project** menu, or select the **Generate HDL** toolbar button as shown below:



A series of processing steps will run in a command window as shown below:

| Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIII/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIII/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIII/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/Xilinx/OPB/bus.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX_mb_fsl.xml" -hwdirhw files "ComplexFilter.c Filter_sw.c" Fill_Acceleratorxic sw/co_init.c Impulse CoDeveloper3/bin/Impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/Xilinx_mb_fsl.xml" -hwdirhw files "ComplexFilter.c Filter_sw.c" Fill_Acceleratorxic sw/co_init.c Impulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. | Build | # > |
|---|--|--|
| Max. Und Deley: 0 Book #11: Stages: 1 Max. Und Deley: 0 Operators: 10 Addrefs/Subtractor(e) (32 bt) 4 Multiple(e) (15 bt) 1 Comparison(e) (2 bt) 5 Comparison(e) (12 bt) 4 Multiple(e) (12 bt) 5 Comparison(e) (22 bt) 4 Multiple(e) (20 Eveloper3/bhr/impulse_genvhill exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mydale C RTL Comport Generator Computer Colleveloper3/bhr/impulse_genvhill exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mydale C RTL Comport Generator Computer Colleveloper3/bhr/impulse_genvhill exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mydale C RTL Colleveloper3/bhr/impulse_genvhill exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mydale C RTL Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-1/mpulse Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-2/mpulse Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-2/mpulse Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-2/mpulse Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-2/mpulse Colleveloper3/Achitectures/Alinx_mb_fill/mill= C-2/mpulse Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_mill= Mill Colleveloper3/Achitectures/Millx_Mills_Mills_Mills_ Mill Colleveloper3/Achitectures/Millx_Mills_Mills_ Mill Colleveloper3/Achitectures/Millx_Mills_Mills_ Mill Colleveloper3/Achitectures/Millx_Mills_Mills_ Mills | | |
| Block #11: Sage: 1 Max. Unit Delay: 0 Operators: 10 Adder[s]: Submetcrofs) (32 bt) 4 Multipler(s) (16 bt) 1 Comparator(s) (2 bt) 5 Comparator(s) (2 bt) Teld Stage: 13 Comparator(s) (2 bt) Teld Stage: 14 Estimated DSFs: 4 Writing outputdone (C./Impulse/Coeveloper3/Ani/Impulse_genvhille.we" FIR_Accelerator.xhw hw/FIR_Accelerator_comp.vhd Impulse C.RTL Component Generator Corporting 1020: 2009, Impulse Accelerated Technologies, Inc. All rights reserved. Software activated- C./Impulse/Coleveloper3/Ani/Impulse_jach" "aC./Impulse/CoDeveloper3/Architectures/Alling.mb. fsl.xml" dc: no_port_bus_connectswdinsw files FTR_Accelerator compited Software activated- C./Impulse/CoDeveloper3/Architectures/Alling.mb. fsl.xml"dc: no_port_bus_connectswdinsw files FTR_Accelerator compited Impulse. Comparetario Compar | Stages: 1 | |
| Block #11: Sage: 1 Max. Unit Delay: 0 Operators: 10 Adder[s]: Submetcrofs) (32 bt) 4 Multipler(s) (16 bt) 1 Comparator(s) (2 bt) 5 Comparator(s) (2 bt) Teld Stage: 13 Comparator(s) (2 bt) Teld Stage: 14 Estimated DSFs: 4 Writing outputdone (C./Impulse/Coeveloper3/Ani/Impulse_genvhille.we" FIR_Accelerator.xhw hw/FIR_Accelerator_comp.vhd Impulse C.RTL Component Generator Corporting 1020: 2009, Impulse Accelerated Technologies, Inc. All rights reserved. Software activated- C./Impulse/Coleveloper3/Ani/Impulse_jach" "aC./Impulse/CoDeveloper3/Architectures/Alling.mb. fsl.xml" dc: no_port_bus_connectswdinsw files FTR_Accelerator compited Software activated- C./Impulse/CoDeveloper3/Architectures/Alling.mb. fsl.xml"dc: no_port_bus_connectswdinsw files FTR_Accelerator compited Impulse. Comparetario Compar | Max Unit Delay: 0 | |
| Sigger 1 Max. Uhr Delay: 0 Operators: 10 Addrej/Subtractor(s) (32 bi) A Multiple(s) (16 bit) Comparator(s) (2 bi) 5 Comparator(s) (32 bi) Max. Uhr Delay: 64 Estimated DSFs: 4 Witting output done C./Impute CoDeveloper3/bin /mputes_genvholl.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mputes 0 SFs: 4 Witting output done C./Impute CoDeveloper3/bin /mputes_genvholl.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mputes 0 SFn: 4 Singerstrain complete Software activated | | |
| Max Until Delay: 0 Operators: 10 Addref(s):Subtractor(s) (32 bt) 4 Multipler(s) (15 bt) 1 Comparator(s) (2 bt) 5 Comparator(s) (23 bt) 2 bt) Total Stages: 19 Max Unit Delay: 0 Writing actortdone | | |
| Operators: 10 Addrejo/Subtractorig) (32 bit) 5 Comparatorig) (32 bit) Total Stages: 13 Waturbeeling (16 bit) 1 Comparatorig) (2 bit) 5 Comparatorig) (32 bit) Total Stages: 13 Witing output done C./Impulse CODeveloper 3/in /impulse_genvihil exe" FIR_Accelerator.tww hw/FIR_Accelerator_comp.vhd mylase Contexponent Generator Miting output done C./Impulse CoDeveloper 3/in /impulse_anch "soC./Impulse/CoDeveloper3/Architectures./alinx_mb_falxml" dc-no_port_bus_connect_ewdraw files Software activated | | |
| 10 Addrey() Subtractor() (32 bit) 4 Multiples() (16 bit) 1 Comparator(s) (2 bit) 5 Comparator(s) (32 bit) Total Stages: 13 Writing output done (/C/mpulse/CoDeveloper3/in/impulse_genvhall.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mpulse C FTL Component Generator Copyright 2022: 2020, Impulse Accelerated Technologies, Inc. All rights reserved. Schware activity and the accelerated Technologies, Inc. All rights reserved. Schware activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the TR_Accelerator is w/FIR_Accelerator. Input Schware Accelerated Technologies, Inc. All rights reserved. Component generation completed Technologies, Inc. All rights reserved. Loading C //mpulse/CoDeveloper3/Architectures/Allinx/ FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architectures/HDL/Xilrix/FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architecture/HDL/Xilrix/FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architectures/HDL/Xilrix/CPB/bus xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kone Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/K | Max. Unit Delay: U | |
| 10 Addrey() Subtractor() (32 bit) 4 Multiples() (16 bit) 1 Comparator(s) (2 bit) 5 Comparator(s) (32 bit) Total Stages: 13 Writing output done (/C/mpulse/CoDeveloper3/in/impulse_genvhall.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mpulse C FTL Component Generator Copyright 2022: 2020, Impulse Accelerated Technologies, Inc. All rights reserved. Schware activity and the accelerated Technologies, Inc. All rights reserved. Schware activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the accelerated Technologies, Inc. All rights reserved. Component generation complete S-Software activity and the TR_Accelerator is w/FIR_Accelerator. Input Schware Accelerated Technologies, Inc. All rights reserved. Component generation completed Technologies, Inc. All rights reserved. Loading C //mpulse/CoDeveloper3/Architectures/Allinx/ FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architectures/HDL/Xilrix/FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architecture/HDL/Xilrix/FIS_Ibus xml Loading C //mpulse/CoDeveloper3/Architectures/HDL/Xilrix/CPB/bus xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kone Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/Kilrix/Hon Kalex/FS/Lytandolone xml Loading C //mpulse/CoDeveloper3/Architectures/K | | |
| 4 Mutplerigh (16 bit) I Comparator(s) (2 bit) 5 Comparator(s) (32 bit) Total Stages: 19 Max. Unt Delay: 64 Estimated DSP: 4 Witing outputdone C./Impulee /CDDEveloper3/bin/impulee_genvhill.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd mydiee CRIT.Component Generator Comparet James and Stages and Stag | | |
| 5 Comparatoria) (32 bit) Total Stages: 19 Miting output done C.//mpulse/CoDeveloper3/in/impulse_genvhill.exe" FIR_Accelerator.xhw hw/FIR_Accelerator_comp.vhd mpulse C RTL Component Generator Copyright 2002: 2009, Impulse Accelerated Technologies, Inc. All rights reserved. Generating B/Oxon B/Oxe Accelerated Technologies, Inc. Component generation complete | | |
| Total Stages: 19 Max. Unit Delay: 64 Estimated DSPs: 4 Writing output:done (C./Impulse/CoDeveloper3/bin/mpulse_genvholl exe" FIR_Accelerator.xhw hw/FIR_Accelerator_comp.vhd impulse CRL: Component Generator Computer CRL: Component Generator Computer CRL: Developer3/bin/mpulse_arch" "aC:/Impulse/CoDeveloper3/Architectures/klinx_mb_fsl.xml" -dc-no_port_bus_connect -swdirsw files Software activated- (C./Impulse/CoDeveloper3/Architectures/klinx_mb_fsl.xml" -dc-no_port_bus_connect -swdirsw files (FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd "FIR_Accelerator_top.vhd mpulse CRL: Developer3/Architectures/klinx_mb_fsl.xml" -dc-no_port_bus_connect -swdirsw files (FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd "FIR_Accelerator_top.vhd mpulse CRL: Developer3/Architectures/klinx_mb_fsl.xml Loading C./Impulse/CoDeveloper3/Architectures/klinx_mb_fsl.xml Loading C./Impulse/CoDeveloper3/Architectures/klinx_mb_fsl.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/VIDL/KlinxYCSL/bus.xml Loading C./Impulse/CoDeveloper3/Architectures/KlinX/MoroBlaze/FSL/standalone.xml Loading C./Impulse/CoDeveloper3/Architectures/KlinX/MoroBlaze/FSL/standalone.xml Loading C./Impulse/CoDeveloper3/Architectures/KlinX/MoroBlaze/FSL/standalone.xml Loading FIR_Accelerator is: ComplexCline relate_coreGenFilesForEDK():Bating Design generation complet chmod -R + w hw mtdir sw C./Impulse/CoDeveloper3/Architectures/Klinx/MoroBlaze/FSL/standalone.xml Loading FIR_Accelerator is: ComplexCline relate_sw code ps sw; done chmod -R + w sw ==================================== | | |
| Max. Umi Delay: 64 Estimated DSPs: 4 Writing output: Writing output: Opport: Computes CRD Component Generator Component generation Component generation Sensenting pel/Complex, fr. Component generation complete -Software activated (C./Inputse/CoDeveloper3/Achitectures/AlmX_mb fsl xml* do: no_port_bus_connect -swdinsw files "FIR_Accelerator_comp with FIR_Accelerator_top.whd "FIR_Accelerator_top.whd Tangtas reserved. Component generation Composed (Part Developer3/Achitectures/AlmX_mb fsl xml*. Loading C/Imputse/CoDeveloper3/Achitectures/AlmX_mb fsl xml Loading C/Imputse/CoDeveloper3/Achitectures/AlmX_MicroBlaze/FSL/bus xml Loading C/Imputse/CoDeveloper3/Achitectures/AlmX/MicroBlaze/FSL/standalone xml <td>5 Comparator(s) (32 bit)</td> <td></td> | 5 Comparator(s) (32 bit) | |
| Max. Umi Delay: 64 Estimated DSPs: 4 Writing output: Writing output: Opport: Computes CRD Component Generator Component generation Component generation Sensenting pel/Complex, fr. Component generation complete -Software activated (C./Inputse/CoDeveloper3/Achitectures/AlmX_mb fsl xml* do: no_port_bus_connect -swdinsw files "FIR_Accelerator_comp with FIR_Accelerator_top.whd "FIR_Accelerator_top.whd Tangtas reserved. Component generation Composed (Part Developer3/Achitectures/AlmX_mb fsl xml*. Loading C/Imputse/CoDeveloper3/Achitectures/AlmX_mb fsl xml Loading C/Imputse/CoDeveloper3/Achitectures/AlmX_MicroBlaze/FSL/bus xml Loading C/Imputse/CoDeveloper3/Achitectures/AlmX/MicroBlaze/FSL/standalone xml <td>Total Stagee: 19</td> <td></td> | Total Stagee: 19 | |
| Estimated DSPs: 4 Writing output done 'C./mpulse/CoDeveloper3/bin/mpulse_genvholl.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd Copyright 2002-2009. Impulse Accelerated Technologies, Inc. All rights reserved. Senerating pell/complex_fr Component generation complete "Software activated" 'C./mpulse/CoDeveloper3/bin/mpulse_acch" "SC:/mpulse/CoDeveloper3/Architectures/kilmx_mb_fsl.xml" dc-no_port_bus_connect -swdraw files 'FIR_Accelerator_top_whd 'FIR_Accelerator_top_whd 'FIR_Accelerator_top_whd 'Inghtse Carbot Developer3/Architectures/Minx_MoreSize/FSU/Subus.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinX/MoreSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MoreSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FSU/Standalone.xml Ladding C:/mpulse/CoDeveloper3/Architectures/MinZ/MinZeSize/FS | | |
| Witing output: done "C:/Impulse/CD Developer3/bin/impulse_genvhdl.exe" FIR_Accelerator.shw hw/FIR_Accelerator_comp.vhd populse CR TL Component Generator Copyright 2002-2009, Impulse Accelerated Technologies, Inc. NI rights reserved. Semerating peloy Complex_fir Component generation complete Software activity and | | |
| "C/Timpulse", CDD-veloper3/bin/impulse_genvihil exe" FIR_Accelerator shw hw/FIR_Accelerator_comp.vhd Dopyingt 2002-2009, Impulse Accelerated Technologies, Inc. All rights reserved. Generating pe0/complex_fir Component generation complete Software activitated | Estimated DSPS: 4 | |
| "C/Timpulse", CDD-veloper3/bin/impulse_genvihil exe" FIR_Accelerator shw hw/FIR_Accelerator_comp.vhd Dopyingt 2002-2009, Impulse Accelerated Technologies, Inc. All rights reserved. Generating pe0/complex_fir Component generation complete Software activitated | Writing output dono | |
| Impulse C RTL Component Generator Senerating pe0/complex_fr Component generation complete -Software activated -Cr/mpulse/CoDeveloper3/bin/impulse_arch" "-aC/Impulse/CoDeveloper3/Architectures/xlinx_mb_fsl.xml"-dc-no_port_bus_connect -swdinsw files -Software activated -Cr/mpulse/CoDeveloper3/Architectures/xlinx_mb_fsl.xml"-dc-no_port_bus_connect -swdinsw files -Software activated Copyright 2002-2012. Impulse Accelerator Jop vhd " FIR_Accelerator xic hw/FIR_Accelerator_Jop vhd Inghts reserved. Loading C/Impulse/CoDeveloper3/Architectures/xlinx_mb_fsl.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Hor/SL/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/SL/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/SL/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/SL/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/SL/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/OPB/bus.xml Loading C/Impulse/CoDeveloper3/Architectures/Xlinx/Micr/Blaze/FSL/standalone.xml E/CoPereident complete -mod H = wn km -Midrisw C/CoDeveloper3/Architectures/Xlinx/Micr/Blaze/FSL/standalone.xmlLoading FIR_Accelerator.cc. - Complex/Filer A Filer.sm v// File - Midrisw C/CoDeveloper3/Architectures/Xlinx/Micr/Blaze/FSL/standalone.xmlLoading FIR_Accelerator.cc. - Crimpulse/CoDeveloper3/Architectures/Xlinx/Micr/Blaze/FSL/standalone.xmlLoading FIR_Accelerator.cc. - Crimpulse/CoDeveloper3/Archite | | when her/EIB. Accelerator, comp vhd |
| Copyright 2002-2005, Impulse Accelerated Technologies, Inc. All right reserved Generating pe0/complex_fir Component generation complete "FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd" "aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsi.xml"-dc-no_port_bus_connect -swdirsw files ("FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd" "IR_Accelerator.ic hw/FIR_Accelerator_top.vhd Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsi.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/Est.Vous.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/FIS_Vous.xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/standalone zml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/XIIIX/MicroBlazer/FSL/standalone xmlLoading FIR_Accelerator xc for in ComplexFilter c: Filter_s wc: do cp | | ann nn n n 2 weeke dei 200mp and |
| All rights reserved. Generating people/complex, fir Component generation complete -Software activated (C/mpulse/CoDeveloper3/bin/impulse_arch" "aC:/impulse/CoDeveloper3/Architectures/kilmx_mb_fsl.xml"-dc-no_port_bus_connect -swdirsw files 'FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd" FIR_Accelerator.xic hw/FIR_Accelerator_top.vhd Impulse C+DDU Design Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/FSL/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/FSL/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/DeBaze/FSL/standalone.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/VHDL/Xilmx/OPB/bus.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx/Impulse/LoBeveloper3/Architectures/Xilmx/Impulse/LoBeveloper3/Architectures/Xilmx/Impulse/CoDeveloper3/Architectures/Xilmx/Impulse/CoDeveloper3/Architectures/Xilmx.mb_fsl.xml" +hwdirfw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator.c. BioGreGorCommon.tcl::GenerateCoreGenFilesForEDK():Exiting Design generation complets -mmulse C:/impulse/CoDeveloper3/Architectures/Xilmx.mb_fsl.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx/MicroBlaze/FSL/cpu.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx.mb_fsl.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx.mb_fsl.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx/MicroBlaze/FSL/cpu.xml Leading C:/impulse/CoDeveloper3/Architectures/Xilmx/MicroBlaze/FSL/cpu.xml L | | |
| Generating pel/complex_fir Component generation complete -Software activated | | |
| Component generation complete | | |
| Software āctivated | | |
| "C./mpulse/Co.Developer3/bin/mpulse_arch" "aC:/mpulse/Co.Developer3/Architectures/alinx_mb_fsl.xml" dc=no_port_bus_connect =swdirsw files "FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd "FIR_Accelerator_top.vhd mpulse C HDL Design Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/mpulse/Co.Developer3/Architectures/VHDL/Xirx/FSL/bus xml Loading C:/mpulse/Co.Developer3/Architectures/VHDL/Xirx/FSL/bus xml Loading C:/mpulse/Co.Developer3/Architectures/VHDL/Xirx/FSL/bus xml Loading C:/mpulse/Co.Developer3/Architectures/VHDL/Xirx/FSL/bus xml Loading C:/mpulse/Co.Developer3/Architectures/VHDL/Xirx/FSL/bus xml Loading C:/mpulse/Co.Developer3/Architectures/Wirx/McroBlaze/FSL/standalone xml Loading C:/mpulse/Co.Developer3/Architectures/Wirx/McroBlaze/FSL/standalone xml Loading C:/mpulse/Co.Developer3/Architectures/Wirx/McroBlaze/FSL/standalone xml Loading C:/mpulse/Co.Developer3/Architectures/Wirx/McroBlaze/FSL/standalone xml Loading C:/mpulse/Co.Developer3/Architectures/Wirx/McroBlaze/FSL/standalone xml Loading FIR_Accelerator xic ibCoreGenCommon tot:/GenerateCoreGenFilesForEDK():Exting Design generation complete shmd -R + vn km krkir sw "C:/mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu xml Loading C:/mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu xml Loading C:/mpu | | |
| "FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd "FIR_Accelerator.xic hw/FIR_Accelerator_top.vhd mpulse ChDuesing Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX_raget xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/FoS1bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/FoS1bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/CoPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/WHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/WHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/WHML/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/WHML/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/WHML/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/MicroBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/MicroBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/MicroBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/MicroBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/MicroBlaze/FSL/pu xml Loading C:/Impulse/CoDeveloper3/Archite | Software activated | |
| Impulse C HDL Design Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/FSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/FSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/FSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/FSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/FSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PSL/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIDL/Xilrax/PB/bus xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIrcvBlaze/FSL/standalone xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIrcvBlaze/FSL/pu xml Loading C://mpulse/CoDeveloper3/Architectures/Xilrax/MIrcvBlaze/ | "C:/Impulse/CoDeveloper3/bin/impulse_arch" "-aC:/Impulse/CoDevelop | per3/Architectures/xilinx_mb_fsl.xml" -dc -no_port_bus_connect -swdirsw -files |
| Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/CoBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/IncoBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/IncoBlaze/FSL/standalone xml IbCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Stating rgc:Sting = libCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Stating Design generation complete chmod R= vm km mkdir sw 'C:/Impulse/CoDeveloper3/Architectures/XilinX/Impulse/LoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirftw-files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co_ int.c Impulse CoDeveloper3/Architectures/XilinX/ImcoBlaze/FSL/pu xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/ImcoBlaze/FSL/pu xml Loading C:/Impulse/CoDev | "FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd " FIR_Accelerato | or xic hw/FIR_Accelerator_top.vhd |
| Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/Fsl.2bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/CoBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/IncoBlaze/FSL/standalone xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/IncoBlaze/FSL/standalone xml IbCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Stating rgc:Sting = libCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Stating Design generation complete chmod R= vm km mkdir sw 'C:/Impulse/CoDeveloper3/Architectures/XilinX/Impulse/LoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirftw-files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co_ int.c Impulse CoDeveloper3/Architectures/XilinX/ImcoBlaze/FSL/pu xml Loading C:/Impulse/CoDeveloper3/Architectures/XilinX/ImcoBlaze/FSL/pu xml Loading C:/Impulse/CoDev | Impulse C HDL Design Generator | Contraction Contra |
| All rights reserved. Loading C://mpulse/Co.Developer3/Architectures/xilinx_mb_fsl.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/McroBlaze/FSL/standalone xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/McroBlaze/FSL/standalone xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/McroBlaze/FSL/standalone xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/McroBlaze/FSL/standalone xml Loading FIR_Accelerator xic lbCoreGornCommon tcl:/GenerateCoreGenFilesForEDK():Starting rgcString = ibCoreGornCommon tcl:/GenerateCoreGenFilesForEDK():Exiting Design generation complete chmod -R +rw hw mkdir sw C://mpulse/Co.Developer3/Architectures/Xilinx_mb_fsl.xml" +hwdirhw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co/c Loading C://mpulse/Co.Developer3/Architectures/Xilinx_mb_fsl.xml" +hwdirhw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co/c Loading C://mpulse/Co.Developer3/Architectures/Xilinx_mb_fsl.xml" -hwdirhw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co/c Loading C://mpulse/Co.Developer3/Architectures/Xilinx_mb_fsl.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C://mpulse/Co.Developer3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml E:// To ComplexFilter h.Filter, h: do cp \$i sw; done cfr in ComplexFilter h | Copyright 2002-2012 Impulse Accelerated Technologies Inc. | |
| Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/FSL/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/FSL/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/CPB/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C://mpules/CoDeveloper3/Architectures/VHDL/Xilinx/OPB/bus xml Loading C://mpules/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone xml Loading C://mpules/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone xml BiCoreGenCommon.tcl:/GenerateCoreGenFilesForEDK():Stating ngcString = biCoreGenCommon.tcl:/GenerateCoreGenFilesForEDK():Stating Design generation complet shmdd R: ww tw mkdir sw treffea/CoDeveloper3/Architectures/Ailrox/mpulse/CoDeveloper3/Architectures/Ailrox_mb_fslxml" hwdirftw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator Xic sw(co_init.c Impulse C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/spu xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone xmlLoading FIR_Accelerator xic for in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in ComplexFilter h Filter h; do cp \$i sw; done cr in Comp | | |
| Lading C:/mpulse/CoDeveloper3/Architectures/HDL/Xilinx/FSL/bus xml Loading C:/mpulse/CoDeveloper3/Architectures/HDL/Xilinx/McroBlaze/FSL/standalone xml Loading C:/mpulse/CoDeveloper3/Architectures/WHDL/Xilinx/OPB/bus xml BiCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Statting ngcString = BiCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Statting Design generation complete chmod -R +nw hw mcdir sw "C:/mpulse/CoDeveloper3/bin/impulse_Jib" "-aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" +hwdirfhw files "ComplexFilter.c Filter_sw.c" IR Accelerator xic sw.vico_init.c Rights CoStravered. Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.sic for in ComplexFilter F. Filter_sw c; d oc p \$i sw; done chmod -R +nw sw =================================== | | |
| Loading C:/mpulse/CoDeveloper3/Architectures/VHDL/Arget.zml Loading C:/mpulse/CoDeveloper3/Architectures/VHDL/Xilinx/derblagus.zml Loading C:/mpulse/CoDeveloper3/Architectures/VHDL/Xilinx/derblagus.zml Loading C:/mpulse/CoDeveloper3/Architectures/VHDL/Xilinx/derblagus.zml Loading C:/mpulse/CoDeveloper3/Architectures/VHDL/Xilinx/derblagus.zml Loading FIR_Accelerator.ic ibCoreCommon tot:/GenerateCoreGenFilesForEDK():Starting rgsString = bibCoreCommon tot:/GenerateCoreGenFilesForEDK():Exting Design generation complete shmdd-R + wr kw mkdir sw 'C:/mpulse/CoDeveloper3/Architectures/Alinx/mpulse/CoDeveloper3/Architectures/alinx_mb_fsl.zml" +hwdirftw-files "ComplexFilter c Filter_sw.c" FIR_Accelerator.ic sw/co_init.c Impulse C:/mpulse/CoDeveloper3/Architectures/alinx_mb_fsl.zml" +hwdirftw-files "ComplexFilter c Filter_sw.c" FIR_Accelerator.ic sw/co_init.c Impulse C:/mpulse/CoDeveloper3/Architectures/alinx_mb_fsl.zml" +hwdirftw-files "ComplexFilter c Filter_sw.c" FIR_Accelerator.ic sw/co_init.c Impulse C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.sic for in ComplexFilter Filter is w c; do cp \$i sw; done chood -R +w sw =================================== | | |
| Lading C:/mpulse/CoDeveloper3/Architectures/XIImX/MicroBlaze/FSL/standalone.xml Lading C:/mpulse/CoDeveloper3/Architectures/XIImX/MicroBlaze/FSL/standalone.xml Lading FIR_Accelerator.xc IbCoreCencCommon td::GenerateCoreGenFilesForEDK():Starting ingcString = IbCoreCencCommon td::GenerateCoreGenFilesForEDK():Exiting Design generation complete chmod R + wh wh mkdir sw "C:/mpulse/CoDeveloper3/bin:/mpulse_IbF "-aC:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" +hwdirfw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c Impulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerator Loading C:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirfw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c Impulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerator Loading C:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter h Filter h: do cp \$i sw; done chmod -R +w sw =================================== | | |
| Laading C://mpulee/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xml Laading FIR_Accelerator xic libCoreCommon tot:/GenerateCoreGenFilesForEDK():Exiting gcString = bibCoreCommon tot:/GenerateCoreGenFilesForEDK():Exiting Design generation complete chmod -R +rw hw rkdr sw C://mpulee/CoDeveloper3//bin/impulse_lib" "-aC://mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" +hwdirhw-files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c mpulse C.Software Interface GenerateT Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C://mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xmlLoading FIR_Accelerator xic sw.c; do cp \$i sw; done for in ComplexFilter c. Filter_sw.c; do cp \$i sw; done for in ComplexFilter Linterh; do cp \$i sw; done chind -R +rw sw =================================== | | |
| Lading FIR_Acceleratorxic ibCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Starting ngcString = libCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Starting ngcString = libCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Starting Design generation complete shmdd R+ wh hw mkdir sw "C./Impulse/CoDeveloper3/Architectures/colleveloper3/Architectures/xilinx_mb_fslxml" +hwdirftw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw.co FIR_Accelerator xic sw.co FIR_Accelerator xic sw.co FIR_Accelerator xic sw.co Int c mpulse CoDeveloper3/Architectures/xilinx_mb_fslxml Loading C:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fslxml Loading C:/Impulse/CoDeveloper3/Architectures/xilinx_Mb_fslxml Loading C:/Impulse/CoDeveloper3/Architectures/xilinx/MicroBlaze/FSL/cpu xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu xml Loading FIR_Accelerator xic for in ComplexFilter c. Filter_sw .c; do cp \$i sw; done cor in ComplexFilter c. Filter_sw .c; do cp \$i sw; done cor in ComplexFilter h Filter h; do cp \$i sw; done cor in ComplexFilter h Filter h; do cp \$i sw; done cor in ComplexFilter bluld' complete ======= Build G Find in Files System | | |
| Laading FIR_Åccelerator.xic ibCoreGenCommon.tcl::GenerateCoreGenFilesForEDK():Statting gcString = ibCoreGenCommon.tcl::GenerateCoreGenFilesForEDK():Batting Design generation complete chmod. R +nv hv mddi sw "C:/Impulse/CoDeveloper3/bin/impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hvwdirhw files "ComplexFilter.c Filter_sw.c" TR_Accelerator.xic sw/co_mit.c TR_Docolerator.xic sw/co_mit.c Hinghas C.Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Leading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Leading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter f. Filter_sw c; d oc p \$\$ sw; done chmod -R +nv sw ==================================== | | |
| biocreSenCommontcl::GenerateCoreGenFilesForEDK():Starting goString = biocreSenCommontcl::GenerateCoreGenFilesForEDK():Starting Design generation complete shmod -R + w hw mkdir sw 'C://mpulse/CoDeveloper3/Achitectures/collinx_mb_fsl.xml" +hwdirftw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw(co_init.c Mingdise CS Developer3/Achitectures/xilinx_mb_fsl.xml A in ghts reserved. Loading C://mpulse/CoDeveloper3/Architectures/xilinx/MicroBlaze/FSL/cpu.xml Loading C://mpulse/CoDeveloper3/Architectures/xilinx/MicroBlaze/FSL/cpu.xml ComplexFilter.c Filter_sw.c; 'Trint_Accelerator.sc 'Trint_ComplexFilter.c 'Tr | | sxml |
| ngeString = ibCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Exiting Design generation complete chmod -R +rw hw mkcdr sw "C./Impulse/CoDeveloper3/bin/Impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirftw files "ComplexFilter.c Filter_sw.c" FIR_Acceleratorxic sw/co_init.c mpulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter Linter h, do cp Si sw; done chmod -R +rw sw ==================================== | Loading FIR Acceleratorixic | |
| ngeString = ibCoreGenCommon tcl::GenerateCoreGenFilesForEDK():Exiting Design generation complete chmod -R +rw hw mkcdr sw "C./Impulse/CoDeveloper3/bin/Impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirftw files "ComplexFilter.c Filter_sw.c" FIR_Acceleratorxic sw/co_init.c mpulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/pu.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter Linter h, do cp Si sw; done chmod -R +rw sw ==================================== | libCoreGenCommon tcl::GenerateCoreGenFilesForEDK0:Starting | |
| bioorcompete chmod -R + w tw mkdir sw (C./mpulse./CoDeveloper3//cbin/impulse_lib" "-aC:/mpulse/CoDeveloper3/Architectures/xdimx_mb_fsl.xml" +twdirftw_files "Complex.Filter.c Filter_sw.c" FIR_Accelerator.xic sw/co_init.c mpulse C.SotWare Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/mpulse/CoDeveloper3/Architectures/xdimx_mb_fsl.xml Loading C:/mpulse/CoDeveloper3/Architectures/xdimx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xdimx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xdimx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex.Filter c. Filter_sw c; do cp \$i sw; done chmod -R +w sw ======== Build G Larget 'build' complete ======= | | |
| Design generation complete chmod -R +nv hw mkdr sw "C./Impulse/CoDeveloper3/bin/impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/xdimx_mb_fsl.xml" +hwdirhw files "ComplexFilter c Filter_sw.c" FIR_Accelerator xic sw/co_jmit.c Impulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilimx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilimx_MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter c Filter_sw.c; do p \$i sw; done chinod -R +nv sw ==================================== | | |
| chmod R +rw hw mkdi sw "C:/Impulse/CoDeveloper3/bin/impulse_lib" "-aC:/Impulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" +hwdirhw files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c TR_Model CoDeveloper3/Architectures/xilinx_mb_fsl.xml" +hwdirhw files "ComplexFilter.c Filter_sw.c" Ingulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter Lifter filter_sw c; d oc p \$i sw; done chmod -R +rw sw ==================================== | | |
| mkdir sw "C:/mpulse/CoDeveloper3/bin./mpulse_lib" "-aC:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirhw-files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c mpulse C Software Interface Generator Copyright 2022-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for i in ComplexFilter.c Filter_sw.c; do cp \$i sw; done cf i in ComplexFilter.c Filter_sw.c; do cp \$i sw; done cf i in ComplexFilter.f Filter.h; do cp \$i sw; done cf i in ComplexFilter.f biller.h; do cp \$i sw; done cf i in ComplexFilter filter.h; ido cp \$i sw; done cf i in GomplexFilter.f biller filter.h; do cp \$i sw; done cf i in Gomplex Filter filter.f is und C. monther filter.f i filter | besign generation complete | |
| "C./mpulse/CoDeveloper3/bin/impulse_lib" "-aC:/mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml" -hwdirhw-files "ComplexFilter.c Filter_sw.c" FIR_Accelerator xic sw/co_init.c Impulse C Statware Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xml cri in ComplexFilter Libter_sw.c; do cp \$\$ sw; done cri in ComplexFilter Libter h; chief Ar +nw sw ==================================== | | |
| FIR_Accelerator xic swi/co_init.c migulae C.Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/CoDeveloper3/Architectures/Xilimx_mb_fsl.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilimx/MicroBlaze/FSL/cpu.xml Loading C:/Impulse/CoDeveloper3/Architectures/Xilimx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter Lifter Switch complexes/Views/Xilims/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter Lifter A: give complexes/Xilims/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter Lifter A: give complexes/Xilims/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter A: Biter A: give complexes ethomod -R +rw sw ======== Build Filter Filter S: System | | |
| Impulse C Software Interface Generator Copyright 2002-2012, Impulse Accelerated Technologies, Inc. In rights reserved. Loading C://mpulse/Co.Developer3/Architectures/Xilimx_MicroBlaze/FSL/cpu xml Loading C://mpulse/Co.Developer3/Architectures/Xilimx/MicroBlaze/FSL/standalone xmlLoading FIR_Accelerator xic for in Complex/Filter /. Filter_sw.c; do cp \$I sw: done for in Complex/Filter /. Filter filter /st done chmod -R +rw sw ==================================== | "C:/Impulse/CoDeveloper3/bin/impulse_lib" "-aC:/Impulse/CoDevelope | r3/Architectures/xilinx_mb_fsl.xml" -hwdirhw -files "Complex-Filter.c Filter_sw.c" |
| Copyright 2002-2012, Impulse Accelerated Technologies, Inc. All rights reserved. Loading C:/Impulse/Co.Developer3/Architectures/Xilmx_mb_fsl.xml Loading C:/Impulse/Co.Developer3/Architectures/Xilmx/MicroBlaze/FSL/spu.xml Loading C:/Impulse/Co.Developer3/Architectures/Xilmx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in Complex/Filter L. Filter .sw c; do cp \$i sw; done for in Complex/Filter L. Filter .sw c; do cp \$i sw; done chmod -R +rw sw ==================================== | | |
| All rights reserved. Loading C://mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter.c Filter, xwo: cdo co g Si sw; done for in ComplexFilter.h Filter.h; do cp Si sw; done chmod -R +rw sw ==================================== | | |
| All rights reserved. Loading C://mpulse/CoDeveloper3/Architectures/xilinx_mb_fsl.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/opu.xml Loading C:/mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter.c Filter, xwo: cdo co g Si sw; done for in ComplexFilter.h Filter.h; do cp Si sw; done chmod -R +rw sw ==================================== | Copyright 2002-2012, Impulse Accelerated Technologies, Inc. | |
| Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/cpu.xml Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter filter_sw c; do cp \$I sw; done chinod -R +rw sw Build of target 'build' complete =================================== | All rights reserved. | |
| Loading C:/Impulse/Co.Developer3/Architectures/Xilino/MicroBlaze/FSL/cpu.xml Loading C:/Impulse/Co.Developer3/Architectures/Xilino/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter. Filter.xw; cdo cp \$i sw; done chmod -R +rw sw =================================== | | |
| Loading C://mpulse/CoDeveloper3/Architectures/Xilinx/MicroBlaze/FSL/standalone.xmlLoading FIR_Accelerator.xic for in ComplexFilter.c Filter_sw.c; do cp \$I sw; done chimod -R +rw sw ==================================== | | /cpu xml |
| for in ComplexFilter C Filter swic; do cp \$1 sw; done for in ComplexFilter C Filter swic; do cp \$1 sw; done chimdor R + w sw Build of target 'build' complete | | |
| ior i in ComplexFilter h Filter h; do cp \$i sw; done chmod -R +rw sw =================================== | | standard stan |
| chmod -R +rw sw Build of target 'build' complete | | |
| Build of target 'build' complete ======== | | |
| Build 🖼 Find in Files 🖸 System | cnmod -ri +rw sw | |
| Build 🖼 Find in Files 🖸 System | Duild of terms in the data and the | |
| | build of larger build complete ======= | |
| | 2003 | |
| | Build Hind in Files L System | |
| | | |

Note: the processing of this example may require a few minutes to complete, depending on the performance of your system.

When processing has completed you will have a number of resulting files created in the **hw** and **sw** subdirectories of your project directory.

See Also

Exporting Files from CoDeveloper

1.2.5 Exporting Files from CoDeveloper

Complex FIR Filter Tutorial for MicroBlaze, Step 5

Recall that in <u>Step 4</u> you specified the directory **EDK** as the export target for hardware and software. These export directories specify where the generated hardware and software processes are to be copied when the **Export Software** and **Export Hardware** features of **CoDeveloper** are invoked. Within these target directories (in this case EDK), the specific destination (which may be a subdirectory under EDK) for each file previously generated is determined from the Platform Support Package architecture library files. It is therefore important that the correct Platform Support Package (in this case Xilinx MicroBlaze FSL) is selected prior to starting the export process.

To export the files from the build directories (in this case hw and sw) to the export directories (in this case the EDK directory), select Project -> Export Generated Hardware (HDL) and Project -> Export Generated Software, or select the Export Generated Hardware and Export Generated Software buttons from the toolbar.

Export the Hardware and Software Files



Note: you must select BOTH **Export Software** and **Export Hardware** before going onto the next step, it does not matter in what order you export them.

You have now exported all necessary files from CoDeveloper to the Xilinx tools environment.

See Also

Creating the Platform Using the Xilinx Tools

1.2.6 Creating a Platform Using Xilinx Tools

Complex FIR Filter Tutorial for MicroBlaze, Step 6

As you learned in the previous <u>Hello World tutorial</u>, **CoDeveloper** creates a number of hardware and software-related output files that must all be used to create a complete hardware/software application on the target platform (in this case a Xilinx FPGA with an embedded **MicroBlaze** processor). This section will walk you through the file export/import process for this example, using the EDK System Builder (Platform Studio) project.

Creating a New Xilinx Platform Studio Project

Now we'll move into the Xilinx tool environment. Begin by launching Xilinx Platform Studio (from the Windows Start ->Xilinx ISE Design Suite 13.4 -> EDK -> Xilinx Platform Studio) and creating a new project. The Xilinx Platform Studio dialog appears as shown below:



Select the Base System Builder wizard (recommended), and click OK.

Next, in the **Create New XPS Project Using BSB Wizard** dialog, click **Browse** and navigate to the directory you created for your **Xilinx EDK** project files. For this tutorial we choose the directory name **EDK**, which is also the directory name we specified earlier in the **Generate Options** dialog. Click **Open** to create a project file called **system.xmp** (you can specify a different project name if desired):

| | c. (sandbox (con | iplexi11C_MICIOBIA2 | e\EDK\system.xmp | | Browse |
|---------------|--------------------------------------|---|--|-----------------------|--------------|
| elect an Int | erconnect Type | | | | |
| C AXI Sy | stem | | | | |
| curre | | ersions of Xilinx IP | opted by Xilinx as the and tool flows. Deta | | |
| PLB Sy | stem | | | | |
| Spart new | an6 and Virtex6. designs that may | . PLB IP will not s y migrate to futur | Xilinx that supports of upport newer FPGA e FPGA families. Det ent on xilinx.com. | families, so is not r | ecommend for |
| | a .bsb Settinas I | File(saved from pr | evious session) | | |
| elect Existin | | | | | |
| elect Existin | | | | | Browse |

Now click OK in the Create New XPS Project Using BSB Wizard dialog. The Base System Builder - Welcome page will appear. Select I would like to create a new design (the default), then click Next to choose your target board.

Choose your development board from the dropdown boxes. This example will use the following board (you should choose the reference board you have available for this step):

Board Vendor: Xilinx Board Name: Virtex 6 ML605 Evaluation Platform Board Revision: D

| 😻 Base System Bi | ıilder | | | | | | ? × |
|--|--|--|---|---|---|---|---|
| Welcome | | Board | System | Processor | Peripheral | Cache | Summan |
| Board Selection Select a target dev | elopment | board. | | | | | |
| Board | | | | | | | |
| • I would like to c | reate a sys | stem for the fo | ollowing developme | ent board | | | |
| Board Vendor | Xilinx | | | | | | • |
| Board Name | Virtex 6 I | 4L605 Evaluati | on Platform | | | | • |
| Board Revision | D | | | | | | - |
| C I would like to c | reate a sys | stem for a cust | tom board | | | | |
| - Board Information - | | | | | | | |
| Architecture | | Device | | Package | Sp | eed Grade | |
| virtex6 | | ▼ хсбух240 | Ot | ▼ ff1156 | ▼ -1 | | Y |
| Use Stepping | | | | | | | ¥ |
| Reset Polarity Acti | ve High | | | | | | 7 |
| Related Information | | | | | | | |
| Vendor's Website | | | | | | | |
| Vendor's Contact In | formation | | | | | | |
| Third Party Board D | efinition Fi | les Download V | Nebsite | | | | |
| FF1156 device. The 128MB of Platform I Additional user desir count (HPC) expans | board inc Flash, 1KB red feature sion conne | ludes Gigabit T IIC EEPROM, U es can be adde ctor or the on | Tri-Mode Ethernet JSB peripheral, CPU ed through daught I-board FMC VITA-5 | MAC/PHY, 512MB DD J Debug connectors er cards attached to 57 low pin count (LP | The ML605 board utiliz DR3 SDRAM SODIMM m , System ACE CF contro the on-board high spe C) connector. Due to A as an option from ML60 | emory, 32MB BPI oller and RS232 se ed FMC VITA-57 Answer Record 39 | Linear Flash, erial port. high pin 430 |
| More Info | | | | | < Back | Next > | Cancel |

Click **Next** to continue with the **Base System Builder** wizard. In the next wizard page, make sure that **Single-Processor System** is selected:



Click Next to continue with the Base System Builder wizard.

Note: the **Base System Builder** options that follow may be different depending on the development board you are using.

In this tutorial we are not using any floating point arithmetic so make sure the "Enable Floating Point Unit" is NOT selected.

| Base System Builder | | | | | | ? : |
|---------------------------|--------------|-------------|-----------|------------|-------|-------|
| Welcome | Board | System | Processor | Peripheral | Cache | Summa |
| rocessor Configuration | า | | | | | |
| onfigure the processor(s) |). | | | | | |
| eference Clock Frequenc | v 200.00 | | | | | → MH: |
| Processor 1 Configuratio | | | | | | |
| Processor Type | MicroBlaze | | | | | • |
| System Clock Frequency | | | | | | ▼ MHz |
| Local Memory | 8 KB | | | | | - |
| Debug Interface | On-Chip HW D | ebug Module | | | | |
| Enable Floating Point | | ebug Modale | | | | |
| Chable Housing Forme | Onic | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Click "Next" to continue with the Wizard which will display a peripheral selection dialog. Add the xps_timer to the configuration by selecting xps_timer from the Peripheral Names pane and clicking the Add button.



Clicking "Next" will display the Cache Configuration dialog. The MicroBlaze is a "Harvard Architecture" processor having separate data and instruction buses and therefore separate cache's if used. This dialog is used to enable cache and to set the size of each cache. Since this tutorial uses a very small application it is not necessary to use either instruction or data caching and the dialog may be left in its default state.

| Welcome | Board | System | Processor | Peripheral | Cache | Sur |
|--|---------------------|---------------------|----------------------|----------------------|---------------------|---------|
| che Configuration | | | | | | |
| ect cache size and ca | che memory for p | rocessor(s). | | | | |
| | | | | | | |
| rocessor 1 (MicroBlaze | | | | | | |
| n MicroBlaze, caches a arge sized caches. | are optional and co | onfigurable. Caches | are implemented usir | ng FPGA LUTs for sma | I caches or Block F | AMs for |
| ☐ Instruction Cache | e | | Data Cad | he | | |
| Instruction Cache Siz | e 2 KB | | T Data Cache | Size 2 KB | | Y |
| Instruction Cache Me | mory | | Data Cache | | | |
| C DDR3_SDRAM C FLASH | | | O DDR3_S | DRAM | | |
| C FLASH | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Click the "Next" button to display the base system Summary dialog.

| Welcome | Depend | Custom | Dragoonar | Desinheral | Casha | Cummer |
|--|----------------------------|---------------|--------------|------------|--------|--------|
| vveicome | Board | System | Processor | Peripheral | Cache | Summa |
| | | | | | | |
| Summary | | | | | | |
| Relow is the summany o | f the system you are cre | ating | | | | |
| below is the summary o | i che system you are cre | acing. | | | | |
| ystem Summary | | | | | | |
| Core Name | Instance Name | Base Address | High Address | | | - |
| Processor 1 | microblaze 0 | | | | | |
| mpmc | DDR3 SDRAM | 0x90000000 | 0x9FFFFFFF | | | |
| xps gpio | DIP Switches 8Bit | 0x81460000 | 0x8146FFFF | | | |
| | Ethernet MAC | 0x81000000 | 0x8100FFFF | | | |
| xps_mch_emc | FLASH | 0x8C000000 | 0x8DFFFFFF | | | |
| xps_iic | IIC DVI | 0x81660000 | 0x8166FFFF | | | |
| xps_iic | IIC EEPROM | 0x81640000 | 0x8164FFFF | | | |
| xps lic | IIC FMC LPC | 0x81620000 | 0x8162FFFF | | | |
| xps_ic | IIC SFP | 0x81600000 | 0x8160FFFF | | | |
| xps_ic | LEDs 8Bit | 0x81440000 | 0x8144FFFF | | | |
| xps_gpio | LEDs Positions | 0x81420000 | 0x8142FFFF | | | |
| xps_gpio | Push Buttons 5Bit | 0x81400000 | 0x8140FFFF | | | |
| xps_gpio | RS232 Uart 1 | 0x84000000 | 0x8400FFFF | | | |
| xps_usrciec | SysACE CompactFlash | | 0x8360FFFF | | | |
| Imb bram if cnt | | 0x00000000 | 0x00001FFF | | | |
| | | uxininininini | UXUUUUUUUUUU | | | |
| ile Location | | | | | | |
| Overall | the life much | | | | | |
| | World\EDK\system.xmp | | | | | |
| | World\EDK\system.mhs | | | | | |
| | World\EDK\data\system. | | | | | |
| | World\EDK\etc\fast_runt | | | | | |
| | World\EDK\etc\downloa | | | | | |
| C:\sandbox\Hello | World\EDK\etc\bitgen.u | t | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | ilder (.bsb) Settings File | | | | | |
| Save Base System Bu | | | | | | |
| | \EDK\system.bsb | | | | | |
| Save Base System Bu C:\sandbox\HelloWorld | \EDK\system.bsb | | | | | |
| | \EDK\system.bsb | | | < Back | Finish | Cancel |

This summary page shows the location of the various output files that will be used later to prepare the FPGA bit (programming) file. It also lists all of the software cores to be instantiated within the MicroBlaze and their base address. These addresses will be automatically used later, you do not need to record them. Click the "Finish" button.

When you clicked the "Finish" button the Wizard created and opened a Xilinx Platform Studio (EDK) project and populated it with the base system you specified with the Wizard.

The next steps will demonstrate how to configure the **MicroBlaze** processor and create the necessary I/ O interfaces for our sample application.

See Also

Configuring the New Platform

1.2.7 Configuring the New Platform

Complex FIR Filter Tutorial for MicroBlaze, Step 7

| | IP Catalog | | ⇔⊡₽× | LLP | Bus Interface | es Ports | Addresses | | <u></u> | Ð < |
|------------------|-------------------------------|------------|----------|----------------------------------|----------------------------------|------------------|-------------------|------------------------------|---------|-----|
| | 2 💿 🙂 🚳 🚳 🔒 | | | MML | Name | Bus Name | IP Type | IP Version | | 7 |
| Design Flow | Description | IP Version | IP Type | 0 | - dimb | | 👷 lmb_v10 | 2.00.b | | -1 |
| - | E EDK Install | | | | - imb | | 🙀 lmb_v10 | 2.00.b | | |
| | E Analog | | | | mb_plb | | 🌟 plb_v46 | 1.05.a | | |
| ~ | Bus and Bridge | | | | microblaze_0 | 1 | 🙀 microblaze | 8.20.b | | |
| Run DRCs | E Clock, Reset | | | | _ Imb_bram | | 👷 bram_block | | | |
| | Communicati | | | ▶ + + | dimb_cntir | | 🙀 Imb_bram | | | |
| nplement Flow | E Communicati | | | | _ ⊕ ilmb_cntlr | | 🙀 Imb_bram | | | |
| Inplement How | DMA and Timer | | | • | E FLASH | | 🙀 xps_mch | | | |
| 4 D | E Debug | | | | DDR3_SDR | | 👷 mpmc | 6.05.a | | |
| 28 | FPGA Reconfi General Purp | | | 4 6 6 | mdm_0 | | 🙀 mdm | 2.00.b | | |
| Record . | General Purp | | | a | Ethernet | | 🌟 xps_ether | | | |
| enerate Netlist | | | | è | DIP_Switch. | | 🙀 xps_gpio | 2.00.a | | |
| - | Memory and | | | a | E LEDs_8Bit | | 👷 xps_gpio | 2.00.a | | |
| 100 | E PCI | | | a | E LEDs_Positi. | <u>, 1</u> | 🙀 xps_gpio | 2.00.a | | |
| and a second | Peripheral Co | | | ê 🍑 | Push_Butt | | 🙀 xps_gpio | 2.00.a | | |
| nerate BitStream | Processor | | | • | □ IIC_DVI | | 🙀 xps_iic | 2.03.a | | |
| | 🕀 Utility | | | è | - IIC_EEPROM | | 👷 xps_iic | 2.03.a | | |
| \wedge | Verification | | | ê 🔶 | IIC_FMC_LPC | £ | 🏫 xps_iic | 2.03.a | | |
| SDK | B Project Local PCo | | | <u>ه</u> | - IIC_SFP | | 🙀 xps_iic | 2.03.a | | |
| Export Design | USER | | | ê 🔶 | B SysACE_Co | | 🙀 xps_sysace | | | |
| Export Dough | 🔤 🔫 fsl_filt | 1.00.a | fsl_filt | a | timer_0 | | 🚖 xps_timer | 1.02.a | | |
| | Project Periphera | | | | | | 👷 xps_uartlite | | | |
| | | | | | - clock_gene | | 🙀 clock_gen | | | |
| 2 | | | | | FLASH_CE | | 🙀 util_vector | | | |
| 6 | | | | | proc_sys_r | | 🙀 proc_sys_r | 3.00.a | | |
| | | | | | | | | | | |
| nerate HDL Files | | | | | | | | | | |
| | | | | 1 | <u>]</u> | | | | | ÷ |
| | | | | Legend Master Slave | Master (Clause in Tarr | ot / Talifatan A | | and Mension | | |
| лл | • | | • | Dreduction | anse (naid) | nse (eval) | Loonl Conconr | ction BBeta Covelopment | | |
| unch Simulator | | | | A Superseded | Discontinued | ise (eval) | Local APre Produc | 2000 vo bera per pevelopment | | |
| | Search IP Catalog: | | Clear | 🗵 Design Sun | | System Asser | mbly View 🛛 🔞 | Graphical Design View | | |
| | | | | Za Design Sun | | System Asser | | Graphical Design View | | |
| | Errors | | | | | | | | ↔ 🗆 | Ē |
| | | | | | | | | | | 1 |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

After completing the previous tutorial step you will be presented with a window similar to this:

It is now time to connect the hardware peripheral to the MicroBlaze processor. We want to use the FSL (fast simplex link) but there is none attached to the processor. In the IP Catalog pane, expand the Bus and Bridge class and select Fast Simplex Link.

| 12 🖸 🕑 🕃 🚳 🗟 | | |
|-----------------|------------|----------|
| Description | IP Version | IP Турє_ |
| 🖻 🐮 EDK Install | 20 | |
| Analog | | |
| Bus and Bridge | | |
| - 📩 AHB-L | 1.00.a | ahblite_ |
| - 🚖 AXI t | 1.00.a | axi2axi_ |
| — 🚖 AXI4 | 1.00.a | axi_ahb |
| - 🚖 AXI4 | 1.00.a | axi_apb |
| - 🚖 AXI I | 1.04.a | axi_inte |
| - 🚖 AXI I | 1.05.a | axi_inte |
| 🚖 AXI t | 2.02.a | axi_plbv |
| - 🚖 Fast S | 2.11.e | fsl_v20 |
| - 📩 Local | 2.00.b | Imb_v1 |

Then right-click the Fast Simplex Link IP to show the instance menu and select Add IP. Do this twice so that two instances of the FSL is added to the system assembly.

| AXI I | 1.05.a 2.02.a | axi_inte axi_plbv | | 3 |
|-------------------------------|--------------------------------------|-------------------------------|----------|------------|
| - Fast S | Add IP | | | |
| | View MPD View IP Mo View PDF D | difications (Cha latasheet | nge Log) | |
| Communicati. DMA and Time | Make This I | P Local | | Ş — |
| E Debug | | | 10.0 | 1 |
| E FPGA Reconfi | | | - | |



The System Assembly should now be changed to appear similar to this:

At this point the two FSL modules do not go anywhere, now it is time to connect them to the MicroBlaze processor. Select the microblaze_0 instance from the System assembly, right-click it and select Configure IP.

| mb_p | lb · | 👷 plb_v46 | 1.05.a |
|----------------|--|-----------|----------|
| | Configure IP | e | 8.20.b |
| | Show Sub-system View | | |
| | Show Ports for selected I | Ps | |
| | Configure Coprocessor | | |
| | View MPD | | |
| ⊡ lmb ⊡ dlm | View IP Modifications (Cha | inge Log) | 1.00.a |
| ± ilmt | View PDF Datasheet | 1 | . 3.00.1 |
| + FLA | Browse HDL Sources | La | 3.01.2 |
| E DDI - | | | 6.05.a |
| 🕀 mdi | Delete Instance | | 2.00.8 |
| Eth - | Make This IP Local | Þr | 4.00.2 |
| E DIP | and the second | | 2.00.2 |
| + LEDS | and a stand | xps_gpio | 2.00.2 |

The Configure IP Wizard for the microblaze will pop-up. Click "Next" until you get to page 4 (PVR and Buses). Then change the number of stream links from zero to two, as shown here:

| Page 4 of 4 - PVR and Buses | | |
|---------------------------------------|-------|------------|
| Processor Version Registers | | |
| Specifies Processor Version Register | | NONE |
| Specify USER1 Bits in PVR | | 0x00 |
| Specify USER2 Bits in PVR | | 0x00000000 |
| Buses | | |
| Select Bus Interface | | PLBv46 |
| Select Stream Interfaces | | FSL 💌 |
| Number of Stream Links | | 2 ÷ |
| Enable Additional Stream Instructions | | |
| Enable Stream Exception | | |
| | | |
| | | |
| Advanced | < Bac | k First |
| | | |
| requency | | BRAM |
| rea National State | | |
| erformance | | DSP48E1 |
| | | |
| | | |

Click "OK" and after a few moments the System Assembly will look similar to this:

| FFLL | P | Bus Interface | s Ports Ad | ldresses | 10 |
|---------|----------|----------------|----------------|--------------|-----------|
| S S M M | L | Name | Bus Name | IP Type | IP Versio |
| | | fsl_v20_0 | | 🙀 fsl_v20 | 2.11.e |
| | | fsl_v20_1 | | 1 fsl_v20 | 2.11.e |
| 880- | | dlmb | | 🙀 lmb_v10 | 2.00.b |
| 1 | | ilmb | | 🙀 Imb_v10 | 2.00.b |
| | | mb plb | | 📩 plb v46 | 1.05.a |
| | | E microblaze_0 | | 🚖 microblaze | 8.20.b |
| | | DLMB | dimb | • | |
| | | ILMB | ilmb | • | |
| | | - DPLB | mb plb | • | |
| | <u> </u> | - IPLB | mb plb | - | |
| | | - SFSL0 | No Connecti | - | |
| | 88 | MFSL0 | No Connecti | • | |
| | | - SFSL1 | No Connecti | • | |
| | | MFSL1 | No Connecti | • | |
| | | DRFSL0 | No Connecti | • | |
| 88.88 | | DWFSL0 | microblaze_0 | | |
| | 88 | DRFSL1 | No Connecti | • | |
| | | DWFSL1 | microblaze_0 | | |
| | | - DXCL | microblaze_0 | | |
| | | - IXCL | microblaze_0_I | | |
| | | DEBUG | microblaze | • | |
| | | TRACE | microblaze_0 | | |
| | | 1 Imb_bram | | pram_block | 1.00.a |
| | | | | | |

Now we can connect the MicroBlaze to the two FSL modules. The easy way to do this is to move the cursor into the pink area for one of the FSL modules. This will cause the potential connections to appear, like this:

| FFLLP | 🕌 Bus Interfa | ces Port | s Ade | dresses |
|--|---------------|------------|---------|------------|
| S S M M L | Name | Bus Nar | ne | IP Ty |
| | | | | t fs |
| | - fsl_v20_1 | | | t fs |
| | dlmb | | | 📩 In |
| | ilmb | | | tr In |
| II | mb_plb | | | † p |
| | E microblaze | _0 | | 1 n |
| | DLMB | dlmb | - | |
| | ILMB | ilmb | - | |
| | DPLB | mb_plb | - | |
| | IPLB | mb_plb | - | |
| | SFSL0 | No Conr | necti 🔻 | |
| 1155 | MFSL0 | No Conr | necti 🔻 | |
| 1122 | SFSL1 | No Conr | necti 🔻 | 1 |
| II A A B B | MFSL1 | No Conr | necti 🔻 | 1 |
| | DRFSLO | No Conr | necti 🔻 | |
| | DWFSL | 0 microbla | 7e 0 | 12 |

From the legend at the bottom of the window you may note that a square indicates a Master and a circle indicates a Slave. Click the circles and squares as shown here to connect fsl_0 and fsl_1 to the correct MicroBlaze ports. The MicroBlaze will be the master on fsl_0 and the slave on fs_1:

| 12 III III III III III III III III III I | IPLB | mb_plb | • |
|--|--------|-------------|---|
| | SFSL0 | No Connecti | • |
| | MFSL0 | fsl_v20_0 | • |
| 122 | SFSL1 | fsl_v20_1 | - |
| | MFSL1 | No Connecti | • |
| | DRFSL0 | No Connecti | - |
| | | | |

The base system is almost complete, but the FSL IP is not connected to a clock or reset. The next step is to configure the clock_generator IP to create the 31.250MHz clock needed for the user logic. Right-click the clock_generator_0 instance in the System Assembly to launch the Configure IP wizard for the clock generator. Configure the CLKOUT4 output as shown here then click "OK".



At this point the base system is fully defined and it is time to import the USER IP and connect the two FSL's , clocks, and resets to the user defined hardware.

See Also

Importing the Generated Hardware

1.2.8 Importing the Generated Hardware

Complex FIR Filter Tutorial for MicroBlaze, Step 8

Import the User Defined Hardware

Return to the Bus Interfaces tab of the System Assembly View. Expand the Project Local PCores and USER modules. Then right-click the fsl_filt IP and select Add IP. Then click OK to put the IP into the system assembly. Expand the fsl_filt_0 IP to display the two related FSL elements. Then as with hooking-up the MicroBlaze in the last topic, connect the user logic to the FSL as shown here. Note that you must match a slave to the MicroBlaze master and a master to the MicroBlaze slave:

| FLLP | Bus Interfaces | Ports Addresse | IS | |
|----------|----------------------|----------------|--------------|------------|
| SMML | Name | Bus Name | IP Type | IP Version |
| - | fs_v20_0 | | 🙀 fsl_v20 | 2.11.e |
| | fsl_v20_1 | | 🙀 fsl_v20 | 2.11.e |
| | - dlmb | | 🚖 Imb_v10 | 2.00.b |
| | ilmb | | 🙀 Imb_v10 | 2.00.b |
| | mb_plb | | 🚖 plb_v46 | 1.05.a |
| | microblaze_0 | | 🙀 microblaze | 8.20.b |
| | 😐 Imb_bram | | 🚖 bram_block | 1.00.a |
| | 😐 dlmb_cntlr | | 🙀 Imb_bram | . 3.00.b |
| | 🕀 ilmb_cntlr | | 🚖 Imb_bram | 3.00.b |
| | - FLASH | | 🚖 xps_mch | 3.01.a |
| <u> </u> | DDR3_SDRAM | | 🚖 mpmc | 6.05.a |
| | mdm_0 | | 🏫 mdm | 2.00.b |
| | = fsl_fit_0 | | 🚜 fsl_filt | 1.00.a |
| <u> </u> | SFSL0 | fsl_v20_0 | - | |
| | MFSL1 | fsl_v20_1 | • | |
| | Ethernet_MAC | 1 | 🙀 xps_ether | 4.00.a |
| 1000 | I do an cuitates and | 13 C | A | 2 00 - |

Connecting Clock and Reset Ports

Next, you need to configure the clock and reset signals for each FSL IP Core. Click on the Ports filter in the System Assembly View and expand fsl_v20_0 and fsl_v20_1. For each FSL link, set FSL_Clk to CLK_S and set SYS_Rst to net_gnd as shown.

| lame | Connected Port | Net | Direction |
|-------------|---------------------------------------|-------------|-----------|
| | connecced rore | Hee | Direction |
| External Po | | | |
| fs/_v20_0 | | | 4 |
| FSL_Ck | External Ports: // External Ports: | CLK_S | · I |
| SYS_Rst | External Ports: | sys_rst_s | · I |
| FSL_M | / | No Connecti | · I |
| FSL_S | / | No Connecti | · I |
| FSL_Full | / | No Connecti | 0 |
| FSL_Ha | / | No Connecti | 0 |
| FSL_Co | 1 | No Connecti | 0 |
| fs/ v20_1 | | | |
| | External Ports: // External Ports: | CLK_S | · I |
| SYS_Rst | External Ports: | sys_rst_s | · I |
| FSL_M | / | No Connecti | I |
| | / | No Connecti | · I |
| - FSL_Full | / | No Connecti | 0 |
| FSL_Ha | 1 | No Connecti | 0 |
| FSL Co | / | No Connecti | 0 |

Select the **Ports** filter in the **System Assembly View** and expand **fsl_filt_0**. This should reveal ports **co_clk** and **FSL_Rst**. The **co_clk** has to be connected to the **CLK_S** clock that we configured in the previous steps. The **FSL_Rst** should be tied to **net_gnd**.

| 11.1 | E fsl_filt_0 | | | | |
|------|--------------|----------------------------|-------------|-----|-----|
| 233 | FSL_Rst | net_gnd / clock_generat | net_gnd | • I | RST |
| 10.0 | co_clk | clock_generat | clock_gener | ▼ I | CLK |
| 0.63 | Ethernet | | | | |

Note: if **co_clk** is missing from the **fsl_filt_0** section, then will need to return to <u>step 3</u> of this tutorial and specify the **Dual Clock** option in the **CoDeveloper Generate Options** page.

Now it is time to create a new clock net and connect the 31.250MHz clock and the co_clk to it.

Specify the Addresses

Now you will need to set the addresses for each of the peripherals specified for the platform. This can be done simply by selecting the **Addresses** tab and clicking on the **Generate Addresses** button. The addresses will be assigned for you automatically:

| instance | Base Name | Base Address | High Address | Size | | Bus Interface(s) | Bus Name | Lock |
|---|-------------|--------------|--------------|------|-------|------------------|----------|------|
| - microblaze_0's Address Map | | | | | 01-02 | | | |
| - dlmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | • | SLMB | dlmb | |
| - ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x00001FFF | 8K | - | SLMB | ilmb | |
| Ethernet_MAC | C_BASEADDR | 0x81000000 | 0x8100FFFF | 64K | - | SPLB | mb_plb | |
| - Push_Buttons_5Bit | C_BASEADDR | 0x81400000 | 0x8140FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_Positions | C_BASEADDR | 0x81420000 | 0x8142FFFF | 64K | - | SPLB | mb_plb | |
| LEDs_8Bit | C_BASEADDR | 0x81440000 | 0x8144FFFF | 64K | - | SPLB | mb_plb | |
| DIP_Switches_8Bit | C_BASEADDR | 0x81460000 | 0x8146FFFF | 64K | - | SPLB | mb_plb | |
| IIC_SFP | C_BASEADDR | 0x81600000 | 0x8160FFFF | 64K | • | SPLB | mb_plb | |
| - IIC_FMC_LPC | C_BASEADDR | 0x81620000 | 0x8162FFFF | 64K | • | SPLB | mb_plb | |
| IIC_EEPROM | C_BASEADDR | 0x81640000 | 0x8164FFFF | 64K | - | SPLB | mb_plb | |
| IIC_DVI | C_BASEADDR | 0x81660000 | 0x8166FFFF | 64K | - | SPLB | mb_plb | |
| SysACE_CompactFlash | C_BASEADDR | 0x83600000 | 0x8360FFFF | 64K | • | SPLB | mb_plb | |
| xps_timer_0 | C_BASEADDR | 0x83C00000 | 0x83C0FFFF | 64K | - | SPLB | mb_plb | |
| - RS232_Uart_1 | C_BASEADDR | 0x84000000 | 0x8400FFFF | 64K | • | SPLB | mb_plb | |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | • | SPLB | mb_plb | |
| FLASH | C_MEM0_BASE | 0x8C000000 | 0x8DFFFFFF | 32M | • | SPLB | mb_plb | |
| DDR3_SDRAM | C_MPMC_BASE | 0x90000000 | 0x9FFFFFFF | 256M | • | SPLB0 | mb_plb | |
| | | | | | | | | |

You have now exported all necessary hardware files from **CoDeveloper** to the **Xilinx** tools environment and have configured your new platform. The next step will be to generate FPGA bitstream. Make sure

that you save your project.

See Also

Generating the FPGA Bitmap

1.2.9 Generating the FPGA Bitmap

Complex FIR Filter Tutorial for MicroBlaze, Step 9

At this point, if you have followed the tutorial steps carefully you have successfully:

- Generated hardware and software files from the CoDeveloper environment.
- Created a new Xilinx Platform Studio project and created a new MicroBlaze-based platform.
- Imported your CoDeveloper-generated files to the Xilinx Platform Studio environment.
- Connected and configured the **Impulse C** hardware process to the **MicroBlaze** processor via the **FSL** bus.

You are now ready to generate the bitmap.

First, from within Platform Studio select the button **Generate Netlist**. Note that this will take several minutes to execute depending on how fast your computer is.

Second, if there are no errors in the netlist generation step then click the **Generate Bitstream** button. Depending on your computer, this may take well over an hour to execute depending on how fast your computer is.

When the bitstream has been generated, click **Export Design** to launch the Xilinx SDK to build your software-side application.



From the pop-up box, select "Export & Launch SDK". This will package the necessary files and export them to the correct locations for the SDK (software development kit) environment. The SDK will then be automatically launched and you can begin the next topic.

See Also

Importing the Application Software

1.2.10 Importing the Application Software

Complex FIR Filter Tutorial for MicroBlaze, Step 10

After clicking the Export Design button the pop-up (shown) is displayed giving you the option to perform an Export only or to Export files and start the Software Development Kit (SDK), for this tutorial please click "Export & Launch SDK".

NOTE: Screen images in this tutorial re-use images from the Hello World tutorial. Where the project "HelloWorld" is seen, please replace it it the project Complex_FIR_Microblaze.

| Export to SDK / Launch SDK | ? × |
|--|------------|
| This dialog allows you to export hardw platform information to be used in SDM | vare K. |
| ✓ Include bitstream and BMM file | |
| (XPS will regenerate bitstream if necessary, and it may take some time to finish.) | |
| Directory location for hardware description file | es |
| | |
| C:\sandbox\HelloWorld\EDK\SDK\SDK_Export | t |

After the SDK tool starts, the following pop-up will be displayed. The export tool will have correctly filled in the project workspace path data. We do not recommend setting this as the "default" since it is not likely to be the default for other projects.

| Browse |
|--------|
| |
| |

Once the SDK window is open you can close the EDK window as it will not be needed further for this tutorial. Your screen should be similar to the one shown:



Configure SDK

It is first necessary to configure the SDK environment to reference the EDK project directory as a "repository" in order to reference the FPGA logic driver by choosing the "Repositories" menu item from the "Xilinx Tools" menu. Click the "New" button next to the "Local Repositories" then navigate to the project's EDK directory and select it. The Preferences dialog should be similar to that shown here:



Click "Apply" and then "OK" to continue.

Create the Complex_FIR_Microblaze Project

Choose File -> New -> Xilinx C Project from the File menu. Then select Empty Application and set the Project name: to Complex_FIR_MIcroblaze_project. The window will appear similar to what is shown here:

| 🛞 New Project | | _ 🗆 🗙 |
|--|---|----------|
| New Xilinx C Project Create a managed make application project. | Choose from one of the sample applications. | G |
| Project name: HelloWorld_project | | |
| ✓ Use default location | | |
| Location: C:\sandbox\HelloWorld\EDK\SDK | \HelloWorld_project | Browse |
| Choose file system: default 💌 | | |
| Target Hardware Hardware Platform: EDK_hw_platform Processor: microblaze 0 | | <u>_</u> |
| Select Project Template | | |
| Dhrystone | Description | |
| Empty Application Hello World IwIP Echo Server Memory Tests Peripheral Tests SREC Bootloader Xilkernel POSIX Threads Demo Zyng FSBL | A blank C project. | X |
| 0 | < Back Next > Finish | Cancel |

Click Next to continue.

On the next dialog window set the Project name: to Complex_FIR_Microblaze_project_bsp_0; the window will now appear similar to what is shown here:

| New Project | | |
|--|--|---------|
| New Xilinx C Project Create a managed make application project. Cho | ose from one of the sample applications. | G |
| Create a new Board Support Package project | | |
| The template provided by application 'Empty | Application' will be used to configure the p | roject. |
| Project name: HelloWorld_project_bsp_0 | | |
| Use default location | | |
| Location: C:\sandbox\HelloWorld\EDK\SDK\I | HelloWorld_project_bsp_0 | Browse |
| Choose file system: default 💌 | | |
| | | |
| 0 | < Back Next > Finish | Cancel |

Click **Finish** to continue.

The **Project Explorer** will now include the two new projects, Complex_FIR_Microblaze_project and Complex_FIR_Microblaze_project_bap_0. Notice that as soon as the files were read into the project that the entire project was built. This is the default behavior of SDK. If you followed this tutorial accurately there should have been no errors requiring edits.

It is now necessary to increase the Stack and Heap memory sizes using the linker script. From the Project Explorer open the lscript.ld file, you will find it in the Complex_FIR_Microblaze_project src directory. Change the Stack and Heap sizes to 0x4000 as shown here:

| A linker script is used to control where different se in this page, you can define new memory regions, Available Memory Regions | | | |
|---|--------------|------------|---------|
| Name | Base Address | Size | Add Mem |
| ilmb_cntlr_dlmb_cntlr | 0x00000050 | 0x00001FB0 | |
| DDR2_SDRAM_MPMC_BASEADDR | 0xC8000000 | 0x08000000 | |
| Stack and Heap Sizes Stack Size 0x40000 Heap Size 0x40000 Section to Memory Region Mapping | | | |

Either click the file save icon or enter <ctrl>s to save the modified linker script.

Now the software files exported from CoDeveloper must be added to the project. We will do this by linking to their location as follows:

Right click the **src** directory in the HelloWorld_project and select **New -> Folder**. Click the **Advanced** button and select **Link to alternate location (Linked Folder)**. Use the browse button to locate the Complex_FIR_Microblaze\EDK\SDK\Complex_FIR_Microblaze directory. The New Folder dialog should look similar to that shown here:

| New Folder |
|---|
| Folder |
| |
| Enter or select the parent folder: |
| HelloWorld_project/src |
| |
| Folder name: HelloWorld |
| C C Use default location |
| Graph Folder is not located in the file system (Virtual Folder) Graph Link to alternate location (Linked Folder) |
| C:\sandbox\HelloWorld\EDK\SDK\Hel Browse Variables Choose file system: default |
| Resource Filters |
| Finish Cancel |

When you click Finish to continue the SDK will automatically rebuild the entire project and prepare the Complex_FIR_Microblaze_project.elf file. In the next step, the BitStream and ELF files will be downloaded into the FPGA.

The Export Design process (previous topic) created an SDK directory and populated it with files needed to add the software to your project. One interesting file is the "system.html" file that is a browsable representation of the overall system and is useful in visualizing and understanding the interactions of the system.

If you accurately followed the tutorial steps the export to SDK process will be fully automated and result in a complete software application with all source files, include files, and libraries correctly setup.

After the SDK opens you will see a number of build activities scrolling in the Console window. It is NOT UNEXPECTED to have the process end with two compiler errors. The first error is an undefined

symbol error and the second error is just a notice that a prior build step failed.

The undefined symbol error will be in the file Filter_sw.c and is an artifact of version differences in the Xilinx tool flow. To correct the error search for the text string "XPAR_XPS_TIMER_1_DEVICE_ID" (without the quotes) and change the "1" to "0" (zero). When you

click the file save button in the toolbar the project will be automatically recompiled successfully.

You can edit other source files and on each file save the project will be automatically updated and incrementally recompiled.

Next, you will run and debug the application using SDK.

See Also

Running the Application

1.2.11 Running the Application

Complex FIR Filter Tutorial for MicroBlaze, Step 11

Setting up Terminal Window and Connecting Cables

Open preferred terminal emulation program. Use the same communication settings you chose when defining the peripheral in **Base System Builder** (15200 baud, 8-N-1). Turn off flow control, if available. These settings were the defaults used.

Connect the serial port of your development machine to the **RS232** interface on your development board. Make sure the download (**JTAG**) cables are connected on the development board. Also ensure that the board is configured to be programmed. Turn on the power to the board. Refer to your development board documentation to correctly configure the board for programming.

Use the **Xilinx Tools -> Program FPGA** menu option to load the bit file into the FPGA. All necessary files will be automatically selected and downloaded. You can change the download files if necessary.

Running Application from SDK

Now let's run the application on the development board.

Select menu **Run** -> **Debug Configurations** select the desired debug configuration from the menu and click **Debug**.

A **Cygwin** bash shell will come up. It runs a script, connecting to the **MicroBlaze** processor and the debugger inside the FPGA. We can learn the base address of the **DDR2_SDRAM** is **0x88000000**.

Now watch your termial window again. You should see the messages generated by the software process indicating that the test data has been successfully filtered. The execution with hardware acceleration is **122** times faster than software only running on **MicroBlaze** microprocessor.

Congratulations! You have successfully completed this tutorial and run the generated hardware on the

development board.

See Also

Tutorial 1: Hello World on the MicroBlaze platform