

3L Diamond FPGA

SDR application – January 2006

Introduction

This application illustrates how you can develop Software Defined Radio (SDR) applications for the [Sundance](#) SMT8036 system using [3L Diamond](#).

The SMT8036 is a development kit for SDR applications consisting of a DSP coupled with an FPGA connected to two ADCs and a DAC.

It is possible to build a wide range of SDR applications using the 3L tools, and this application shows one such implementation. The system consists of tasks running on the DSP and FPGA. All the tasks used to construct this system have been carefully chosen to allow re-use in your own application.

DSP tasks are implemented in C while FPGA tasks may be implemented directly in VHDL as shown in this application or by using high level design tools such as [Celoxica's](#) DK toolset.

The SMT8036 system

The SMT8036 comprises a SMT365 connected to a SMT370. Both modules are held on a SMT310Q carrier board.

The SMT365 module provides the FPGA and DSP that are used for the processing. It has the following features:

- a 600Mhz Texas Instrument C6416 DSP
- a Xilinx VirtexII 2000 FPGA
- 8MB ZBTRAM connected to the DSP
- Twos 400MB/s high-speed bus interfaces (SHBs)
- Six 20MB/s comports
- 8MB flash



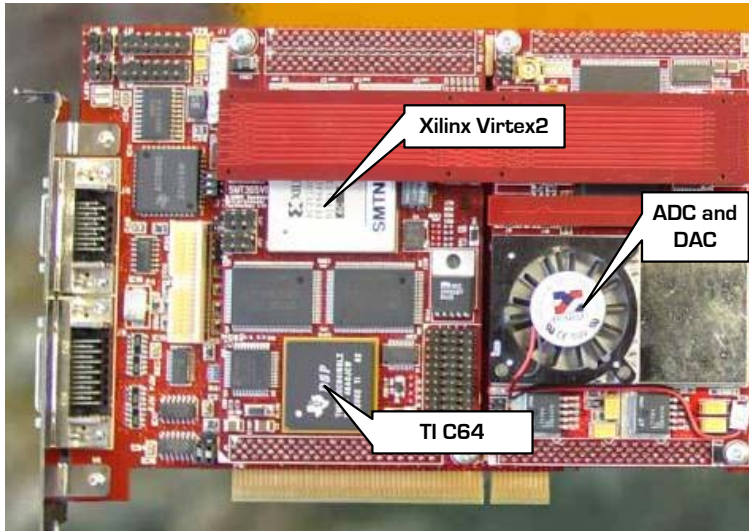


Figure 1: the SMT8036 system

The SMT370 has the following features:

- Two 14-bits ADCs at 105Mhz
- One 16-bits dual DAC at 400Mhz (interpolation)
- 4MB of NtSRAM at 160Mhz
- Two SHBs
- Two comports
- Xilinx VirtexII 1000 FPGA
- Xilinx PROM to configure the FPGA

The ADC of the SMT370 is connected to the FPGA of the SMT365 via its SHBA. The SMT365 is connected to the DAC of the SMT370 via its SHBB. The SMT365 sends control words to the SMT370 via its comport 0.



Overview

The picture gives an overview of the SDR application. Samples from the ADC are sent to the FPGA on the SMT365 where the processing is implemented. The DAC is either used to output the result of the processing or to output a pattern specified by the DSP.

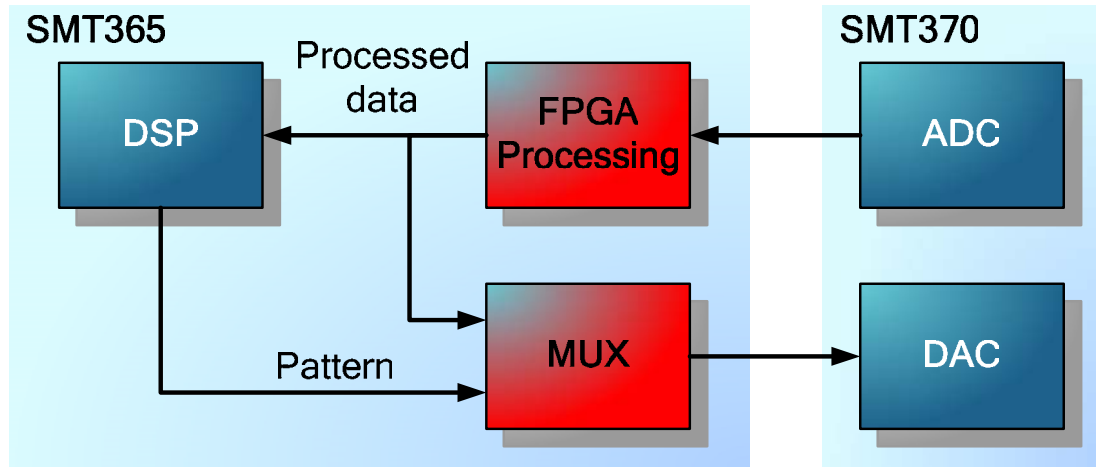


Figure 2 - Overview of the SDR application

The operational mode is selected by a mux as shown.

In **open loop mode** the data processed by the FPGA is not provided to the DAC. In this configuration users typically connect the analog output of the DAC to the analog input of the ADC.

In **closed loop mode** the MUX provides the data processed by the FPGA to the DAC. This assumes a user provided ADC signal.

In both modes a copy of the processed data is presented to the DSP for further processing or display.

A dialog running on the host machine allows configuring the operating mode of the application. This dialog allows the user to change settings of the SMT370.



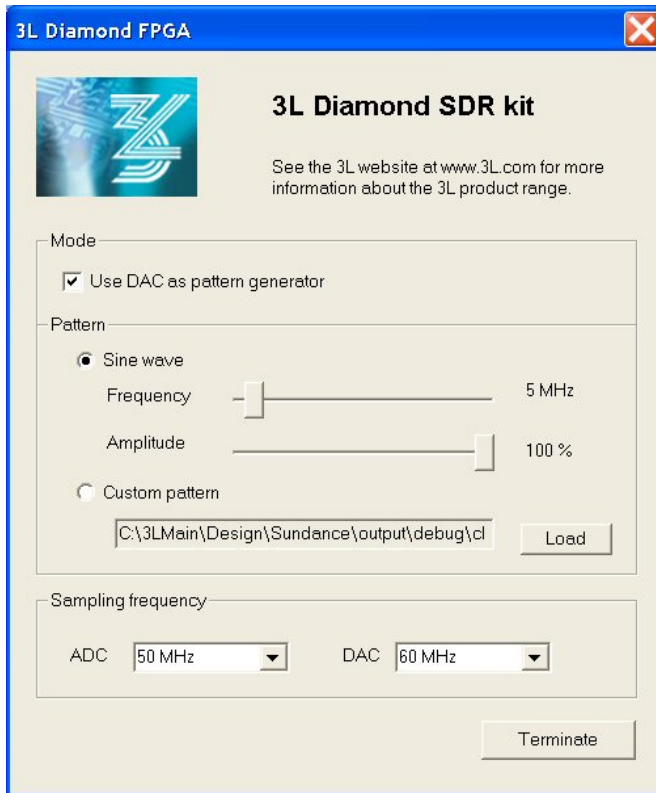


Figure 3 –Graphical User Interface

When “*Use DAC as pattern generator*” is selected, the SDR demo is in the open loop mode. In this case, you can configure the DAC to generate a pattern (sinewave) for which you select the parameters(amplitude and frequency). Or the DAC can generate a waveform loaded from file.

When it is unselected, the application is in the closed loop mode.

The dialog is used to select the sampling frequency of the ADC and of the DAC. ¹

Block diagram

The following diagram shows how the application is made up from a number of tasks connecting by control channels (dashed lines) and data channels (black lines). Tasks in blue execute on the DSP; some of them send graphical output to the host PC for display. Tasks in red execute on the SMT365 FPGA.

¹ Note that data corruption will occur if you choose a sampling frequency greater than 100MHz for the ADC. This is because the SHB cable used to connect the SMT365 and SMT370 can not operate at frequencies above 100MHz.



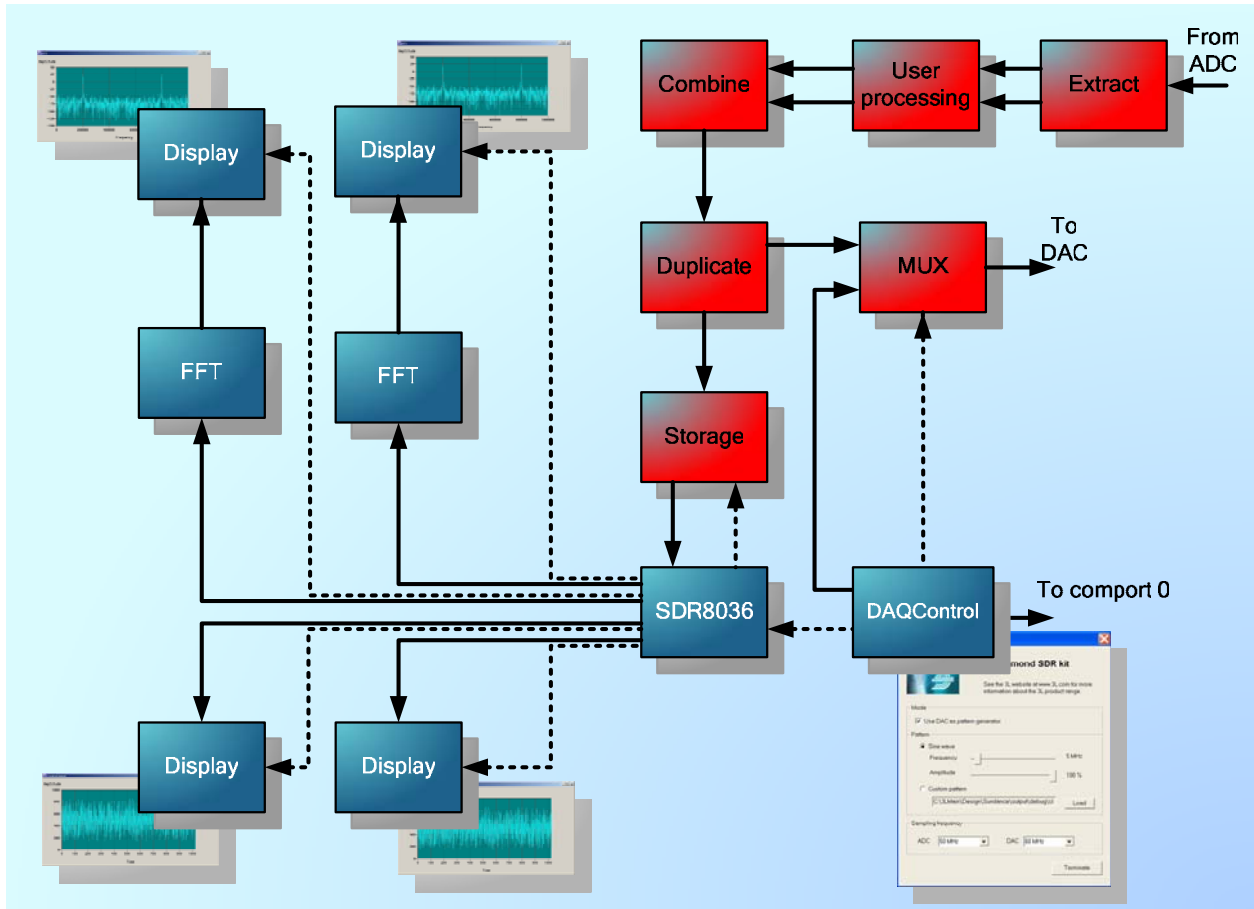


Figure 4 - Block diagram of the SDR application

The *extract* task splits 32-bit values received from the ADC into two streams of 16-bit sample values. (One stream per ADC channel) The user processing task takes these streams, transforms them, and passes the results on to the *combine* task which reconstitutes a single 32-bit stream. This stream is sent to the *duplicate* tasks. The *duplicate* task makes a copy of the data stream and provides the original to the *mux* task and the copy to the *storage* task.

The *mux* task selects the data flow based on the operational mode, open or closed loop, as set by the *DAQControl* task. In addition, the *DAQControl* task configures the SMT370 by sending control data from comport 0 of the SMT365.

The *storage* task is used to capture a contiguous data sample. This is implemented in the FPGA to ensure that the capturing can be done even if the data stream is very fast. In your applications you might choose to omit this task, but then you will need to ensure that the data flow between the FPGA and DSP is slow enough so that data does not get lost.



The **SDR8036** task receives the processed data and sends them to the **FFT** tasks for processing. The results, along with the original processed data, are then displayed. The **SDR8036** task is also responsible for configuring the axis and display properties of the graphs using control channels.

The building blocks

The example directory contains a number of DSP and FPGA tasks. The DSP directory contains the building block tasks executing on the DSP. The FPGA directory contains the building block tasks for the FPGA.

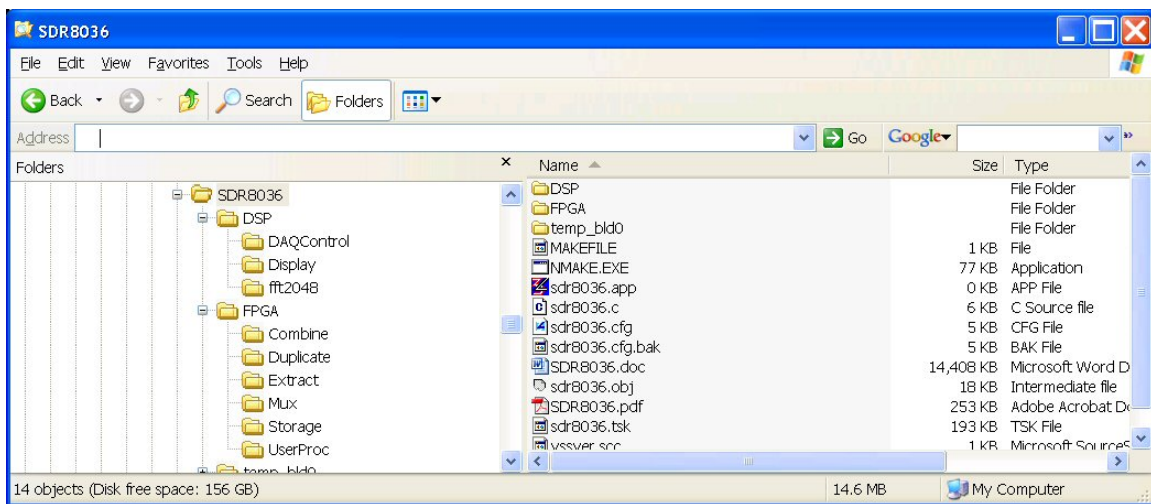


Figure 5 - Directory structure

The following tasks are used to build the SDR application.

FPGA tasks

Extract task

The samples of the two ADCs are interlaced on the 32-bits data arriving on SHBA of the SMT365. The low-order sixteen bits represent one sample from ADCA and the high-order sixteen bits represent one sample from ADCB.

The **extract** task separates these data streams and sends the ADCA samples to output channel 0 and the ADCB samples to output channel 1.



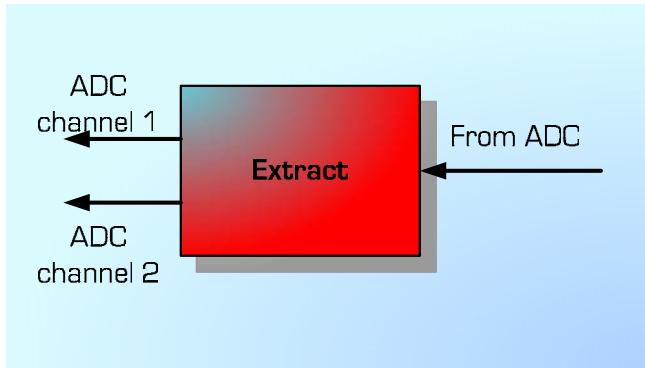


Figure 6 - Extract task

User processing task

This task performs a processing on the data it receives. The task is left empty for you to implement your own processing.

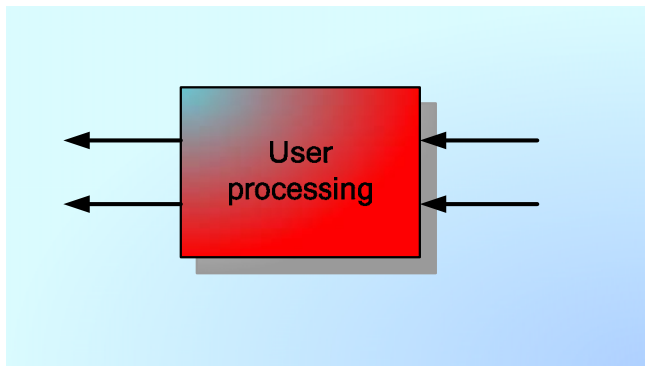


Figure 7 - User processing

Combine task

This task combines the samples to be sent to the DAC over SHBB into a stream of 32-bit values. The samples for DACA are packed in the low-order sixteen bits of each value and the samples for DACB are packed into the high-order sixteen bits.

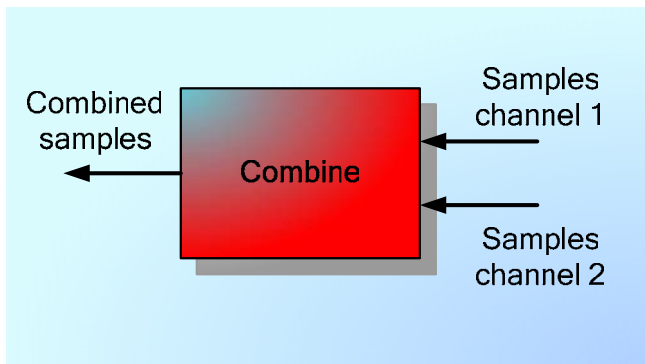


Figure 8 - Combine task



Duplicate task

This task outputs copies of the data from its input channel onto its two output channels.

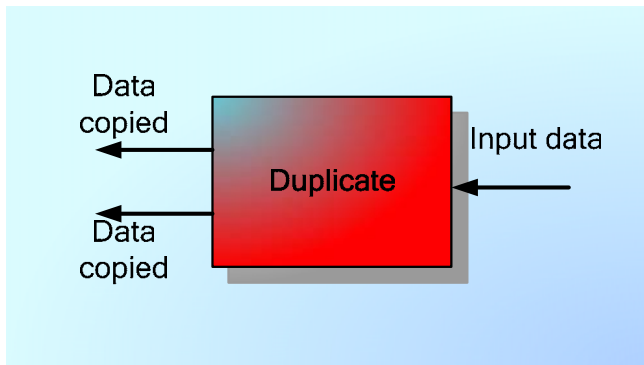


Figure 9 - Duplicate task

MUX task

This task implements a 2→1 channel multiplexer. Input channel 0 selects which of input channels 1 or 2 is presented on output channel 0. Writing a '1' selects input channel 1 and writing a '0' selects input channel 2.

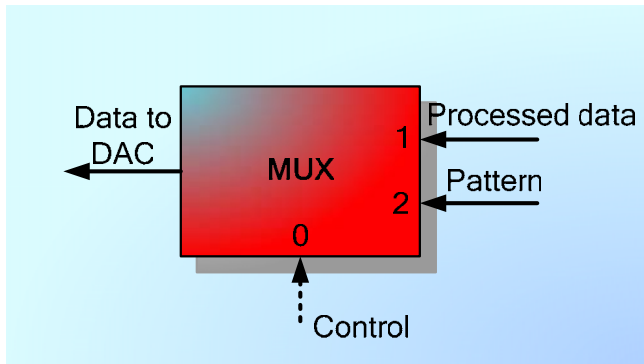


Figure 10 - MUX task

This task is used to select between the real time data coming from the processing block and the data loaded from a pre-existing waveform.

Storage task

This task stores the data coming on its input channel 0 in a 16KB memory. The data stored are sent to output channel 0.



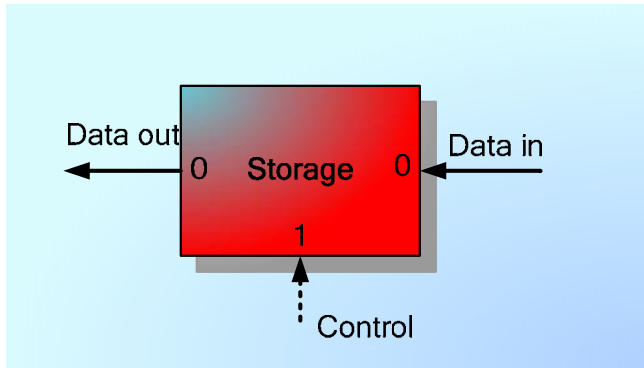


Figure 11 - Storage task

The user specifies the amount of data to be stored using input channel 1. Writing a value to this channel also starts the storage of the data.

DSP tasks

SDR8036 task

This task receives data from the two ADCs as processed by the FPGA and sends them to both the FFT and display tasks. It is also responsible for configuring the various display tasks (scale, etc...) and presenting status information to the user interface.

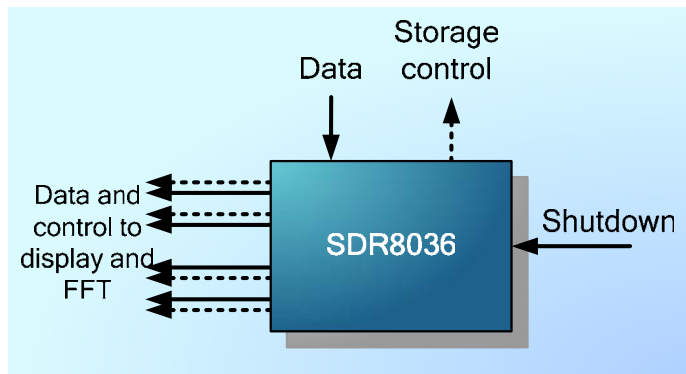


Figure 12 - SDR8036 task

DaqControl task

This task configures the ADCs and the clock synthesizers of the SMT370, and configures and manages the generation of waveforms by the DAC.

A Graphical User Interface allows to change the parameters of the SMT370.



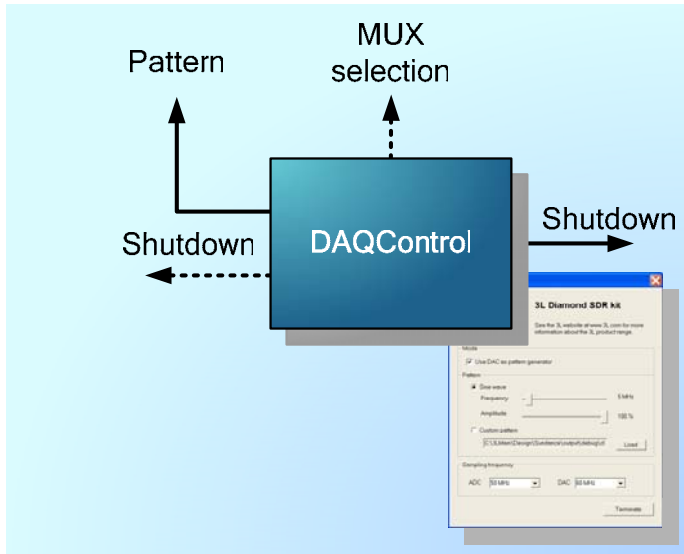


Figure 13 - DAQControl task

Display task

The *Display* task is responsible for displaying graphic data on the host PC. For each instance of this task in the DSP, there is a host graphics window created on the host machine.

The source code for this task is not installed, and the task is distributed as a .task file.

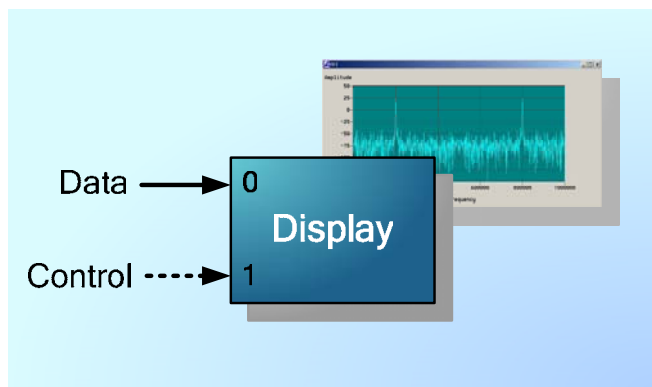


Figure 14 - Display task

The task has two inputs. Channel 0 is the data to display and channel 1 receives control information to configure the properties of the graphic display.



FFT2048 task

The *FFT2048* task expects 2048 floating point values and calculates the FFT of the data. The source for this task is not installed with Diamond FPGA, and the task is provided as a .tsk file.

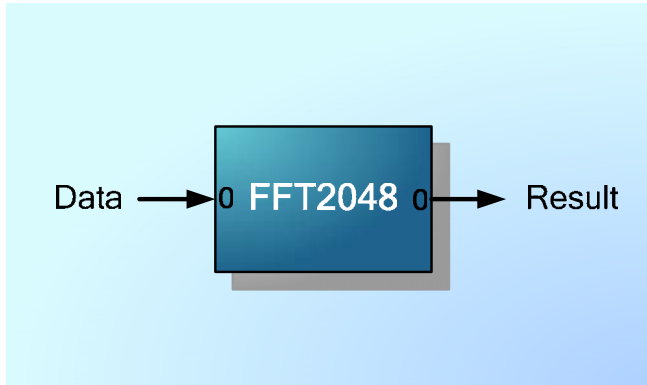


Figure 15 - FFT2048 task

Building and running the example

Make sure that you have Xilinx ISE 7.1.4 or later version as well as Diamond 3.1 or later installed on your PC.

Run “nmake.exe” to build the application; it can take few minutes to compile the application. Double click the file *SDR8036.app* to start the application.

