International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 5, August 2014, PP 51-59
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

AMBA Protocol for ALU

K Swetha

Student,
Dept of Electronics and Communication Engineering,
Sri Venkateswara College of Engineering & Technology
(Autonomous), Chittoor, A.P, India
kswetha.ec@gmail.com

G Ramakrishna

Professor,
Dept of Electronics and Communication Engineering,
Sri Venkateswara College of Engineering & Technology
(Autonomous), Chittoor, A.P, India
svec_grk@rediffmail.com

Abstract: *The proposed project implementation is on AMBA IO System. The bus interface used in the project is AXI Interface Bus. Master and Slave interfaces will be developed according to the proposed AMBA design. The description of the AMBA protocol and various blocks are mentioned below. The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers AXI acts as the high-performance system back bone bus. AXI supports the efficient connection of processors, on-chip memories and off-chip outer memory interfaces with low-power peripheral macro cell functions. is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques. Memory devices such as SRAM, ROM and FIFO are used as Slave devices and Processor Control Designs are used as master devices for AMBA IO System. The intercommunication will be performed using AMBA Bus with Arbiter. We can have Single Master Single Slave design as well as Multiple Master Multiple Slave devices as per user requirements. The design is implemented using Verilog HDL language. The verification of the design is done using Verilog Test bench. Verification has become one of the time consuming task in design and verification cycle and hence takes up the major chunk of the resources. Since the serial communication protocols are preferred means of data communication, application of Verilog environment for the verification of AMBA IO System has tremendous implementation scope. For the designed AMBA IO system, it is tested automatically by reusable reliable layer verification platform.*

Index Terms: *AMBA protocol, SRAM, system on chip memories, ALU, AXIS bus, FIFO*

1. Introduction

The AMBA protocol is an open standard, on-chip interconnect requirement for the connection and management of functional blocks in a System-on-Chip (SoC). It facilitates easy implementation of different macro functions operating at different frequencies (High frequency). The AMBA protocol is technology independent, since we implement this standard for any operating frequency range.

The scope of AMBA has gone far beyond microcontroller devices, and is at the present widely used on a range of ASIC and SoC parts including applications processors used in modern portable mobile devices like Smartphone’s

AMBA is a registered trademark of ARM Limited, and is an unlock standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC). It facilitates right-first-time development of multi-processor designs with large numbers of controllers and peripherals.

AMBA was introduced by ARM Ltd in 1996. The first AMBA buses were highly developed System Bus (ASB) and Advanced Peripheral Bus (APB). In its 2nd version, AMBA 2, ARM added AMBA High-performance Bus (AHB) that is a single clock-edge procedure. In 2003, ARM introduced the 3rd generation, AMBA 3, with AXI to reach even higher performance inter-connects and the Advanced Trace Bus (ATB) as part of the Core Sight on-chip debug and trace solution.

AMBA IO is used as interface between peripherals by using APB bridge. AMBA can be used as multi master and multi slave device by using AXI bus for its implementation. In this project the design consists of single master and multi slaves.

The AMBA Bus matrix is a Self-Motivated Arbitration scheme proposed three methods for data transmitting from master to slave for on chip communication. Multilayer advanced extensible interface bus (ML-AXI) bus matrix employs slave-side arbitration. Slave-side arbitration is special from master-side arbitration in terms of request and grant signals as, in the former, the master merely starts a burst transaction and waits for the slave response to proceed to the then transfer. Therefore, in the former, the unit of arbitration can be a transaction or a transfer. However, the ML-AXI bus matrix of ARM offers only transfer-based fixed-priority and round-robin arbitration schemes. One can design and implement a flexible arbiter for the ML-AXI bus matrix to support three priority policies fixed priority, round robin, and dynamic precedence and three data multiplexing modes transport, transaction, and desired transfer length.

2. LITERATURE SERVEY

The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. It is supported with the ARM Limited Corporation with wide cross-industry participation.

Four different buses are defined within the AMBA specification:

- The Advanced extensible interfaces(AXI)
- The Advanced High-performance Bus (AHB)
- The Advanced System Bus (ASB)
- The Advanced Peripheral Bus (APB).

  A test methodology is integrated with the AMBA specification which provides an Infrastructure for modular macro cell test and diagnostic access.

AMBA 3 requirement defines four buses/interfaces:

- Advanced eXtensible Interface (AXI3 or AXI v1.0) - generally used on ARM Cortex-A processors including Cortex-A9
- Advanced High-performance Bus Lite (AHB-Lit v1.0)
- Advanced Peripheral Bus (APB3 v1.0)
- Advanced Trace Bus (ATB v1.0)

AMBA 2 requirement defines three buses/interfaces:

- Advanced High-performance Bus (AHB) - widely used lying on ARM7, ARM9 and ARM Cortex-M based designs
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB2 or APB)

AMBA specification (First version) defines two buses/interfaces:

- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)

  The timing aspects and the voltage level on the bus are not dictated by the specifications.

Advanced eXtensible Interface (AXI)

AXI, the third generation of AMBA interface defined in the AMBA 3 specification, is targeted at more performance, high clock frequency system designs and includes features that make it suitable for high speed sub-micrometer interconnect:

Objectives of the newest generation AMBA interface are to:

- be suitable for more-bandwidth and low-latency designs
- enable more-frequency operation without using complex bridges
- meet the interface supplies of a wide range of components
- be suitable for memory controllers with more initial access latency
- provide flexibility in the execution of interconnect architectures
- Be backward-compatible through existing AHB and APB interfaces.

The key features of the AXI protocol are:

- separate address/manage and data phases
- support for unaligned data transfers with byte strobes
- burst-based transactions through only start address issued
- separate read and write data channels to allow low-cost Direct Memory Access (DMA)
- ability to issue many outstanding addresses
- out-of-order deal completion
- easy addition of register steps to provide timing closure.
- As well as the data transfer protocol, the AXI protocol includes optional extensions that
- cover signaling for less-power operation.

Advanced High-performance Bus (AHB)

The AMBA AHB is for more-performance, high clock frequency system modules.

The AHB acts as the high-performance system *backbone* bus. AHB supports the able connection of processors, on-chip memories and off-chip outside memory interfaces with low-power peripheral macro cell functions. AHB is and specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

In addition to before release, it has the following features:

- single edge clock protocol
- split transactions
- several bus masters
- burst transfers
- pipelined operations
- single-cycle bus master handover
- non-tri state implementation
- Large bus-widths (64/128 bit).

A simple transaction on the AHB consists of an address phase and a subsequent data phase (without wait states: only two bus-cycles). Access to the aim device is controlled through a MUX (non-tri state), thereby admitting bus-access to one bus-master at a time.

AHB-Lite is a subset of AHB formally defined in the AMBA 3 standard. This subset simplifies the design used for a bus with a single master.

Advanced System Bus (ASB)

The AMBA ASB is for high-performance system modules. AMBA ASB is an alternative system bus proper for use where the high-performance features of AHB are not required. ASB and supports the efficient connection of processors, on-chip memories and off-chip outer memory interfaces with low-power peripheral macro cell functions**.**

Advanced Peripheral Bus (APB)

The AMBA APB is for less-power peripherals. AMBA APB is optimized used for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction through either version of the system bus.

APB is designed for low bandwidth control accesses, meant for example register interfaces on system peripherals. This bus have an address and data phase like to AHB, but a much reduced, low complexity signal list (for example no bursts). It have to support 32bit and 66 MHz signals.

Objectives of the AMBA specification:

- to facilitate the *right-first-time* improvement of embedded microcontroller products with one or more CPUs or signal processors
- to be technology-independent and ensure that highly reusable peripheral and system macro cells can be migrated across a diverse range of IC processes and be appropriate for full-custom, regular cell and gate array technologies
- to encourage *modular system design* to improve processor independence, providing a improvement road-map for advanced cached CPU cores and the development of peripheral libraries
- to reduce the silicon infrastructure required to support efficient on-chip and off-chip communication for both operation and manufacturing test.

AMBA Products:

A family of synthesizable intellectual property (IP) cores, AMBA Products licensable from ARM Limited that implement a digital highway in a SoC (System On Chip) for the efficient moving, for high performance, high clock frequency and storing of data using the AMBA protocol qualifications. The AMBA family includes AMBA Network Interconnect (NIC-301), SDRAM, FLASH memory controllers (DMC-34x, SMC-35x), DMA controllers (DMA-230, DMA-330), level 2 cache controllers (L2C-310), etc

A Typical AMBA-Based Microcontroller

An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the outside memory bandwidth, on which the CPU, on-chip memory and further Direct Memory Access (DMA) devices reside. This bus provides a more-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the more performance bus is a bridge to the lower bandwidth APB, where the majority of the peripheral devices in the system are located.

Migration from AHB to AXI

With modern Systems on Chip including multi-core clusters, other DSP, graphics controllers and other sophisticated peripherals, the system fabric poses a critical presentation bottleneck. The AHB protocol, even in its multi-layer configuration cannot keep up with the demands of now a day's SoC. The reasons for this include:

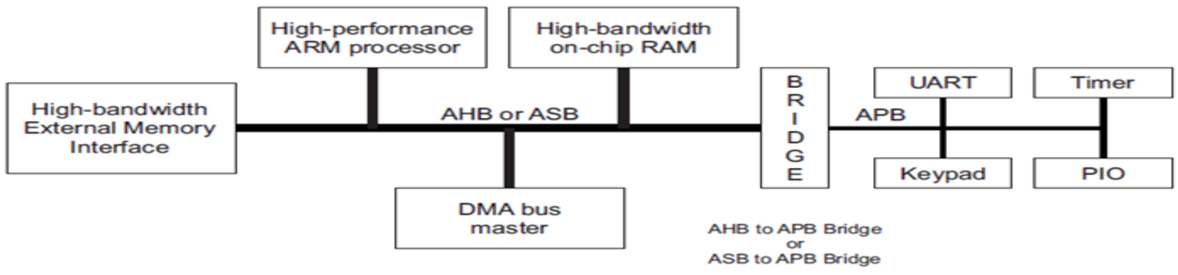

High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
APB
UART
Timer
Keypad
PIO
AHB to APB Bridge or ASB to APB Bridge

Figure: 1 A typical AMBA system

1. AHB is transfer-oriented. With every one transfer, an address will be submitted and a single data item will be written to or read from the selected slave. All transfers will be initiated with the master. If

the slave cannot respond immediately to a transfer request the master will be stalled. Each master can have single one outstanding transaction.

2. Sequential accesses (bursts) consist of consecutive transfers which indicate their relationship by asserting HTRANS/HBURST accordingly.

3. Although AHB systems are multiplexed and thus have independent read and write data buses, they cannot work in full-duplex mode.

An AXI interface consists of up to five channels which can operate largely independently of each other. Each channel uses the similar trivial handshaking between source and destination (master or slave, depending happening channel path), which simplifies the interface design.

Unlike AHB concept is not an afterthought but is the central focus of the protocol design. In AXI3 all connections are bursts of lengths between 1 and 16. The addition of byte allow signals for the data bus supports unaligned memory accesses and store merging.

The communication between master and slave is transaction-oriented, where each contract consists of address, data, and reply transfers on their corresponding channels. Apart starting rather liberal ordering rules there is no strict protocol-enforced timing relation between individual phases of a transaction. Instead each transfer identifies itself as part of a specific transaction by its transaction ID tag. Transactions may entire out-of-order and transfers belonging to different transactions may be interleaved. Thanks to the ID that each transfer carries, out-of-order transactions can be sorted out at destination.

3. PROPOSED BLOCK-DIAGRAM

The AMBA Bus matrix is a Self-Motivated Arbitration scheme proposed three methods for data transmitting from master to slave for on chip communication. Multilayer advanced extensible interface bus (ML-AXI) bus matrix employs slave-side arbitration. Slave-side arbitration is special from master-side arbitration in terms of request and grant signals as in the former, the master merely starts a burst transaction and waits for the slave response to proceed to the after that transfer. Therefore, in the former, the unit of arbitration can be a transaction or a transfer. However, the ML-AXI bus matrix of ARM offers only transfer-based fixed-priority and round-robin arbitration schemes. One can design and implement a flexible arbiter for the ML-AXI bus matrix to support three priority policies fixed priority, round robin, and dynamic precedence and three data multiplexing modes transfer, transaction, and preferred transfer length.

The Basic IO Architecture Blocks are:

- Master: - The master device is a ALU (Arithmetic and Logic Unit). This ALU can do arithmetic operations on complex numbers. Hence it’s the main computation block in the design.
- Slave: - From the architecture we can see that the design comprises of three slaves. The slaves that are used are as follows.
- ROM: - This is a memory block that can store the address and data that are needed in the operation. The data can only be read at any moment of time but can’t be modified.
- SRAM: - This is a memory block that can store the address and data that are needed in the operation. The data can be read and modified at any moment of time.
- FIFO: - FIFO stands for First In First Out and is a register. It’s a kind of stacked memory where data can be read and modified at any instant of time.

Basic Operation of AMBA-IO Blocks:

AMBA-IO system design can be implemented by the ALU, FIFO, SRAM and ROM. This is simple AMBA-IO system where implemented by the AXI bus as the interface

ALU:

In this project ALU performs 22 operations, the output of the ALU can be seen in corresponding slaves FIFO, ROM, SRAM sequentially with the help of arbiter.ALU output is given to the AXI bus as the data input.

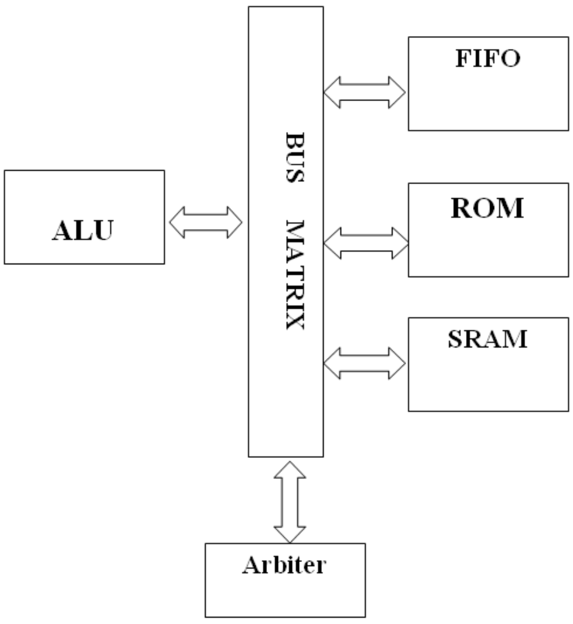


Figure 2. *proposed block diagram*

AXI-BUS:

This AXI-BUS acts as the interface between the master (ALU) and The slave (FIFO, ROM and SRAM) AXI-BUS input is taken from the ALU and corresponding AXI-BUS output is given as the data input to the slaves (FIFO, ROM and SRAM).

FIFO:

FIFO (first in, first out) is the memory which is used as the slave in this project. Input is taken from the AXI-BUS as the data-input and the corresponding information or data is stored in the FIFO memory.

ROM:

ROM (Read only memory) is the memory which is used as the slave in this project. Input is taken from the AXI-BUS as the data-input and the corresponding information or data is taken as the address line, the data in that address line which is stored initially that data is taken as the Rom output because ROM is just a reads data.

SRAM:

SRAM (static random access memory) is the memory which is used as the slave in this project. Input is taken from the AXI-BUS as the data-input and the corresponding information or data is stored in the SRAM memory.

ARBITER:

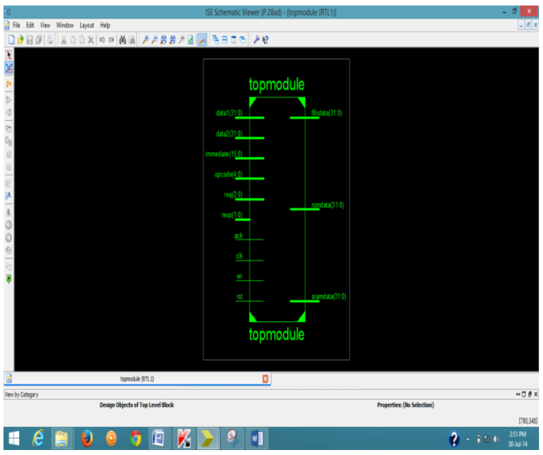
ARBITER is used to select slaves sequentially .In this project round robin-arbiter is used. When request is given to the arbiter, arbiter gets activated and gives the corresponding grant signals as output. By the grant signals the slaves are activated sequentially grant signal is the 3 bit signal when 001 is granted FIFO slave is active and then information id writes or stored in the FIFO memory .when 010 is granted by the arbiter ROM is activated at which address line is selected that address line data will be taken as the output. when 100 is granted SRAM will be selected and the data a is stored in that SRAM data.

BAUD RATE:

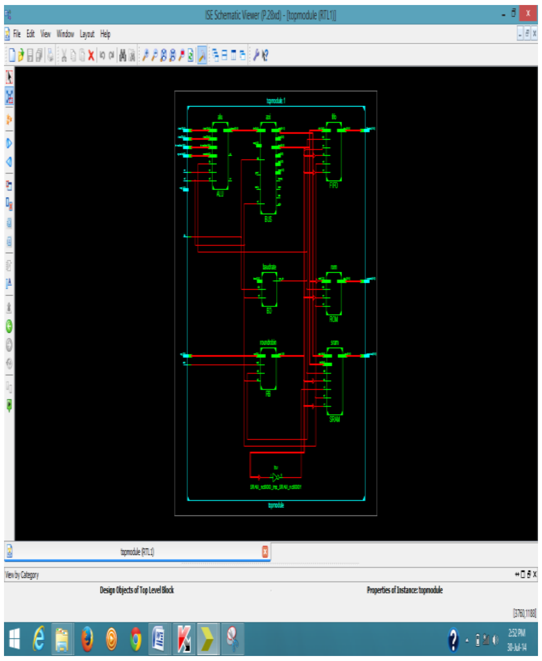
This baud rate is used to synchronize all the blocks that means in this project AXI-BUS produces data-out after certain clock plus to synchronize this clock plus delayed .when clock signal is given as the input to the BAUD RATE and produced synchronized clock signal which is known as the SYS_CLK .this SYS_CLK is given instead of clock signal for the ALU, FIFO, ROM, SRAM, ROUND ROBIN ARBITER.

4. Simulation Results

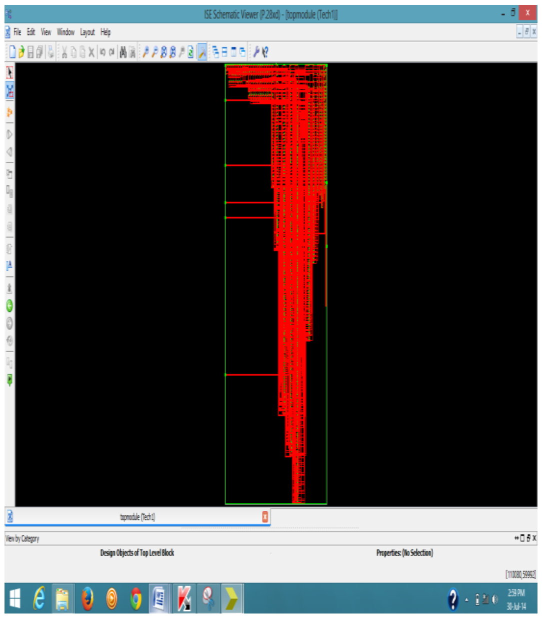
Block Diagram



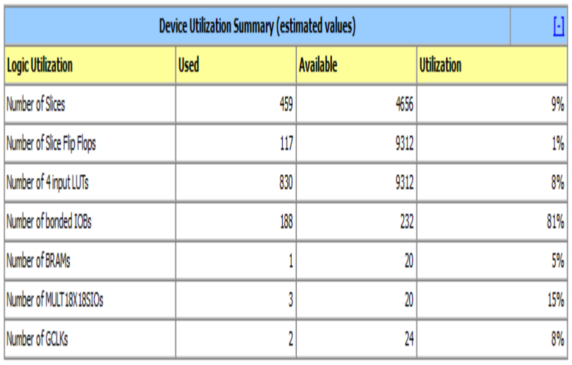
RTL Schematic



Technology Schematic

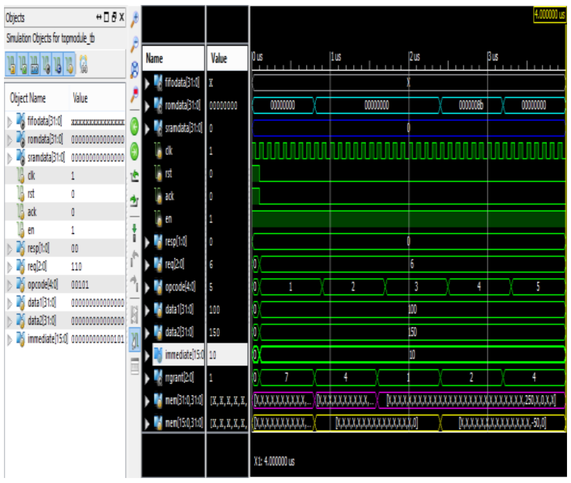


Design Summary



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	459	4656	9%
Number of Slice Flip Flops	117	9312	1%
Number of 4 input LUTs	830	9312	8%
Number of bonded IOBs	188	232	81%
Number of BRAMs	1	20	5%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	2	24	8%

Output Waveform



5. Conclusion

AMBA-IO system is implemented by the AXI-INTERFACE where master is ALU where 23 operations are implemented .this ALU output is seen in the slaves .slaves are FIFO ,SRAM,ROM .The selection of slaves is done by the arbiter which is known as round robin arbiter by the sequential selection of slaves. The axi bus delay is reduced by the baud rate mechanism .the sequential synchronization is implemented by the system clk. all the results are shown .By implementing this techniques controlling of all functional block are done.

References

[1] Jerraya, W. Wolf. Multiprocessor Systems-on-Chips [M], 2007

[2] D.Flynn,AMBA:enabling reusable on-chip designs[J].IEEE Micro Magazine,July-Aug.1997.Volume:17

[3] R.Hofman and B.Drerup.Next generation CoreConnect processor local bus archit-ectture[A],Annual IEEE International ASIC/SOC Conference[C],2002: 221-225.

[4] Silicore.http://www.silicore.net [5] Avalon Bus Specification Reference Manual[S/OL]. :http://www.altera.com. 2003,7.

[5] D.Wingard.MircoNetwork-based intergration for SOCs[A].In:Design Automation Conference,2001.Proceedings[C].2001:673-677.

[6] B,Cordan,An efficient bus architecture for system on-chip design[A].In: Custom Integrated Circuits, Proceedings of the IEEE 1999[C].1999:623-626.

[7] Open Core protocol specification[S/OL].: http://lad.dsc.ufcg.edu.br/ip/OCP-IP- _Open Core Protocol Specification-1.0.pdf

[8] Yang tao, Study on Key Technology of 10M/100M/1000M Adaptive MAC controller[Master thesis], Chinese Academy of Science Institute of Computing Technology, 2008,12. (in Chinese)

[9] Zhou yanchao, Research on Design of On-chip Bus and IP core Integeration Technology of the SOC platform [master thesis]□ Northwestern Polytechnical Univesity, 2004,12.(in Chinese)

[10] ARM,Inc.AMBA® AXI Protocol Version:2.0 Specification[S].2003

[11] Haalen R,Malhotra R,de-Heer A.Optimized routing for providing thernet LAN services[J].IEEE Communications Magazine,2005,43(11):158-164.

[12] Koopman P,Chakravarty T.Cyclic redundancy code(CRC) polynomial selection for embedded networks[J].Dependable Systems and Networks,2004,7:145-154.