Implementation of 4x4 Reconfigurable Crossbar Switch for Network Processor

Prashant K. Wanjari
Student, Mtech
YCCE, Nagpur

Prof. Anagha Choudhari
Lecturer,
YCCE, Nagpur

Indrayani Patle
Student, Mtech
SRKNEC, Nagpur

ABSTRACT

This paper presents the proposal and implementation of a 4x4 reconfigurable crossbar switch (RCS) architecture for network processors. Its main purpose is to increase the performance, and flexibility for environments with multiprocessors and computer clusters. In this paper we show the result of simple 4x4 crossbar switch .The results include VHDL simulation of RCS on Modelsim software and the use of it in a broadcast function implementation, found in message passing support middleware. This reconfigurable crossbar Switch used in Network Processor to connect the various circuits which are used to perform the various task.

Keywords

Network Processor, FPGA, and Model sim

1. INTRODUCTION

At the end of nineties, network equipments normally used general-purpose processors. However, the need of quality-of-service and the high speed of data transmission demanded a rapid evolution of network equipments. Thus, the Network Processor (NP) was Created to increase the data transmission speed & also used to perform the various operations. Network processors are used in place of some GPPs (General-Purpose Processors) and ASICs (Application Specific Integrated Circuits) in network equipments, targeting two important issues: flexibility and performance. As these features are essential to process the packets, a network processor is the best choice to get them. The main motivation is the necessity of increasing two features cited before. With the use of a reconfigurable crossbar switch in a network processor it could be achieved. Thus, using network processor with a reconfigurable crossbar switch as interconnection structures, it is possible to increase the throughput and reduce the latency in communications with shared memory and message transference.

Therefore, the main objective of this paper is to present the RCS-2, a reconfigurable crossbar switch architecture used to connect different inputs and outputs in interconnection and communication networks, The reconfigurable crossbar switch was described in VHDL (VHSIC Hardware Description Language) and we want to implemented it on FPGA (Field

Programmable Gate Array) .in this paper we got the result about crossbar switch but it will not the reconfigurable.

Many chips are communications processors but not network processors. Communications processors, such as Free scale's Power QUICC chips, are closely related to network processors but serve applications with lower data rates Data rates for communications processors range from a few megabits per second to 1Gbps (for instance a single gigabit Ethernet channel). Their lower prices mean they have more integration than most network processors. For example, communications processors typically contain a RISC processor core that runs a standard MIPS, PowerPC, or ARM instruction set. By contrast, most NPUs don't include such a processor. In a communications processor, it's common for Layer 3 processing and above to be handled by this RISC processor, whereas NPUs commonly handle Layers 3 and above with proprietary packet engines. Many communications processors integrate Layer 1 and Layer 2 processing; most NPUs don't. These differences in price and performance between communications processors and NPUs mean systems designers typically specify them for widely different application. .

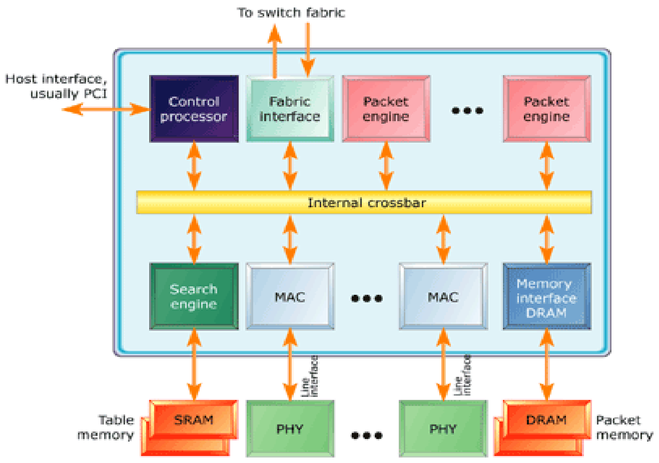


Figure: 1 Generalised Block diagram for Network Processor

The second interface is the fabric interface, which connects the NPU to the external switch fabric or, in some cases, directly to another NPU. This interface is important for building a line card but might not be used in other designs. The fabric interface should have at least as much bandwidth as the line interface, because nearly all packets in a line card must move across both interfaces. In fact, it should have enough extra bandwidth, typically at least 25%, to support fabric headers, in-band communication, and other fabric overhead. The memory interface often consists of several separate physical connections. One or more typically connect to packet memory, where packet headers and payloads are stored during processing. Packet memory also holds packets queued for speed-matching reasons or, in quality of service applications, to support multiple priority levels. As a rule of thumb, the packet memory should be 256MB for OC-48, 1GB for OC-192 (or 10 GB Ethernet), and 4GB for OC-768 data rates. For these large arrays, NPUs typically use low-cost DRAM rather than fast but expensive SRAM. Sustained bandwidth to this memory must be at least double the line bandwidth, because each packet must be written to the packet memory and later read back.

2. CHALLENGES FACED IN NPU DESIGN

2.1 Organization of computational power one challenge facing NPU designers is how to organize all this compute power. Although state-of-the-art silicon manufacturing can squeeze dozens of small packet engines on a single chip, it's difficult to connect more than 16 engines to a single memory using an on-chip bus or crossbar. Adding too many engines to a bus can cause contention, delays, and electrical problems, slowing down the entire chip. One solution is to limit the number of packet engines to 16 while increasing the performance of each engine. Most early packet engines were simple scalar (one instruction at a time) RISC processors. Some new designs have shifted to VLIW (very long instruction word) packet engines to get more performance per packet engine. The superscalar techniques you see in PC and server processors like Opteron, Pentium 4, or SPARC are less efficient than VLIW and aren't needed unless software compatibility is important. Another approach is to pipeline packet engines in such a way that each group of engines performs a specialized task. EZchip's NP-1 and Agrees APP750, to name two examples, have one group of engines that connect to lookup-table memory, while another group connects to the packet queues. The number of connections to any particular on-chip resource is thus reduced. This approach however can limit certain potential applications that use a well chosen pipeline design. For example, these pipelined chips are well designed for processing IP packets, but it's more difficult for them to perform higher-layer functions such as ISCSI and TCP termination. Yet another approach is to combine pipelining of VLIW packet engines in a dataflow fashion that increases efficiency. For this, a single instruction, multiple data (SIMD) technique can be used in order to organize hundreds of stripped-down packet engines.

2.2 Fixed or programmable:

Another factor you have to consider when choosing an NPU is the use of fixed-function coprocessors to supplement the performance of packet engines. AMCC, for example, uses only six scalar packet engines in its simplex 10Gbps NPU, so there's room for the company to add more engines in future chips. AMCC can get away with so few programmable engines because its fixed-function policy engines, search engines, and traffic managers perform much of the packet-processing task. Fixed-function logic is generally more efficient for any given task than programmable packet engines, so using coprocessors can increase performance. The obvious downside is reduced flexibility; applications have to fit the capabilities of the fixed-function blocks. A more subtle issue is relative design difficulty. Fixed-function logic typically implements complex state machines; this complexity balloons as the system designers add more features and options. On the other hand, you can replicate a single packet engine many times and can program it for a variety of tasks. Fixed-function logic should however be used judiciously. Finally, it must be considered that many networking customers have unique needs that can be handled only through programmability. A flexible, programmable device can more easily support whatever algorithms, protocols, or services a customer might want to implement. A highly programmable NPU will serve the broadest possible market. On the other hand, a well-designed NPU with appropriate use of fixed-function blocks is likely to be more efficient for mainstream applications.

3.3. Programmability spectrum

Because of these different design choices, network processors offer various levels of programmability, as Figure 3 shows. At one end of the spectrum is the entirely fixed-function logic of a "net ASIC." As programmable processor elements are added, the design moves to the right. A fully programmable NPU with few, if any, fixed-function blocks sit at the far right. Intel's IXP architecture is a good example of a fully programmable design, using packet-engine software to do almost all the work. EZchip has a highly programmable NPU chip, but its traffic-manager chip is not programmable, so the total product is less programmable than Intel's. AMCC's NP chips [8] also use a fixed-function traffic manager, and even the NPU combine’s limited packet-engine horsepower with plenty of coprocessors. Net ASICs, such as Marvell's Prestera-MX, have no packet engines at all; it's all hardwired. One limitation to programmability is the amount of hardware and software requirement. Intel's IXP family of chips requires eight to 10 times more software than AMCC's does to perform the same tasks. Tasks that are done in AMCC's hardware must however be written in software for the IXP In the extreme case, a hardwired ASIC can be used. Owing to the efficiency of fixed-Function logic, hardwired ASICs can also provide cost, power, and integration advantages. For example, the Prestera-MX delivers full-duplex 10Gbps throughput including media-access controllers (MACs), search engines, and egress traffic management in a single chip costing about $600 and consuming only 7W. With the exception of Xelerated's unique PISC (Packet Instruction Set Computer) architecture, all the available programmable solutions require at least twice as many chips, with more than twice the cost and twice the power. Programmability enables easy differentiation between NPUs with and without ASICs. Programmable NPUs also assure that with the change in technology, new features can be easily added to the existing NPUs. It comprises of three parts, with the first being the Receive to Memory part (Rx2Mem) which is tasked with counting the number of bytes in the frame being received and determining its initial byte of data. The second part is the Control part, which is tasked with forwarding frame data to the Process module and storing both the received frames as well as the processed ones. Finally, the Process module is the last part of our architecture, which is responsible for processing all the frame data by performing all the instructions supported by the design.

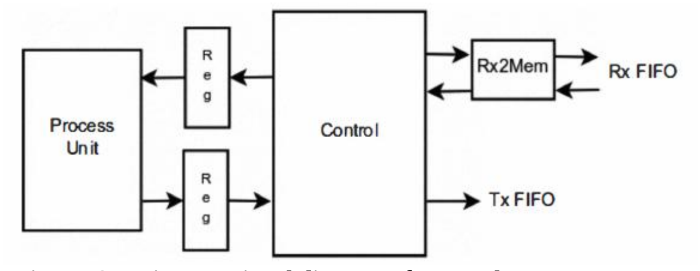


Figure: 2 Basic operational diagram of Network Processor

A. Rx2Mem Module:

This is the first module of our design, as shown in figure 2; it is responsible for three simple yet essential operations: first, it keeps a count of all the frames we have received so far through the network adapter; second, it calculates the length of each frame it receives which is essential when processing frames and altering their contents; finally, it is responsible for signalling the beginning of each frame, once again a necessary operation, since we need to be able to determine where the first byte of a frame is stored in the design's data memory.

B. Control Module

This module is the heart of our architecture (figure 2). As we have already mentioned its purpose is to forward the proper frame data to the Process module and to store received and processed frame data in the data memories. It can be abstractly divided in four parts, each on a different part of the architecture's data path. The first part is called Received to Memory (R2M), receiving data from the Rx2Mem module and storing them in the memory, along with information about each frame (its starting address in the data memory and its length in bytes).The second is the Memory to Process part (M2P),responsible of forwarding proper frame data to the Process module in order for them to be processed; this module is capable of sending the whole frame or a specific byte of the frame to the Process module, thus optimizing the performance of some of the instructions supported by the design. The third part is called Process to Memory (P2M) and it is tasked with writing processed frame data back to the design's data memory. It is also responsible for calculating any changes in the frame's length, since an Add or Remove instruction can alter a frame's length. Finally, the Memory to Transmit (M2T) part, which is the final part of the Control module, is responsible of transmitting processed frames back to the client through the board's network adapter.

C. Process Module

This module is where all the frame processing takes place (figure 2). Instructions are loaded from the instruction memory and according to each instruction; different actions are taken in order to accomplish the desired effect on the frame data. The frame data that are to be processed are requested from the Control module and after being processed according to the loaded instruction, are forwarded back to the Control module for storage on the on board memory.

3. RELATED WORKS

There are lots of commercial network processors of different companies. Some companies and respective network processors are: IBM (NP4GS3), Motorola/CPort (C-5 Family), Lucent/Agree (FPP/RSP/ASI), and Sitera/Vitesse (IQ2000), Chameleon (CS2000), EZChip (NP-1), Intel (IXP1200) and others. None of them presents reconfigurability, except the CS2000 of Chameleon However, it does not have reconfigurable crossbar switch.NP architectures have dedicated blocks to execute specific functions as an embedded ASIC (NPSoC – Network Processor System-on-Chip). Some blocks are: PCI units, memory units, packet classifiers, policy engines, metering engines, and packet transform engines, pattern processing engine, queue engine, QoS engines and other blocks.

4. RECONFIGURABLE CROSSBAR SWITCH ARCHITECTURE

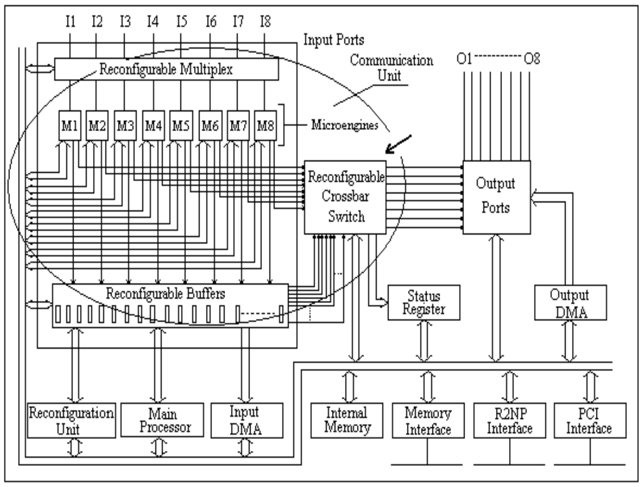


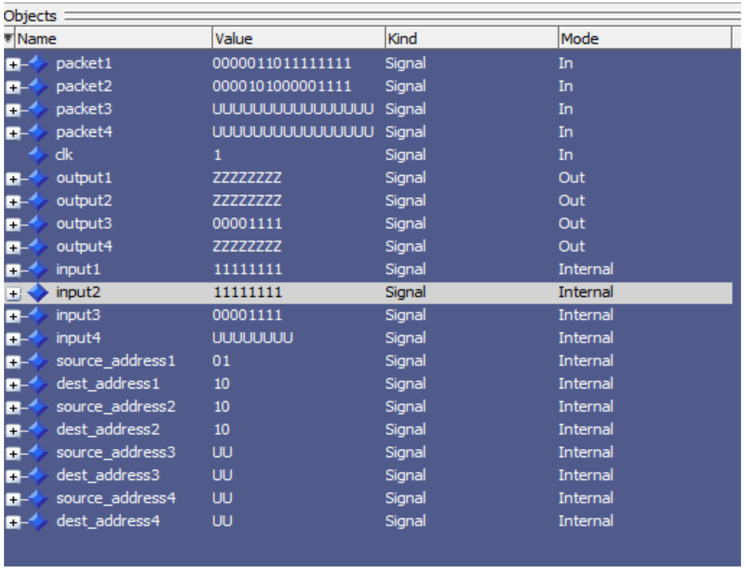
Figure: 3 Reconfigured Crossbar switch

RCS-2, presented in figure, has three main blocks: (1) connection matrix, where the topologies are implemented ;(2)decoder, that converts the reconfigurable bits for a matrix bits set and (3) pre-header analyzer (PHA). NPs can add a pre-header in the packet with the output destination. Reconfigurable crossbar switch (RCS-2) uses reconfiguration bits to implement the topology in the space. That topology actually maintains the created connections as a circuit. The reconfiguration bits set are capable of reconfiguring or implement a new topology in RCS-2 whenever necessary. RCS-2 architecture is based on two reconfiguration levels. Using these two levels it is possible to reconfigure and to readapt the crossbar switch to many network topologies and different workload situations. The first level is based on static reconfiguration using a reconfigurable device, like FPGA, Programming this device, it is possible to implement a RCS-2 with number of in and out ports (and consequently rows and columns – circuit and logic gates) limited by the device capacity. The second level of reconfiguration makes possible the implementation of different network topologies. It could be done by dynamically reconfiguration of the connection matrix nodes. These nodes determine which connections will be closed and consequently which paths exist through the crossbar switch. RCS-2 has two bits of reconfiguration to each node, which define the current topology. Only the Reconfiguration Unit and the instruction set of the network processor are able to change those bits in order to implement new topologies. Although one instruction can modify a reconfigurable bit, it only modifies the 01 and 10 formats the 00 and 11 formats are restricted to Reconfiguration Unit. A. RCS-2 Implementation to allow the first level of reconfiguration, defined in section III, and to verify the proposal of reconfigurable crossbar switch, the RCS-2 architecture was codified using VHDL. Beyond the portage of code, it was necessary to build a parameterized code to achieve the first level. To allow the second level of reconfiguration, the decoder module was codified. This module receives three sets of bits: configuration type, address, and data. The configuration type determines which kind of configuration must be made: node, line or column. When it is a node configuration type bits, the decoder fills the configuration bits of a connection matrix node indicated by address with the data bits. When it is line or column configuration type bits, the entire line or column (all nodes) indicated by address bits is filled by data bits. In this way, these types of reconfiguration can be made in one reconfiguration time, configuring in Parallel all nodes. This time is defined by the technology of the target device. The code and the architecture were verified through behavioural simulation using the Model Sim XE II from Model Technology.

5. ADVANTAGES & DISADVANTAGES

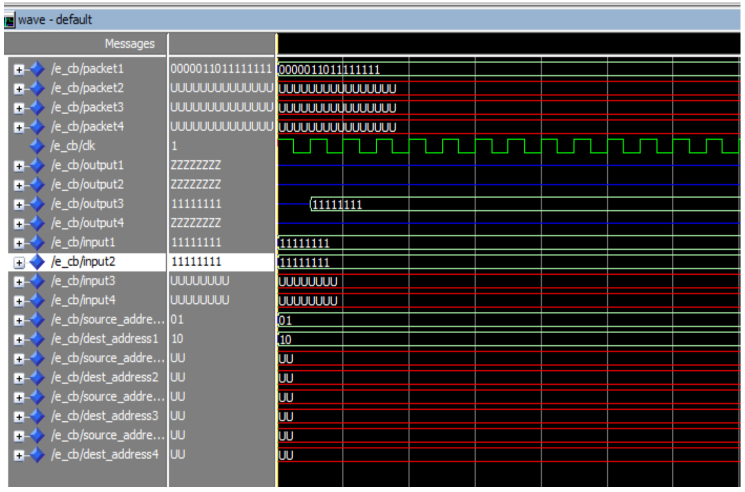
The main competitors to NPUs are general-purpose microprocessors and custom ASICs. In previous networking systems, microprocessors were used to perform routing functions in low-end devices because of their low cost, general availability, and ease of programming. Microprocessors don't have enough performance for high-bandwidth devices, so these boxes used custom ASICs. ASICs provide ultimate control over the design. Using ASICs, a networking designer can create highly differentiated products. On the other hand, ASICs have long design cycles(9 -18 months), long debug cycles, and high development costs (millions of dollars). As a result, ASIC development is the riskiest portion of system development. Like standard microprocessors, network processors are programmable and available off the shelf, yet they can match the performance of ASICs in demanding networking applications. NPUs replace fixed-function ASICs with a programmable design, providing additional advantages. A programmable device shortens the design cycle and is more easily modified to support new or evolving standards. Programmability not only accelerates time to market, it can even enable an NPU-based router to be field-upgraded with a new protocol something that can't be done with a hardwired solution.

6. RESULTS


Objects

Name	Value	Kind	Mode
packet1	0000011011111111	Signal	In
packet2	0000101000001111	Signal	In
packet3	UUUUUUUUUUUUUUUU	Signal	In
packet4	UUUUUUUUUUUUUUUU	Signal	In
clk	1	Signal	In
output1	ZZZZZZZZ	Signal	Out
output2	ZZZZZZZZ	Signal	Out
output3	00001111	Signal	Out
output4	ZZZZZZZZ	Signal	Out
input1	11111111	Signal	Internal
input2	11111111	Signal	Internal
input3	00001111	Signal	Internal
input4	UUUUUUUU	Signal	Internal
source_address1	01	Signal	Internal
dest_address1	10	Signal	Internal
source_address2	10	Signal	Internal
dest_address2	10	Signal	Internal
source_address3	UU	Signal	Internal
dest_address3	UU	Signal	Internal
source_address4	UU	Signal	Internal
dest_address4	UU	Signal	Internal

Result (1) OBJECT WINDOW


wave - default

Messages		
/e_cb/packet1	0000011011111111	0000011011111111
/e_cb/packet2	UUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
/e_cb/packet3	UUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
/e_cb/packet4	UUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
/e_cb/clk	1	
/e_cb/output1	ZZZZZZZZ	
/e_cb/output2	ZZZZZZZZ	
/e_cb/output3	11111111	11111111
/e_cb/output4	ZZZZZZZZ	
/e_cb/input1	11111111	11111111
/e_cb/input2	11111111	11111111
/e_cb/input3	UUUUUUUU	UUUUUUUU
/e_cb/input4	UUUUUUUU	UUUUUUUU
/e_cb/source_addre...	01	01
/e_cb/dest_address1	10	10
/e_cb/source_addre...	UU	UU
/e_cb/dest_address2	UU	UU
/e_cb/source_addre...	UU	UU
/e_cb/dest_address3	UU	UU
/e_cb/source_addre...	UU	UU
/e_cb/dest_address4	UU	UU

Result (2) WAVE WINDOW

7. CONCLUSIONS

The developed reconfigurable crossbar switch architecture presented advantages due to its flexibility and high performance. This fact justifies its employment in a network processor. The capability of adapting the topology implemented on crossbar switch to the environment changes generates high performance for data processing in several situations as multiprocessors and computer clusters could be reached with modifications in The contribution of this paper is the proposed RCS-2 architecture. The first level of reconfiguration of the RCS-2 could be reached through the codification of the architecture using a hardware description language, allowing it to be implemented in several devices with dimensions determined by device capacity. The second level of reconfiguration the matrix of connections. These modifications generate an overhead. However, through the experiments, it was evidenced that the overhead time is less than the speedup obtained through the topologies implementation in RCS-2. Therefore, the RCS-2 has a better performance when compared to a TCS.

8. REFERENCES

[1] D. E. Comer, "Network Systems Design using Network Processors", Prentice Hall, 2003

[2] D. Kim, K. Lee, S. Lee and H. Yoo, "A Reconfigurable Crossbar Switch with Adaptive Bandwidth Control for Networks-on-Chip", IEEE International Symposium on Circuits and Systems, 2005

[3] G. Lawton, "Will Network Processor Units Live up to Their Promise?", IEEE Computer, Volume 37, Number 4, April, 2004,

[4] I. A. Troxel, A. D. George and S. Oral, "Design and Analysis of a Dynamically Reconfigurable Network Processor", IEEE Conference on Local Computer Networks, November 6-8, 2002

[5] J. Chang, S. Ravi, and A. Raghunathan, "FLEXBAR: A crossbar switching fabric with improved performance and utilization", IEEE Custom Integrated Circuits Conference, May 2002,

[6] L.E.S. Ramos and C.A.P.S. Martins, "A Proposal of Reconfigurable MPI Collective Communication Functions". Third International Symposium on Parallel and Distributed Processing and Applications, LNCS 3758, Nanjing, China, November 2-5, 2005

[7] S. Young, et al., "A High I/O Reconfigurable Crossbar Switch", 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, California, April .