XRC Board Level Application Library v2.0 (For Simulink and System Generator)

ALPHA DATA parallel systems ltd

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@alphadata.co.uk

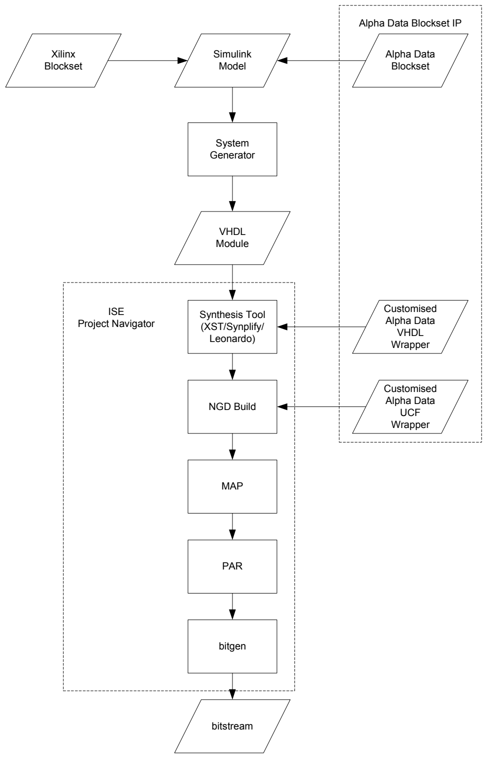
1 Introduction

The Alpha Data XRC Board Level Application Library 2.0 is a collection of embedded IP, VHDL, Simulation Models, Wrapper Models and Constraint files designed to ease the design of FPGA applications on Alpha Data Embedded PMC and PCI cards. These modules are designed to work with Xilinx System Generator and Simulink from the Mathworks.

1.1 Whats New

Users of the XRC Board Application Blockset version 1.7 or earlier should note that there are a number of substantial changes in this release. The most significant is that the automatic generation of bitstreams has been removed. While automatic generation makes it easy for novice users to produce their first bitstream, it can limit the control over the many tools used in this process, and tools such as Xilinx Project Navigator are much more suited to this task.

The design flow is now as follows:



The Alpha Data Library therefore provides 2 distinct pieces of IP to the user.

1. A selection of embeddable modules is provided to handle board specific tasks such as PCI and Memory I/O. These are included in the Simulink Model and can be built into a VHDL module by System Generator.
2. Example Wrapper VHDL and UCF files are provided to show how to connect the outputs of the System Generator Module to the Pins on the FPGA. This top level wrapper is provided to allow tri-state I/O pins to be interfaced to System Generator Module (which only supports Inputs and Outputs). The wrapper can also be used to generate DCM locked clocking for the off-chip RAM modules. Finally any other difficult I/O tasks such as asynchronous input of data can also be handled in VHDL at this level, and these files are entirely customisable by the user.

2 Installation Instructions

The XRC Board Level Application Library is provided as a zip file and a Matlab installation script:

xrc_application_blockset.zip
setup_xrc_application.m

1) Open Matlab

2) Change Directory to where xrc_application_blockset.zip and setup_xrc_application.m have been downloaded.

3) Type setup_xrc_application

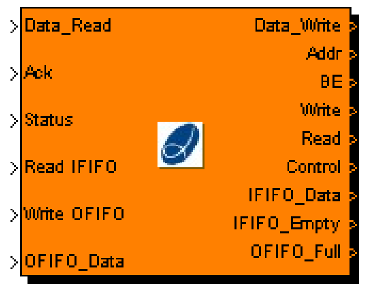
4) Quit Matlab

3 XRC Simulink Blockset

The XRC Blockset provides simulation and synthesis models to handle the most common I/O tasks in the ADM-XRC Series of cards. Communication with the host is through a Local Bus bridge and PCI. The modules provided have control and status registers (for simple control; tasks), a slow, asynchronous, memory mapped, slave interface (for more sophisticated control and small data transfers), and input and output demand mode DMA FIFO's for large high speed data transfers to and from the host. The other modules provided interface to the off-chip memory on each card (ZBT, DDR-SDRAM, DDR-II SSRAM). Other I/O is generally unidirectional and synchronous to the System Generator clock domain, and this can be implemented using the standard System Generator I/O ports.

3.1 Local Bus Interface Modules

3.1.1 LBIF_PLX32



PLX 32 Bit Interface

This interface connects to the PLX9656 PCI bridge chip used on the ADM-XRC-II. It provides the following ports:

Control : (UFix_32_0) -- control register value – set by host at address 0x40000
Status : (Ufix_32_0) -- status register – read by host at address 0x40040

The slave memory mapped interface uses the following ports:
Data_Write: (Ufix_32_0) -- data written by host
Addr: (UFix_16_0) -- 32 bit address for read or write
BE: (Ufix_4_0) -- byte enables
Write: (Boolean) -- host wants to write data (Data_Write) to Address (Addr)
Read: (Boolean) -- host wants to read data from Address(Addr)
Data_Read: (UFix_32_0) -- data to be returned to host
Ack: (Boolean) -- indicates that Data_Read is valid – must be asserted in response to Read
The slave memory is mapped to local bus addresses 0x000000-0x03FFFC

The Demand Mode DMA FIFOs are connected to the PLX9656 DMA channels, with writes from the host on DMA channel 0 resulting in Data Being pushed into IFIFO, and reads from the host on DMA channel 1, pops data out of OFIFO. Since Demand Mode DMA is used the host process will block if IFIFO is full or OFIFO is empty.

IFIFO Ports are:
IFIFO_Data: (UFix_32_0)
IFIFO_Empty: (Boolean)
Read_IFIFO: (Boolean)

OFIFO Ports are:
OFIFO_Data: (UFix_32_0)
OFIFO_Full: (Boolean)
Write_OFIFO: (Boolean)

The following C-Code for accessing these can be used. The Demand Mode DMA application in the ADM-XRC SDK should also be consulted to see how to set up Demand Mode DMA transfers. With this module the DMA counters at Local Bus locations 0x040080 and 0x0400C0, and these should be set before starting the DMA transfer.

```
void sendData(int offset, int size)
{
  ADMXRC2_STATUS              status;
  HANDLE                              event;

  event = CreateEvent(NULL, TRUE, FALSE, NULL);
  /*
  ** Program the PCI-to-local transfer counter.
  */
  fpgaSpace[65568] = size;
  fpgaSpace[65568];

  /*
  ** Perform the DMA transfer.
  */
  status = ADMXRC2_DoDMA(card,sendbufHandle,offset,size,0x40,
                      ADMXRC2_PCITOLOCAL,0,mode,0,NULL,event);
  CloseHandle(event);
}

void recvData(int offset, int size)
{
  ADMXRC2_STATUS              status;
  HANDLE                              event;

  event = CreateEvent(NULL, TRUE, FALSE, NULL);

  /*
  ** Program the local-to-PCI transfer counter.
  */
  fpgaSpace[65584] = size;
  fpgaSpace[65584];

  /*
  ** Perform the DMA transfer.
  */
  status = ADMXRC2_DoDMA(card,recvbufHandle,offset,size,0x80,
                      ADMXRC2_LOCALTOPCI,1,mode,0,NULL,event);
   CloseHandle(event);
}
```

Two parameters are provided for the Local Bus Interface Block, the first “Local Bus script Program” is used to simulate Local Bus transactions. The second parameter “Clock Ratio” sets the ratio used in simulation between Local Bus Clock Cycles and the System Generator System Clock e.g. if this is set to 1.5 then the Local Bus program will update its outputs every 1.5 System Clock cycles.

The Local Bus Script Program can be specified as a string or as a Matlab variable. If either of these is specified as load <filename.txt> then the file <filename.txt> will be loaded and parsed in place of the string.

The following commands are parsed:

slave_read <address>
reads the contents of a local bus address and displays the result

slave_write <address> <data> [be]
write 32 bit integer <data> to address with optional byte enable signal

dma_read <length> [matlab variable]
reads <length> words from OFIFO and either displays them or stores them in a a [matlab variable]

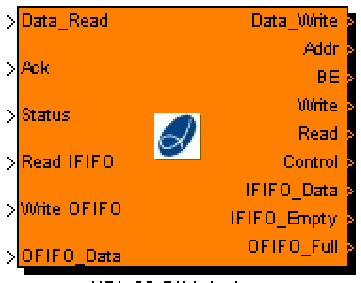
dma_write <length> [matlab variable]
writes <length> words into IFIFO. If a [matlab variable] is specifed then this is used as data otherwise integers 1..<length> are sent

sleep <count>
this pauses the simulation program for <count> LCLK cycles

Example Program:
```
slave_write 0x40000 0xabcd1234
slave_read 0x40000
slave_read 0x40040
slave_write 0x0 0x1234abcd
dma_write 8 myfifo_in
dma_read 8 myfifo
slave_read 0x0
```

N.B. There is a 32K limit on the size of the program.

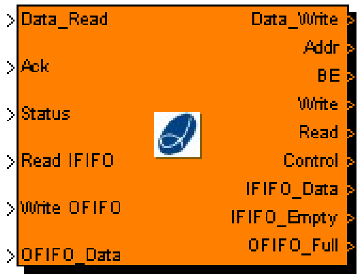
3.1.2 LBIF_XPL32



XPL 32 Bit Interface

This interface connects to the XPL PCI-Local Bus Bridge used on the ADM-XPL and ADM-XP. Functionally this module performs identically to the LBIF_PLX32 module. In implementation there are minor differences in the VHDL top level ports produced, as the data and address pins are multiplexed with the XPL bridge, unlike the PLX9656.

3.1.3 LBIF_XPL64



XPL 64 Bit Interface

This module is similar to LBIF_PLX32 and LBIF_XPL32, however the data widths are 64 bits wide. The ports therefore are

Control : (UFix_64_0)
Status : (Ufix_64_0)
Data_Write: (Ufix_64_0)
Addr: (UFix_16_0)
BE: (Ufix_8_0)
Write: (Boolean)
Read: (Boolean)
Data_Read: (UFix_64_0)
Ack: (Boolean)
IFIFO_Data: (UFix_64_0)
IFIFO_Empty: (Boolean)
Read_IFIFO: (Boolean)
OFIFO_Data: (UFix_64_0)
OFIFO_Full: (Boolean)
Write_OFIFO: (Boolean)

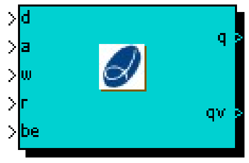
The host application should set the Local Bus Space up to allow 64 bit access. The DMA Mode bit for 64 bit access should also be set.

A similar format is used for the Local Bus Script Program but with one minor changes to deal with 64 bit values:
slave_write <address> <data0> [data1] [be]
will write data0 to the lower 32 bits and data1 to the upper 32 bits if specified

N.B. the address input does not change and is still refers to a 32 bit word address. So bit 0 may be ignored if 64 bits transfers are being used.

3.2 Memory Modules

3.2.1 ZBT SRAM (32 bit)

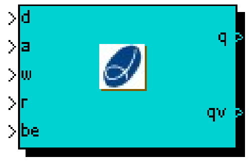


ZBT SRAM 0

These modules implement the 32bit ZBT SRAM on the ADM-XRC-II.
Ports are:
d : UFix_32_0
a : UFix_20_0
w : Boolean
r : Boolean
be : UFix_4_0
q : UFix_32_0
qv : Boolean

If “w” is asserted then data of “d” is stored in the SRAM at address “a”.
If “r” is asserted then SRAM address “a” is read and the output appear on “q” 4 clock cycles later, with “qv” going high to indicate that it is valid.

3.2.2 ZBT SRAM (64 bit)

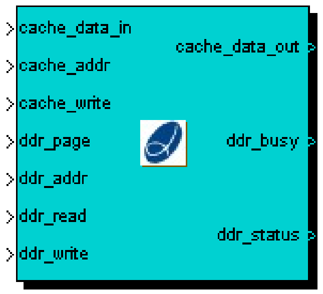


ZBT SRAM 64

This module implement the 64bit ZBT SRAM on the ADM-XPL.
Ports are:
d : UFix_64_0
a : UFix_20_0
w : Boolean
r : Boolean
be : UFix_8_0
q : UFix_64_0
qv : Boolean

If “w” is asserted then data of “d” is stored in the SRAM at address “a”.
If “r” is asserted then SRAM address “a” is read and the output appear on “q” 4 clock cycles later, with “qv” going high to indicate that it is valid.

3.2.3 XRM-DDR SDRAM (64 bit)



XRM DDR SDRAM

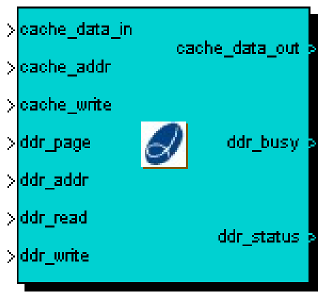
This module implements and interface to the XRM-DDR SDRAM on the XRM-DDR module. The data interface to the System Generator module is via an 8K cache memory block RAM, with a 1 clock cycle latency.

Ports are:
cache_data_in : UFix_32_0
cache_data_out : UFix_32_0
cache_addr: UFix_11_0
cache_write: Boolean

Control signals are used to burst the data from the 8K cache to the DDR
ddr_page : UFix_14_0 is used to select an 8K page in the DDR
ddr_addr : UFix_4_0 Selects a 256 byte block within the cache
ddr_read : Boolean – when asserted the 256 byte block is copied from DDR to Cache
ddr_write : Boolean – when asserted the 256 byte block is copied from Cache to DDR
ddr_busy : Boolean -- Indicates whether a burst is in operation or not. While ddr_busy is high, ddr_read and ddr_write will be ignored.
ddr_status : UFix_3_0 – indicates status info for the DDR controller

The time for a burst will vary as it may have to wait for a refresh to finish, however each burst should take between 36 (usual) and 43 (worst case) clock cycles.

3.2.4 DDR SDRAM (32 bit)



DDR SDRAM

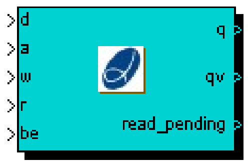
These modules implement the interface to the DDR SDRAM on the ADM-XPL and ADM-XP. The data interface to the System Generator module is via a 4K cache memory block RAM, with a 1 clock cycle latency.

Ports are:
cache_data_in : UFix_32_0
cache_data_out : UFix_32_0
cache_addr: UFix_10_0
cache_write: Boolean

Control signals are used to burst the data from the 4K cache to the DDR
ddr_page : UFix_14_0 is used to select an 4K page in the DDR
ddr_addr : UFix_4_0 Selects a 128 byte block within the cache
ddr_read : Boolean – when asserted the 128 byte block is copied from DDR to Cache
ddr_write : Boolean – when asserted the 128 byte block is copied from Cache to DDR
ddr_busy : Boolean -- Indicates whether a burst is in operation or not. While ddr_busy is high, ddr_read and ddr_write will be ignored.
ddr_status : UFix_3_0 – indicates status info for the DDR controller

The time for a burst will vary as it may have to wait for a refresh to finish, however each burst should take between 36 (usual) and 43 (worst case) clock cycles.

3.2.5 DDR-II SSRAM



DDRII SRAM 0

These modules implement the interface to the DDR-II SSRAM on the ADM-XP. The 32 bit DDR ports on the devices are mapped to single clock 64 bit wide ports.

Ports are:
d : UFix_64_0
a : UFix_21_0
w : Boolean
r : Boolean
be : UFix_4_0
q : UFix_64_0
qv : Boolean
read_pending : Boolean

If “w” is asserted then data of “d” is stored in the SRAM at address “a”.
If “r” is asserted then SRAM address “a” is read and the output appears on “q” 2 clock cycles later, with “qv” going high to indicate that it is valid.
“read pending” will be high 1 clock cycle after “r” is asserted. If “w” is asserted in this clock cycle it will be ignored and no write will take place.

For correct operation the DDR-II-SSRAM must be clocked at over 75MHz and less than 133MHz.

4 Wrapper Files

The VHDL produced by System Generator cannot handle all the interfacing requirements for the ADM-XRC series cards. Therefore the VHDL module produced must be wrapped before synthesizing and generating a bitstream.

4.1 Wrapper VHDL

Wrapper VHDL files are provided for each different card. These files are located in the directory vhdl-wrappers. Each file should be copied to the target directory and then edited to match the requirements.

The VHDL files contain a declaration and instantiation of a component sysgen_design. This needs to be renamed and edited so that it matches the entity declaration of the <design>_clk_wrapper.vhd produced by system generator.

The files contain definitions for every port defined in all the Alpha Data library block modules. If a module has not been used then the port declaration will need to be removed. If you have included your own custom I/O ports – such as connection to the front I/O – then you must add this to the wrapper.

The wrapper contains declarations of IOBUFs, which convert the three ports groups of signals exported from System Generator into tri-state signals needed for memory and bus interfaces.

The wrapper also contains some DCMs, which lock the system clock to the MCLK pin input, as well as lock the RAM clocks to the system clock, and provide phase shifted versions of the clock for DDR RAM modules.

4.2 Wrapper UCF Files

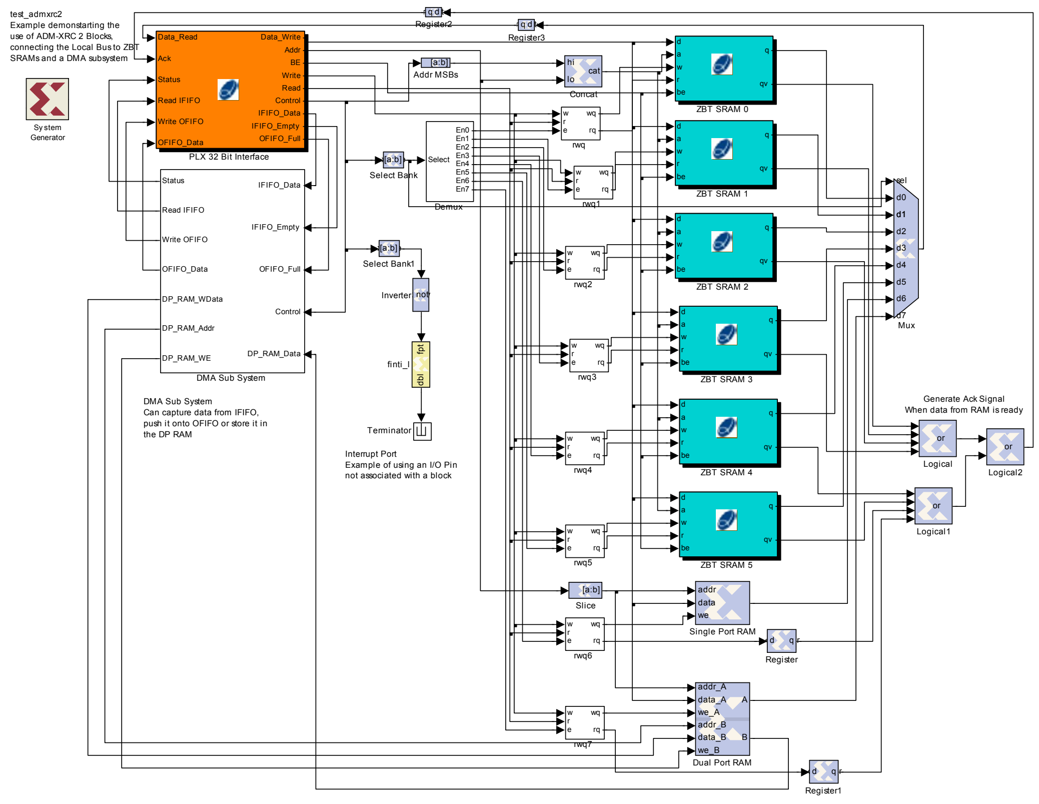
Wrapper UCF files are provided in the directory ucf-wrappers. These files contain the basic pin constrains for all the ports used by the modules. This file should be copied to the target directory and edited to select the pins used in the design. If you have added other I/O pins then they should also be constrained in the UCF file. (constraints set in System Generator might not be propagated).

The editing of the VHDL and UCF files will be initially laborious, however once modified for a specific I/O configuration (e.g. local bus, RAM and user I/O). They will not need to be changed further.

5 Example Applications

Five example applications are provided which demonstrate how to connect up the Local Bus Interface, the DMA FIFOs and the ZBT SRAM, DDR SDRAM and DDR-II SSRAM. These examples are located in the directory test.

5.1 "test_admxrc2"



This example is for the ADM-XRC-II and connects up all 6 Banks of ZBT SRAM to the Local Bus Interface. The control register is used to select which of the 6 banks the Local Bus accesses, or if it accesses one of 2 block RAMs. The DMA FIFO's are connected up, and these can loop the data round (IFIFO -> OFIFO), capture the data (IFIFO -> Block RAM 7) or output the Block RAM Data (Block RAM 7 -> OFIFO).

Control Register Bits:

(2:0) Select RAM Bank
000 ZBT SRAM 0
001 ZBT SRAM 1
010 ZBT SRAM 2
011 ZBT SRAM 3
100 ZBT SRAM 4
101 ZBT SRAM 5

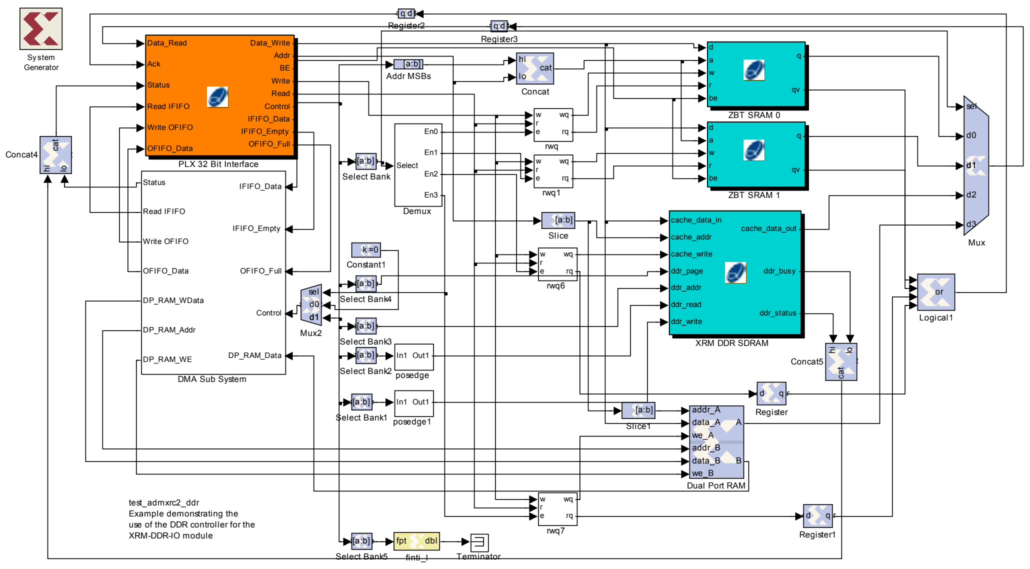
110 Block RAM 6
111 Block RAM 7 (DMA FIFO Capture)

(7:4) MSBs of ZBT Address
These bits are used as a page register for the ZBT Bank being accessed

(16) DMA FIFO WriteThrough – copies any input on the IFIFO to OFIFO
(17) DMA FIFO Capture – writes any data input on IFIFO to Block RAM 7
(18) DMA FIFO Burst – triggers a burst of data from Block RAM 7 to OFIFO

(24) Interrupt Bit – connected to the FPGA interrupt pin – demonstrates how to connect an output signal which is not part of the module.

5.2 *"test_admxrc2_ddr"*



This example is for the ADM-XRC-II with an XRM-DDR I/O module attached. The main purpose of this module is to demonstrate the operation of the DDR SDRAM controller. This module has a simple cache interface which is connected to the local bus and reads and writes between this and the DDR is controlled using the control register.

Control Register Bits:
(1:0) Select RAM Bank
00 ZBT SRAM Bank 0
01 ZBT SRAM Bank 1
10 DDR SDRAM Cache
11 Block RAM 3 (DMA FIFO Capture)

(2) Trigger DDR Read (DDR Data to Cache)
(3) Trigger DDR Write (Cache Data to DDR)

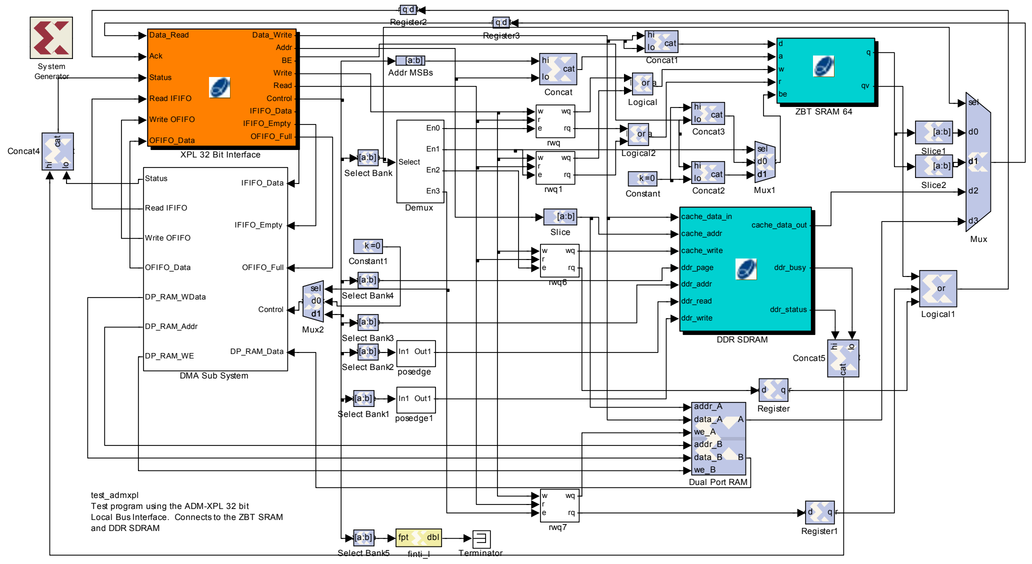
(7:4) MSBs of ZBT Address
These bits are used as a page register for the ZBT Bank being accessed

These bits are also used to control the DMA when bits (1:0) = “11”
(4) DMA FIFO WriteThrough – copies any input on the IFIFO to OFIFO
(5) DMA FIFO Capture – writes any data input on IFIFO to Block RAM 3
(6) DMA FIFO Burst – triggers a burst of data from Block RAM 3 to OFIFO

(11:8) DDR Addr – In Page address of burst
(8K Cache is divided into 16 blocks each of which can be burst to/from the DDR)

(29:16) DDR Page
Page in DDR Memory used for burst

5.3 *“test_admxpl”*



This example is for the ADM-XPL and shows how to connect the ZBT SRAM and the DDR SDRAM to the 32 bit Local Bus module. This also contains the DMA FIFO interface used in the previous example. The 64 bit ZBT SRAM is divided into 2 32 bit banks accessed independently using Byte Enables

Control Register Bits:
(1:0) Select RAM Bank
00 ZBT SRAM Lower 32 bits
01 ZBT SRAM Upper 32 bits
10 DDR SDRAM Cache
11 Block RAM 3 (DMA FIFO Capture)

(2) Trigger DDR Read (DDR Data to Cache)
(3) Trigger DDR Write (Cache Data to DDR)

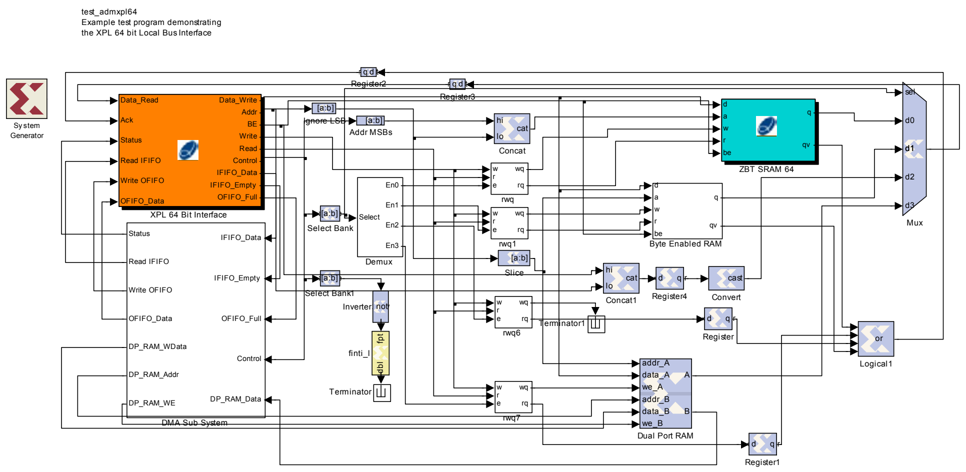
(7:4) MSBs of ZBT Address
These bits are used as a page register for the ZBT Bank being accessed

These bits are also used to control the DMA when bits (1:0) = “11”
(4) DMA FIFO WriteThrough – copies any input on the IFIFO to OFIFO
(5) DMA FIFO Capture – writes any data input on IFIFO to Block RAM 3
(6) DMA FIFO Burst – triggers a burst of data from Block RAM 3 to OFIFO

(11:8) DDR Addr – In Page address of burst
(4K Cache is divided into 16 blocks each of which can be burst to/from the DDR)

(29:16) DDR Page
Page in DDR Memory used for burst

5.4 *“test_admxpl64”*



This model is also for the ADM-XPL. This example demonstrates how to use the 64 bit Local Bus. This is connected to the 64 bit ZBT, and to block RAMS, some configured so that the byte enables work.

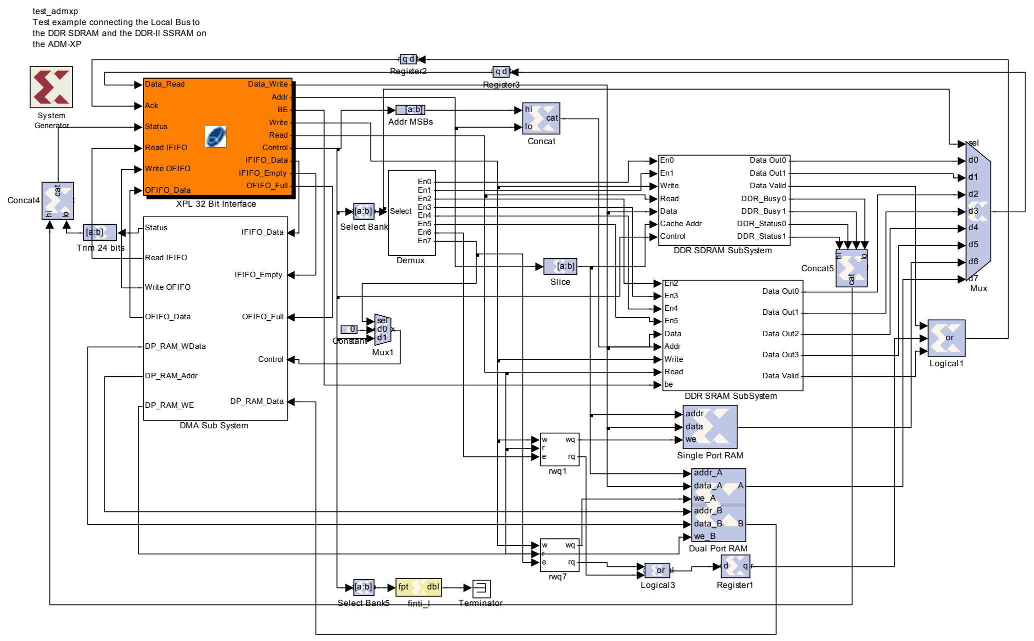
Control Register Bits:
(1:0) Select RAM Bank
00 ZBT SRAM
01 Byte Enabled Block RAM
10 Address Feedback Register
11 Block RAM 3 (DMA FIFO Capture)

(8:4) MSBs of ZBT Address

These bits are used as a page register for the ZBT Bank being accessed

(16) DMA FIFO WriteThrough – copies any input on the IFIFO to OFIFO
(17) DMA FIFO Capture – writes any data input on IFIFO to Block RAM 3
(18) DMA FIFO Burst – triggers a burst of data from Block RAM 3 to OFIFO

5.5 *"test_admxp"*


test_admxp
Test example connecting the Local Bus to the DDR SDRAM and the DDR-II SSRAM on the ADM-XP

This model is for the ADM-XP. It connects the 32 bit Local Bus interface to the DDR SDRAM and DDR-II SRAM modules.

Control Register Bits:

(2:0) Select RAM Bank
000 DDR SDRAM 1
001 DDR SDRAM 2
010 DDR-II SRAM 0
011 DDR-II SRAM 1
100 DDR-II SRAM 2
101 DDR-II SRAM 3
110 Block RAM 6
111 Block RAM 7 (DMA FIFO Capture)

(8:4) MSBs of DDR-II SDRAM Address
These bits are used as a page register for the ZBT Bank being accessed

These bits are also used to control DDR to Cache Transfers when either DDR SDRAM bank is active.
(4) Trigger DDR Read (DDR Data to Cache)
(5) Trigger DDR Write (Cache Data to DDR)

These bits are also used to control the DMA when bits (2:0) = "111"
(4) DMA FIFO WriteThrough – copies any input on the IFIFO to OFIFO
(5) DMA FIFO Capture – writes any data input on IFIFO to Block RAM 3
(6) DMA FIFO Burst – triggers a burst of data from Block RAM 3 to OFIFO

(11:8) DDR Addr – In Page address of burst
(4K Cache is divided into 16 blocks each of which can be burst to/from the DDR)

(29:16) DDR Page
Page in DDR Memory used for burst

5.6 Building The Test Examples

These test examples have additional files to help build a bitstream from the model.

1) top_test_adm???.vhd – top level wrapper for the design
2) top_test_adm???.ucf – top level UCF file for the design
3) netlist\top_test_adm???.npl – Project Navigator File for design

After running Generate using the System Generator, the resulting VHDL files should appear in the netlist directory. The Project Navigator File should then be able to link these with the wrapper files and build the bitstream.

5.7 Running the Test Examples

Example C code is also provided to load these bitstreams into an FPGA and test all the interfaces implemented. This code is located in directory test\c-src. The code test_admxrc2.c will test test_admxrc2.bit and test_admxrc2_ddr.bit on an ADM-XRC-II. The code test_admxpl.c will test test_admxpl.bit and tes_admxpl64.bit on an ADM-XPL. The code test_admxp.c tests test_admxp.bit on an ADM-XP.