Alpha Data, System Generator Board Support Package v3.1

ALPHA DATA

Copyright © 2009 Alpha Data Parallel Systems Ltd. All rights reserved.

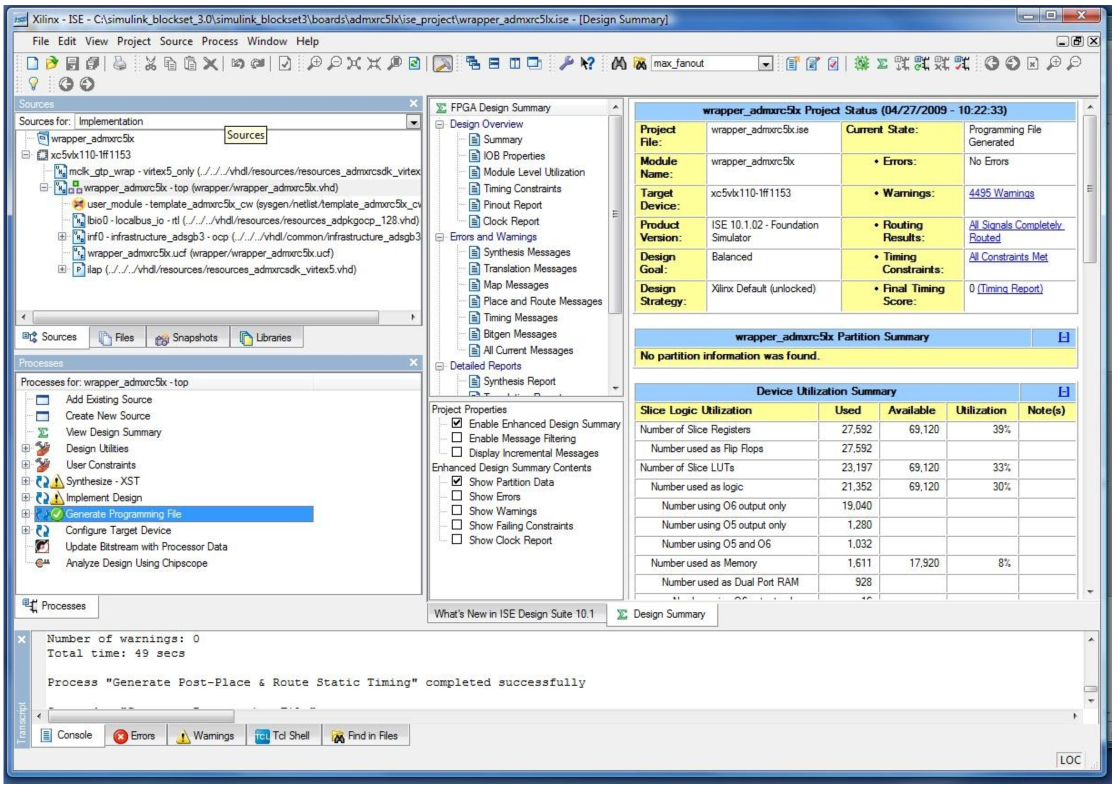
This publication is protected by Copyright Law, with all rights reserved. No part of this publication may be reproduced, in any shape or form, without prior written consent from Alpha Data Parallel Systems Limited.

Alpha Data Parallel Systems Ltd.
4 West Silvermills Lane
Edinburgh EH3 5BD
Scotland, UK

Phone: +44 (0) 131 558 2600
Fax: +44 (0) 131 558 2700
Email: support@alpha-data.com

1 Introduction

The Alpha Data, System Generator, Board Support Package, version 3.1 is a collection of design examples and VHDL code to simplify the development process of using Xilinx System Generator and Xilinx ISE for DSP development. This board support package is ISE based, using Xilinx's main FPGA design GUI for the top level implementation, but the provided examples also contain a template "user module" System Generator project which can be launched from within ISE, allowing the DSP core design to be captured using the Matlab/Simulink environment tools.



Xilinx - ISE - C:\simulink_blockset_3.0\simulink_blockset3\boards\admxrc5lx\ise_project\wrapper_admxrc5lx.ise - [Design Summary]

wrapper_admxrc5lx Project Status (04/27/2009 - 10:22:33)			
Project File:	wrapper_admxrc5lx.ise	**Current State:**	Programming File Generated
Module Name:	wrapper_admxrc5lx	**• Errors:**	No Errors
Target Device:	xc5vlx110-1ff1153	**• Warnings:**	4495 Warnings
Product Version:	ISE 10.1.02 - Foundation Simulator	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0 (Timing Report)

wrapper_admxrc5lx Partition Summary

No partition information was found.

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	27,592	69,120	39%	
Number used as Flip Flops	27,592			
Number of Slice LUTs	23,197	69,120	33%	
Number used as logic	21,352	69,120	30%	
Number using O6 output only	19,040			
Number using O5 output only	1,280			
Number using O5 and O6	1,032			
Number used as Memory	1,611	17,920	8%	
Number used as Dual Port RAM	928			

```
Number of warnings: 0
Total time: 49 secs

Process "Generate Post-Place & Route Static Timing" completed successfully
```

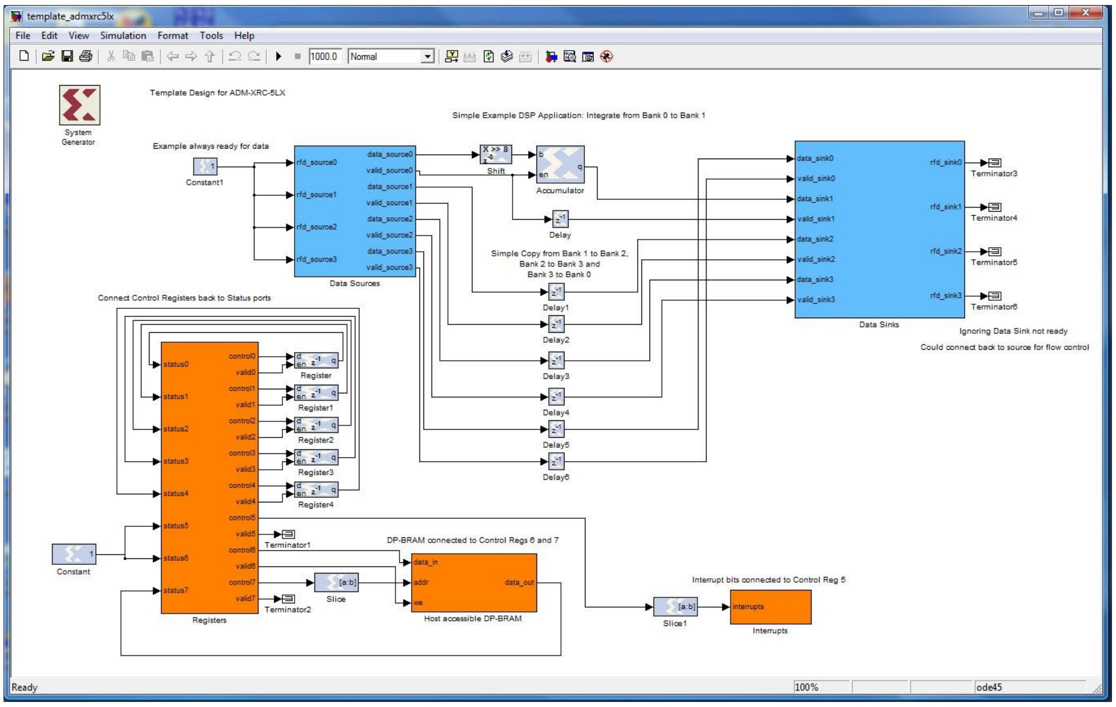
File Edit View Project Source Process Window Help
max_fanout
Sources
Sources for: Implementation
wrapper_admxrc5lx
xc5vlx110-1ff1153
Sources Files Snapshots Libraries
Processes for: wrapper_admxrc5lx - top
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Update Bitstream with Processor Data
Analyze Design Using Chipscope
FPGA Design Summary
Design Overview
Errors and Warnings
Detailed Reports
Project Properties
What's New in ISE Design Suite 10.1
Design Summary
Console Errors Warnings Tcl Shell Find in Files

1.1 Design Flow

The design flow using this board support package is different from previous Alpha Data System Generator Blocksets, as the objective here is now to use ISE as the top level FPGA GUI and Matlab/Simulink/System Generator as Module Design tool for the DSP specific part of the design.

The top level ISE design contains wrapper VHDL and constraints for handling the low level complex IO issues such as bi-directional buses to memory and to the host, as well as the asynchronous clocking of the design modules.

A template design containing a useful set of IO connections is provided to embed the DSP design. The principal dataflow expected is a stream of data from one bank of memory through a DSP pipeline, and stored in another bank of memory.


template_admxrc5lx
File Edit View Simulation Format Tools Help
Template Design for ADM-XRC-5LX
System Generator
Simple Example DSP Application: Integrate from Bank 0 to Bank 1
Example always ready for data
Constant1
Data Sources
Shift
Accumulator
Delay
Simple Copy from Bank 1 to Bank 2, Bank 2 to Bank 3 and Bank 3 to Bank 0
Delay1
Delay2
Delay3
Delay4
Delay5
Delay6
Data Sinks
Terminator3
Terminator4
Terminator5
Terminator6
Ignoring Data Sink not ready
Could connect back to source for flow control
Connect Control Registers back to Status ports
Register
Register1
Register2
Register3
Register4
Registers
Constant
Terminator1
Terminator2
Slice
DP-BRAM connected to Control Regs 6 and 7
Host accessible DP-BRAM
Interrupt bits connected to Control Reg 5
Slice1
Interrupts
Ready
100%
ode45

This dataflow approach should keep the DSP design interfacing simple but with high performance. The wrapper VHDL contains IP to allow the host to upload data to and download data from the memory banks, and to control the data flow.

As well as this data flow interface there are also a few configuration registers for control, a BRAM bank for configuration data (e.g. a table of filter coefficients), and an interrupt register for scheduling host/FPGA interaction.

1.2 Version Support

When developing with System Generator, version compatibility issues are common. With potentially 4 Operating Systems (Windows XP 32 bit, Windows XP 64 bit, Windows Vista 32 bit and Windows Vista 64 bit), many Matlab Releases (currently using 2008a) and many ISE/SysGen releases (currently 10.1) finding a fully working combination can be difficult. This package aims to avoid further compatibility issues by not implementing a library of Simulink blocks with S-function (like previous Blocksets) or even using MEX functions – both of which can easily cause (OS or Matlab version) compatibility problems. The template models will require a Matlab version compatible with the version used to create them (R2008a), but otherwise should prevent few other difficulties.

1.3 Advanced Designs

The provided templates will not cater for all users, but they should provide a reasonable degree of flexibility for users wishing to avoid changing any VHDL.
To achieve many of the more difficulty design tasks, however some modification of the wrapper may be necessary. The project has been structured to make this as simple as possible. If all that is required is the use of a different external clock or the addition of some IO synchronous to the user clock, then only very minor edits to the wrapper VHDL and UCF (constraints file) are required. For more complex IO tasks,

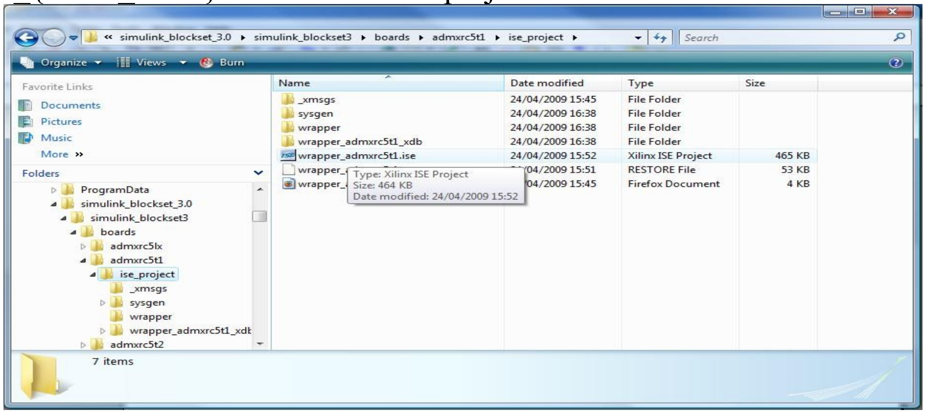
e.g. adding an asynchronous interface to connect to an A2D device or dropping in a high speed Serial IO CoreGen block (Aurora) then while the modifications may be more significant, only the wrapper file need be changed. Adjusting the data widths of the dataflow paths can also be achieved with minor modifications to the VHDL and the template IO blocks. It is not possible to predict every requirement, so the source code for all the wrapper interfacing logic is provided, and can be modified to achieve the desired system performance.

2 Tutorial

This tutorial provides a step-by-step guide to building a simple DSP design using the template designs.

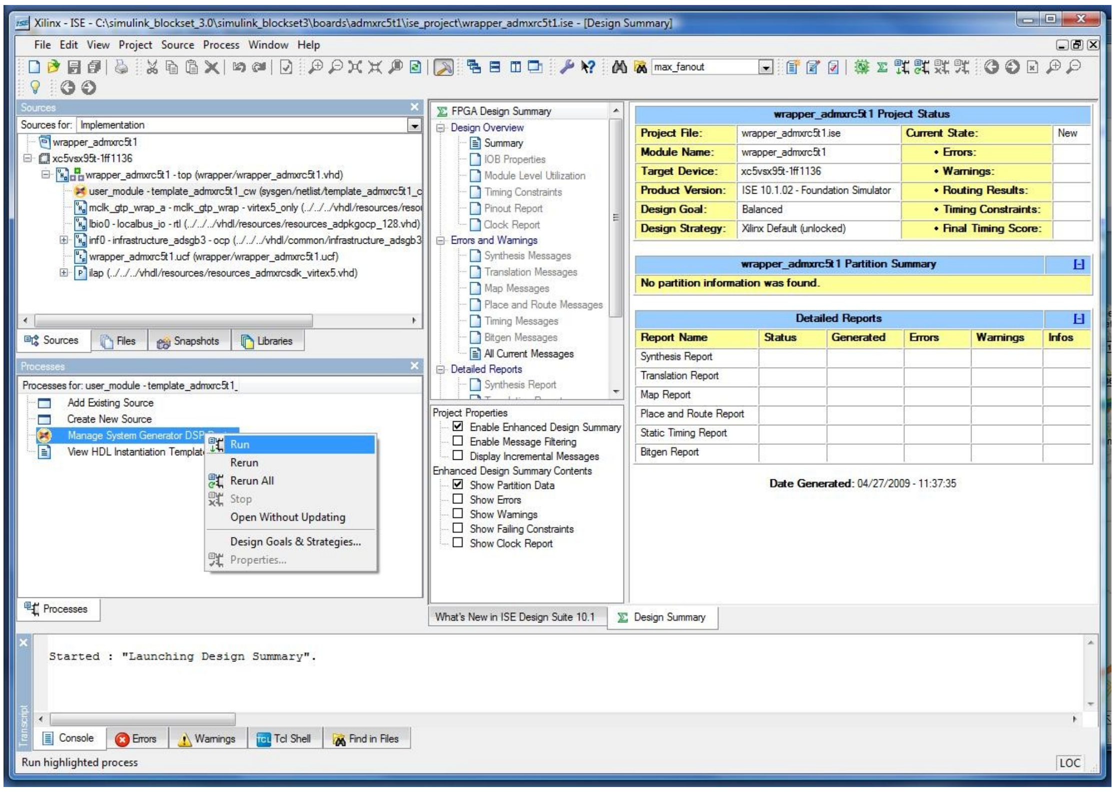
2.1 Getting Started

First open the ISE template project for the card. The projects are located in the boards->[board name]->ise_project directory. Double click on wrapper_{board_name}.ise to launch the project.



Name	Date modified	Type	Size
_xmsgs	24/04/2009 15:45	File Folder	
sysgen	24/04/2009 16:38	File Folder	
wrapper	24/04/2009 16:38	File Folder	
wrapper_admxrc5t1_xdb	24/04/2009 16:38	File Folder	
wrapper_admxrc5t1.ise	24/04/2009 15:52	Xilinx ISE Project	465 KB
wrapper_	/04/2009 15:51	RESTORE File	53 KB
wrapper_	/04/2009 15:45	Firefox Document	4 KB

This will open then ISE Project Navigator Project.


Xilinx - ISE - C:\simulink_blockset_3.0\simulink_blockset3\boards\admxrc5t1\ise_project\wrapper_admxrc5t1.ise - [Design Summary]
File Edit View Project Source Process Window Help
max_fanout
Sources
Sources for: Implementation
wrapper_admxrc5t1
xc5vsx95t-1ff1136
wrapper_admxrc5t1 - top (wrapper/wrapper_admxrc5t1.vhd)
user_module - template_admxrc5t1_cw (sysgen/netlist/template_admxrc5t1_c
mclk_gtp_wrap_a - mclk_gtp_wrap - virtex5_only (../../../vhdl/resources/reso
lbio0 - localbus_io - rtl (../../../vhdl/resources/resources_adpkgocp_128.vhd)
inf0 - infrastructure_adsgb3 - ocp (../../../vhdl/common/infrastructure_adsgb3
wrapper_admxrc5t1.ucf (wrapper/wrapper_admxrc5t1.ucf)
ilap (../../../vhdl/resources/resources_admxrcsdk_virtex5.vhd)
Sources
Files
Snapshots
Libraries
Processes
Processes for: user_module - template_admxrc5t1_
Add Existing Source
Create New Source
Manage System Generator DSP
View HDL Instantiation Template
Run
Rerun
Rerun All
Stop
Open Without Updating
Design Goals & Strategies...
Properties...
Processes
FPGA Design Summary
Design Overview
Summary
IOB Properties
Module Level Utilization
Timing Constraints
Pinout Report
Clock Report
Errors and Warnings
Synthesis Messages
Translation Messages
Map Messages
Place and Route Messages
Timing Messages
Bitgen Messages
All Current Messages
Detailed Reports
Synthesis Report
Project Properties
Enable Enhanced Design Summary
Enable Message Filtering
Display Incremental Messages
Enhanced Design Summary Contents
Show Partition Data
Show Errors
Show Warnings
Show Failing Constraints
Show Clock Report
What's New in ISE Design Suite 10.1
Design Summary
Console
Errors
Warnings
Tcl Shell
Find in Files
Transcript
Run highlighted process
LOC

wrapper_admxrc5t1 Project Status			
Project File:	wrapper_admxrc5t1.ise	**Current State:**	New
Module Name:	wrapper_admxrc5t1	**• Errors:**	
Target Device:	xc5vsx95t-1ff1136	**• Warnings:**	
Product Version:	ISE 10.1.02 - Foundation Simulator	**• Routing Results:**	
Design Goal:	Balanced	**• Timing Constraints:**	
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	

wrapper_admxrc5t1 Partition Summary

No partition information was found.

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

Date Generated: 04/27/2009 - 11:37:35

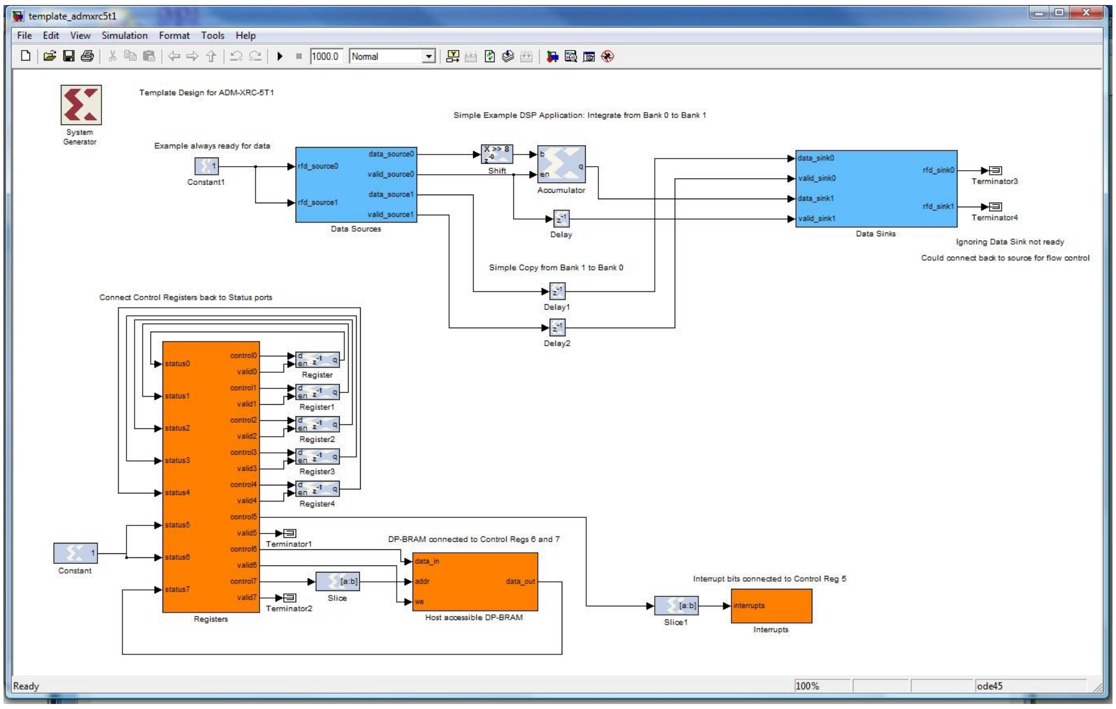
```
Started : "Launching Design Summary".
```

In the Sources window select “user_module – template_admxrc5t1_cw” to select the System Generator sub-module

This will bring up a number of options in the Process Window.

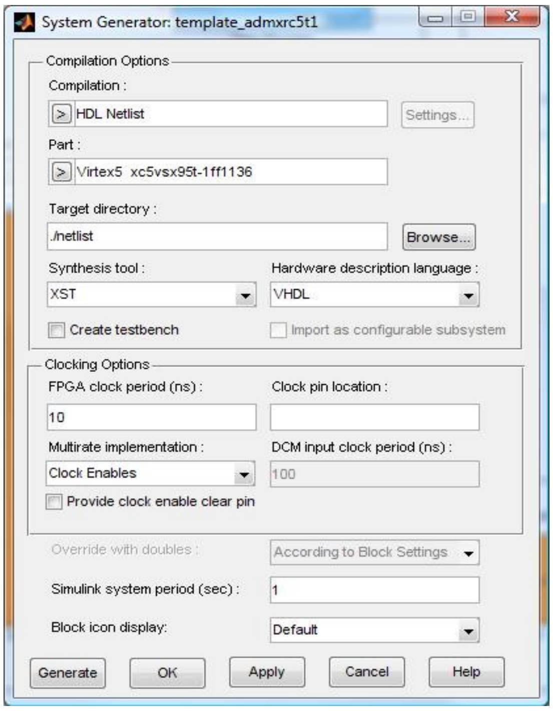
Select “Manage System Generator DSP Design”

Right click on this and select Run to launch Matlab, Simulink and System Generator.


template_admxrc5t1
File Edit View Simulation Format Tools Help
Template Design for ADM-XRC-5T1
System Generator
Simple Example DSP Application: Integrate from Bank 0 to Bank 1
Example always ready for data
Constant1
Data Sources
Shift
Accumulator
Delay
Data Sinks
Terminator3
Terminator4
Ignoring Data Sink not ready
Could connect back to source for flow control
Simple Copy from Bank 1 to Bank 0
Delay1
Delay2
Connect Control Registers back to Status ports
Register
Register1
Register2
Register3
Register4
Registers
Constant
Terminator1
Terminator2
Slice
DP-BRAM connected to Control Regs 6 and 7
Host accessible DP-BRAM
Interrupt bits connected to Control Reg 5
Slice1
Interrupts
Ready
100%
ode45

This will open the Simulink Model containing the template design. This contains 6 special blocks which are required to correctly generate the VHDL module.

The “System Generator” block contains the compilation options for generating the VHDL module for the Simulink model. The “HDL Netlist” option should be used. Note that the FPGA clock period is **not** propagated to the top of the design.


System Generator: template_admxrc5t1

Compilation Options

Compilation : HDL Netlist

Part : Virtex5 xc5vsx95t-1ff1136

Target directory : ./netlist

Synthesis tool : XST

Hardware description language : VHDL

Create testbench

Import as configurable subsystem

Clocking Options

FPGA clock period (ns) : 10

Clock pin location :

Multirate implementation : Clock Enables

DCM input clock period (ns) : 100

Provide clock enable clear pin

Override with doubles : According to Block Settings

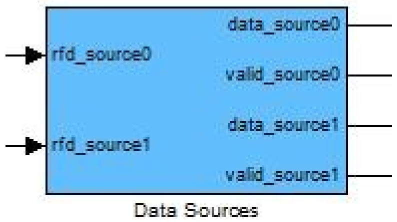
Simulink system period (sec) : 1

Block icon display: Default

The other 5 special blocks are Simulink Sub-systems containing the IO blocks necessary for the generated VHDL to match the component socket in the top level VHDL wrapper in the ISE project.

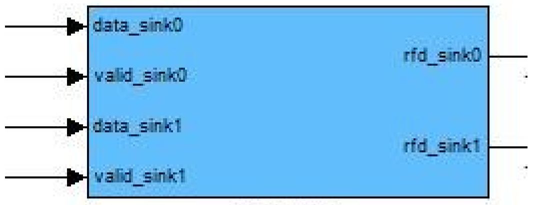
These blocks are:

Data Sources



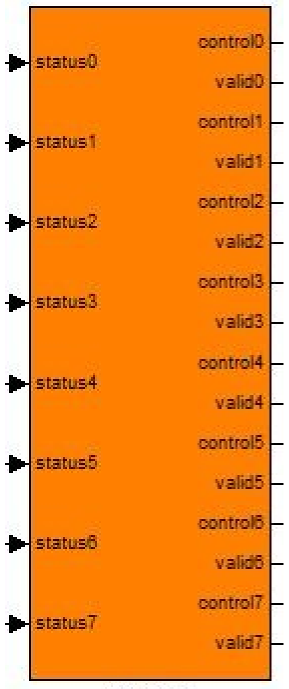
Data Sources

Data Sinks



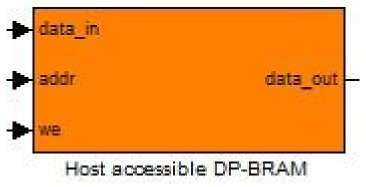
Data Sinks

Registers

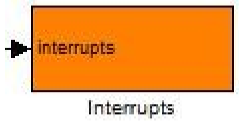


Registers

Dual Port BRAM



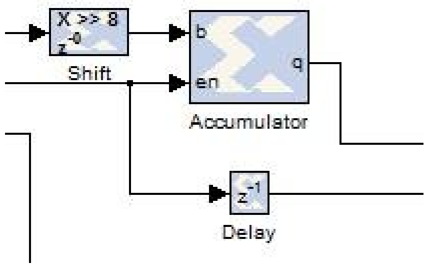
Interrupts



These blocks contain the Gateway In, Gateway Out and assertion blocks required to ensure that System Generator creates the correct module definition.

They also contain a simple Simulink model for simulation. These models (outside of the SysGen signal domain) can be changed to model any signal behaviour required.

The other logic in the template is included to connect up all the signals, however this should be replaced by the DSP design.



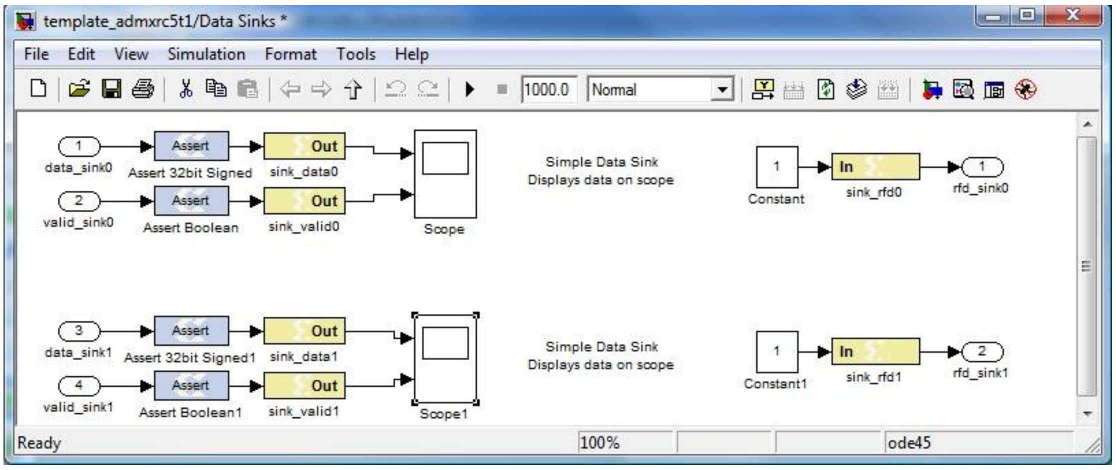
In the example, the data flow interfaces from data_source0 (data read from SDRAM bank #0) to data_sink1 (data written into SDRAM bank #1) are connected by a simple integrator (accumulator) circuit. For a real design obviously a much more sophisticated digital filter could be placed in here to achieve the desired functionality.

Editing of System Generator designs is not covered here, as there are plenty of examples supplied with System Generator.

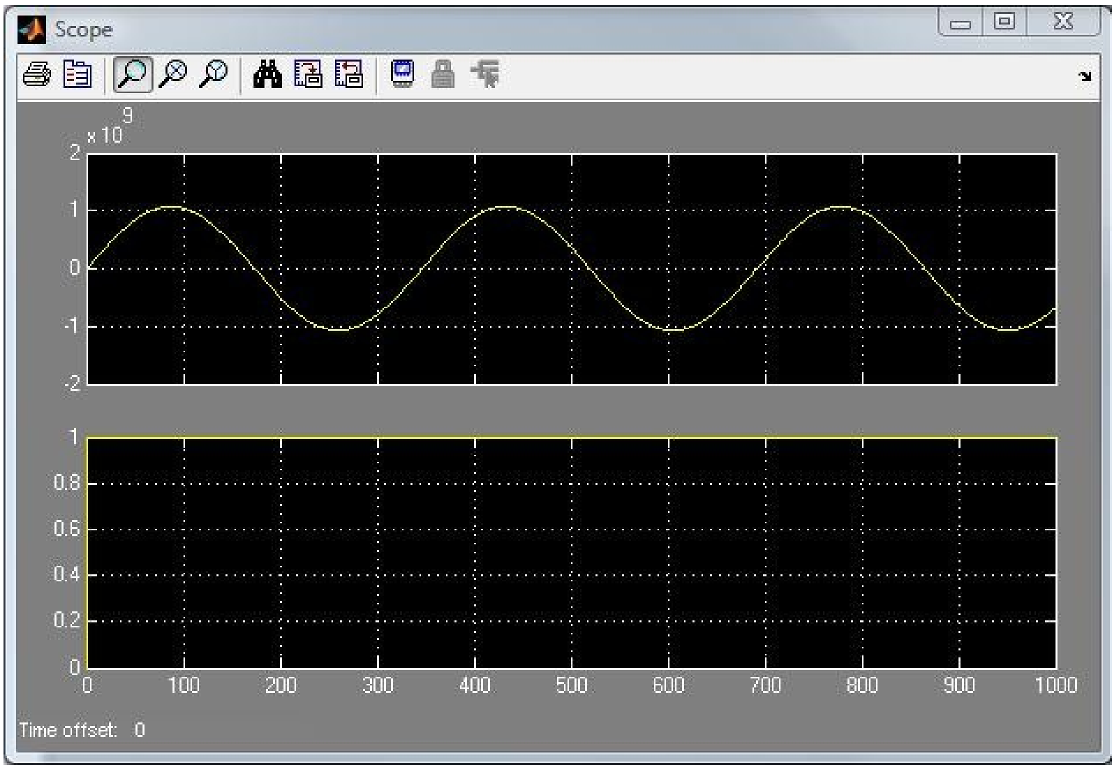
2.2 Simulation

Pressing the "play" button (start_simulation) at the top of the template model, and this will simulate the design for 1000 clock cycles. The templates assume a normalised clock frequency of 1Hz as this simplifies the specification of digital filtering.

If you double click on the data sinks block, you will see that the simulation model does not attempt to emulate memory, but simply displays the output data on a scope.

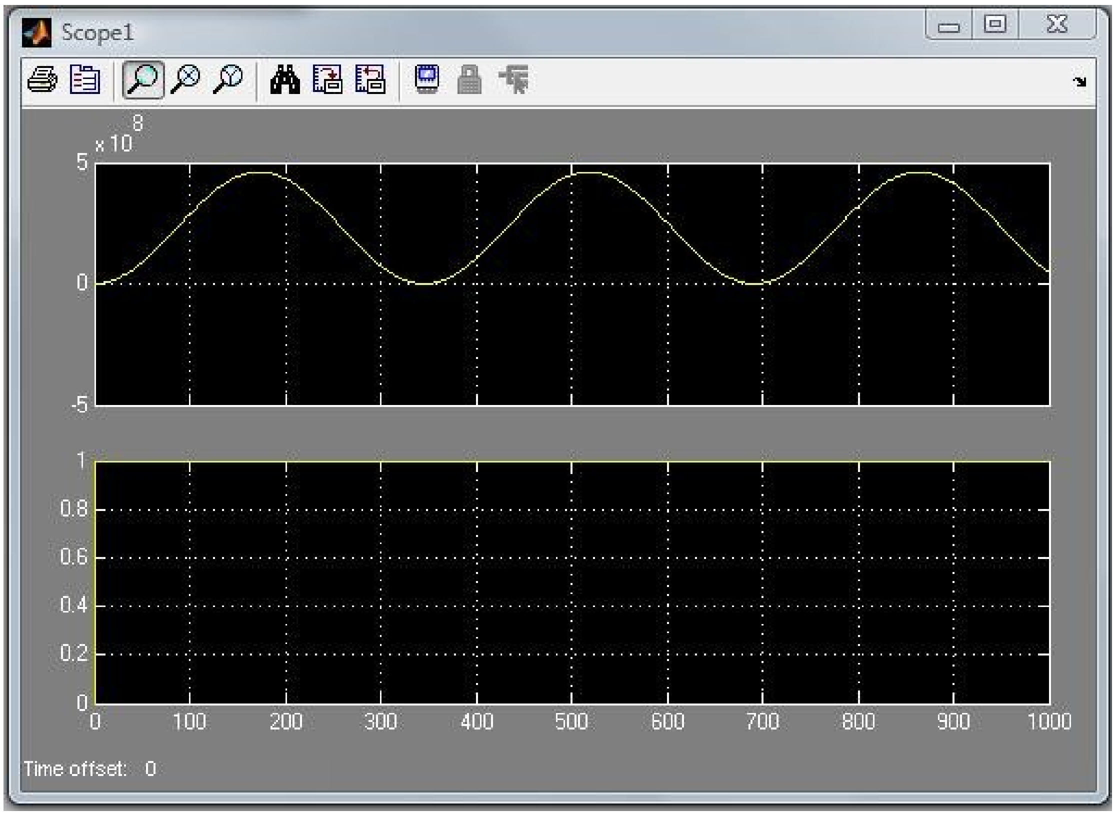


The output on scope 0 is a straight copy of the input from data_source1 (a sine wave).



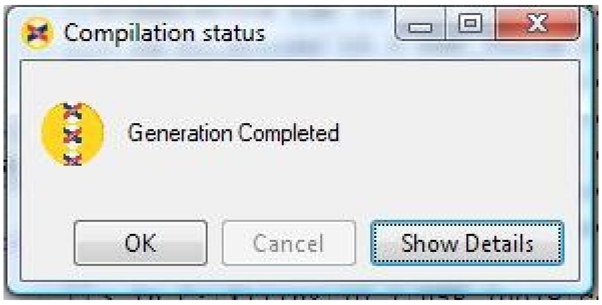
The output on scope 1 is the integrated version of the input from data_source0 (also a sine wave). The output is an offset cosine wave.

The second signal in both scopes is the data valid signal, which is held constant in this example.



2.3 Generating The SysGen Module

Selecting the “Generate” option on the SysGen module will take the template model file and output a VHDL project. This will update and modify the existing sub-project.


Compilation status

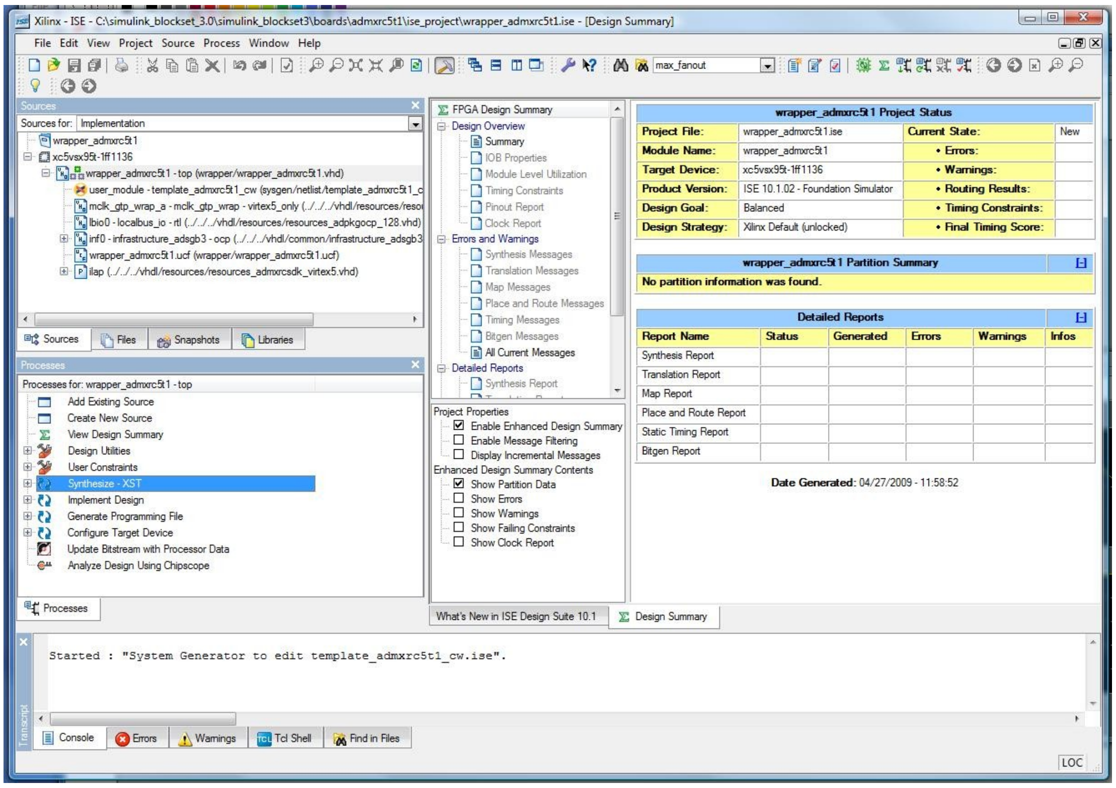
Generation Completed

At the end of this, you will get the generation complete message.

2.4 Synthesis, Implementation and Bitstream Generation

The final stage is the generation of the FPGA configuration file (bitstream).

The Xilinx ISE has to run a number of individual tools. This flow can be stepped through or run to completion.


Xilinx - ISE - C:\simulink_blockset_3.0\simulink_blockset3\boards\admxrc5t1\ise_project\wrapper_admxrc5t1.ise - [Design Summary]
File Edit View Project Source Process Window Help
max_fanout
Sources
Sources for: Implementation
wrapper_admxrc5t1
xc5vsx95t-1ff1136
wrapper_admxrc5t1 - top (wrapper/wrapper_admxrc5t1.vhd)
user_module - template_admxrc5t1_cw (sysgen/netlist/template_admxrc5t1_c
mclk_gtp_wrap_a - mclk_gtp_wrap - virtex5_only (../../../vhdl/resources/reso
lbio0 - localbus_io - rtl (../../../vhdl/resources/resources_adpkgocp_128.vhd)
inf0 - infrastructure_adsgb3 - ocp (../../../vhdl/common/infrastructure_adsgb3
wrapper_admxrc5t1.ucf (wrapper/wrapper_admxrc5t1.ucf)
ilap (../../../vhdl/resources/resources_admxrcsdk_virtex5.vhd)
Sources
Files
Snapshots
Libraries
Processes
Processes for: wrapper_admxrc5t1 - top
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Update Bitstream with Processor Data
Analyze Design Using Chipscope
Processes
FPGA Design Summary
Design Overview
Summary
IOB Properties
Module Level Utilization
Timing Constraints
Pinout Report
Clock Report
Errors and Warnings
Synthesis Messages
Translation Messages
Map Messages
Place and Route Messages
Timing Messages
Bitgen Messages
All Current Messages
Detailed Reports
Synthesis Report
Project Properties
Enable Enhanced Design Summary
Enable Message Filtering
Display Incremental Messages
Enhanced Design Summary Contents
Show Partition Data
Show Errors
Show Warnings
Show Failing Constraints
Show Clock Report
What's New in ISE Design Suite 10.1
Design Summary
Console
Errors
Warnings
Tcl Shell
Find in Files
Transcript
LOC

Project Properties
- [✓] Enable Enhanced Design Summary
- [ ] Enable Message Filtering
- [ ] Display Incremental Messages

Enhanced Design Summary Contents
- [✓] Show Partition Data
- [ ] Show Errors
- [ ] Show Warnings
- [ ] Show Failing Constraints
- [ ] Show Clock Report

wrapper_admxrc5t1 Project Status

Project File:	wrapper_admxrc5t1.ise	**Current State:**	New
Module Name:	wrapper_admxrc5t1	**• Errors:**	
Target Device:	xc5vsx95t-1ff1136	**• Warnings:**	
Product Version:	ISE 10.1.02 - Foundation Simulator	**• Routing Results:**	
Design Goal:	Balanced	**• Timing Constraints:**	
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	

wrapper_admxrc5t1 Partition Summary

No partition information was found.

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

Date Generated: 04/27/2009 - 11:58:52

```
Started : "System Generator to edit template_admxrc5t1_cw.ise".
```

By selecting the wrapper_admxrc5t1 in the sources module, a range of options appears.

The first stage is Synthesis, where the VHDL generated by SysGen is combined with the template VHDL and synthesised into an netlist of FPGA primitives.

This is followed by the implementation stage where timing constraints are added and other netlists may be included. The netlist is then mapped onto the FPGA hardware, placed and routed to produce a mapped FPGA.

This mapped and placed FPGA design is then converted using “Generate Program File” into a configuration bitstream which can be downloaded to the device.

Right click on “Generate Program File” and select run to run all these tasks.

2.5 Testing

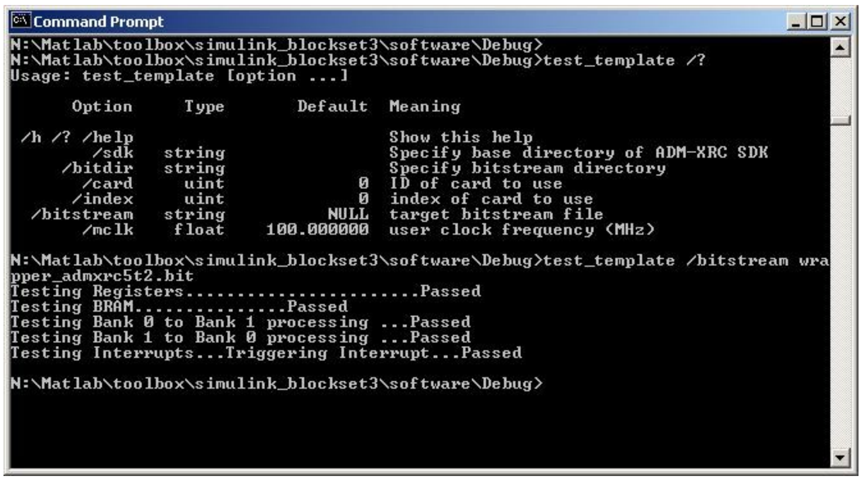
In the software directory, is a test program written in C : test_template.c. This can be compiled using Microsoft Visual C++ to generate an executable.

This program runs 4 different tests on the generated FPGA:

- Registers, a read/write test is performed on the 5 SysGen accessible registers that are looped back
- The BRAM bank is written and read directly by the host. The access from the SysGen Application side is also checked using the connections set up in registers 6 and 7.

- The main dataflow DSP operations are tested for each pair of SDRAM banks, with a simple sawtooth test pattern generated and the output checked against the expected values.
- The interrupt logic is also tested.

A number of options are selectable from the command line, including the card ID or index to allow the selection of an individual card in a system hosting several FPGA boards. The bitstream to upload to the FPGA can be specified, if not the software will use
"..\..\boards\{detected_board_type}\ise_project\wrapper_{detected_board_type}.bit" as the bitstream file name. An error will be reported if the file cannot be found. The user data clock can also be programmed using the /mclk switch.



```
N:\Matlab\toolbox\simulink_blockset3\software\Debug>
N:\Matlab\toolbox\simulink_blockset3\software\Debug>test_template /?
Usage: test_template [option ...]

      Option     Type       Default  Meaning

 /h /? /help                         Show this help
        /sdk   string                Specify base directory of ADM-XRC SDK
     /bitdir   string                Specify bitstream directory
       /card     uint             0  ID of card to use
      /index     uint             0  index of card to use
  /bitstream   string          NULL  target bitstream file
       /mclk    float    100.000000  user clock frequency (MHz)

N:\Matlab\toolbox\simulink_blockset3\software\Debug>test_template /bitstream wra
pper_admxrc5t2.bit
Testing Registers.......................Passed
Testing BRAM...............Passed
Testing Bank 0 to Bank 1 processing ...Passed
Testing Bank 1 to Bank 0 processing ...Passed
Testing Interrupts...Triggering Interrupt...Passed

N:\Matlab\toolbox\simulink_blockset3\software\Debug>
```

3 User Designs

To create a user specific application, the best starting point is the template design. This can be modified to meet the requirements. The 6 special blocks in the Simulink template model should be kept, unless their removal is absolutely necessary. Removing any of “Data Sources”, “Data Sinks”, “Registers”, “Host Accessible DP-BRAM” or “Interrupts” will also require an equivalent modification in the top level VHDL (note that it is the Gateway blocks that must stay here, Simulink Blocks outside of the SysGen domain can be modified to generate the appropriate signal data for simulation purposes.) The System Generator token also needs to be present in the design and should not be modified significantly (note that the clock period specified in here is not used by the top level synthesis.) The following section documents how to modify the design to achieve many common design objectives.

3.1 Creating a simple DSP Data Flow design

To create a simple data flow design, the simplest way is to connect up the data source ports from one SDRAM bank to the inputs of your DSP logic and the outputs to the data sink ports of another SDRAM bank.

It is generally a good idea to connect a FIFO block to the data source ports to buffer the data. The rfd port should be connected to the nfull flag as there is a 2 cycle latency between rfd being de-asserted and data_valid going low.

A FIFO block is also useful for interfacing with IP (e.g. Xilinx FFT Core) which requires a guaranteed continuous burst of data (e.g. 1024 words), but which can also throttle inputs and outputs as required i.e. the start flag of the FFT has to be delayed till the FIFO has the first burst.

The registers (and the BRAM if necessary) can be used to provide addition control and configuration for the DSP algorithm. They can also provide some status information back to the host as well. The interrupts can be used to notify the host of events occurring in the FPGA, although the completion of a data transfer from one bank to another does not require an explicit interrupt as the sources and sink generate their own interrupts (using bits 24-31 of the interrupt register.)

Simulation and validation of your DSP module should be fairly straight forward using the Simulink environment. The simple LUT data sources could be replaced with more complex data, read from the Matlab environment.

The data_sources and sinks are fixed to 32 bits wide. This may be suitable for many situations. Increasing/decreasing the bit width can be achieved fairly trivially within the System Generator Simulink design. For advanced users, it is also possible to modify the top level VHDL and the Gateway blocks to support 8, 16 or 64 bit wide data.

The synthesis flow should be identical to the template tutorial example.

3.2 Changing the Clock Frequency

The user clock frequency can be changed fairly easily. Select the wrapper_{board}.ucf file in the top level project. You should be presented with the option Edit Constraints (text). Alternatively you can edit this file with any text editor.

The last line should be modified, to set the desired period for the user clock.
NET "mclka_ibufg" PERIOD = 5.0ns HIGH 50%;

3.3 Changing the User Clock Input

Changing the clock input is slightly more complex than simply changing the input pin. The user clocks supplied on the Alpha Data boards are differential, so have 2 pins associated with the clock. With the 5LX card, these feed into an IBUFDS and then into a BUFG for global clock routing. With the 5T1 and 5T2 cards, the user clock is a dedicated GTP clock input, and has to be routed through an unused tile (mclk_gtp_wrap module) as well, before going onto a BUFG for global clock routing. If another clock input is desired (such as an external , then the top level input logic must be changed in the wrapper VHDL file wrapper_{board}.vhd (this can be edited in ISE or with a standard text editor), and the associated pins and clock period changed in the ucf file.

3.4 Adding some simple IO

Simple IO can be added to the System Generator design with few changes. For example, a simple data capture system can be easily designed by connecting the IO pins into a data sink block, possibly with a register used to control the capture. (although the internal registers in the data_sink module also provide some capture control)

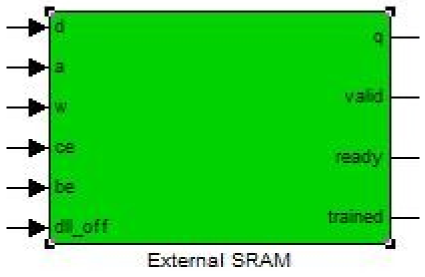
In the Simulink Model, a System Generator Gateway in block can be added.

When synthesizing, the wrapper_{board}.vhd now must also be modified, as there will be an extra port on the top level of the output generated by System Generator. This port should be added at the entity declaration for the wrapper (to create the port at the FPGA top level) and onto the template_{board}_cw component declaration and the instantiation, with signals connected from the instantiation to the top level ports.

Finally, the UCF file wrapper_{board}.ucf will need to have the pin location constraints and any other IO standard information (voltage levels, drive currents, speed (FAST/SLOW)) added for that port.

3.5 SRAM Support

Some boards also have DDR2_SRAM chips fitted. These devices are generally much smaller capacity than DDR2_SDRAM. They are however better suited for non-burst applications, and have much better random access, and read/write turnaround performance than similar speed SDRAM. To support these devices from within System Generator, additional templates have been provided with different wrapper and infrastructure files. These memory banks are only connected to the System Generator User Application and cannot be accessed directly by the host.



Each SRAM has ports for write data (d - 64 bits), address (a – 18 bits), write (w – boolean), command enable (ce – boolean), byte enable (be – 8 bits), disable SRAM dll (dll_off -boolean), read data (q – 64 bits), read data valid (valid – boolean), command FIFO able to accept data (ready – boolean) and SRAM IO delays trained (trained – boolean).

The actual SRAM is clocked at the fixed memory clock frequency. Therefore an asynchronous buffer sits between the user clock domain signals and the actual SRAM interface. This will increase the latency on any read commands. Note that since the USER clock is programmable and runs completely asynchronously from the memory clock, it is impossible to provide an accurate simulation model. The provided Simulink memory model is deliberately not accurate (writes only output to a scope, reads are from a lookup table) to emphasize this. This could be replaced with a more complex model if limited accuracy modelling of data storage and recovery is required.

An additional software program test_sram_template.c is provided to test the SRAM functionality.

3.6 Adding Complex IO and IP

More complex IO can also be handled in a similar manner. Typically this IO will require some HDL interfacing between the simple data flow types of Simulink and the IO pins which may be bi-directional or possibly high speed serial pins, which cannot be supported by System Generator.

For example high speed serial interfacing using Xilinx Aurora Cores can be quite tricky. Xilinx CoreGen can be used to provide some basic reference IP. This will include some complex clocking modules as well as wrappers from the GTP devices. This IP must be embedded in the design at the wrapper level, by modifying wrapper_{board}.vhd and wrapper_{board}.ucf. Only the simple parallel data port interfaces will connect to the System Generator sub-system, and the user clock may have to be fixed by the Aurora logic.

Adding complex IP within the System Generator design presents a number of problems. Only IP which has a specified interface is supported, ruling out most verified IP without minor changes or wrapping. Bi-directional ports are not supported. Simulation is either by, independently developed simulation S-functions (which may not accurately model the HDL) or by ModelSim plug in co-simulation of the model. Because of these limitations, when handling complex IP and IO, the best place for this complexity is in the top level HDL FPGA wrapper, with System Generator used only for the DSP sub-system.

3.7 Changing the data source and data sink widths

These widths can be changed fairly simply in the VHDL top level wrapper file, by modifying the constants:

```
 constant sink_data_width        : natural           := 32;
 constant source_data_width      : natural           := 32;
```

Other acceptable values are 8,16,64 and 128.

The component declaration for template_{board}_cw also needs to be changed to match the new widths.

In the Simulink Model, the Gateway blocks will also need to be changed to match the new data width.

4 Application Register Map

The generated FPGA design has the following areas mapped into a 4MB window on the PCI bus. Accessing this directly from the host using processor IO read and writes is relatively slow. The SDRAM banks are not mapped into this window. They are accessed indirectly using DMA functions which transfer large chunks of data between the SDRAM and the host in a highly efficient manner. Detailed examples of how to write software to interact with Alpha Data FPGA cards can be found in the ADM-XRC-SDK.

4.1 User Application Registers (0x080000-0x08001C)

The 8 user application registers are mapped into the second 512k Window in the 4MB

Address	Function
0x080000	User Register #0
0x080004	User Register #1
0x080008	User Register #2
0x08000C	User Register #3
0x080010	User Register #4
0x080014	User Register #5
0x080018	User Register #6
0x08001C	User Register #7

4.2 User Application DP-BRAM (0x100000-0x1007FF)

The User Application dual-port Block RAM occupies 512x32 bit words starting at location 0x100000 (the 3rd 512k Window in the 4MB FPGA Space)

4.3 Data Sinks (0x180000-0x183FFF)

The Data Sinks are mapped into the 4th 512k window. The Data Sink for each SDRAM has a register base address of 0x1000*(SDRAM bank number)

Address	Function
0x180000	Data Sink SDRAM Bank 0 Registers
0x181000	Data Sink SDRAM Bank 1 Registers
0x182000	Data Sink SDRAM Bank 2 Registers
0x183000	Data Sink SDRAM Bank 3 Registers

Offset from these Base Addresses, each data sink has a number of control registers:

Offset	Function
0x00	Start Address of Data
0x04	Word Limit – Number of (128 bit) Words to Record
0x08	Interrupt Limit – Interrupt generated after this number of (Data

	Width = 32 bit) words have been received.
0x0C	CONTROL/STATUS Register: Bit 0: Write 1 to Start, Read indicates if Running Bit1: Enable Continuous Operation, Once Word Limit is reached the address pointer will reset to the start address Bit 2: Clear FIFO, Word Count, and stop data capture. Bit 4: Enable Single Step Operation
0x10	Current Address (Read Only)
0x14	Current Memory Write Word Count (Read Only)
0x18	Current Interrupt Data Word Count (Read Only)
0x20	Single Step Least Significant Word
0x24	Single Step Most Significant Word

4.4 Data Sources (0x200000-0x203FFF)

The Data Sources are mapped into the 4th 512k window. The Data Source for each SDRAM has a register base address of 0x1000*(SDRAM bank number)

Address	Function
0x200000	Data Source SDRAM Bank 0 Registers
0x201000	Data Source SDRAM Bank 1 Registers
0x202000	Data Source SDRAM Bank 2 Registers
0x203000	Data Source SDRAM Bank 3 Registers

Offset from these Base Addresses, each data source has a number of control registers:

Offset	Function
0x00	Start Address of Data
0x04	Word Limit – Number of (128 bit) Words to Output
0x08	Interrupt Limit – Interrupt generated after this number of (Data Width = 32 bit) words have been output.
0x0C	CONTROL/STATUS Register: Bit 0: Write 1 to Start, Read indicates if Running Bit1: Enable Continuous Operation, Once Word Limit is reached the address pointer will reset to the start address Bit 4: Enable Single Step Mode
0x10	Rate : If set, Data will be output and Valid asserted every “x” clock cycles.
0x20	Single Step Least Significant Word
0x24	Single Step Most Significant Word

4.5 LB2OCP Bridge Registers (0x000000-0x07FFF8)

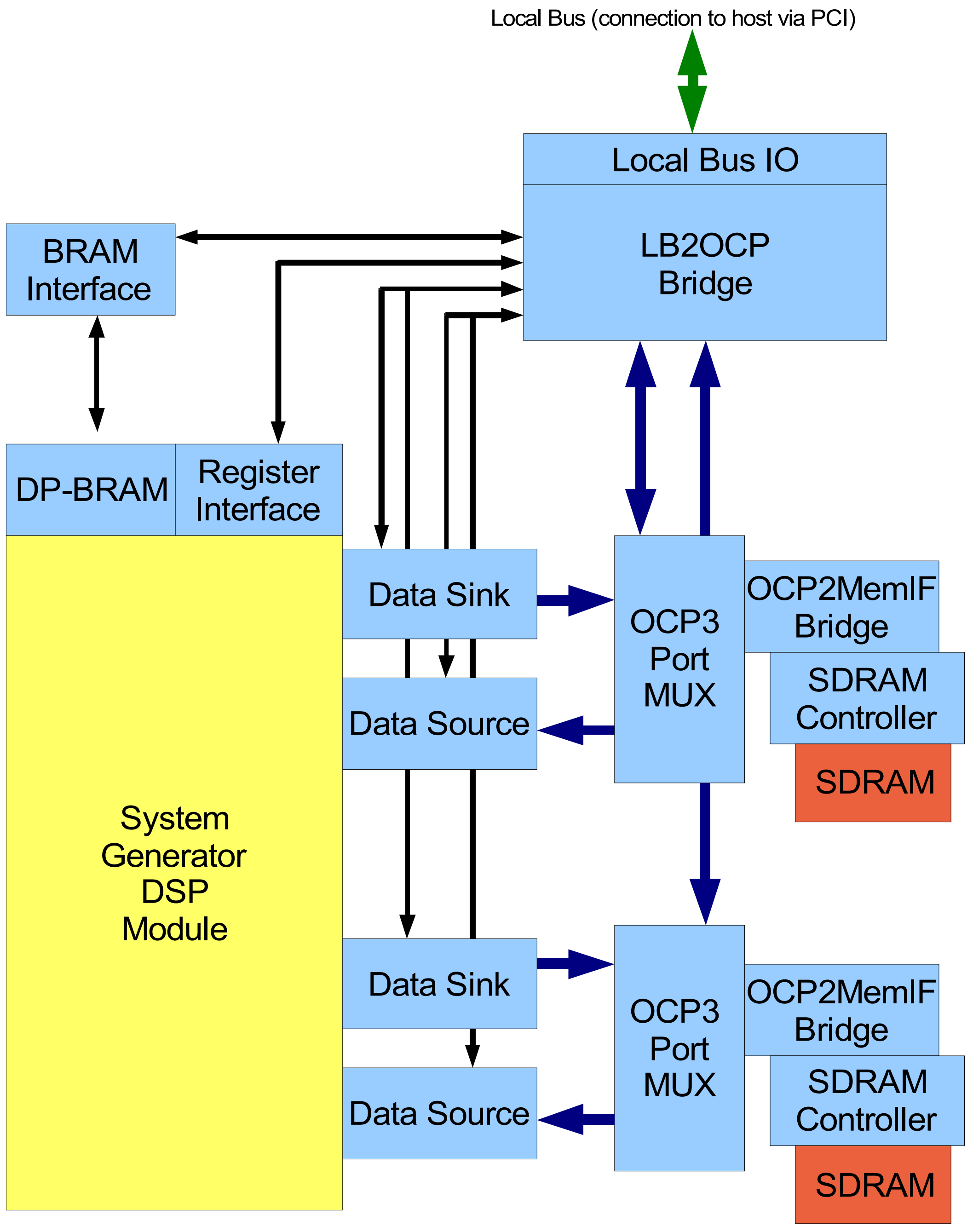
The following registers are used in the Localbus2OCP bridge at the core of the infrastructure module.

Address	Function
0x000000	Demand Mode DMA channel 0 Xfer Size (Engine #0 - Write)
0x000004	Demand Mode DMA channel 1 Xfer Size (Engine #0 - Read)
0x000008	Demand Mode DMA channel 2 Xfer Size (Engine #1 - Write)
0x00000C	Demand Mode DMA channel 3 Xfer Size (Engine #1 - Read)
0x000010	Not Used
0x000014	Register Space #1 Page
0x000018	Register Space #2 Page
0x00001C	Register Space #3 Page
0x000020	Register Space #4 Page
0x000024	Register Space #5 Page
0x000028	Register Space #6 Page
0x00002C	Register Space #7 Page
0x000030	DMA Engine #0 Write Target/Burst Length
0x000034	DMA Engine #0 Write Row Size
0x000038	DMA Engine #0 Write Number of Rows
0x00003C	DMA Engine #0 Write Column Increment
0x000040	DMA Engine #0 Write Start Address
0x000044	DMA Engine #0 Read Target/Burst Length/FGSG
0x000048	DMA Engine #0 Read Row Size
0x00004C	DMA Engine #0 Read Number of Rows
0x000050	DMA Engine #0 Read Column Increment
0x000054	DMA Engine #0 Read Start Address
0x000058	DMA Engine #1 Write Target/Burst Length
0x00005C	DMA Engine #1 Write Row Size
0x000060	DMA Engine #1 Write Number of Rows
0x000064	DMA Engine #1 Write Column Increment
0x000068	DMA Engine #1 Write Start Address
0x00006C	DMA Engine #1 Read Target/Burst Length/FGSG
0x000070	DMA Engine #1 Read Row Size
0x000074	DMA Engine #1 Read Number of Rows
0x000078	DMA Engine #1 Read Column Increment
0x00007C	DMA Engine #1 Read Start Address
0x000080	IRQ Enable Register
0x000084	IRQ Clear Register
0x000088	System Status Register
0x00008C	IRQ Status Register
0x000090	Demand Mode DMA channel 0 Xfer Left (Engine #0 - Write)
0x000094	Demand Mode DMA channel 1 Xfer Left (Engine #0 - Read)
0x000098	Demand Mode DMA channel 2 Xfer Left (Engine #1 - Write)
0x00009C	Demand Mode DMA channel 3 Xfer Left (Engine #1 - Read)
0x0000A0	DMA Engine #0 Write Status LSW
0x0000A4	DMA Engine #0 Write Status MSW
0x0000A8	DMA Engine #0 Read Status LSW
0x0000AC	DMA Engine #0 Read Status MSW
0x0000B0	DMA Engine #1 Write Status LSW

0x0000B4	DMA Engine #1 Write Status MSW
0x0000B8	DMA Engine #1 Read Status LSW
0x0000BC	DMA Engine #1 Read Status MSW
0x07F800-07FFFC	Information ROM
0x080000-0FFFFC	Register Space #1
0x100000-17FFFC	Register Space #2
0x180000-1FFFFC	Register Space #3
0x200000-27FFFC	Register Space #4
0x280000-2FFFFC	Register Space #5
0x300000-37FFFC	Register Space #6
0x380000-3FFFFC	Register Space #7

These registers are best accessed using the LB2OCP.C and LB2OCP.H software functions.

5 Top Level Block Diagram



The block diagram shows the location of the DSP module in the system design, along with the main modules used to build the rest of the system (for handling host transfers and multi-ported SDRAM control)

6 Single Step Testing

It is sometimes desirable to simply test the DSP module operation from within the Matlab or Simulink environment. To enable this, single-step operation is available for the data source and data sink ports. In this mode, instead of bursting data from or to and SDRAM bank, a single register is used to generate a single word of data on the data_source and strobe the valid line for 1 clock cycle. The data_sinks will capture the last valid word and this can be read from a register.

Note for this to work as a single step operation, the valid signals must be used to enable the DSP operation. Note that the clock still runs freely.

In the directory single_step, the Matlab function run_single_step.m provides a Matlab script for pushing a Matlab array of data into the data sources a word at a time and reading the output a word at a time.

This function has the following inputs:
card_index : PCI enumeration index of the ADM-XRC-xxx board
bitstream : the bitstream generated by ISE
user_clk : the user clock frequency
no_sinks : number of data sinks to monitor
source_data : a [number_of_samples x number_of_sources] input signal array
registers : an array of up to 8 register initialisation values

The function outputs a [number_of_samples x no_sinks] array.

Two Simulink S-Functions and example Models are also provided to allow single step processing of data using a Simulink data source and output.

The card_index, bitstream, user_clock frequency and register initial values must be set in the Model Parameters.

These functions all use the ADMXRC Matlab Toolbox Functions to access the hardware.

The S-Functions are provided in Matlab M-file format to allow user customisations such as initialising the BRAM, or interactive register updates.