Communicating Process Architectures 2003
Jan F. Broenink and Gerald H. Hilderink (Eds.)
IOS Press, 2003

A Single Chip Solution for Distributed Processing Systems

B.C. O'NEILL, P.W. MOORE and S. CLARK

School of Engineering, The Nottingham Trent University,
Burton Street, Nottingham, NG1 4BU,UK
Email : Brian.Oneill@ntu.ac.uk

Abstract. This paper describes a processor and an inter-processor communications interface integrated on a single chip for use in a distributed processing system. The system is based on work of the electronic systems design and parallel processing group at the Nottingham Trent University**.** The four main elements of the chip design are processor, memory, communication interface and packet routing switch all integrated unto one chip. This design is achieved by the use of the ALTERA ARM based Excalibur system on a programmable chip (SOPC) containing an embedded processor and programmable logic. The paper describes the communication features and implementation carried out by the research group to achieve this single chip processor.

1 Introduction

The first major processor that was designed, primarily for a multi-processing system, was the transputer. From the hardware perspective one major advantage was that systems were relatively easy to build. The inter-connection between processors was via a pair of serial links and this interface was an integral part of the processor chip. The design was very successful and in the initial stages it was used in a wide range of applications from large number crunching tasks to embedded systems. However, with time other competing processors diminished the market for the transputer and by the time the second generation of transputer was introduced its market share was not sufficient to be economically viable. There are many viewpoints on reasons for the transputer demise. One reason is the very high cost of processor development, which in the author's opinion is the most significant. Continuous development can only be sustained from the sale of processors for single use applications. The multi-processor market is not sufficient to recover the cost of development. As the technology of fabrication progresses towards higher densities, the initial cost of development are increasing and the unit costs of production are reducing. The consequence of this trend is that there are less processor types under development today than any period in the previous twenty years.

Currently many multi-processing systems are built using standard PCs linked by standard network cards. There are many exceptions [1,2,3,4] to this but the difficulty for designers of multi-processor systems is the high unit cost and lifetime of systems. This paper describes the design and construction of a multi-processor system using 'System On a Programmable Chip' (SOPC) technology which in part overcomes some of the cost difficulties of developing such systems. The SOPC has an embedded processor and programmable logic, which is used in this project to add specialist communication

functions. The user of an SOPC device benefits from the efficiency of ASIC design for core functions, in this case the processor, timers and others, and the flexibility of using PLD technology to provide specialist functions. The same SOPC device can be used for a wide range of applications thus obtaining the economy of scale, which any one of these alone cannot provide. To offer the same functionality (processor and specialist logic) on a PLD device would require the use of a very large and costly device and also access to processor IP.

This paper will outline the design of a distributed parallel processor network, which has at its core a single chip processor node. It will focus on the inter-processor communication features designed by the electronic systems design and parallel processing group at the Nottingham Trent University.

2 Background

The research work started by concentrating on the inefficiency of the serial communication network which has point to point terminations. Figure 1 shows a typical network under consideration, which is configured, from a single host processor node. The processor nodes can have a minimum set of common features that are relatively easy to design and implement. Any additional design work need only focus on the I/O interface requirements for a particular specialist application. In the 17 node network of figure 1 any message which is sent across the network will have to pass into and out of several processor node.

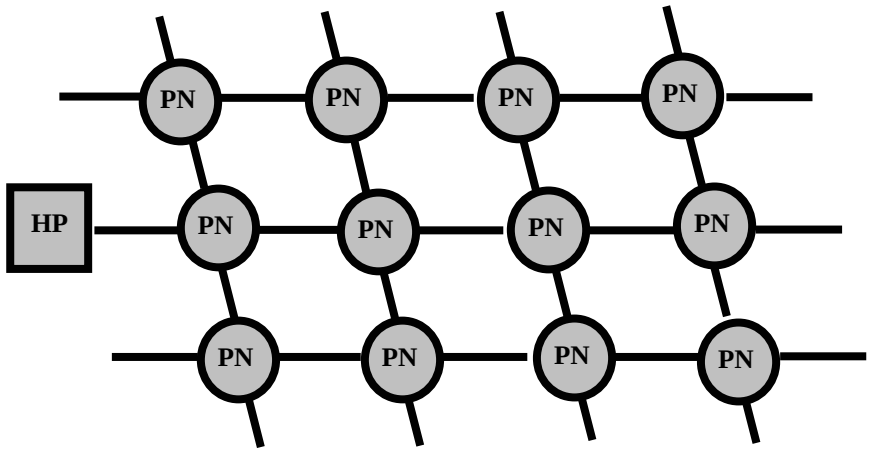

Figure 1- A parallel system with point to point communications.

To address the communication bottleneck the research group developed a packet routing switch. Figure 2 shows a 16 channel packet routing switch (ICR C416) which was designed to interface to the first generation of INMOS transputers [5]. The use of this device, in some real-time control systems, has shown the efficiency of routing as a solution to medium scale, low-cost, inter-processor communications.

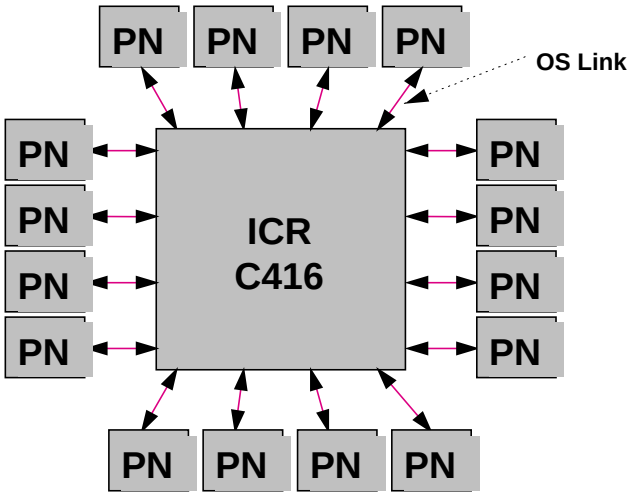

Figure 2- A parallel network with an ICR C416 packet routing switch.

Over a period of several years the group has developed components for this network to build a complete distributed parallel processing systems. The components contain many modular functional block designs captured in VHDL and implemented on FPGA devices. This has allowed for design upgrade and reuse over a period of time and the porting of design to the newer devices. The main features of the current network will now be described. The router network specification is similar to the ESA SpaceWire standard [6].

3 Outline of the communication network

Routers provide direct communication, among processors, similar in function to that of a telephone exchange, supporting the simultaneous transfer of messages from any input to any output of the router. Each packet/message has header bytes, which are used to configure the path of the packet through the network. Control at the core of the device prevents one message from blocking the path of another. If two or more packets request the same output then all except the first are queued in a FIFO. All communications take place over a two wire serial link. The block diagram of figure 3 shows the main functions of the routing device.

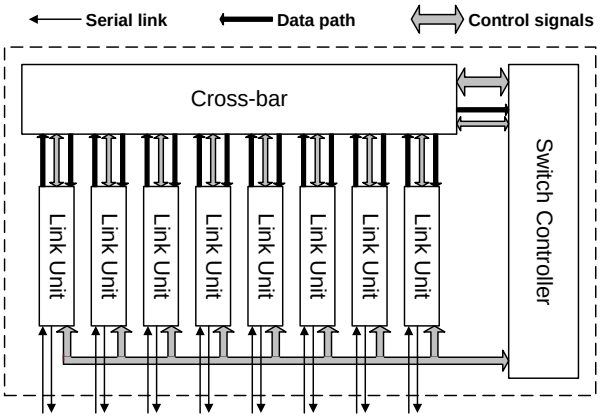

Figure 3 : High level block diagram of the basic router

For large systems routers can be cascaded. A system of up to 14 processors can be linked by two 8 link routers and there is no limit to the size of system that can be built. The system scales without loss of bandwidth for moderate size networks (i.e. networks where

the average message passes through 4 or 5 routers). For large networks a strategy of store and forward of packets achieves better performance than the wormhole routing method of this packet switch [7,8]. Figure 4 shows an example of a 12 processor network linked by two 8 link routers and each processor node (PN) can automatically configure the routing switches to transmit a data packet to any other node. Two of the eight links are used to interconnect the router to improve bandwidth. The host processor node (HP) uses the same link connections as all other connections. In most typical applications this would be a standard PC containing boot programmes for all other processing nodes and would handle the external I/O of the network.

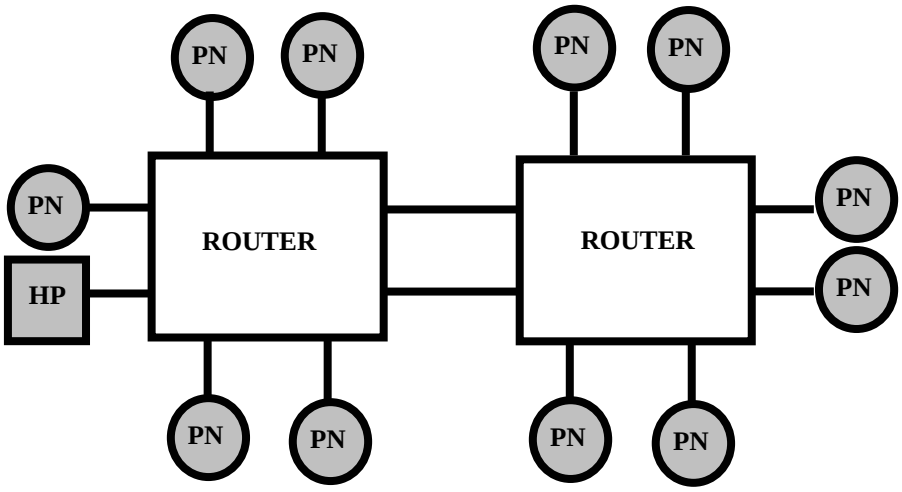

Figure 4: A network configuration of 12 distributed processors

The time taken to connect an input channel to an output channel is approximately 1.5μs. The receiver input link uses an oversampling technique for data recovery of the signal [9]. This has the advantage that a single FPGA device can support many communication links as no phase lock loop circuitry is required for clock recovery. This link is referred to as an OS link and requires only two electrical signals (Tx & Rx) compared to the four required for the DS method of SpaceWire.

3.1. Message types and addressing modes

The router supports both uni and multi cast messages. Uni-cast messages have a single source and destination and multi-cast may start from one of several sources but is transferred to two or more destinations.

Each message contains one or more header bytes of routing information at the start of the serial byte stream. The switch supports two header addressing modes. When physical mode is used the header value directly relates to the hardware value of the output port and this header byte is stripped from the message as the message is transmitted from the device. Logical addressing requires prior configuration of the routing device so that any valid logic value can connect a message to a particular output port. The header value may or may not be stripped at the output. The advantage of logical addressing is to allow several adjacent routers to be configured as one virtual router.

3.2. Packet Protocol/Tokens

The low-level link protocol defines a full-duplex, two-wire system (Tx and Rx), that utilises eleven-bit tokens to transmit information on a serial bit stream. Tokens have eight data bits, a start bit, a stop bit and a control bit, as shown in Figure 5. The start (always

logic high) and stop bit (always logic low) are used for framing, the type bit is used for differentiating between control and data tokens (logic low and high respectively). The stop bit can be extended by any amount when there is no further information to transmit.

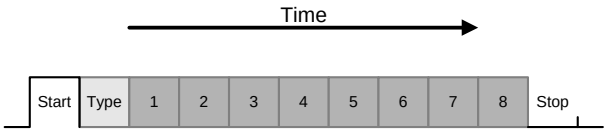

Figure5: Physical link token format

Using this format for the token it is possible to define 256 separate control tokens. However, at present only six tokens are defined. They include an initialisation token, two flow control tokens, and three message terminator tokens. The initialisation token is used to establish a validated connection between the link ends. As the system uses a two-wire asynchronous link, a flow control technique must be employed to prevent buffer overflow that utilises the same resources, but is transparent to the data connection. The protocol uses two flow control tokens to implement a stop/go (X-ON/X-OFF) technique compared to the credit based method of SpaceWire. Thus, if the buffer of a receiving device is approaching maximum, a 'stop' token can be sent on the return line to inhibit further transmission. Once enough space in the buffer has been cleared a 'go' token is sent to re-enable the transmission. The stop/go compared to credit based technique was found by simulation to be more efficient for many bi-directional flow conditions [10]. The flow control tokens are also used in the fault detection mechanism, which will be described later. Finally, the three message terminator tokens are defined for used in the network layer of the protocol, that is, 'end of packet' and 'end of message' and 'bad end of packet' (BEOP). The BEOP token is used in the recovery mechanism of the network after an error has been detected.

3.3. Cascading of router

When routers are connected together it is possible to use several link connections between two routers and group these connections together too logically appear as one link. This technique is referred to as 'Group Adaptive Routing'. Control functions on the router will automatically allocate a message to any free link of the group. This effectively increases the bandwidth of the connection between two routers at the expense of using additional link connections.

3.4. Configuration Port

The routing device contains an internal configuration port that can be accessed from any input port of the device. On power up the default configuration allows physical mode addressing and therefore its is possible to remotely configure the device from any node in the network.

3.5. Fault Detection and response

The router provides a level of fault tolerance for events that were thought to be the predominant causes of failure; namely, disconnection, host failure, device failure, and FIFO overflow. Detection of faults is based on knowing the status of the link at all times. By monitoring activity at the receiver, any operation that did not conform to a basic set of rules could identify faults easily. The two main states are identified as the basis of the detection mechanism namely, 'asleep' and 'awake'. The link enters the 'asleep' state on reset or on the occurrence of a fault. The link remains in the 'asleep' state until the successful completion of an initialisation sequence when the 'awake' state is entered.

A failure is detected in one of four ways. The first method is a synchronisation error. As previously described, the start and stop bits frame each token. If the sampling circuits detect an incorrect frame, a synchronisation error is flagged. The second method of failure detection is a 'receiver buffer overflow' flag. Although the flow control system should prevent such a situation occurring, a buffer overflow signals a breakdown of the flow control mechanism, which indicates a corrupt interconnection. The third relates to the frequency of link activity, where activity refers to the movement of any tokens over the link. Each side of the link monitors the link activity, and if no activity occurs within a predefined time period, the connection is said to be lost and an error is flagged. If there is no data to transfer across the link then idle tokens are inserted to maintain the link active flag. The final detection method triggers if the link receives an initialisation token while in the 'awake' state. As this token is not expected in normal operation, it indicates a failure in the transmission line at the other transceiver.

On detection of a fault a sequence of events occurs on the faulty link. The link is first flagged as unavailable and further input data is ignored. Then no further data is transmitted out and the remainder of the packet is spilt. On the receiver side of the link a BEOP token is added to the receiver FIFO. Finally a reset token is transmitted to signal failure to the other end of the link pair. If the fault is transient the link will recover from this state and resume transmission of further messages. The main use of this fault recovery procedure is to prevent messages sent to non-operating links from blocking the rest of the network.

3.6. Interface to processing node

To build a communication network, in addition to routers, processor interface components are required. Previously, an interface to the PCI bus was developed to connect to a host PC and provided the base for the specification of an interface to the AMBA bus to connect to the ALTERA ARM Excalibur [11] processor. A block diagram of the PCI interface is shown in figure 6.

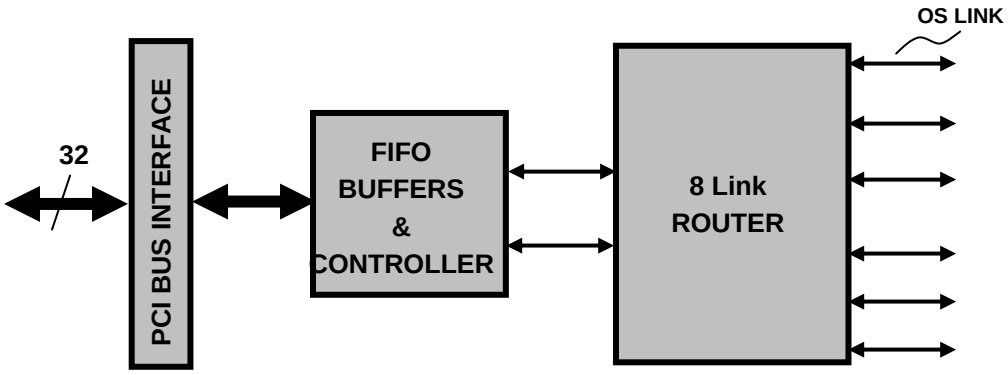

Figure 6: Block diagram of PCI bus interface to router.

The PCI interface supports two link connections between the FIFO buffers and the router. Each link has a separate transmit and receive channel and has a 32bit wide by 64 deep FIFO. Messages are transferred by DMA in data bursts of up to 64 double words, until the message is completed. The rate of transfer is approximately 70Mbytes/sec across the PCI bus interface. The overhead on the processor is very low and, because of the high rate of transfer, bus utilisation is very low. The hardware is optimised for the efficiency of data transfer and the latency of transfer was not part of the design specification.

The data is then transferred to a byte stream FIFO, byte token information and control tokens are added, before the packet is transferred to transmitter module. There is a similar

structure for the received data. Currently the baud rate of the serial data link is 22Mbit/s and in addition some successful work has been carried out at link speeds of 44Mbit/s at cable lengths of 100 m [9].

The two link interfaces from the host processor are connected internally to an eight link router which is integrated in the same FPGA device.

At the block diagram level of figure 6 the AMBA bus interface is similar to the PCI interface but at a lower level the AMBA bus contains four separate 32 bit interfaces for data transfer. Nevertheless the conversion of the PCI interface to the AMBA interface of the ALTERA ARM Excalibur was relatively straightforward. Many of the modules were reused with little modification. The number of PLD logic elements (LE) of the Excalibur EPXA1 was half that used for the PCI interface device and therefore contains only one link connection between the FIFO buffers and a five link router to provide four external link connections. Full measurements of the data transfer rate across the AMBA bus have yet to be carried out. It is anticipated that a rate of transfer, similar to that obtained with the PCI bus, will be measured as the same 33 MHz clock frequency and FIFO size is used.

To transmit a packet of data it is necessary to write to three registers of the interface module. First, a 32 bit header containing routing and packet ID information second, a 32 bit start address of the data packet and finally, a 20 bit length value measured in bytes. The start of each packet must be double word aligned but the length can be of any value within the 20 bit field. The transfer is then carried out automatically under DMA by the PLD module allowing the CPU to proceed with other tasks. On completing, an interrupt flag is asserted to flag the end of the transfer. A similar structure is used for the receiver. Any packet entering the receiver is stalled pending writing to the two receive registers (address & length). Once a packet has been received and the interrupt asserted the process must be repeated for any further next packets.

The design of the communication interface is optimised to have a minimum overhead on the processor node to setup and receive messages and also to produce a low overhead on the processor's data bus. The design provides hardware support for different messages types (i.e. virtual channels) in contrast to the single message type of the transputer. However the addition of the FIFO buffers needed to obtain the high bust rates of 70Mbytes/s has added to the latency of transfer.

4 Implementation

The Excalibur EPXA1 device is used as a stand alone processor with the communication interface implemented in the programmable logic elements of the device to provide a single chip solution for the processing node. These elements are configured from a small non volatile memory device (ECP2). Programmes can be booted from the link on power up. Addition SDRAM can be added via a standard connection. The Excalibur EPXA1 without router function requires 933 LEs or 22% of total. With a five link router 2838 LEs are required which is 68% of total.

To obtain access to a host PC a PCI interface is required and this is implemented on an ALTERA APEX 20k 200 device [11]. The two link interface without the router requires 2765 LEs or 33% of total. With an eight link router 5722 LEs are required which is 68% of the total LEs.

5 Conclusions

The network provides a scalable solution for inter-processor communication in a distributed multi-processor network. The benefits of SOPC technology allow the integration of embedded and custom functions in a single chip processing node similar to that of the transputer. The level of integration achieved in the Excalibur SOPC device and development tools have led to cost effective solution for this project.

The porting and reuse of VHDL design modules has been demonstrated to effect.

6 Acknowledgement

The authors would like to thank ALTERA Corporation and IC-Routing LTD for their financial support for this project.

Reference:

[1] N.J. Boden, et al.; "Myrinet – A Gigabit-per-Second Local Area Network"; IEEE Computer, vol. 15, No. 1; 1995; pp. 29-36.

[2] Analog Devices, URL: http://www.analog.com/products/descriptions/2154_0.html, summary: The ADSP-21061 is a member of the powerful SHARC ® family of DSP processors.

[3] TEXAS Instruments, URL: http://dspvillage.ti.com/docs/dspproducthome.jhtml, DSP Platforms both fixed- or floating- point platforms to support many design needs.

[4] CRAY RESEARCH INC.; "The Cray T3D System Architecture Overview"; Cray Research Inc.; Technical document HR-04033; 1993.

[5] J W Ellis, B C O'Neill & S Clark, 'A Router Design for T800 Compatible Transputer Arrays' - Transputer Applications, ISSN 0969-9341, 1993, Vol. 1(2) pp12-18.

[6] European Space Agency, ECSS-Space Engineering SpaceWire – Links, Nodes, Routers and Networks (ECSS-E-50-12) URL: http://www.estec.esa.nl/tech/spacewire.

[7] A.M.JONES, N.J.DAVIES, M.A.FIRTH, C.J.WRIGHT; "PACT The Network Designer"s Handbook"; IOS Press, Ohmsha, Concurrent systems engineering series, Vol. 51; ISSN 1383-7575; 1997; pp 17-18

[8] L.M. Li, P.K. McKinley; "A survey of wormhole routing techniques in direct networks"; IEEE Computer, vol. 26, no. 2; 1993; pp. 62-76

[9] B C O'Neill, S Clark and K L Wong, 'Serial Communication Circuit with Optimized Skew Characteristics', IEEE Comms Letters, IEEE computer Society, Vol.5, No.6, pp 260-262, 2001.

[10] R Hotchkiss, PhD thesis, Nottingham Trent University, Nov 2000.

[11] ALTERA CORPORATION, http//www.altera.com.