XRC Co-simulation Library v2.0

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 XRC Co-simulation Library is a Simulink Blockset designed for use in conjunction with Xilinx System Generator 6.3 and Matlab Release 14.1. This blockset enables System Generator Models to be Co-simulated using Alpha Data ADM-XRC-II-L, ADM-XRC-II, ADM-XRC-II-Pro-Lite (ADM-XPL), ADM-XRC-II-Pro (ADM-XP), ADP-DRC-II and ADP-WRC-II FPGA reconfigurable computing cards. The System Generator design methodology incorporating Co-simulation allows a very rapid move from Simulation in Simulink to Verification on Hardware. This design methodology can be used incrementally, with Co-simulated Hardware in the Loop modules used to accelerate the development and simulation of larger designs. This new release includes a number of updates to support external shared memory blocks and interfaces.

2 New features

The main new feature included in this release is support for external memory devices on the ADM-XRC-II, ADM-XPL and ADM-XP. The co-simulation models of these devices are simple to use and both SRAM and DRAM devices are supported with a similar consistent interface. These modules interface with Xilinx Shared Memory blocks within the co-simulation allowing high speed DMA transfers from the Simulink model to the external memory.

External general purpose IO ports are also supported, providing bi-directional connections to front and read panel pins on the ADM-XRC-II, ADM-XPL and ADM-XP.

3 Installation Instructions

The XRC Co-simulation Library is provided as a zip file and a Matlab installation script:

xrc_cosim_blockset.zip
setup_cosim.m

1) Open Matlab

2) Change Directory to where xrc_cosim_blockset.zip and setup_cosim.m have been downloaded.

3) Type setup_cosim

4) Quit Matlab

4 Basic use of the Library

Once installed, board options for the ADM-XRC-II, ADM-XRC-II-Lite, ADM-XRC-II-Pro-Lite, ADP-DRC-II, ADP-WRC-II will appear in the Compilation->Hardware In The Loop menu of the System Generator GUI (accessed by double clicking on the System Generator token in your design). You can select the correct Alpha Data board and the correct FPGA specification form this GUI.

Now when Generate is pressed in the System Generator GUI, the VHDL is still created as normal, but afterwards, the Xilinx (and possibly 3rd party) synthesis tools are run to create a bitstream for the device and board selected. After this is finished, a new library containing a hardware cosimulation block will appear, as shown in figure 2.

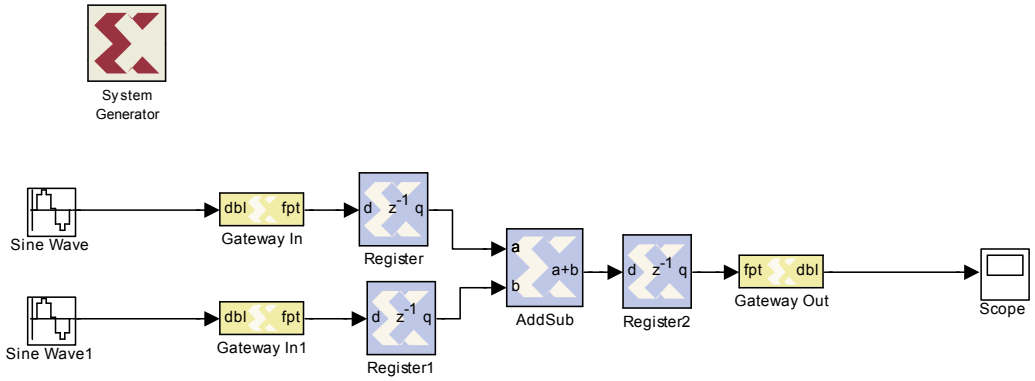
Figure 1: Hardware Cosimulation Block

This block can be inserted into a System Generator Simulink design in place of the model synthesized, for testing and verification or even as a simulation component, to accelerate the simulation of a higher level design. The block has 3 parameters, specifying the clock mode, free running frequency, the generated bitstream and which Alpha Data board (if multiple boards are installed in the system) to use. The maximum recommended frequencies for co-simulation designs for each board are 50MHz for ADM-XRC-II, 62 MHz for ADM-XPL and 58MHz for the ADM-XP. The block also has some parameters specifying the External Memory access modes.

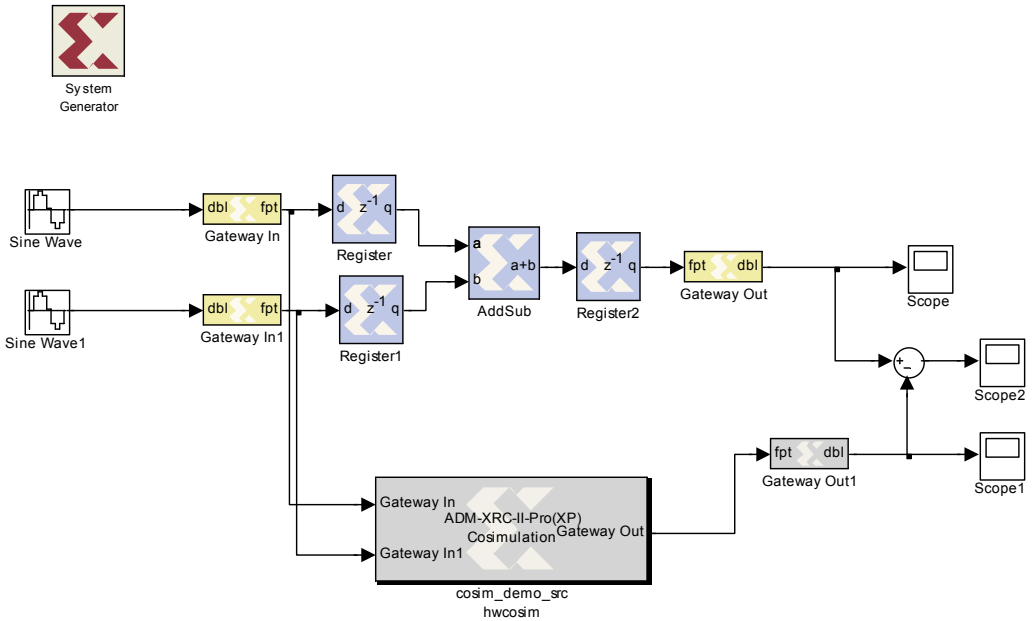
5 A Simple Example

A simple example for demonstrating how to use the cosim blockset is provided in: `%MATLAB%\toolbox\alphadata\xrc_cosim_blockset\examples\cosim_demo\cosim_demo_src.mdl`.

This example adds two sine waves together, and displays the outputs on a scope.



A second model cosim_demo.mdl is provided to show how to insert the Hardware in the Loop block into a simulation, and compare the results:

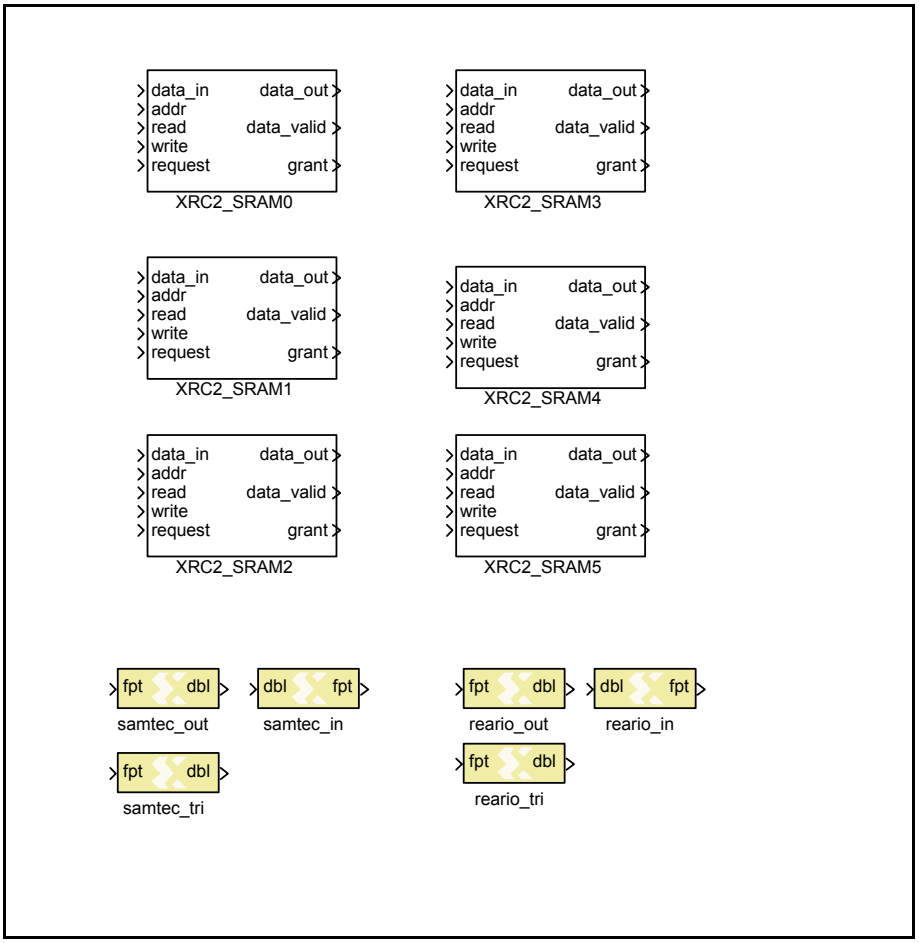


Note that the conversion between doubles and Fixed Point data types is handled using the Gateway blocks. The cosimulation block can accept doubles as inputs, but this may cause slight mismatches between Simulation and hardware in the loop results.

6 ADCOSIM Library

A new ADCOSIM library is provided containing external memory and IO blocks. Note that due to the single stepped nature of Cosimulation, it is not possible to use the Alpha Data Applications Blockset. The ADCOSIM library can be accessed through the Simulink library browser.

6.1 ADM-XRC-II Blocks



For the ADM-XRC-II, 6 SRAM memory blocks are provided. These have the usual read, write, address (20 bit for 4Mb) and data lines (32 bit wide). They also have a data valid line to indicate when a read output is valid. Generally data will be valid 4 clock cycles after a read, however to maintain compatability with other RAM blocks the data_valid line should always be used to qualify data as the 4 clock cycle response is not guaranteed.

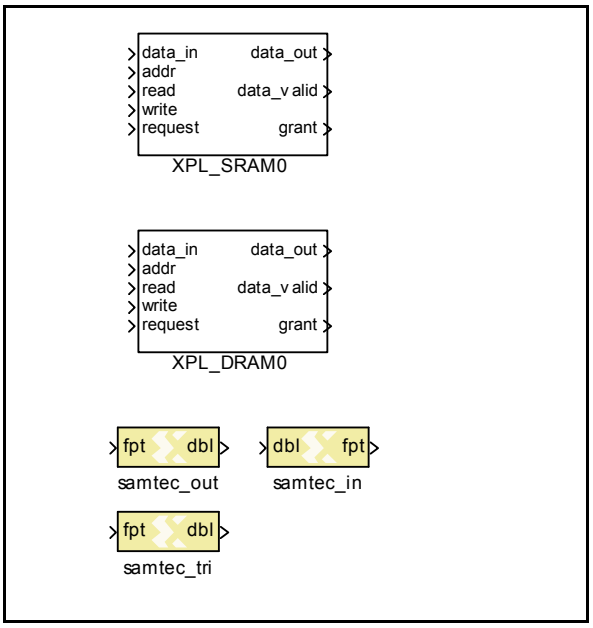
In addition to the usual signals, the memory blocks also have request and grant lines, to allow interaction with the shared memory. The FPGA process must assert a request to gain control of the memory and will not be able to access it until grant goes

high. By default if no other process tries to access the shared memory, the grant signal will always go high after request is asserted.

IO blocks are provided for the front and rear IO connectors. 64 pins are available on the Pn4 reario connector. These can be configured as inputs or outputs through the reario_tri port. N.B. you need to include reario_tri and drive it with a 64 bit wide signal to use the reario_in and/or reario_out ports. Drive bits with a constant '1' to set the pin as an input or to tri-state it. Drive with a constant '0' to enable the pin as an output. Note that reario_out also needs to be driven with a 64 bit wide signal, even if some of the pins are inputs. These signals will be optimised away in the implementation phase.

The samtec_in, samtec_out and samtic_tri pins provide a similar 156 bit wide port connection to the XRM connector. You will need to consult with the XRM module manual to determine which XRM pin connects to the SAMTEC pin used. Note that on the XRC-II Samtec Pins 88-94 are not connected.

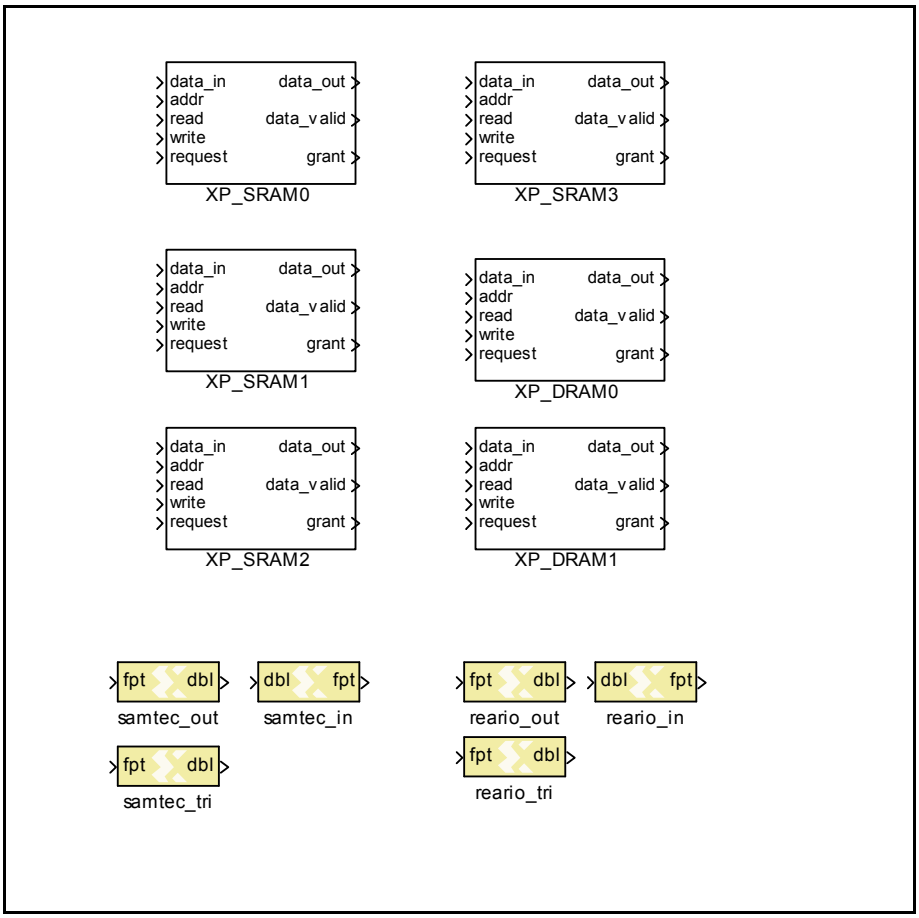
6.2 ADM-XPL Blocks



For the ADM-XPL, there are 2 memory blocks, one for the SRAM and one for the DRAM. The interface is the same for both blocks. The only significant differences is that the time between the read and data_valid signal is no longer 4 clock cycles on the DRAM controller. The actual delay time can vary depending on memory refreshes. To implement a model similar to the SRAM, the DRAM only uses data on one clock edge of its DDR cycle. Therefore the capacity of RAM available for cosim is half that of which is available i.e. only 32MB on a 64MB XPL (23 address pins are provided). The capacity of the SRAM is also half as it is physically 64 bits wide device, however only 32 bits are implemented.

The XPL does not have the Pn4 connector wired to the VIIPro FPGA. The samtec IO connections should be connected in the same way as the XRC-II reario and samtec interfaces: the samtec_tri port must be instantiated and driven by a signal specifying all port directions. (Specify as input for unused ports). There are 147 pins available on the XPL Samtec connector.

6.3 ADM-XP Blocks



The ADM-XP has 6 banks of memory, 4 banks of SRAM and 2 banks of DRAM. The DRAM are configured as 32MBx32bit (23 address pins), and have unpredictable latency. The SRAM are configured as 16MBx32bit (22 address pins), although smaller devices may be fitted to the card.

The samtec and reario connections are similar to those of the ADM-XRC-II. The samtec bus is 160 pins wide, however pins 88 to 99 are not connected.

6.4 SRAM and DRAM simulation

The SRAM and DRAM modules use a matlab S-function to simulate the memory. This is provided in the blockset directory as modelram_cosim.m. This does not fully model all the RAM modules in the system. The main restriction in the model is that it is restricted to modelling 4MB of data. The main memory model is in a Matlab Global variable SRAM, which is a 6x1048576 matrix. This can be written to before the Simulation and results read after Simulation, using Matlab. The model also assumes a read to valid delay of 4 clock cycles. Timing of data output from the RAM should be based on the data valid signal, and not relative to the read input.

Two Shared Memory access blocks are also provided. One for writing to data in an SRAM and one for reading data from an SRAM. The detailed operation of these blocks can be viewed in their Matlab S functions (readram_cosim.m and writeram_cosim.m). The write block sequentially writes a frame of data into the SRAM and waits for the FPGA to process it before writing the next frame. The read block waits for the FPGA to process a block of data and then reads back the data a sample at a time. Each frame is written to data starting at address 0.

Each block has 2 parameters 'SRAM No' and 'frame_size'. Frame size indicates how many samples are written to data before the 'FPGA' is granted control over the memory. In the case of the read SRAM block, 'frame_size' indicates how many samples of data are read from the RAM after the FPGA has released the block.

The basic write RAM loop is:

```
do
    for i=0:frame_size-1
      write 'Data' to SRAM  address i
    end
    wait for FPGA 'req' = 1
    assert FPGA 'gnt'
    wait for FPAG 'req' = 0
    deassert FPGA 'gnt'
loop
```

The enable data signal is valid when the frame of data is being written to the SRAM.

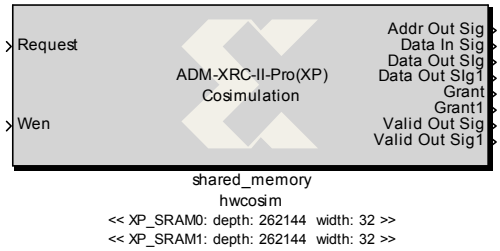
The basic read RAM loop is:

```
do
    wait for FPGA 'req' = 1
    assert FPGA 'gnt'
    wait for FPAG 'req' = 0
```

```
    deassert FPGA 'gnt'
    for i=0:frame_size-1
      read 'Data' from SRAM  address i
    end
loop
```

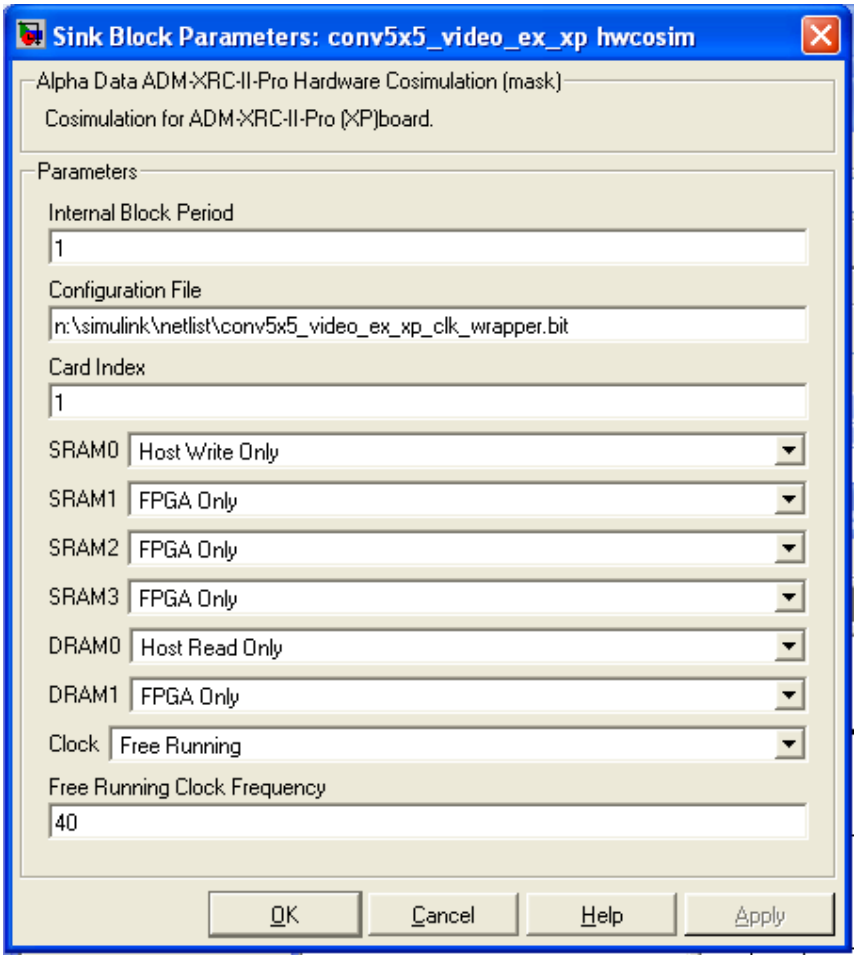
The data valid signal indicates when the data is being read out of the SRAM.

6.5 SRAM and DRAM Cosimulation



When a co-simulation bitstream is created with the memory modules in the design, the external shared memories will appear as Shared Memories on the block in the same way as internal shared block RAMs appear. These memories can be accessed using the standard Xilinx shared memory blocks. The memory names are specified in the format "<board>_<S/D>RAM<i>" , e.g. XRC2_SRAM3. This name should be entered into any shared memory block to connect to the FPGA Board Memory.

Some additional parameters have been added to the cosimulation block:
An access mode parameter for each memory is available, with 4 options:
FPGA always granted access (default),
Shared Memory only performs DMA Read,
Shared Memory only performs Write,
Shared memory performs write and read back.
The first option is for FPGA applications not sharing the memory. The second and third options can speed up performance in systems that do not need to both read and write the same memory. The last option lets the shared memory operate in the same way as the internal shared memories.

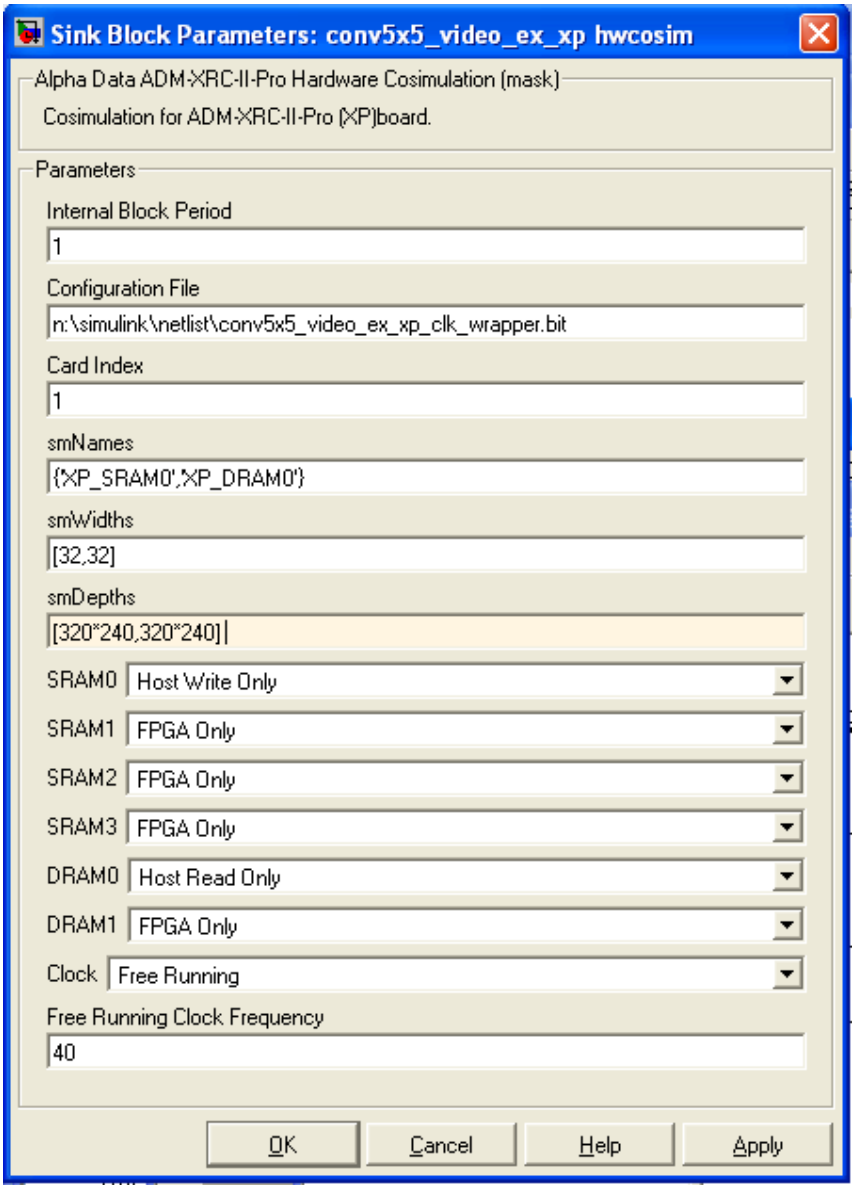


There are a few hidden options that can be used to speed up the operation further. These are not displayed by default, and to display them requires editing of the cosim block mask. Right click on the blcok and select edit Mask. Click on the Parameters Tab. Scroll down and click on the smNames parameter, click the Show Parameter tick box to make the property visible. Also make smWidths and smDepths visible.

This now allows you to edit the 3 fields. The most useful field to edit is smDepths. This can be increased or reduced to change the size of memory transferred from the host to and from the FPGA. This can obviously speed up the algorithm if only a small part of the memory is being used.

The smWidths field can also be modified if only part of the 32 bit data width is being used.

The smNames parameters should not be changed unless multiple cards are being used. In that case you can extend the names to allow shared memories on different cards to be distinguished, e.g XRC2_SRAM0_card1 and XRC2_SRAM0_card2. It is important that the first part of the name (e.g. XRC2_SRAM0) is not changed.



In this example smDepths has been modified to reduce the DMA transfer size to match that of a 320x240 image.

6.6 Burst Limitations of DRAM

DRAM devices have a more complicated control interface than SRAM devices, requiring separate clock cycles for setting up the row and column addresses. They also require periodic refresh cycles. This can make the read latency of the interface vary. The device is only capable of bursting within a single row (512 words).

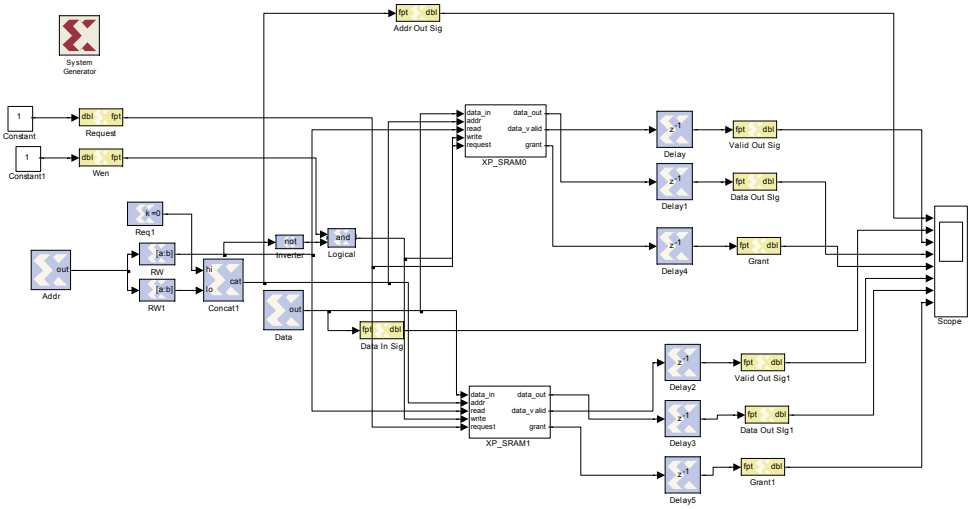
The interface module has 16-word command, read data and write data FIFOs. The interface also runs at twice the clock frequency of the system. This should allow the

memory to burst continuously over long address ranges, as although the FIFO will fill up during refreshes and row changes, it should be emptied faster than it is filled. The large latency associated with reads can in some cases still cause an overflow and therefore it is advisable to restrict the length of continuous bursts to less than 512 words.

Random accesses to memory across multiple rows will increase the turn around time for the operation. Continuous switching between reads and writes is also not advisable. Both these operations may fill up the FIFO and therefore should not be started continuously on consecutive clock cycles. Therefore random accesses and read and write changes should be restricted to bursts of less than 16 words.

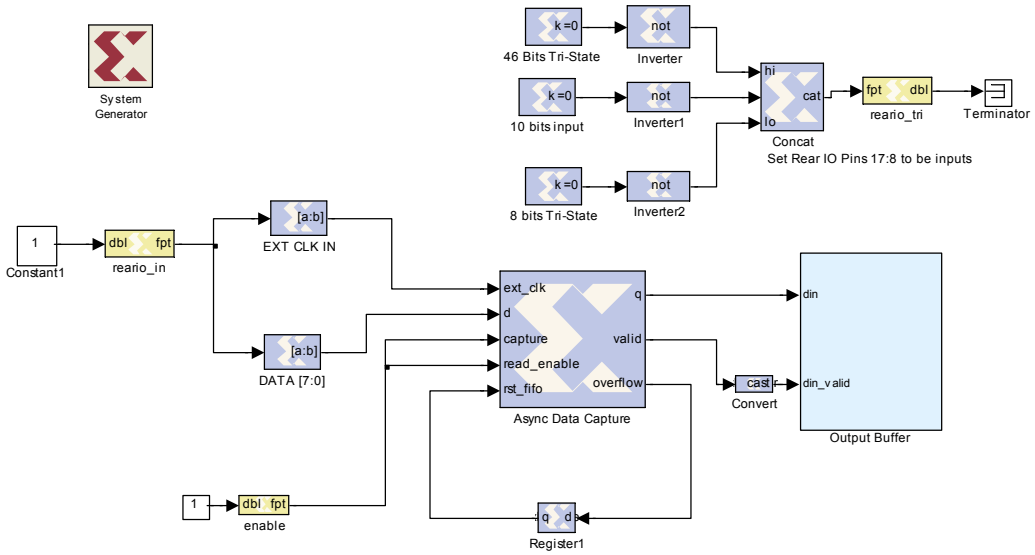
7 Advanced Examples

7.1 External Memory

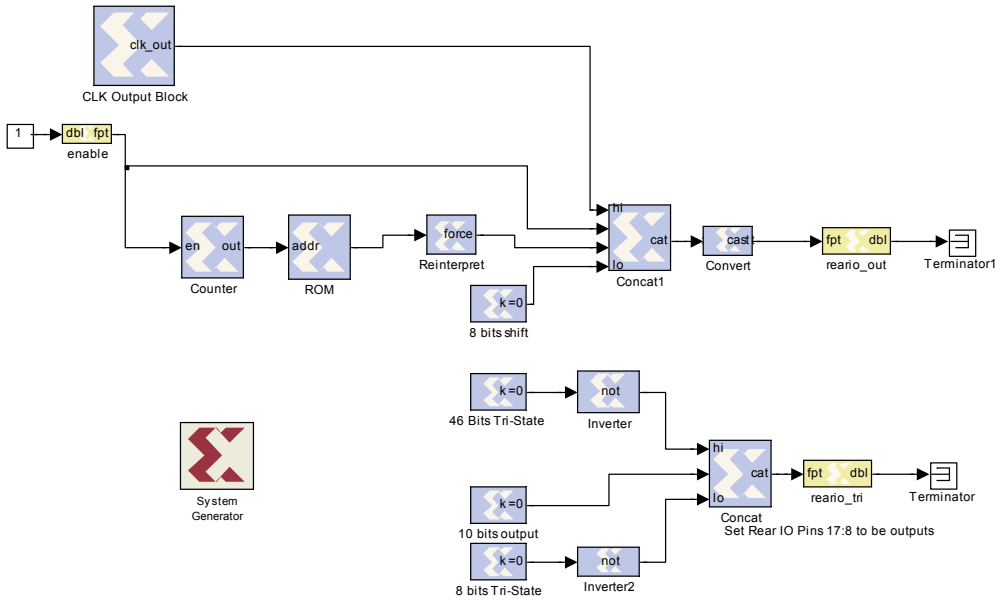


This example very simply demonstrates how to wire up the external memories in the FPGA and run them in single stepped mode. Counters generate address, data signals and read and write strobes. External inputs can mask the write strobe and the mutual exclusion request.

7.2 Data Capture

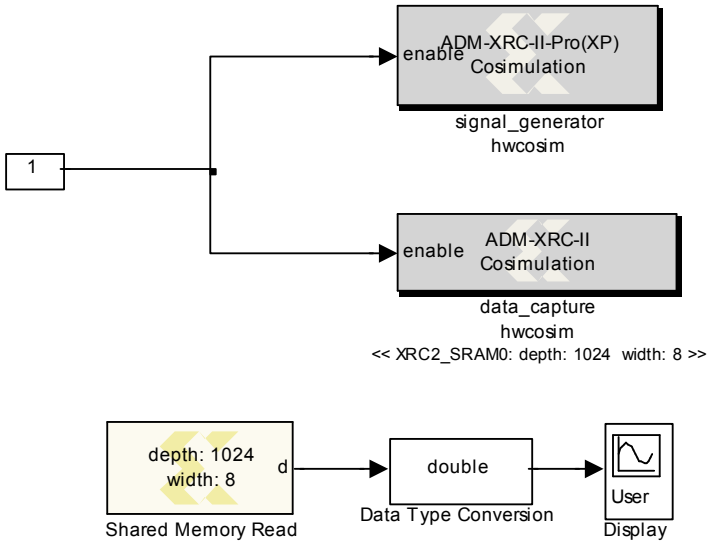


This example demonstrates how to set up I/O pins in an application. In this case an 8 bit wide data signal is captured from the Rear IO (Pn4) pins. The data signal has its own clock, so a VHDL black box is included to sample the data on the negative edge of the external clock and transfer it to the Simulink clock domain using an asynchronous FIFO. The data is then stored in a shared external memory, which can be read into Simulink by the host.



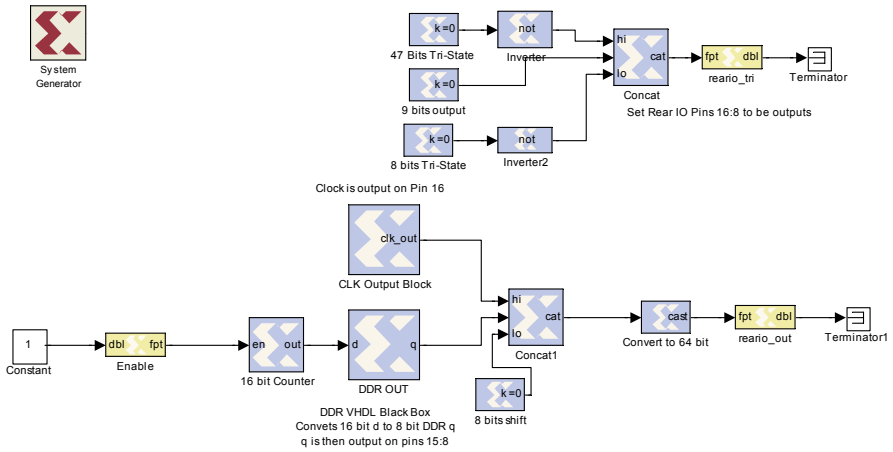
A companion signal generator application generates a sine wave on the 8 Rear IO (Pn4) pins. This example also includes a simple VHDL black box to output the

System Generator on a clock pin using a DDR register to implement clock forwarding.



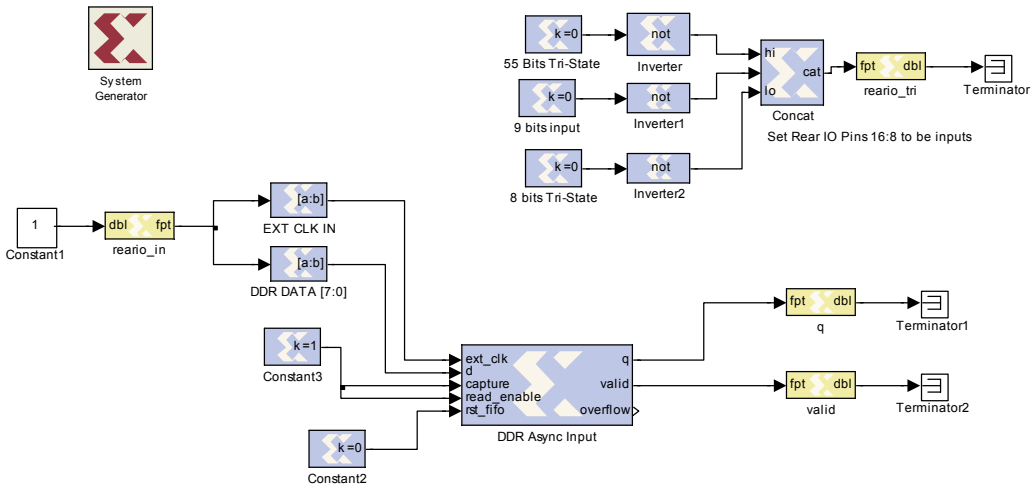
The testbench model runs the signal generator on one ADM-XRC series PMC and the data capture application on another ADM-XRC series PMC. Both are installed on the same ADC-PMC carrier, so that the Pn4 Rear IO pins are connected, and the data generated in the first FPGA is captured by the second FPGA and stored in the external shared memory which is then read by Simulink and the sine wave displayed.

7.3 DDR IO



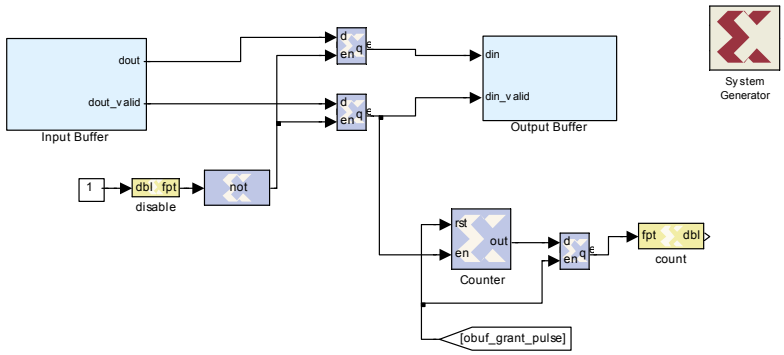
The DDR IO example consists of a DDR transmitter and a DDR receiver. A VHDL black box is used to convert a 16 bit wide data bus to 8 bits running at DDR which are

connected directly to the pins. The CLK output block is also used to forward the System Generator clock to the pins.

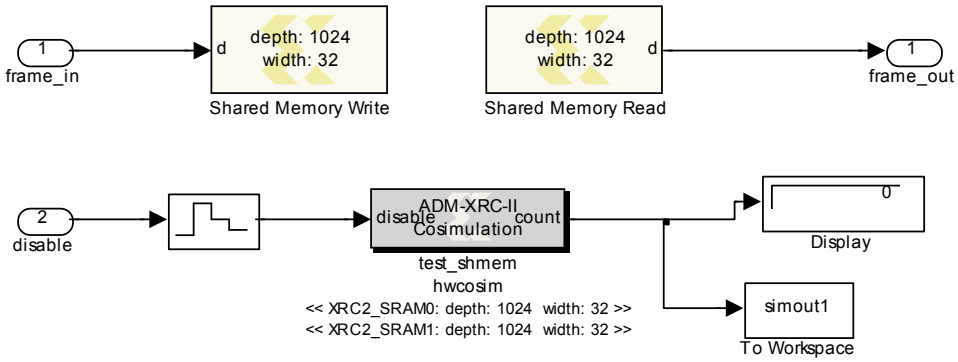


The receiver block has an asynchronous data capture VHDL black box similar to the single data rate module. In this case however the ideal clock sample time is 90 degrees after the positive and negative edges. To support operation in both single stepped clock mode and free running clock mode, the phase shift is generated by delaying the external clock through 3 LUTs on the FPGA. This delay may not be appropriate for all free running clock frequencies and may need modification. If the input clock is free running then a DCM can be used to accurately generate an appropriate phase shift.

7.4 Shared Memory



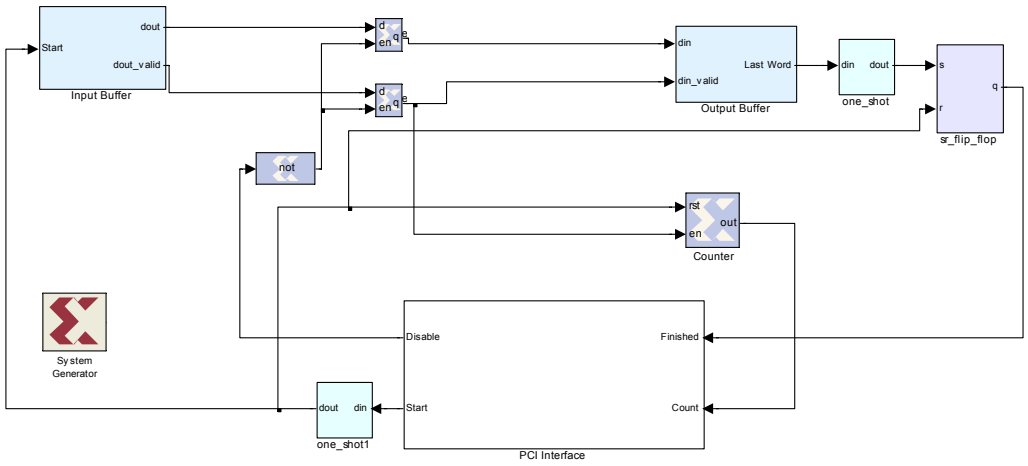
This shared memory example shows how to set up an application which processes data in one SRAM and stored it in another. No application is included, instead a register is instantiated and the example simply copies data from one bank of memory to another. A count is kept of all the data words received.



The testbench example copies data from Matlab into one shared memory and reads it out from the other shared memory back into Matlab for checking. The simin variable is provided in the simin.mat file.

This example shows how to use the cosimulation blockset to stream data through an algorithm. To build a user application, simply replace the registers in the data and valid pipelines with the processing algorithm.

If the speed provided by the cosimulation blockset is insufficient it is also possible to port the algorithm to use the Alpha Data Applications blockset. This blockset allows higher clock rates (as it does not support single stepping mode) and is targeted at the development of the final embedded application.



The Matlab script file apps_shmem_test.m can be used to configure and control the FPGA using the bitstream generated.