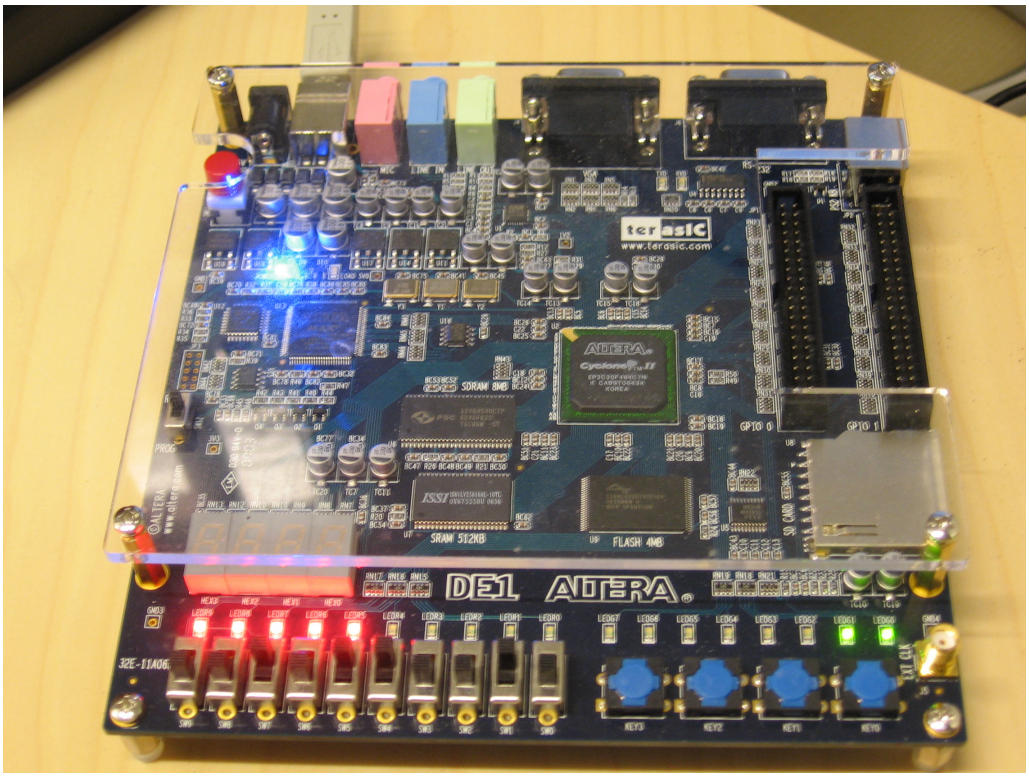
Altera Innovate Nordic 2007

Final Project Report

8/28/2007



Project name:	Leon3 MP on Altera FPGA
Team name:	Team Hervanta
Team ID:	IN00000033
Team members:	Hannu Penttinen
	Tapio Koskinen
Email Addresses:	hannu.penttinen@tut.fi
	tapio.koskinen@tut.fi
Contact No:	00358503708348
Instructor:	Marko Hännikäinen

Introduction 4

Symmetric multiprocessor system 5

eCos 7

Installation 7

Configuring and compiling eCos 8

Test software 8

Other information 8

AMBA-to-HIBI interface 9

Dma_ctrl 10

Dma_tx 11

Dma_rx 13

Timetable 15

Distribution of work 15

Conclusions 15

References 16

INTRODUCTION

Growing need for increasing computational power while keeping power consumption low in handheld devices, such as mobile phones and PDAs, has lead to increased usage of multiprocessor systems-on-chip. By using multi-processor systems the required performance can be achieved at lower power consumption than in single-processor systems. The most power efficient solution would be to use application-specific hardware, but it takes a lot of effort to design them. Multiprocessor systems provide a reusable and versatile yet energy efficient solution with much less design effort.

The goal of our project was to implement a configurable SparcLeon3 based multiprocessor system on Altera Cyclone FPGA with arbitrary number of SparLeon3-processors. Ecos was chosen as the operating system since it's been ported to SparcLeon3 by Gaisler research and it's royalty-free. Another goal was to implement an AMBA-to-HIBI-interface to provide a DMA access from AMBA to HIBI-bus [1]. The HIBI-bus could be used for inter-processor communication and therefore the AMBA-bus could be used as processor's internal bus. Our system can be used for prototyping various multi-processor systems with arbitrary number of processors on FPGA. This enables concurrent software and hardware development which in turn decreases the development time.

The starting point for our work was a project we carried out at Tampere University of Technology's Project course last spring. In the course we implemented a similar system on Nios Development Board, Stratix Professional Edition.

In this project, we achieved the following results:

- Easily configurable symmetric multiprocessor system of 2 SparcLeon3-processors with eCos RTOS
- Implemented AMBA-to-HIBI DMA interface
- Implemented AMBA-to-HIBI DMA driver for eCos

Each of the results is discussed in more detail in the following chapters.

SYMMETRIC MULTIPROCESSOR SYSTEM

Our goal was to implement a similar system to a system described in [3]. Our planned system is also depicted in figure 1. The idea was that both processors had their own instruction and data memories and that they both were totally independent of each other. This way the software would be easy to compile and link, since there would be no issues with linking addresses etc. as it would be with shared memories. However, due to the memory requirements of eCos, we noticed that there was not enough on-chip memory for implementing the system.

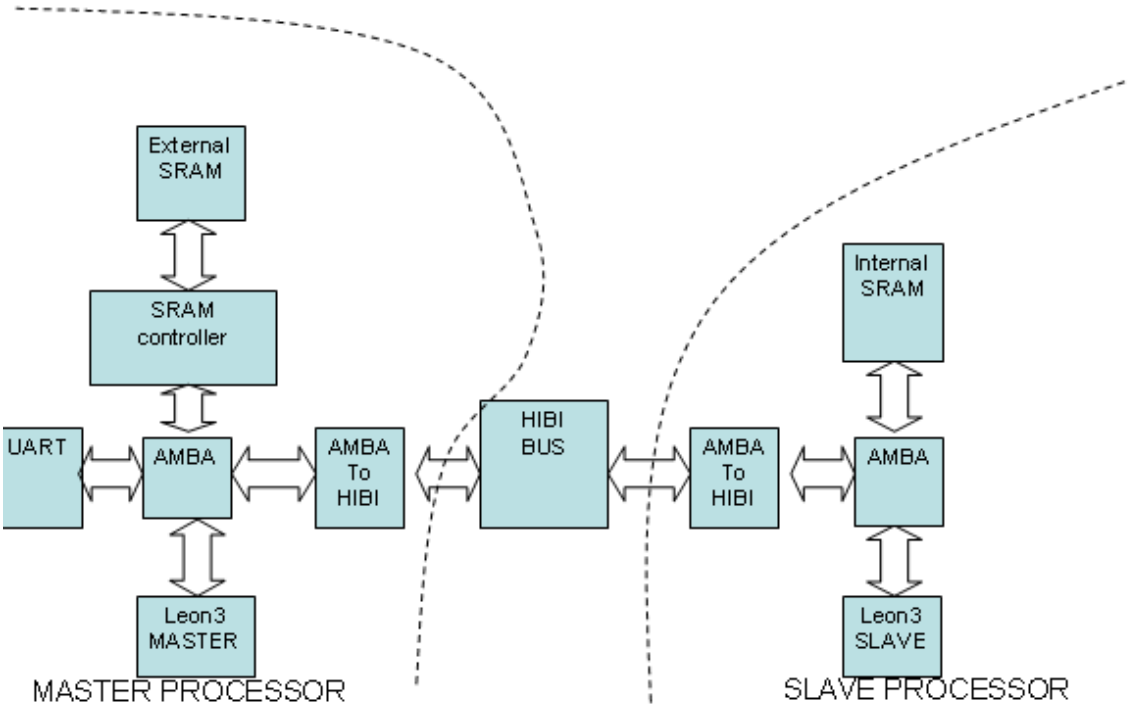


Figure 1. Planned multiprocessor system

Our next proposal was to use external SRAM as shared program memory for the processors. Data memories would still be non-shared. In order to do this we planned to connect the slave processor's AMBA bus to master's AMBA bus with uni-directional AMBA-bridge to provide slave access to external SRAM program memory. This attempt failed, since the AMBA-bridge was available only in the commercial version of Gaisler Reasearch's GRLIB.

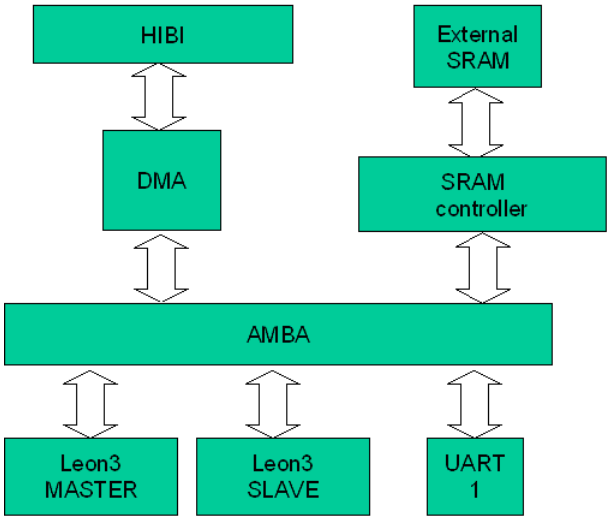


Figure 2. Realized multiprocessor system

Finally we decided to implement a symmetric multiprocessor system depicted in figure 2. The number of Leon3's can be selected by changing one VHDL-generic. Due to limited resources of Cyclone FPGA, only two processors were implemented in our system. The system has two AMBA-to-HIBI DMA units for testing the AMBA-to-HIBI interface. In a real system only one DMA would be used.

The originally planned system could be derived from this system with small effort if an AMBA-bridge was available: first implementing one instance with one Leon3, AMBA-to-HIBI DMA, external SRAM and an AMBA-bridge and then implementing second instance with one Leon3, AMBA-to-HIBI DMA and an AMBA-bridge. Finally these two instances could be connected with AMBA-bridge to provide program memory access to slave processor. This system would be the same kind of architecture that is used in [3] and it's depicted in figure 3.

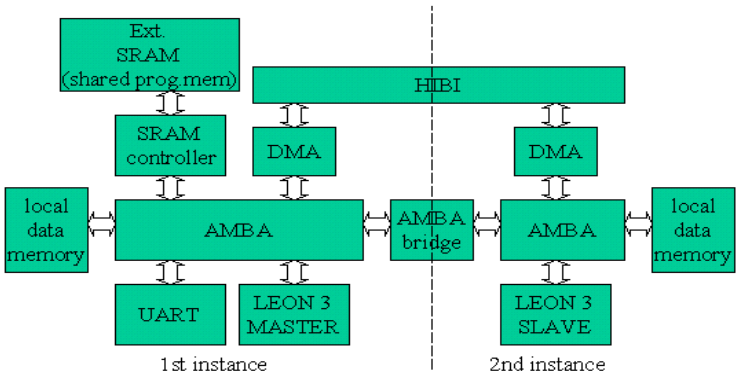


Figure 3. Proposed system

ECOS

INSTALLATION

Five things are needed to get eCos running on FPGA:

1. Cygwin
2. eCos configuration tool
3. eCos repository
4. Cross-compiler for Sparc architecture
5. Grmon for downloading software to FPGA.

These can be found in following addresses: http://www.cygwin.com/setup.exe,
http://www.ecoscentric.com/snapshots/configtool-060710.exe.bz2
ftp://gaisler.com/gaisler.com/ecos/src/ecos-rep-1.0.7.tar.gz,
ftp://gaisler.com/gaisler.com/bcc/bin/windows/sparc-elf-3.4.4-1.0.29d-cygwin.tar.bz2,
ftp://gaisler.com/gaisler.com/grmon/GrmonRCP-eval-0.8.zip.

It is also possible to configure eCos on Linux, in which case other versions of these programs are needed from Gaisler's website (and naturally Cygwin is not needed at all).

Installation is pretty straightforward. Cygwin is fine with default settings. Others can be unpacked basically anywhere ("tar xvf <package name>", configtool: "bunzip2 <file>"). eCos configuration tools needs the location of eCos repository (packages-directory) and the location of cross-compiler (sparc-elf-3.4.4/bin). Repository location can also be specified with ECOS_REPOSITORY environment variable. GrmonRCP needs Java, but if that is a problem there is also a command line tool for downloading software to FPGA (ftp://gaisler.com/gaisler.com/grmon/grmon-eval-1.1.21.tar.gz).

Before starting with the configuration tool, we install our own modifications to eCos repository. There is Dmahibi-package/driver, which should be unpacked to ecos-repository/packages/io/dmahibi. Then there is a modified ecos.db, which has definitions for dmahibi-package and should be copied to ecos-repository/packages-directory. Now we are ready to start the eCos configuration tool.

CONFIGURING AND COMPILING ECOS

For a quick start there is leon3.ecc file enclosed, which has ready-made configuration. These are the steps to create it: Open configuration tool and choose Build-> Templates. Choose Leon3 processor. Then choose Build->Packages, Dma for Hibi, Add, OK. Now let's enable SMP support. Go to eCos HAL / Sparc architecture and check SMP support. Then go to eCos kernel and check SMP support. Now save the configuration with File->Save as.

To compile eCos choose Build->Generate Build Tree and Build-> Library. In the directory where configuration was saved (ecc-file) there will be created directories xyz_build and xyz_install. Xyz_build contains object files which are not necessarily needed anymore (naturally if left alone they speed up the following compiling processes). In the install directory there are many header files (include-directory) and in the lib directory there is a library libtarget.a which is linked into applications and target.ld, a linker script.

Applications can be compiled and linked with eCos on Cygwin command line with the following command:

sparc-elf-gcc -g -Ixyz_install/include -Lxyz_install/lib -Ttarget.ld -nostdlib hello.c -o hello

-I tells where headers are located. -L tells where library is. -Ttarget.ld specifies the linker script which tells the memory addresses to linker. -nostdlib tells the compiler not to use standard libraries (everything should be in libtarget.a). -g just tells to add debugging information. Hello.c is the code file and -o hello tells linker to name binary file as "hello".

TEST SOFTWARE

Test software starts three threads which sleep somewhat random (but short) time. First thread sends every now and then data over HIBI to thread number three. Second thread does basically nothing. eCos schedules these threads automatically on all CPUs. Source code and binaries are attached.

Downloading compiled software to FPGA is done with Grmon. After the FPGA board is programmed, reseted (key0) and debugger enabled (key1), GrmonRCP can be started. Choose Initialize target, Serial, com1 (this can vary) and 115200. Also enable UART loopback. Load program file and click Run.

OTHER INFORMATION

There are good eCos manuals Sourceware's website. There is a user guide, a reference manual and a component writer's guide. These can be found at http://ecos.sourceware.org/docs-2.0/. Then there is also a mailing list for questions: http://ecos.sourceware.org/ml/ecos-discuss/.

AMBA-TO-HIBI INTERFACE

This block provides a DMA connection from AMBA-bus to HIBI-bus. It is divided into three sub-blocks: dma_ctrl, which functions as a configuration unit, dma_tx, which implements one transmit (tx) channel and dma_rx, which implements an arbitrary number of receive (rx) channels.

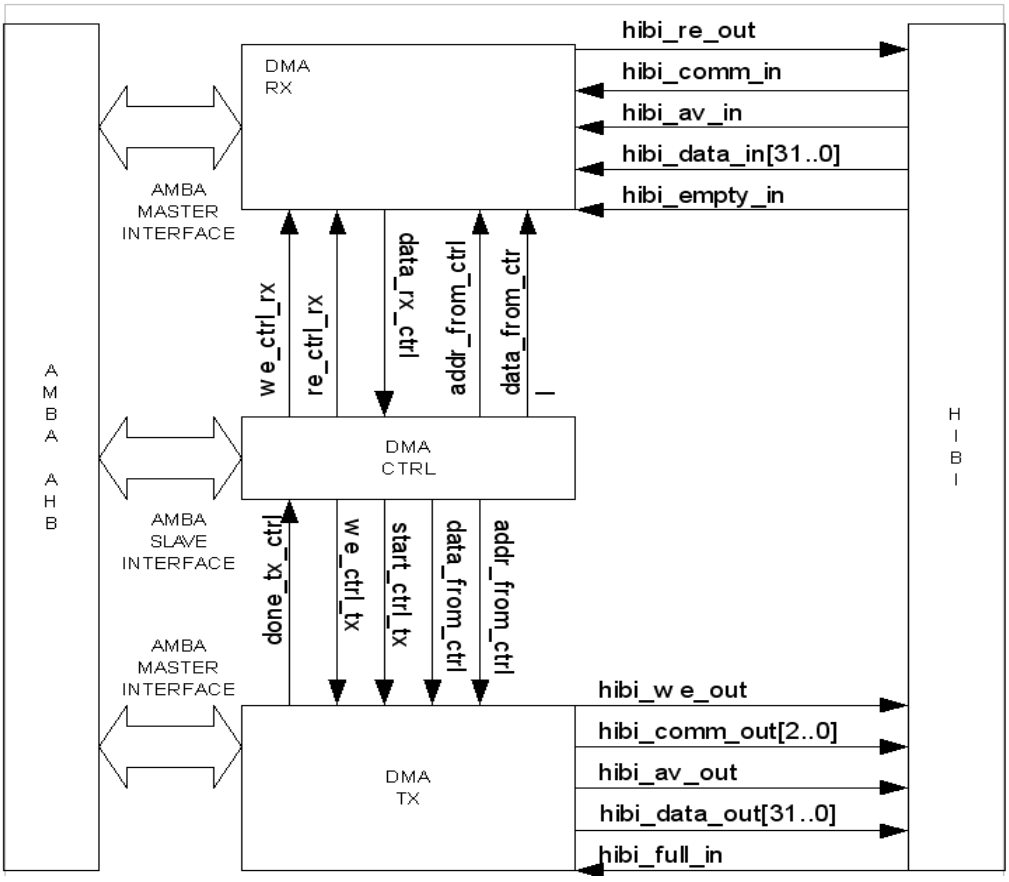


Figure 4. Block diagram of AMBA-to-HIBI interface

DMA_CTRL

This block is connected to AMBA AHB-bus with slave interface. Its interface ports are described in table 1. Dma_ctrl functions as a configuration unit and the processor can read and write the tx- and rx-channels' configuration registers through this block. Dma_ctrl includes one state-machine which is depicted in figure 5.

Port	Direction	Function
clk	in	clock
rst_n	in	reset
amba_slv_in	in	AMBA slave interface inputs
amba_slv_out	out	AMBA slave inteface outputs
rx_data_in	in	Data from dma_rx
tx_done_in	in	Tx-done from dma_tx
rx_re_out	out	Read enable to dma_rx
rx_we_out	out	Write enable to dma_rx
tx_we_out	out	Write enable to dma_tx
rx_addr_out	out	Channel- and register address to dma_tx
tx_addr_out	out	Register address to dma_tx
chan_data_out	out	Write data to both dma_rx and dma_tx
tx_start_out	out	TX-start for dma_tx

Table 1. Dma_ctrl interface

The function of the block is following. The block waits for read/write requests from AMBA and propagates them to either dma_tx or dma_rx depending on the request address. If request was a read request, dma_tx/dma_rx block returns the value of requested register to dma_ctrl which in turn sends it to requester.

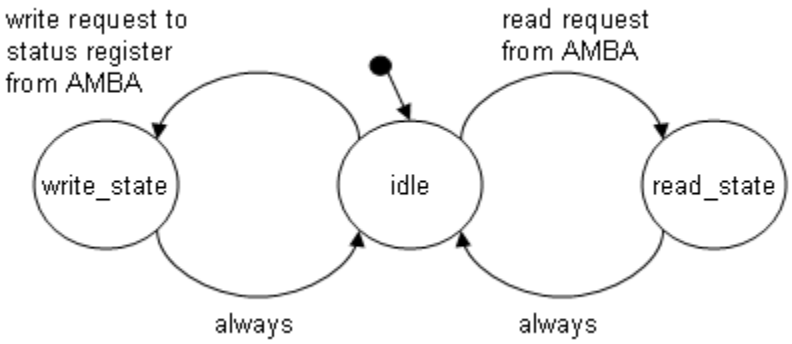


Figure 5. Dma_ctrl state-transitions

DMA_TX

Dma_tx implements one transmit channel. The interface of this block is described in table 2. The block contains tx-configuration registers (table 3) and two state-machines. One state machine handles the AMBA interface and the other the HIBI interface. The state-transitions of state machines are depicted in figures 6 and 7. The configuration registers are set according to write commands from dma_ctrl.

Port	Direction	Function
clk	in	clock
rst_n	in	reset
amba_mst_in	in	AMBA master interface inputs
amba_mst_out	out	AMBA master interface outputs
ctrl_data_in	in	write data from dma_ctrl
ctrl_addr_in	in	register address from dma_ctrl
ctrl_we_in	in	write enable from dma_ctrl
tx_done_out	out	tx-done to dma_ctrl
tx_start_in	in	tx-start from dma_ctrl
hibi_we_out	out	HIBI write enable
hibi_comm_out	out	HIBI command
hibi_full_in	in	HIBI full
hibi_data_out	out	write data to HIBI
hibi_av_out	out	HIBI address valid

Table 2. Dma_tx interface

The transmission begins when dma_ctrl sets tx_start_in signal high. Next the AMBA-state machine starts requesting tx-data from address stored in mem_addr-register. When data is available from AMBA, HIBI-state machine sends data to HIBI and mem_addr is incremented. When the configured amount of data is successfully sent to HIBI, transmission ends and tx-start signal goes to value '1'. The value of tx-start can be read from status register bit 16.

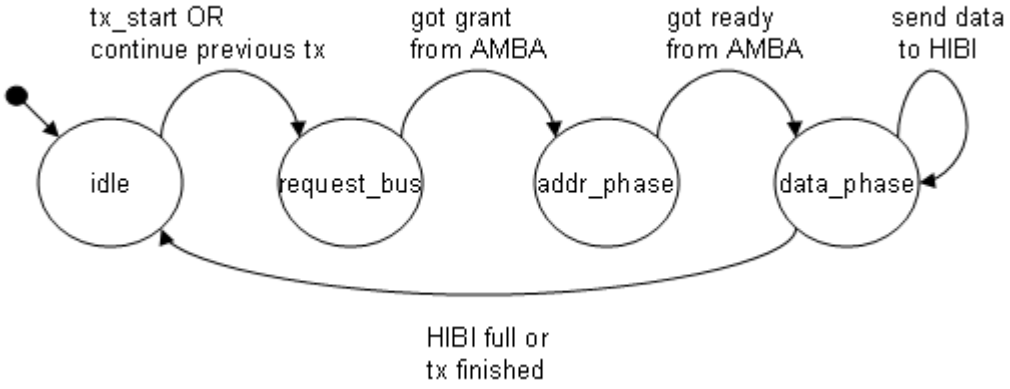


Figure 6. Dma_tx AMBA state machine

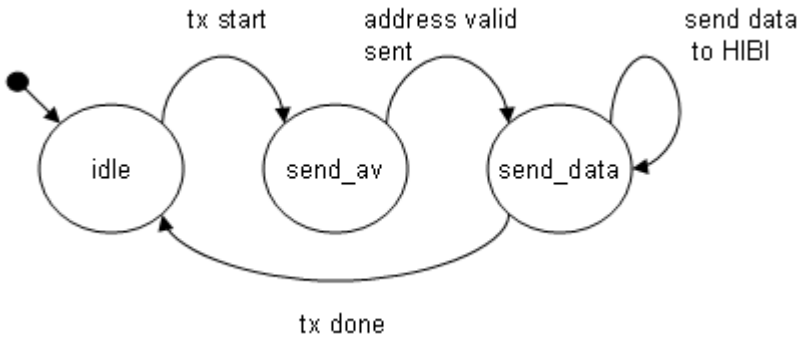


Figure 7. Dma_tx HIBI state machine

Address	Register	Purpose
0x20	mem_addr	Source address of data
0x24	amount	Amount to send
0x28	comm	HIBI command
0x2C	HIBI_addr	HIBI destination address

Table 2. Dma_tx registers

DMA_RX

This block implements the receive channels of AMBA-to-HIBI interface. The interface of this block is depicted in table 4. The block contains an arbitrary number of rx-channels (number is chosen by VHDL-generic) and their configuration registers (table 5). The block also contains one state-machine which handles the AMBA interface. State machine is depicted in figure 8.

Port	Direction	Function
clk	in	clock
rst_n	in	reset
amba_mst_in	in	AMBA master interface inputs
amba_mst_out	out	AMBA master interface outputs
ctrl_data_in	in	write data in from dma_ctrl to register
ctrl_addr_in	in	channel and register address from dma_ctrl
ctrl_we_in	in	write enable from dma_ctrl
ctrl_re_in	in	read enable from dma_ctrl
ctrl_data_out	out	register value output to dma_ctrl
hibi_re_out	out	HIBI read enable
hibi_empty_in	in	HIBI write enable
hibi_data_in	in	HIBI data input
hibi_av_in	in	HIBI address valid input

Table 4. Dma_rx interface

The configuration registers function the same way as in dma_tx except the request address includes the channel number in addition to register address. The bits 4..0 contain the register address and bits [4 + n_chan_bits_g .. 4] contain the channel number. N_chan_bits_g is a VHDL-generic, which tells the number of bits used to index rx-channels. It's value should be set to ceil(log2(number_of_channels)).

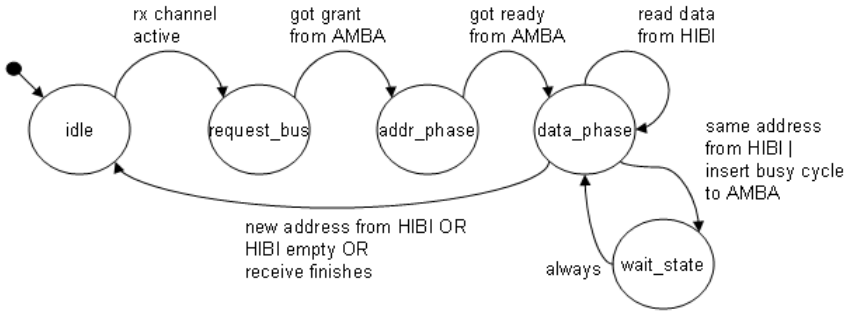


Figure 8. Dma_rx state machine

Receive channel is activated by writing '1' to a bit corresponding channel number in init_chan-register. After this, the activated channel waits until an address matching to receive_addr-register arrives from HIBI. When the address arrives, state-machine requests AMBA-bus and, after getting grant from AMBA, starts sending data from HIBI to AMBA. The destination address is in mem_addr-register. If AMBA is not ready to receive data, HIBI read is stalled until AMBA is ready. Data receiving continues until a new address comes from HIBI, HIBI gets empty or all expected data is received.

When receive finishes, the bit corresponding to channel number goes to '1' in IRQ source-register. If interrupts are allowed in config-register(table 6), an interrupt is given to AMBA. The interrupt is acknowledged by writing '1' to bit corresponding the channel number in IRQ source-register

Address	Register	Purpose
0x00	mem_addr	Destination start address for received data
0x04	receive_addr	HIBI-address where to receive data
0x08	amount	Amount of data to be received
0x0C	curr_addr	Current destination address for received data
0x10	config/status	Lower 16-bits are status register, upper configuration register(table 6)
0x14	init_chan	Initialize RX-channel (one-hot encoded, common for all RX-channels)
0x18	(RESERVED)	-
0x1C	IRQ source	RX-channel IRQ register(one-hot encoded, common for all RX-channels)

Table 5. Dma_rx registers

Bit	31..18	17	16	15..2	1	0
Purpose	(RESERVED)	RX-busy	TX-done	(RESERVED)	RX IRQ enable	TX-start

Table 6. Status/Config register

TIMETABLE

What has been done for this competition relates to a course done at the TUT. The work for that project was started at the end of 2006, and some 600 hours have been used in total for that TUT-project and this competition. Naturally these projects differed and not all work used on TUT-project was useful on this competition, and also some extra work was needed for the competition. Implementing AMBA-to-HIBI interface took roughly 200 hours, eCos 100 hours and multiprocessor system 100 hours.

DISTRIBUTION OF WORK

Hannu Penttinen worked on implementing the Leon3 Sparc processor and other hardware elements on the FPGA. Tapio Koskinen worked on eCos and test software. Aarno Tenhunen didn't participate in the project after all.

CONCLUSIONS

The goal of the project was to create an asymmetric multiprocessor system using SparcLeon3-processors and Altera Cyclone FPGA. However, due to the problems encountered, a symmetric multiprocessor (SMP) system was implemented. SMPs have also many applications and they are used in growing numbers nowadays. Our final system is an excellent platform for prototyping such systems on FPGA and it enables concurrent software and hardware designing on SMPs which can reduce design time considerably. Our system is also a good starting point for implementing a prototyping platform for asymmetric processor systems.

REFERENCES

[1] Erno Salminen, Vesa Lahtinen, Tero Kangas, Jouni Riihimäki, Kimmo Kuusilinna, Timo D. Hämäläinen, **"HIBI v.2 Communication Network for System-on-Chip"**, Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS IV), Samos, Greece, July 18-20, 2004, Vol.LNCS 3133 Computer Systems: Architectures, Modeling, and Simulation, A.D. Pimentel, S. Vassiliadis, (eds.), pp. 412-422, Springer-Verlag, Berlin, Germany.

[2] Ari Kulmala, **"Multiprocessor System with General-Purpose Interconnection Architecture on FPGA"**, Tampere, Finland, 2005, 73 pages, Tampere University of Technology

[3] Olli Lehtoranta, Erno Salminen, Ari Kulmala, Marko Hännikäinen, Timo D. Hämäläinen, **"A Parallel MPEG-4 Encoder for FPGA Based Multiprocessor SoC"**, 15th International Conference on Field Programmable Logic and Applications (FPL 2005), Tampere, Finland, August 24-26, 2005, pp. 380--385, Springer LNCS.