XAPP655 (v1.1) January 17, 2003

Mixed-Version IP Router (MIR)

Author: Gordon Brebner

Summary

This application note describes a reference design for a mixed-version IP router (MIR) servicing up to four gigabit Ethernet ports. MIRs are useful where several gigabit Ethernet networks are operating with a mixture of IPv4 and IPv6 hosts and routers attached directly to the networks, and further nodes reached via the routers. A particular benefit of an approach based on the Virtex-II Pro™ family is that the router's functions can evolve smoothly, maintaining router performance as the organization migrates from IPv4 to IPv6 internally, and also as the Internet migrates externally.

The router's range of functions includes straightforward IPv4-to-IPv4 and IPv6-to-IPv6 routing, encapsulation of IPv6 packets within IPv4 packets (and vice versa), and conversion of packets from IPv4 to IPv6 (and vice versa). The choice of function on a per-packet basis and the execution of these functions are carried out by a combination of logic and processor functions. The basic aim is to carry out more frequent, less control-intensive functions entirely in logic, leaving the other functions in the processor.

The reference design targets the XC2VP7 device resources and is configured for the bulk of the incoming packets to be version 4. Such packets are processed and switched entirely by logic, with no internal copying of packets between buffers and no delay between packet receipt and onward forwarding. This involves a specially tailored interconnection network between the four ports. It also requires processing performed in parallel with packet receipt, i.e., multithreading in logic. Version 6 packets, or some rare version 4 cases, are passed to the PowerPC processor. In essence, the PowerPC processor acts as a slave to the logic, rather than the more common master-slave relationship.

The reference design demonstrates the many features of the Virtex-II Pro device: RocketIO™ transceivers, the FPGA fabric (logic and memory), and the embedded PowerPC™ processor, which uses Processor Local Bus (PLB) and Device Control Register (DCR) CoreConnect™ architectures.

Introduction

The MIR **Reference Design** instantiates one point in a broad configurable design space of MIR routers. It is intended to provide a strategic illustration of a Virtex-II Pro Platform FPGA design. Refer to: **http://www.xilinx.com/products/platform/** for additional information on Virtex-II Pro Platform FPGAs.

The main features of the reference design are:

- Exhaustive testing using functional simulation
- Synthesized
- Placed and routed with the timing and resources checked
- Tested on a real device with the main internal component blocks verified

The reference design is not suitable for immediate use as a service IP router. To provide a compact demonstration vehicle, certain routine technical aspects of the networking interface have been omitted. The reference design was not tested with live external network interfaces.

Some of the research results of the MIR design were published at two international conferences held in 2002[1, 2]. These papers are useful sources of additional background and information on the MIR **Reference Design**.

Technical Background

Internet Protocol (IP) Versions 4 and 6

Since 1981, IPv4 has been the basic Internet protocol. IPv6 emerged in 1994. Since 1998 it has been a proposed standard. One of the main motivations was a much-needed extension of the 32-bit IPv4 address space to a 128-bit address space, although there are various other extensions and simplifications. Inevitably migration is a slow process, given the large legacy IPv4 capability. This includes recommendations on mixed-version router and host capabilities during the transition. An initial practical step is the IPv6 test-bed, 6bone, described at: **http://www.6bone.net**

Three Internet requests for comments (RFCs) are particularly relevant. RFC2893[3a] concerns transition mechanisms for IPv6 hosts and routers. It includes mechanisms for dual IPv4/IPv6 nodes and for encapsulation of IPv6 packets over IPv4 tunnels. RFC 2473[3b] concerns generic encapsulation of packets over IPv6 tunnels. RFC 2765[3c] contains a stateless algorithm for translating packets between IPv4 and IPv6 formats. It does not cover address assignment and routing, but this is covered in RFC2766[3d]. A complicating issue in each case is reconciling the differing packet fragmentation mechanisms used. Tunneling over IPv6 is less important at first, since it only becomes a major concern later in the transition to IPv6 when islands of IPv4 remain in a sea of IPv6. Translation between the two IP versions should only happen as a last resort, where an IPv6-only node needs to communicate with an IPv4-only node.

Packet Handling Using Programmable Logic

In summary, the published literature contains examples of programmable logic being used for:

- End-station protocol handling of specific stacks (e.g., TCP/IP by Fallside and Smith[4]) and generalized stacks (e.g., protocol harness of Brebner[5]);
- Simple packet routing (e.g., Lockwood[6,7] et al) with low-width non-CAM lookup; and
- Simple packet filtering (e.g., Dirmar[8] et al and Guccione[9] et al) with high-width CAM lookup.

This is in addition to cases where FPGAs are used as straight ASIC replacements in existing hardware implementations. The key issue probed by MIR is to identify in an overall system the types of packet handling that will benefit from a programmable logic implementation.

Network Processors

The MIR reference design demonstrates that a Virtex-II Pro solution is a viable competitor in the network processor market. The performance attributes, including programmability, and flexibility for use in other applications make it a general-purpose device. The Network Processing Forum[10] is concerned with standardization in the network processor area. This design does not aim for conformance with any of these emergent standards. Rather, the standardization works both to rationalize the competing paradigm of network processors, as well as to point at general architectural features required to support specialized network processing within a whole-system context.

Virtex-II Pro Attributes

The reference design capitalizes on the combination of the four central Virtex-II Pro attributes: programmable logic, PowerPC processors, RocketIO high-speed links, and block memory. The design explores logic/processor interaction and trade-offs for high-speed communications using just the one FPGA in isolation. This application note illustrates some of the possible choices.

The Virtex-II Pro family has a wide range of different attributes, from the XC2VP2 with no PowerPC cores and four RocketIO interfaces, to the XC2VP125 with four PowerPC cores and

24 RocketIO interfaces. The number of system gates and block RAM bits increases across the family. For MIR, precisely one processor is used, thus avoiding multiprocessor operation complications. The target devices are the XC2VP4 or the XC2VP7. The reference design includes up to four RocketIO interfaces, corresponding to the capabilities of the XC2VP4. However, it is targeted at the XC2VP7 to benefit from increased logic real estate.

Virtex-II Pro Development Kit (PDK)

The PDK provides SWIFT simulation models of the PowerPC 405 core and the RocketIO transceiver, IBM CoreConnect bus models, compiler/assembler/linker tools and three example designs. In the kit, Verilog is used for the hardware and C is used for the software (with small amounts of assembler for low-level system initialization).

MIR Specification

Two key demonstrations made by the MIR routers are:

1. The exploitation of the programmability of the hardware, allowing focus on different functions over time.
2. The exploitation of the conventional processor as a slave to the logic for carrying out less-frequent or less time-critical functions, as opposed to just being a master control device.

The design of a mixed-version IPv4/IPv6 router is a demonstration of these criteria. Both protocols have some relatively rare functions that allow tuning the logic to the most frequent functions at a particular time and using a processor to perform the less-frequent functions.

While preserving performance, the design goes beyond the past work on packet handling mentioned earlier and demonstrates the flexibility of a network processor. Although the focus of the design is specifically on the Internet protocol, an attempt is made to provide a framework capable of extending to other protocols, including higher-layer protocols.

The target product is a mixed-mode IP router servicing four gigabit Ethernet ports. This is aimed at organizations operating several gigabit Ethernet networks, with a mixture of IPv4 and IPv6 hosts and routers attached directly to the networks, and further nodes reached via the routers. Gigabit Ethernet is one of the communication standards supported by the RocketIO transceiver.

The range of functions for the MIR router include:

- Straightforward IPv4-to-IPv4 and IPv6-to-IPv6 routing
- Encapsulation of IPv6 packets within IPv4 packets
- Encapsulation of IPv4 packets within IPv6 packets
- Conversion of packets from IPv4 to IPv6, and from IPv6 to IPv4

The choice of function on a per-packet basis can be inferred from the source address, the input port, the output port, and/or the destination address. This choice is associated with the normal basic router choice of the output port for packet forwarding. The decision process and the execution of functions are carried out by a combination of logic and processor. To identify the best balance between logic and processor, depends on the work profile of the router. More frequent, less control-intensive functions are carried out in the logic, and other functions are performed in the processor. Management of the router, including routing tables, is expected to be on the processor.

As is conventional in networking, assume an average IP packet size of 1600 bits. The router needs to maintain an average rate of about 1.25 million packets per second to maximize the use of the four ports. Comparative figures for contemporary ultra-fast IP routers (for example, see reference [11]) place this target figure at the conservative end of the scale.

Networking Environment

The intended operating environment of the router is particularly important because this both restricts the scale of generality necessary and also affects the degree of configurability in response to environmental changes. As previously stated, MIR is designed to be connected to

various gigabit Ethernets. In terms of Internet connectivity, the operating organization is assumed to have a single connection to the global Internet through a router on one Ethernet. MIR is not intended to be this router to the external world, although in principle it could be and without major functional changes.

Network Addressing

The operating organization is assumed to have a single Class B IPv4 address and a single 001/TLA/NLA IPv6 address[3e] allocated. Variation of the IPv4 address type, for example, to one or more Class C addresses, would not change the required functionality of the reference design significantly. Within this address space, the organization allocates addresses to the devices connected to the Ethernet. IPv4 addresses have an 8-bit network identifier for the Ethernet and an 8-bit interface identifier. Again, this partition into two 8-bit quantities could be varied, without requiring major functionality changes. IPv6 addresses have a 16-bit network identifier for the Ethernet and a 64-bit interface identifier. Some upper bounds on the size of the address space follow from the limit on the number of Ethernets (four, in this case, being the number of MIR ports) and the standard limit on the number of stations on an Ethernet (1024, in general, for this networking technology).

Derived Address Table Format

For each connected device, MIR must store information on whether the device is IPv4-only, IPv6-only, or mixed IPv6/IPv4. In particular, MIR must also store this information for the connection to the external Internet. The role of MIR is to allow as much IP connectivity as possible in this mixed-mode setting. However, to simplify the reference design, only support for unicasts is implemented, with no support for multicasts and broadcasts.

A critical issue for a system relying only on on-chip memory involves the storage required for address lookup tables. Taking into account the assumptions about the IPv4 and IPv6 address spaces given above, a compact scheme was devised for MIR, using only 64 bits for each device included. Thus, it is possible to hold routing information for 256 devices in a single block RAM, which is very compact. The 64-bit format is:

Number of bits	Purpose
1	Indicates whether IPv4 protocol is spoken
1	Indicates whether IPv6 protocol is spoken
4	MIR port number (0, 1, …) – only 2 bits used currently
10	Interface number on MIR port, using lower 8 bits of the IPv4 address, with the other 2 bits unused
48	IEEE MAC-48 address, which equals the concatenation of the first and last 24 bits of the lower 64 bits of the IPv6 address

The first two bits are used to guide the multi-version router on what functions are required of it. From the remaining 62 bits (58 bits actually used), all three of: the Ethernet MAC address, IPv4 address (if any), and IPv6 address (if any) can be looked up.

The Ethernet MAC address is available in the bottom 48 bits.

The IPv4 address is assumed to be a Class B address space with a fixed 16-bit prefix. The next 8 bits correspond to a network number within the organization and is found using a simple 4-input lookup table driven by the MIR port number where each port corresponds to a single network. The final 8 bits correspond to an interface number on the network and are available exactly from the 10-bit interface number field.

The IPv6 address has a fixed 48-bit prefix, corresponding to the allocated 001/TLA/NLA value. The next 16 bits correspond to a network number within the organization and is found using a simple 4-input lookup table driven by the MIR port number where each port corresponds to a single network. The final 64 bits correspond to an interface number on the network and are mandated to be in EUI-64 encoding defined as the two halves of the MAC-48 address separated by a fixed 16-bit pattern (`FFFEh`).

The 4-bit size of the port number field is large enough to accommodate most members of the Virtex-II Pro family, allowing up to 16 ports. The 10-bit interface number field accommodates the maximum number of Ethernet devices (1024).

System Functions

There are matching sets of required functions for IPv4 and IPv6 packet handling, respectively.

IPv4 Packet Handling

- IPv4 address extraction and matching
- IPv4 local address lookup
- IPv4 to IPv6 packet translation
- IPv4 over IPv6 tunnel lookup
- IPv4 in IPv6 packet encapsulation
- Fragmentation of over-size IPv4 packets
- IPv4 from IPv4 decapsulation
- IPv6 from IPv4 decapsulation

IPv6 Packet Handling

- IPv6 address extraction and matching
- IPv6 local address lookup
- IPv6 to IPv4 packet translation
- IPv6 over IPv4 tunnel lookup
- IPv6 in IPv4 packet encapsulation
- Fragmentation of over-size IPv6 packets
- IPv6 from IPv6 decapsulation
- IPv4 from IPv6 decapsulation

Additionally, the internal routing of IPv4 and IPv6 packets to the hub and ports within MIR for further processing or for transmission is an important function. The mechanisms for performing these actions are described in the next section.

Codesign and Trade-offs

A service-based system architecture model forms the basis for the study of codesign and trade-offs. In consideration of which components are used to carry out services for other components, there are four top-level MIR permutations.

- Coarse-grain services (on a per-packet basis)
  - Logic off-loads a packet to processor as server
  - Processor off-loads a packet to logic as server
- Fine-grain (in-packet functions)
  - Logic uses processor to perform function(s)
  - Processor uses logic to perform function(s)

In traditional systems, the occurrence is obvious. For example, the two coarse-grain permutations match packet reception (logic) and packet transmission (processor), while the second fine-grain permutation matches logic being used in an accelerator capacity. The first fine-grain permutation is perhaps the most interesting since it illuminates a non-processor-centric design space. With a logic-centric design, the MIR is a system driven entirely by input demands from the MIR environment.

For MIR, two function types have a fairly immediate assignment as a logic or processor implementation. These are classifying incoming packets as either IPv4 or IPv6 (or something else) and performing simple address matching for format and locality/non-locality. Unless a

completely coarse-grain processor implementation is intended, classification is done by logic circuitry.

The remaining system functions are classified into three categories:

1. Lookup-style functions - searching routing tables and tunneling tables
2. Routing-style functions - internal buffering and forwarding of packets
3. Massage-style functions - encapsulation and decapsulation, fragmentation, or translation.

These three orthogonal dimensions are offered to explore the codesign space for the MIR system. However, in practice, there are interactions between them when building an integrated system, for example, between combining routing and massage harmoniously.

A selective approach was used to define the MIR reference design instead of a systematic study of all of the possibilities for each of the three categories. The focus involved a few novel design options with a logic-centric system view.

For table lookups, either the logic or the processor can act as a fine-grain slave for the other. Other studies have looked at logic lookup options.

- Hardwired algorithms, with the lookup table encoded in the programmable logic
- Use of content-addressable memory
- Use of random-access memory (maybe with hashing prior to lookup)

Previous studies using the processor already show good ways of searching tables. Future investigation using the processor as a slave, in particular, examining the overhead involved in activating a “remote function call” from the logic may reveal other ways of searching.

System Architecture

Figure 1 is a block diagram of the MIR System.

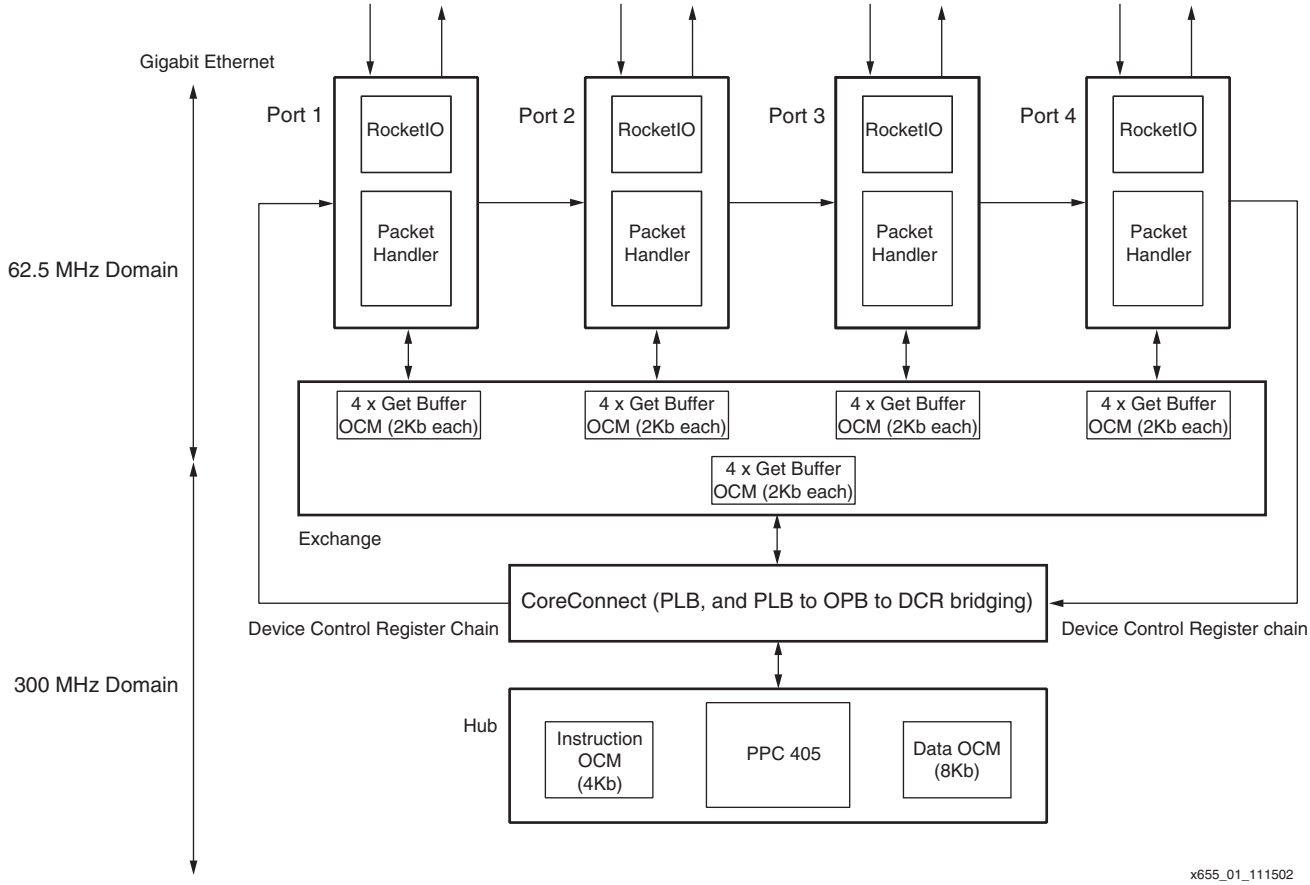

Figure 1: **MIR Block Diagram**

Reference Design Instantiation

The reference design illustrates an advanced design for the full capabilities of the Virtex-II Pro device. It includes a range of novel packet processing features. It is targeted at a particular point in a configured multi-version IP space:

- Pass-through IPv4 packets and IPv4 packets heading for IPv6 tunnels are handled entirely in logic
- Other IPv4 packets and all IPv6 packets are handled by the processor

This particular implementation trade-off though artificial is suitable in a largely IPv4-based organization operating in the context of a largely IPv6 Internet. In fact, it was chosen as a minimal setting allowing the demonstration of all the key system attributes, while simplifying the surrounding and a more routine detail of packet handling. Demonstrating the more exotic logic-related features for internal packet switching and packet analysis/modification is emphasized.

Module Details

Appendix A contains a list of all system modules. The main system modules are:

- The top module - the overall system
- The port module - external interfacing and packet handling in logic
- The exchange module - exchange and buffering of packets between ports and hub
- The coreconnect module - instantiates CoreConnect architecture
- The hub module - the PowerPC processor part of the system

Test Framework

The reference design has a simple functional simulation test bed, allowing observation and investigation of system behavior. The design also features optionally included diagnostic output lines suitable when testing the design on real hardware.

Clock Domains

Figure 1 shows the two clock domains. The 62.5 MHz domain applies to the logic circuitry for packet handling and is synchronized with the gigabit Ethernet interface. The 300 MHz domain applies to the PowerPC system design. The clocking arrangements use one Virtex-II Pro Digital Clock Module (DCM) for each clock domain.

RocketIO Transceivers

The RocketIO interface is configured for gigabit Ethernet operation, with a 32-bit wide interface between the serializer-deserializer and the rest of the system. Automatic generation and checking of Ethernet CRCs is enabled within the interface. To allow testing, it is also possible to configure the system with internal parallel or serial loopback within the transceiver as an option selectable by software executing in the PowerPC processor.

For demonstration purposes, the logic circuitry connected to the transceiver implements full gigabit Ethernet PHY framing and deframing, allowing packets with arbitrary byte lengths (i.e., not restricted to just multiples of four). This is used in back-to-back testing of MIR systems. However, the logic circuitry does not contain support for full gigabit Ethernet interaction with a standard network switch. Specifically, it does not deal with extra requirements beyond the basic packet framing and deframing, including initial auto-negotiation at startup and dealing with pauses in transmission for flow control.

There is discussion of the higher-level Ethernet MAC layer handling in **Packet Handling in Port Circuitry**.

CoreConnect Architecture

The standard IBM CoreConnect architecture (Reference 12) is used to interface the PowerPC processor to the logic circuitry. The bus structure is used to match the operational requirements of the PowerPC processor in this system to keep it from forming a bottleneck in parts of the system where processing is entirely in logic. The CoreConnect architecture serves two functions:

1. PowerPC processor access to packet buffers located within the exchange interconnection network
2. PowerPC processor access to control and status registers located within the ports

The backbone of the architecture, the processor local bus (PLB), allows 64-bit transfers. This is needed for the access to packet buffers. For access to control and status registers, there is a bridge from the PLB to the on-chip peripheral bus (OPB) with a bridge to the device control register (DCR) bus. Standard blocks for all three buses and the bridges between them are available for the Virtex-II Pro Platform FPGA.

The CoreConnect module of the system instantiates a PLB arbiter, PLB devices, an OPB arbiter, and OPB devices. It also connects a DCR chain to access various control and status registers. There are two PLB master devices, the CPU ICU and DCU ports. There is one slave device, the PLB-to-OPB bridge. There is one OPB master device, the PLB-to-OPB bridge and two slave devices, the OPB arbiter's control registers, and the OPB-to-DCR bridge. The DCR master is the OPB-to-DCR bridge (not the CPU). There are five DCR slaves: the PLB arbiter's control registers, the PLB-to-OPB bridge, the MIR chain of port control and status registers, a non-critical interrupt controller, and a critical interrupt controller. Some of these components are not actively used in the current system, acting as hooks for future extensions. Specifically, they are the OPB arbiter (only one master) and the two interrupt controllers.

The use of the CoreConnect architecture offers practicable solutions for the two interfacing points between the PowerPC processor and the logic. In the case of packet buffer access, an alternative would be to use direct data-side on-chip memory (DSOCM) access from the PowerPC processor. However, this is almost certain to lead to timing problems for the PowerPC clock domain due to a combination of routing delays and decoding logic delays at the buffers. In the case of control and status register access, an alternative is to use the *mfdcr* and *mtdcr* machine code instructions for direct access to DCR registers. However, indirection through the CoreConnect provides a mapped memory interface between the PowerPC processor and the DCR registers. This allows simplified and more flexible program access to the registers by direct memory access.

Main System Components and Functions

The following three sections describe the three main parts of the system:

1. The movement and buffering of packets within the exchange module including mechanisms to support the avoidance of copying packets between buffers at all stages of processing.
2. The handling of packets by logic circuitry within the port module includes the use of multithreading of control flows within the logic.
3. The operation of the software within the PowerPC processor, and the logic interface circuitry, implemented using the **CoreConnect Architecture**.

Exchange Network

The exchange network is concerned with the aspects classified earlier as "routing-style functions." In the MIR architecture, there are five computational loci: the four ports and the hub. The basic requirement for internal routing is the movement of packets (or, possibly, the movement of parts of packets) from one locus to another. The familiar problem of all-ways interconnection is frequently seen at various levels of granularity, for example, computer networking, parallel multiprocessor computers, and even on a single device. A large amount of earlier research is available as guidance.

One extreme point in terms of interconnection network is where a logical star topology is centered on the hub. A potential shortcoming of this approach is the serialization effect at the hub, potentially cancelling out any benefits gained from parallelism in handling the port. In general, there are various less extreme points to avoid complete centralization, for example, using a bus or ring, or more exotic alternatives like hypercubes and their derivatives. In the MIR reference design, an opposite extreme to the star is used, one where there is a complete interconnection scheme of dedicated links between all points. In the reference design, there is a total of 20 unidirectional links (five sources times four destinations). This n^2 resource scheme is not an issue with only four ports, but it will likely be too expensive for larger numbers of ports. The actual scheme is packaged as a black box in the system. This allows the substitution of alternative, less resource-intensive interconnection schemes.

The precise scheme implemented involves using a dual-port buffer, occupying one block RAM on each of the 20 internal links, one memory port used for reading, the other for writing. Thus, each communication port, and the hub, has four "get" buffers for reading, one each writable by the other four components. Since the full 36-bit word size available in each block RAM is used, the data paths for the interconnection network are also 36-bits wide.

When a write is attempted to a full buffer, the word to be written is discarded and an overflow flag is raised. A commitment mechanism for storing complete packets in buffers is included, mainly for other reasons described in the next section. However, this also ensures that no packet fragment is left behind if a buffer overflow occurs in the middle of a packet.

Each port, or the hub, selects packets for transmission or for further processing on a round-robin basis. The selection process is implemented in logic at the ports and by program in the hub.

In the hub, the interface to the contents of each relevant buffer (four for reading and four for writing) is random access, reflecting the particular nature of processors. Access to the 36-bit words is through the 64-bit PLB interface in the CoreConnect architecture. In addition to the direct access to the contents of each buffer, the processor also has access, through DCR registers, to the get and put pointers denoting the current positions of packet reading and writing from and to the buffers. There is read access to all pointers and write access only to those controlled by the hub.

In the ports, the interface is FIFO-like, reflecting the nature of logic circuitry. In essence, 36-bit words can be read from and written to buffers on a FIFO basis, with no explicit memory addressing by the port logic. As explained in the next section, the term "FIFO-like" is used because the writing mechanism actually implemented is subtle, although still serial and address-less in nature; the reading mechanism is entirely FIFO-like.

Packet Handling in Port Circuitry

Receiving Packets

Massage-Style Functions

When a packet is received, the bulk of the work requires "massage-style" functions. This group of functions is the main focus for investigation of trade-offs between implementation in logic and by program. Any final system would have a mix of implementation mechanism. Depending on the network environment and load mix, the more frequently required functions in logic, the less frequently required functions by program. The architecture of the MIR reference design has support for certain types of packets to be completely handled using functions only implemented in logic. Independent of how functions are implemented, one key underlying aim in MIR is to minimize, ideally to eliminate, the copying of packets between buffers, since this is well-known to be a significant cause of inefficiency in both routers and end systems.

Implementations of all required functions by program are included in the PowerPC software section of the reference design. Only a subset is implemented in logic to provide a sufficiently convincing demonstration. However, there is an emphasis on supplying a solid basic infrastructure for the integration of functions implemented in logic into the packet handling flow. This infrastructure involves three significant advanced features in the packet handling logic:

1. Internal Packet Broadcasting

The first component tackles the issue of avoiding the copying of packets between buffers. At first sight, at least one copy seems necessary because, after receiving a packet in an input buffer, it must be placed in the appropriate output buffer for onward transmission or handling by the processor independently of the necessity of massage. The solution adopted is initially to receive packets by placing their contents into all four of the possible output buffers, using a broadcasting style on the exchange network. As soon as a sufficient portion of the packet header is received and analyzed, the broadcasting is turned off and subsequent words of the packet are written only to the correct buffer.

2. Multithreading in Logic Circuitry

The second component tackles reducing latency by allowing packet massage in parallel with receipt of the packet. Some multithreading is introduced into the processing being carried out at an individual port. While having different logic blocks operating in parallel is simple, arbitration of access to the packet is needed. The solution adopted is to operate the threads on differing clock phases, knowing the buffer memory access time is much shorter than the processing clock cycle. As an example, a thread receiving a packet can be clocked in step with the RocketIO receiver, while a thread massaging a packet can be clocked in between each receiving step. The write interface to buffers is equipped to allow this dual access (to one port of a dual-ported block RAM) at alternating memory clock cycles and also to allow the massage thread to circumvent a normally strict FIFO discipline.

In **MIR Operation Examples**, there are two examples of the use of multiple threads including two cases involving encapsulation of packets for onward transmission.

Some general principles of multithreading in logic circuitry have been distilled from the MIR system design, and reported in detail in a research paper[2]. Each thread corresponds to a finite state machine (FSM) within the logic. Individual threads are activated and deactivated by other threads by means of one-bit activation flags that enable and disable transitions out of the initial state of the FSM. Communication between threads is via shared registers in the logic, with careful synchronization to prevent conflicts.

3. Buffer Handling Details

Detail buffer issues beyond the emulated multi-port access just described are in this component. Since the initial broadcast of packets to all feasible buffers would leave abandoned fragments after the true buffer had been established, it was necessary to include a mechanism for committing packets to buffers once they were fully received, massaged, and destined. At this point, immediate onward transmission is possible. The four extra bits available in the 36-bit memory word were used in the first and last (32-bit data) words of packets. In the first word, they indicate an offset (0 to 15 words) before the packet data actually starts to allow for presence or absence of an encapsulation header. In the last word, they form a 4-bit mask denoting the position of the final byte of the packet in the 32-bit word.

Lookup-style Functions

In addition to massage-style functions, address lookup is required for incoming packets. Additional dual-ported memory is included in the exchange module to support dual access to lookup tables for the logic to do lookups in each port and for the PowerPC processor to initialize and update tables. A similar mechanism to the shared packet buffers is used including 64-bit access through the PLB. The format used for address tables is described in **Derived Address Table Format**. Each table entry occupies 64 bits, thus matching the memory access mechanism. Two block RAMs are used in the reference design allowing up to 512 table entries.

For demonstration purposes, the lookup logic added to each port just uses a simple hash on an IPv4 address to index into the table, giving a lookup within a single clock cycle. The use of more subtle searching algorithms is not in development. One further option for exploration is the use of the PowerPC processor as a slave to perform searching. Critical and non-critical interrupt controllers had been incorporated as optionally configurable features in the reference design, disabled by default.

Transmitting Packets

Transmission of packets at a port is considerably simpler than receiving, since no analysis and massage is required. A simple finite state machine is used to output successive words of Ethernet PHY framing and packet contents to the RocketIO interface. The packet contents are retrieved in sequence from the appropriate output buffer at the port. The state machine takes account of the packet length in bytes, generating correct PHY framing with respect to 32-bit word boundaries at the end of transmission.

PowerPC Software

Integrated Test and Service Software

The software can be configured to operate in either service mode or test mode, using a single parameter by the mechanism described in **Appendix A**. Service mode is the normal mode of operation, where the PowerPC processor is a slave for handling incoming IP packets appropriately. In test mode, the PowerPC processor has a more active role. It initially generates a set of packets for transmission by the ports with the port inputs and outputs connected into a ring. All internal buffers and routes within the router are exercised and tested.

The program implementation of the required router functions is almost complete, apart from an assumption that no fragmentation of IP packets will be necessary. Those functions that are implemented by logic circuitry in the ports in the MIR reference design can be commented out, since no PowerPC processor assistance is required.

For test mode, one of three different sets of test packets is initially generated, depending on whether the system is configured in 2-, 3- or 4-port mode. The test packets carry IPv4

destination addresses that are tuned to the address lookup mechanism included in incoming packet handling. The choice of addresses ensures that all 20 paths through the exchange network are exercised by incoming packets (incoming from looped-back connections from output ports). In some cases, packets arrive back at the PowerPC hub, in which case they travel no further. In other cases, packets loop forever through a sequence of two or more ports. Minimum-size Ethernet packets are used, to supply the most stringent timing conditions.

Instruction and Data Memory

All instructions and data for the processor are held in on-chip memory in block RAM, consistently with the single-chip system design approach. The program code can be just accommodated within 4K of memory, that is in two block RAMs. However, to allow for any expansion, the system can be configured to have 8K of memory. In principle, the program code could be located entirely within the (16K) instruction cache, avoiding the need for the separate memory during execution. Currently, instruction caching is disabled. The program data, stack, etc. fits comfortably in the 8K of memory allocated, although there is a configuration option to double this. Data caching for this program data is disabled.

PLB Accessing and Data Caching

For 36-bit buffer accesses and 64-bit lookup table accesses, 64-bit operations over the PLB are used. To enable this, given that the processor has a basic 32-bit data path, data caching, which is disabled for the program data memory, is enabled for the packet buffer area and the lookup table area. The 64-bit operations then result from cache line fetches and flushes. Note that the PLB is also used as the conduit to the DCR bus, used to read/write packet buffer pointers in the exchange network, and to read/write control and status registers in the ports.

MIR Operation Examples

To illustrate the operation of the reference design MIR system when a packet is received, two contrasting examples are described here. Both involve the encapsulation of an incoming packet within a new packet for onward transmission. However, one is handled entirely by logic circuitry, whereas the other relies on assistance from the PowerPC processor.

In the packet handling logic, there are seven threads that are activated as required. One is a main thread concerned with receiving the raw 32-bit words of the packet. It is clocked on the leading edge of a 31.25 MHz clock, since received words are available on the trailing edge of the clock. The other six threads are concerned with analysis and/or massage, and are all clocked on the trailing edge of the clock, in order to synchronize with the main receive thread.

1. MAC/SNAP header recognition thread
2. IPv4 header recognition thread
3. IPv6 header recognition thread (vacuous in reference design)
4. IPv4 address lookup thread
5. IPv6 encapsulation thread
6. MAC header construction thread

In all cases, as each packet is received, as soon the PHY framing words have been received, a MAC/SNAP header recognition thread is started. In addition, a speculative (at this stage) IPv6 encapsulation thread is started. The incoming packet is always stored at an offset of 10 words within its buffer, which leaves room for a speculative header to be built on front without contention. The early start ensures prompt completion before the time required for a minimum-length Ethernet packet to be received. The choice of threads activated after these ones depends on what is found in the packet, as the two examples below illustrate.

Example 1: Encapsulation of IPv4 Packet Within an IPv6 Packet

The following is a sequence of thread activities, after the above standard basics:

1. When packet is identified as being IPv4 by the MAC/SNAP header analysis, the IPv4 header recognition thread starts.
2. When IPv4 destination address is captured, the IPv4 address lookup thread starts.

In this case, all threads run to normal completion. The massage effect is that a new six-word MAC/SNAP header followed by a new ten-word IPv6 header has been constructed, immediately before the start of the original IPv4 header. Note that the original MAC/SNAP header is over-written. All massage is complete by the time that the packet is fully received, and thus it is ready for immediate outgoing transmission via the appropriate port.

Example 2: Encapsulation of IPv6 Packet Within an IPv4 Packet

The following is the single thread activity, after the above standard basics.

1. When packet identified as being IPv6 by the MAC/SNAP header analysis, IPv6 header recognition threads starts (although this does not do anything). The speculative IPv6 encapsulation thread continues to completion redundantly, but harmlessly. When the packet is fully received, it is available to the PowerPC software in the hub to parse, encapsulate in an IPv4 packet and then transmit onwards via the appropriate output port.

Reference Design

The reference design is available in Verilog for download from the Xilinx ftp site at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp655.zip

Functional Simulation

The reference design is fully verified by functional simulation using ModelSim SE 5.5e. The simulation involved the PowerPC processor running in test mode, with the RocketIO interfaces connected into a ring.

Synthesis: Real Resource Requirements and Timing

As stated earlier, the reference design is targeted at the XC2VP7 member of the Virtex-II Pro family. For testing purposes on a real XC2VP7, a minimal two-port version has been used, both to minimize the design flow cycle time and to minimize the use of on-chip resource. This version uses around 70% of the available slices on the XC2VP7 part and presents no place-and-route difficulties. For testing, the processor clock domain has been operated at half the target speed (150 MHz, rather than 300 MHz). The design satisfies this relaxed clock constraint, but would not satisfy anything much higher without some attempts at tuning of the placement. However, this timing shortfall is not a major concern, assuming the processor plays a part-time role in the system. The logic clock domain has been operated at the target speed (62.5 MHz), and the design easily satisfies this. A 3-port version also fits on the XC2VP7 with no place-and-route problems, albeit using almost all of the slices. No detailed timing experiments have been done with this version. A full 4-port version, after synthesis, is measured at just over 100% of slices. A slightly trimmed version has been placed and routed for the XC2VP7, but only when timing constraints are absent.

An approximate apportioning of the logic resource required for the 4-port version is:

- Packet handling 10% per port (40% total)
- Exchange 35%
- CoreConnect 20%
- Other logic 5%

Status

The reference design is provided primarily as an advanced example of design for the Virtex-II Pro family. The internal architecture, including the PowerPC software, is complete and suitable for adoption and use. However, to become a practical router, some further detail work is required. First, some missing aspects of gigabit Ethernet protocol handling are required. Second, and given the first, testing on a real Virtex-II Pro device connected to a real Ethernet switch is needed.

References

1. Brebner G, "Single-chip Gigabit Mixed-version IP Router on a Virtex-II Pro device", Proc. 10th IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, 22-24 April 2002, IEEE Computer Society Press 2002, pp. 35-44.
2. Brebner G, "Multithreading for logic-centric systems", Proc. 12th International Conference on Field-Programmable Logic and Applications, Montepellier, France, 2-4 September 2002, Springer LNCS 2438, pp.5-14.
3. Internet Requests for Comments (RFC) Accessible at **http://www.rfc-editor.org/rfc.html**
   a. RFC 2893 "Transition Mechanisms for IPv6 Hosts and Routers"
   b. RFC 2473 "Generic Packet Tunneling in IPv6 Specification"
   c. RFC 2765 "Stateless IP/ICMP Translation Algorithm (SIIT)"
   d. RFC 2766 "Network Address Translation - Protocol Translation (NAT-PT)"
   e. RFC 2374 "An Aggregatable Global Unicast Address Format"
4. Fallside H and M Smith, "Internet Connected FPL", Proc. 10th International Conference on Field-Programmable Logic and Applications, Springer LNCS 1896, 2000, pp.48-57.
5. Brebner G, "Highly reconfigurable communication protocol multiplexing element for SCOPH", Reconfigurable Technology, Proceedings of SPIE, 4525, 2001, pp.99-106.
6. Braun F, J Lockwood and M Waldvogel, "Reconfigurable router modules using network protocol wrappers", Proc. 11th International Conference on Field-Programmable Logic and Applications, Springer LNCS 2147, 2001, pp.254-263.
7. Lockwood J, N Naufel, J Turner and D Taylor, "Reprogrammable network packet processing on the Field Programmable Extender (FPX)", Proc. 9th International Symposium on Field Programmable Gate Arrays, ACM Press 2001, pp.87-93.
8. Ditmar J, K Torkelsson and A Jantsch, "A dynamically reconfigurable FPGA-based content addressable memory for Internet protocol characterization", Proc. 10th International Conference on Field-Programmable Logic and Applications, Springer LNCS 1896, 2000, pp.19-28.
9. Guccione S, D Levi and D Downs, "A reconfigurable content addressable memory", Proc. 7th Reconfigurable Architecture Workshop, Springer LNCS 1800, 2000, pp.882-889.
10. **http://www.npforum.org**
11. **http://www.whnet.com/giga.html**
12. **http://www.chips.ibm.com/products/coreconnect/**

Appendix A

MIR System Parameters

This reference design features easy parameterization of various system features relating to optional architectural components, internal bus addressing, and external network addressing. The configuration of parameters and optional features is controlled through two global parameter include files, one for the hardware (*global_params.v*) and one for the software (*global_params.h*), both shown in the **Appendix C** file listing. The parameters featured in both files *have to be kept consistent manually*, as indicated by comments within the files themselves. In some case, there are also consistency issues with other files, and again this is signalled by comments.

A summary of the system parameters is listed:

- Architectural parameters:
  - Number of gigabit input/output ports (two, three or four)
  - Selection of service mode or test mode operation
  - Optional doubling of instruction memory size for PowerPC processor (4 Kb to 8 Kb)
  - Optional doubling of data memory size for PowerPC processor (8 Kb to 16 Kb)
  - Optional inclusion of interrupt controller logic
  - Optional connection of input/output ports into a ring for simulation testing
  - Optional inclusion of diagnostic output signals from modules
  - Optional inclusion of AFX board diagnostic outputs for real hardware testing
- System address space parameters: PLB, OPB, and DCR addressing
- Network addressing parameters:
  - Own MAC, IPv4, and IPv6 addresses
  - Relevant router and tunnel addresses within local network

The full details of all these parameters can be studied within the global parameter *include* files.

Appendix B MIR Circuitry Modules

Table 1: **Module Summary (for n-port configuration, n = 2, 3, or 4)**

File	Module(s)	Instantiated By
top.v	top	– (if real); testbench (if simulated)
exchange.v	exchange_module	top
	port_select_module	exchange_module (n times)
	hub_port_buffer_module	exchange_module (n times)
	port_hub_buffer_module	exchange_module (n times)
	port_port_buffer_module	exchange_module (n(n-1) times)
hub.v	hub_module	top
port.v	port_module	top (n times)
packet.v	packet_module	port_module
	DCR_module	packet_module
	transmitter_module	packet_module
	receiver_module	packet_module
ocm.v	isocm_module	hub_module (one or two times)
	dsocm_single_module	hub_module (one or two times)
	dsocm_dual_module	hub_port_buffer_module, port_hub_buffer_module
	gsocm_dual_module	port_port_buffer_module
	lutocm_dual_module	exchange_module (n times)
coreconnect.v	coreconnect_module	top
	plb_bus_logic	coreconnect_module
	opb_bus_logic	coreconnect_module
clocks.v	basic_clocks_module	top
	GIO_clocks_module	basic_clocks_module
Files from PDK	PPC405	hub_module
	GT_ethernet_4	port_module
	GT	GT_ethernet_4
	RAM16_S36	isocm_module (two times)
	RAM16_S9	dsocm_single_module (four times)
	RAM16_S36_S36	{ds,gs,lut}ocm_dual_module (x 1,1,2)
	IBUFG, DCM, BUFG	basic_clocks_module, GIO_clocks_module
braminit.v	– (BRAM initialization)	(included by) top (func.sim. only)

Notes:

1. The coreconnect_module is strongly derived from the Embedded PPC405 Reference System. It uses other modules including arbiter, plb2opb_bgo, plb_bram_cntlr, opb_arbiter, opb_dcr_brg_top, and INTC (and their sub-modules); also, plb_bus_logic and opb_bus_logic have some (simple) internal sub-modules not listed here. Inclusion of the two INTC modules is a configurable option.

Appendix C

MIR Source Files

The MIR reference design folder uses all of the "make" facilities provided by the PDK. The folder contains a *flow.cfg* file, together with subfolders: *par, sim, sw, syn, and sys*. The source files specific to the reference design are listed below. The PDK *make* processes generate many other standard-named derived files.

Circuitry Source Files (Corresponding to Module Summaries in Appendix B)

In subfolder "sys/verilog":

- clocks.v
- coreconnect.v
- exchange.v
- global_params.v (see **Appendix A**)
- hub.v
- ocm.v
- packet.v
- port.v
- src.lst (used by PDK make processes)
- top.v

Program Source Files, and Related Files

In subfolder "sw":

- global_params.h (see **Appendix A**)
- ppc.h
- kernel.S
- main.c
- mapfile
- code.bmm
- Makefile
- (plus generated files: kernel.o, main.o, code.elf, code.lst, code.map, bram_init.ucf, bram_init.v)

Simulation Files

In subfolder "sim/testbench/verilog":

- testbench.v

In subfolder "sim/func_sim":

- wave.do

Place-and-Route Constraint File

In subfolder "par":

- top.ucf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/19/02	1.0	Initial Xilinx release.
01/17/03	1.1	Added link to **Reference Design**.