Accessing Memory on the MFC VXI Module

Slot 0 Memory Map

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Memory Space | Slot 0:  Offset to First | Slot 0:  Offset to Last | Data Width | No. of Locations Accessible | No. of Locations Possible |
| DSP Internal DM RAM | 0x0000 0000 | 0x003F FFFF | 32 | 294,912 |  |
| DSP SDRAM (MS0) | 0x0040 0000 | 0x0050 0000 | 32 | 262,144 | 4 Meg |
| MAX II A32 Registers | 0x0050 0000 | 0x005F FFFF | 32 | 9 | -- |
| FLASH Memory | 0x0060 0000 | 0x015F FFFF | 8 | 1,048,576 | 128 x 64k sectors |
| FPGA RAM | 0x0160 0000 | 0x016F FFFF | 32 | 36,000\* | 1,048,576 |
| FPGA SDRAM | 0x0170 0000 | 0x03FF FFFF | 8 | 42,991,620 | 64 Meg |

\*Cyclone FPGA has 1,152,000 RAM bits available

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Memory Space | DSP:  Offset to First | DSP:  Offset to Last | Data Width | No. of Locations Accessible | No. of Locations Not Accessible |
| DSP Internal DM RAM | 0x000B 8000 | 0x000F FFFF | 32 | 294,912 |  |
| DSP SDRAM (MS0) | 0x0020 0000 | 0x0400 0000 | 32 |  |  |
| FLASH Memory (MS1) | 0x0400 0000 | 0x07FF FFFF | 8 | 128 x 64k sectors | 128 x 64k sectors |
| MAX II A32 Registers | 0x0800 0000 | 0x0BFF FFFF | 32 | 9 |  |
| FPGA RAM | 0x0C00 0000 | 0x0C0F FFFF | 32 | 36,000\* |  |
| FPGA SDRAM | 0x0C10 0000 | 0x0FFF FFFF | 8 | 66,060,290 | 64 Meg |

CONSTANT SDRAM\_U\_LIMIT : integer := 5; -- Upper Boundary in MB of DSP\_Internal and DSP\_SDRAM Memory Addresses

CONSTANT MAX2\_U\_LIMIT : integer := 6; -- Upper Boundary in MB of Max2 Register Addresses

CONSTANT FLASH\_U\_LIMIT : integer := 22; -- Upper Boundary in MB of Flash Memory Addresses

CONSTANT FLASH\_BASE\_ADDRESS : std\_logic\_vector(23 downto 0) := "000110000000000000000000"; --0x180000

CONSTANT FPGA\_BASE\_ADDRESS : std\_logic\_vector(23 downto 0) := "010110000000000000000000"; --0x580000

-- A32 address space from Offset Register after configuration

A32baddr <= A32Offset(15 downto 10); -- A32Offset is the OFFSET register set by the Slot 0 Resource Manager

A32board\_select <= '1' when ((addr\_l(31 downto 26) = A32baddr) and (amsingle = '1')) else '0';

-- Decode address and determine device selected

IntAddrSeg <= CONV\_INTEGER(addr\_l(25 downto 20)); -- addr\_l(31 downto 1) <= vxi\_a(31 downto 1);

SDRAM\_select <= '1' when (A32board\_select='1' and IntAddrSeg < SDRAM\_U\_LIMIT ) else '0';

Max2\_select <= '1' when (A32board\_select='1' and IntAddrSeg >= SDRAM\_U\_LIMIT and IntAddrSeg < MAX2\_U\_LIMIT) else '0';

Flash\_select <= '1' when (A32board\_select='1' and IntAddrSeg >= MAX2\_U\_LIMIT and IntAddrSeg < FLASH\_U\_LIMIT) else '0';

FPGA\_select <= '1' when (A32board\_select='1' and IntAddrSeg >= FLASH\_U\_LIMIT) else '0';

-- A16 address space from Logical Address Register

-- Set to 0xFF before RM (dynamic configuration) writes the final LA to the LA register

A16baddr <= Logical\_Addr; --For DC

-- A16baddr <= LA\_FF; -- For SC

-- Before LA is assigned, look at modid input also to determine if board is selected

-- because all DC cards will have the same LA of 0xFF before configuration is complete

-- After LA is assigned, modid input is ignored

A16board\_select <= '1' when (((addr\_l(13 downto 6) = A16baddr) and (amshort = '1') and (LA\_Assigned = '1' ))

or ((addr\_l(13 downto 6) = A16baddr) and (amshort = '1') and (LA\_Assigned = '0') and (vxi\_ModId = '1' ))) else '0';

--===================================================================================================

-- State machine logic to select desired memory access process

if (DSP\_Write = '0') then -- DSP write access to Max2 A32 Registers

DSP\_ACK\_Int <= '0';

vxi\_sm <= DSPWrMax1;

elsif (DSP\_Read = '0') then -- DSP read access to Max2 A32 Registers

DSP\_ACK\_Int <= '0';

vxi\_sm <= DSPRdMax1;

elsif (asq = '1' and write\_l = '0') then -- address strobe went low / slot0 read

if (A16board\_select = '1') then vxi\_sm <= ConfigRd1;

elsif (Max2\_select = '1') then vxi\_sm <= Max2Rd1;

elsif (FPGA\_select = '1') then vxi\_sm <= FPGARd1;

elsif (Flash\_select = '1') then vxi\_sm <= FlashRd1;

elsif (SDRAM\_select = '1') then vxi\_sm <= SDRAMRd1;

end if;

elsif (asq = '1' and write\_l = '1') then -- address strobe went low / slot0 write

if (A16board\_select = '1') then vxi\_sm <= ConfigWr1;

elsif (Max2\_select = '1') then vxi\_sm <= Max2Wr1;

elsif (FPGA\_select = '1') then vxi\_sm <= FPGAWr1;

elsif (Flash\_select = '1') then vxi\_sm <= FlashWr1;

elsif (SDRAM\_select = '1') then vxi\_sm <= SDRAMWr1;

end if;

else

vxi\_sm <= root; -- Next state is root (this state)

end if;

--== Configuration Register Access Slot 0 A16 VME transfer

Config\_Reg\_addr(1 downto 0) <= addr\_l(2 downto 1);

vxi\_data\_out(15 downto 0) <= Config\_Reg\_data\_out(15 downto 0); -- Config Reads

Config\_Reg\_data\_in(15 downto 0) <= vxi\_data\_in(15 downto 0); -- Config Writes

--== Max2 A32 Register Access from Slot 0

A32\_Reg\_addr(3 downto 0) <= addr\_l(5 downto 2);

vxi\_data\_out(31 downto 0) <= A32\_Reg\_data\_out(31 downto 0); -- Max2 Reads

A32\_Reg\_data\_in(31 downto 0) <= vxi\_data\_in(31 downto 0); -- Max2 Writes

--== FPGA Memory Access (32 bit words) from Slot 0

\*Reads and Writes with the FPGA must interrupt the DSP to request the bus.

CONSTANT FPGA\_BASE\_ADDRESS : std\_logic\_vector(23 downto 0) := "010110000000000000000000"; --0x580000

local\_addr\_out (23 downto 0) <= addr\_l(25 downto 2) - FPGA\_BASE\_ADDRESS; -- 0x0160 0000 divided by 4 = 0x0058 0000

Cyclone\_CSn <= '0';

Wait 2 Clocks -- MAXII\_CLK 31.24 MHz clock from U6 oscillator

vxi\_data\_out(31 downto 0) <= local\_data\_in(31 downto 0); -- FPGA Reads

local\_data\_out(31 downto 0) <= vxi\_data\_in(31 downto 0); -- FPGA Writes

--== FASH Memory Access (8 bit data) from Slot 0

\*Reads and Writes with the FLASH must interrupt the DSP to request the bus.

\*\* Unlocking, erasing, programming are accomplished via subroutines on the slot 0 processor.

CONSTANT FLASH\_BASE\_ADDRESS : std\_logic\_vector(23 downto 0) := "000110000000000000000000"; --0x180000

local\_addr\_out (23 downto 0) <= addr\_l(25 downto 2) - FLASH\_BASE\_ADDRESS; -- 0x0060 0000 divided by 4 = 0x0018 0000

Flash\_ChipSeln <= '0';

Wait 4 Clocks -- MAXII\_CLK 31.24 MHz clock from U6 oscillator

vxi\_data\_out(7 downto 0) <= local\_data\_in(7 downto 0); -- FPGA Reads

local\_data\_out(7 downto 0) <= vxi\_data\_in(7 downto 0); -- FPGA Writes

--== DSP\_Internal and SDRAM Memory Access (32 bit data) from Slot 0

\*Reads and Writes with the SDRAM must interrupt the DSP to request the bus.

\*\*DSP internal RAM and SDRAM is accessed indirectly through the DSP processor.

The address bits (23 downto 0) are delivered to the DSP via the databus and IRQ0 interrupt

(READ\_INTERRUPT or WRITE\_INTERRUPT) to the DSP.

local\_data\_out(31 downto 0) <= "00000000" & addr\_l(25 downto 2); -- initial delivery of SDRAM address

-- READ DSP/SDRAM: Wait for DSP to read address, fetch data and write data to Max2

vxi\_data\_out(31 downto 0) <= local\_data\_in(31 downto 0);

OR

-- WRITE DSP/SDRAM: Wait for DSP to read address, then Max2 puts data on local databus.

local\_data\_out(31 downto 0) <= vxi\_data\_in(31 downto 0);

/\*! DSP Interrupt Service Routine

NAME: DSP\_Read - Routine to carryout Slot0 Read request

DESCRIPTION:

This function is used to read on board memory. This memory may

be within the SHARC itself or external to the SHARC, i.e. SDRAM, FPGA,

etc. The address to read is first retreived from the MAX II and

translated from Slot 0 VXI space to DSP space using Addr\_decode()

and then the data is read from the MAX II and written to the

decoded address.

Once the memory is accessed the value is read by the

SHARC and then written back to the MAX II data register

at address VXI\_DATA\_REG

INVOKED BY : IRQ0 (010 interrupt)

--==========================================================================================

/\*!

NAME: Addr\_decode - Decoding routine from VXI space to DSP space

DESCRIPTION:

This function supports reading and writing memory on the MFC board, serving as a support function to DSP\_Read() and

DSP\_Write(). The function first translates the Slot 0 VXI address which is a byte address into a 32-bit word

address by dividing the VXI address by 4.

The resulting DSP address is compared to the memory map of the board to find the memory region requested, DSP internal memory, DSP SDRAM, FPGA, etc. Once the region is determined the actual address in DSP space is calculated and returned

to the calling function.

INVOKED BY : Multiplexed interrupt DSP\_Read() or DSP\_Write()

OUTPUTS : i0 - "pointer" to the exact address

AUTHOR : P. Varghese

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Addr\_decode: //Returns decoded address in i0

//r2 = FEXT r1 BY 2:24; //Bit 25 to 2 represents the offset from the base address for 32-bit word addresses

r2 = r1; //Extraction now done in MaxII

r3 = DSPMEM\_UPPR\_LMT; //Check if DSP internal memory access

comp(r2,r3); // #define DSPMEM\_UPPR\_LMT 0x100000; // = 0x0040 0000 divided by 4

if lt jump AccessDSPIntMem;

r3 = SDRAM\_UPPR\_LMT; //Check if SDRAM access

comp(r2,r3); // #define SDRAM\_UPPR\_LMT 0x140000; // = 0x0050 0000 divided by 4

if lt jump AccessSDRAM;

i0 = r3; //Address is outside range, so use upper address limit

rts;

AccessDSPIntMem:

r4 = DSP\_DM\_BASE\_ADDRESS; // #define DSP\_DM\_BASE\_ADDRESS 0xB8000; // = DSP memory space seg\_dmda DM RAM

r4 = r4 + r2;

i0 = r4; //Further decoding may be required

rts;

AccessSDRAM:

r3 = DSPMEM\_UPPR\_LMT; // Subtract base address

r2 = r2- r3;

//Add SDRAM address offset

r4 = SDRAM\_BASE\_ADDRESS; // #define SDRAM\_BASE\_ADDRESS 0x00200000;

r4 = r2 + r4;

i0 = r4;

rts;

Addr\_decode.end:

nullRoutine:

r0 = 0;

dm(m6,i0) = r0; // Write a 0 to MSG\_REG0

rts;