Design of Bufferless On-Chip Routers Providing In-Order Packet Delivery

Anh T. Tran and Bevan M. Baas
University of California - Davis, USA
{*anhtr, bbaas*}*@ucdavis.edu*

***Abstract*—Previous bufferless router designs require to drop and retransmit packets or deflect them each time a network channel get conflicted. These approaches, unfortunately, make data packets and even their flits arrive at destinations out-of-order. In this work, we present a new bufferless router architecture that provides in-order packet delivery. The key idea is to utilize pipeline registers at channel links as storage elements that allow the router to operate as a wormhole router without physical buffers. The router employs a dimension-ordered deterministic routing policy without packet dropping. To obtain higher performance, we also propose a new flow control technique called Express Flow Control (EFC) that allows all flits of an in-flight packet to synchronously forward each time its head flit wins the output port arbitration. Experimental results show that both proposed router architectures guarantee in-order packet delivery. BufferlessEFC routers are 23% less latency and 60% greater throughput than bufferless routers while have 2.5% smaller area and 4.7% lower power.**

I. Introduction

As technology continues to scale with Moore's law, designers add more processing elements onto a single chip to obtain higher performance through increased parallelism [1]–[3]. Networks on chip were shown to be feasible and easy to scale for supporting a large number of processing elements rather than point-to-point interconnect wires or shared buses [4]. Routers of an on-chip network typically have buffers at their input or output ports for temporarily storing data packets while they are being forwarded to destinations [5]. Unfortunately, on-chip router's buffers can easily consume more than half of a router's total area and power budget [6]. Bufferless router designs that allow removing buffers from routers are therefore very attractive.

There are two approaches for designing bufferless routers proposed in the literature. The first approach is to drop packets each time a channel confliction occurs; routers then notify the source processor to retransmit the dropped packets [7]. Another approach is to deflect packets to different output channels of routers with a defection policy that must guarantee all packets will eventually arrive at their destinations [8]. In these two methods, unfortunately, the transmitted packets often arrive at destinations out-of-order. Therefore, additional buffers are needed to reorder these packets before they are consumed by processors. The size of these reordering buffers may be larger than the buffers originally removed from routers, thus negating the benefit in some cases.

In this paper, we propose new bufferless router architectures that provide in-order packet delivery. The key idea is to utilize pipeline registers at channel links as storage elements that allow the router to operate as a wormhole router without physical buffers. This also reduces router latency because buffer write and read stages are fully removed from the router datapath. To obtain higher performance, we also propose a new flow control technique called Express Flow Control (EFC) that allows all flits of an in-flight packet to synchronously forward each time its head flit wins the output port arbitration.

This paper is organized as follows: Section II describes the proposed bufferless router architecture and its performance analysis. New express flow control for improving the performance of bufferless router is presented in Section III. The experimental results are shown in Section IV with evaluation and comparison between two proposed bufferless routers. Finally, Section V concludes the paper.

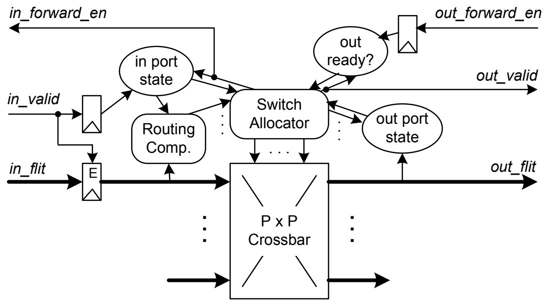

Fig. 1. The proposed in-order bufferless router that utilizes pipeline registers for storing data flits at input ports.

II. Proposed Bufferless Router Providing In-Order Packet Delivery

A. Bufferless Router Architecture

Instead of using a buffer at each input port, we utilize a pipeline register to keep only at most one flit at the time. Input flit register is flip-flops with enable signal which are normally available in the standard cell library (or each can built from a standard D flip-flop and a 2-input MUX). Between two nearest neighboring routers, an associated valid signal is sent along with data flit bits as shown in Fig. 1. A backward flow control signal is sent back to the upstream router to avoid overwriting at its input register.

At each clock raising edge, input flit register at each input port catches a new flit from *in_flit* bits if the corresponding *in_valid* is high (otherwise, its old flit value is maintained). When the head flit is written to the input register, its output port is computed by a routing computation logic (RC), then the request is sent to a switch allocator (SA) to access whether it is allowed to traverse through the crossbar (Xbar) and the output link forward to next router. We use a XY deterministic routing algorithm that guarantees all packets to traverse on the same route for each pair of a source and a destination. In addition, flits and packets are not allowed to drop, therefore keep all packets to arrive at destinations in order. Round-robin arbiters are used for the SA. We can pipeline the router to multiple stages, but in this work, for simplicity, we combine all these activities (RC, SA, Xbar and link traversal) in only one clock cycle.

Once its SA request is granted, the corresponding flow control signal *in_forward_en* is asserted to notice its upstream router that it is ready to accept a new flit. Otherwise, *in_forward_en* is deasserted. The flow control signal is pipelined at upstream router as shown in Fig. 1 to keep the whole internal router datapath working stable in one clock cycle. Each input port or output port of the router has a finite state machine to keep trace its states (IDLE, WAIT or BUSY). Head flit of a packet will set states of these ports and setup the Xbar once it is granted, then body flits will inherit these states to travel to the output port without the needs of RC and SA accesses. One the tail flit is sent, it also resets the states of its input and output port so that they are ready for serving new packets.

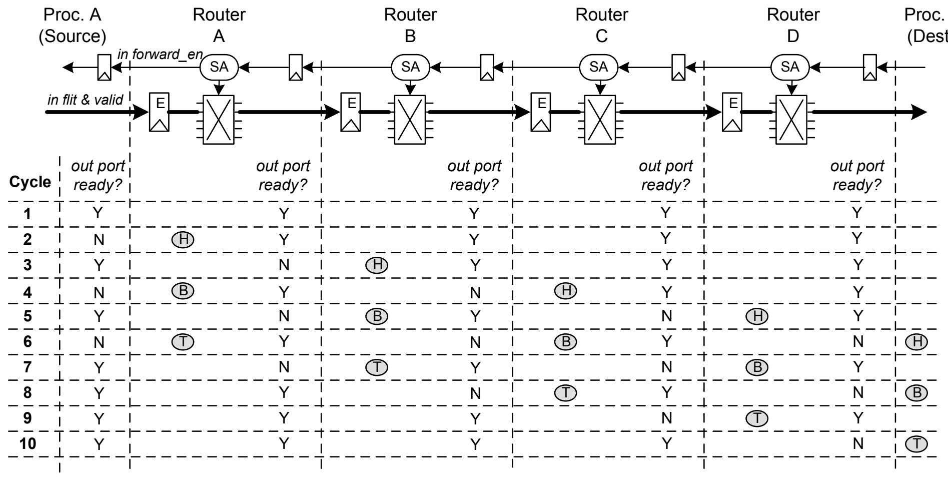

Fig. 2. Illustrated time diagram of activities of bufferless routers on the path from source to destination while transferring a packet of three flits.

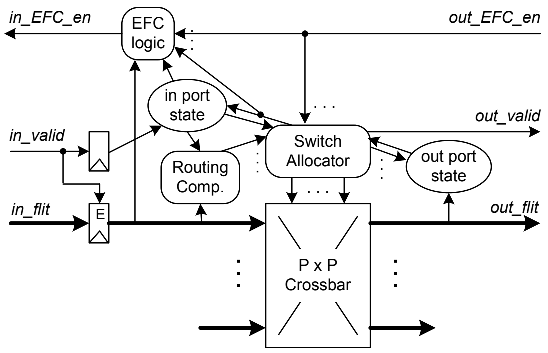

Fig. 3. BufferlessEFC router architecture.

B. Performance Analysis

Figure 2 illustrates an example when a packet of three flits is sent across four routers from Proc. A to Proc. D assuming no conflict at output ports for simplicity. At cycle 1, all states are reset, Proc. A notices the input local port of its associated router A being ready, then it sends the head flit. At cycle 2, the head flit is written to the input register of router A, and the processor output port is set to be not ready. Because router A's output port now is ready, the head flit is allowed to traverse to router B, and *forward_en* is set high. At cycle 3, the head flit is written into the input port of Router B and the output port of Router A is now not ready. At the same time, Proc. A now notices its output port is ready again (due to the corresponding pipelined *forward_en*), so it sends a new body flit. At cycle 4, the head flit is written to the input port of router C while body flit is written to the input port of router A. The process is continue until the whole packet is received by Proc. D that takes 10 cycles in total.

There are two important points shown in this example. First, each input register of a router only accepts a new flit each two cycles. This fact increases packet latency even without network congestion. In general, at low load traffic, a packet with L flits needs $(N + 2L)$ cycles to travel from a source to its destination with distance of N routers. Second, as observed from horizontal axes of the figure, two consecutive flits of a packet separates together one router; therefore each packet with L flits spreads across $2L$ routers. Because the output links of routers must be held for the whole packet until its tail flit is sent, other packets have to wait in more cycle times to be served by these links. This situation is much worse when the network traffic becomes heavy; thus the network throughput quickly reaches its saturation, especially for long packets as will be shown in experimental results in Section IV.

III. Express Flow Control for Bufferless Routers

A. Architecture

For removing stall cycles between flits of a packet, we propose a new bufferless router as shown in Fig. 3. In this design, the pipelined *forward_en* signals are removed. An express flow control (EFC) logic signal is added to each input port which allows all body and tail flits of a packet to notice whether its head flit has gotten granted to advance. We name this router design a bufferlessEFC router. All other parts of the bufferlessEFC router are kept the same as the bufferless router described in section II except the output ready logic circuits are also removed along with *forward_en* signals.

EFC_en signals is only set by the head flit of each packet once it is granted to traverse to the next router. To avoid this *EFC_en* signal propagating through a long path across more than one packets, the tail flit should reset this signal. An empty input flit register should assert its corresponding *EFC_en* signal so that allows its upstream router to forward a flit in next cycle if any.

B. Performance Analysis

A similar example of sending a packet of three flits across four bufferlessEFC routers is illustrated in Fig. 4. At cycle 1, all *EFC_en* signals are set high because all input flit registers are empty; therefore Proc. A sends a head flit to its Router A. At cycle 2, the head flit is written to the input register of Router A. Because no output conflict as assumed, the head flit is granted to traverse to Router B. The corresponding *EFC_en* remains high (if the head flit is not granted, this signal is deasserted), so Proc. A sends a new body flit. At cycle 3, the head flit is written to the input register of Router B while the body flit is written to the input register of Router A. Once the head flit gets granted to traverse to router C, its *EFC_en* signal propagates back to Proc. A that allows the body at Router A to traverse to Router B and Proc. A to send the tail flit. At cycle 4, the head flit is written to Router C, body flit is written to Router B while the tail flit is written to Router A. This tail flit resets its *EFC_en* signal so that it does not allow Proc. A to send new flits. The process continues until all flits are received by Proc. D after 8 cycles.

As observed, there is no stall cycle between flits of a packet that reduces the overall packet latency. Because now the in-flight packet also spreads on less number of bufferlessEFC routers than bufferless routers, more packets are allowed to advance in the network hence improve much network throughput. A notice is that, because the tail flit resets its corresponding *EFC_en* signal, there is always at least one stall cycle between two consecutive packets. Therefore, the throughput would be higher if we send longer packets. However, because the EFC signal propagates across multiple routers from the head flit to its tail flit, longer packet will need a lower clock frequency to avoid timing violation as analyzed follows.

C. On The Clock Frequency

Utilizing express flow control improves performance of bufferless routers in terms of clock cycle times, but gets hurt from clock rate. For bufferless router, a flit performs routing computation, requests switch

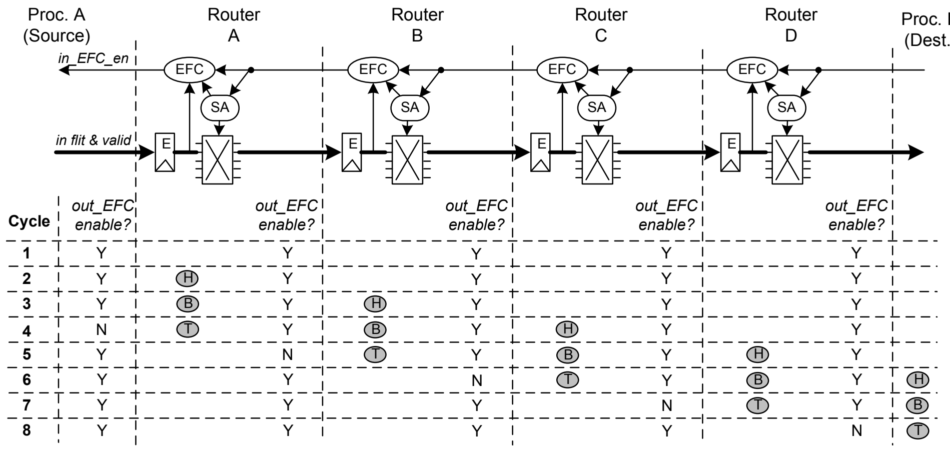
Fig. 4. Illustrated time diagram of activities of bufferlessEFC routers on the path from source to destination while transferring a packet of three flits.

arbitration, then traverses the crossbar and output link to reach the next router; therefore the clock period must satisfy:

$$T_{clk} \geq t_{RC} + t_{SA} + t_{Xbar} + t_{link} \tag{1}$$

where t_{RC}, t_{SA}, t_{Xbar} and t_{link} are delay of routing logic, switch allocation, crossbar and output link, respectively [1].

For bufferlessEFC router, the critical path starts from the time the head flit of a packet gets granted until its tail flit notices the *EFC_en* signal and then traverses to the next router. This path includes RC and SA delays at the head flit, then *EFC_en* signal is issued and propagates across $(L-1)$ routers to reach the tail (assuming the packet has L flits), then the tail flit notices this enable signal allowing it to traverse through its crossbar and output link to the next router. Thus, the clock period for bufferlessEFC router must satisfy condition:

$$T_{clk} \geq t_{RC} + t_{SA} + L \cdot t_{link} + (L-1)t_{EFC} + t_{Xbar} \tag{2}$$

where t_{EFC} is the delay of EFC control logic. Transferring longer packets should require greater clock period or lower clock rate in order for bufferlessEFC routers can be correctly functional. However, as will be shown in our experimental results in Section IV, due to their much lower network latency, running at slower clock rates still allows them to achieve higher performance compared to bufferless routers in term of absolute time.

IV. Experiments and Analysis

A. Experimental Setup

We developed cycle-accurate simulators in Verilog for both proposed bufferless and bufferlessEFC routers. Experiments were performed on a 8x8 2-D mesh network where each network node consists of a processor and a router. We evaluate the network performance over random traffic patterns for both fixed and variable packet lengths from 1 flit to 5 flits per packet. For each simulation run, we inject 100,000 packets into the network and the evaluation is performed after 10,000 warmup cycles. Latency of a packet is measured from the time its head flit is generated by the source to the time its tail flit is consumed by the destination. Average network latency is the mean of all packet latencies in the network.

B. Latency and Throughput

1) At the Same Clock Rate: Fig. 5 shows the latency vs. throughput curves in term of clock cycles for fixed-length packets from 1 to 5 flits at the same clock rate. We only show in the figure the cases of packets with 1, 3 and 5 flits for easy view (the curves with packet length of 3 and 4 should be in the middle). For the packet length of 1 flit, both bufferless and bufferlessEFC routers have the same

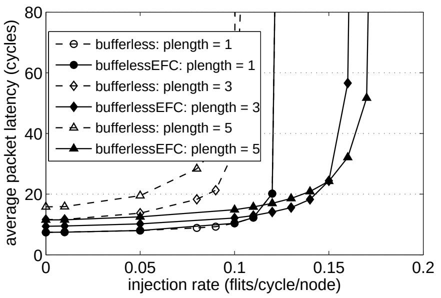

Fig. 5. Latency vs. injection rate curves in term of cycles at the same clock rate

TABLE I
Latency of router's components in term of FO4

components	t_{RC}	t_{SA}	t_{Xbar}	t_{link}	t_{EFC}
latency (FO4)	20	10	8	5	4

average low-load latency of 7 cycles and saturation throughput of 0.12 flits/cycle/node [2].

If packet length increases by one flit more, the low-load latency of bufferless routers increases 2 more cycles while bufferlessEFC routers increase only one cycle. This is because for bufferless routers, two consecutive flits of one packet are separate together at least one stall cycle. While packets in the network of bufferlessEFC routers are forwarded consecutively like as a wormhole router. Likewise, when packet length increases the saturation throughput of bufferless routers reduces quickly while that of bufferlessEFC routers increases. For packet length of 5 flits, throughput of bufferless routers is 0.09 while that of bufferlessEFC routers is 0.17 flits/cycle/node, a 90% speedup.

2) At the Maximum Clock Rate: As analyzed in Section III, for longer packets, bufferlessEFC routers need lower clock rates to avoid timing violation due to the EFC signal propagation across multiple routers from a head flit to its tail flit. Table I lists the delay of router components derived by Peh and Dally [9]. Assuming the interconnect copper wires between nearest routers are 1 mm in length. Link delay is simulated in Spice using a Π3 distributed wire model taking crosstalk effects into account [10] that is around 5 FO4 delay. At 65 nm CMOS, our Spice simulation using PTM card [11] gives FO4 delay to be around 15 ps.

From Eqn. (1) and Eqn. (2), the absolute latency vs. throughput curves of bufferless and bufferlessEFC routers are shown in Fig. 6. As shown, both bufferless and bufferlessEFC routers now have throughput (in term of flits/ns) reduced when packet length increases. For packet length of 3 flits, the throughput of bufferless router and bufferlessEFC router are 0.16 and 0.18 flits/ns/node,

[1]We ignore here flip-flop propagation, setup time and hold time delays which are small compared to element circuits mentioned above.

[2]saturation throughput is measured at the average packet latency of around 60 cycles

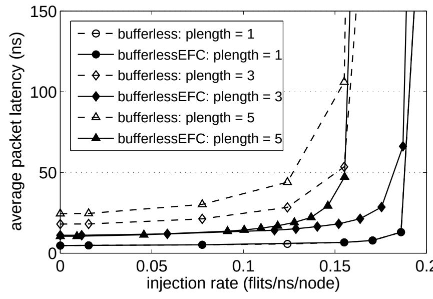

Fig. 6. Latency vs. throughput curves of bufferless and bufferlessEFC routers in absolute time (ns) at their maximum clock rates

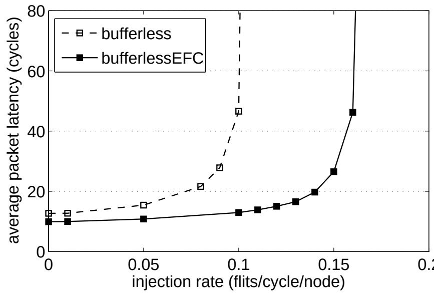

Fig. 7. Latency vs. throughput curves in term of cycles with variable packet lengths at the same clock rate

respectively. For packet length of 5 flits, their throughputs are 0.14 and 0.15 flits/ns/node, respectively. Although they have almost same throughput while transferring 5-flit packets, bufferlessEFC routers have low-load latency of 10 ns that is 60% less than bufferless routers (25 ns).

3) Variable Packet Length: We also simulated network performance for variable packet lengths. Packets are randomly injected into the network with random lengths from 1 to 5 flits. Fig. 7 shows the performance curves of bufferless and bufferlessEFC routers running at the same clock rate. BufferlessEFC router has non-load latency of 10 cycles, a 23% less than bufferless router. It also achieves a throughput of 0.16 flits/cycle/node that is 60% greater than bufferless router.

To avoid timing violation, the clock rate for bufferlessEFC routers is set to support the worst case length that is 5 flits ($L = 5$ in Eqn. (2)). As shown in Fig. 8, both routers have the same average non-load latency of 9 ns. However, now the bufferless router achieves a 0.16 flits/ns/node throughput that is 7% higher than bufferlessEFC. We can improve the performance of bufferlessEFC routers by using a fast interconnect medium for EFC_en signals. Because this signal is only 1 bit, a simple solution is shielding this wire to protect it from crosstalk. Our simulation in Spice shows that delay of 1 mm shielded copper wire is around 3 FO4. Applying the shielded wire scheme for EFC signals, now bufferlessEFC routers achieve a throughput of 0.17 flits/ns/node that is 6% higher than bufferless routers.

C. Synthesis Area and Power

The Verilog models of both routers are synthesized targeting STM 65 nm CMOS standard cells. Both routers have the same flit width of 32 bits. The synthesis results are reported in table II. Because bufferlessEFC router totally removes the circuits for detecting output port ready (which are actually finite state machines), the total register area of bufferlessEFC router is 4.3% less than that of bufferless router. Although EFC logic circuits are added to bufferlessEFC router, they are smaller than the savings of removing the $forward\_en$ logic and output port ready detecting circuits; therefore, in total, bufferlessEFC router is 2.5% smaller area and 4.7% lower power compared to a bufferless router.

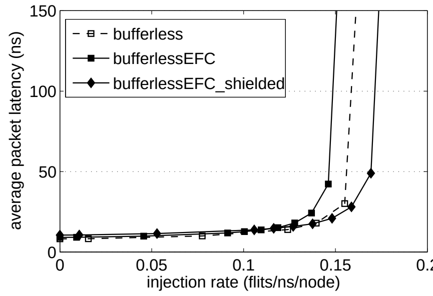

Fig. 8. Latency vs. throughput curves in absolute time (ns) with variable packet lengths at the maximum clock rates

TABLE II
SYSTHESIS AREA AND POWER RESULTS

	Area (μm^2)			Power (mW)
	registers	comb.	total	
Bufferless	3660.80	3404.44	7065.24	1.93
BufferlessEFC	3502.20	3384.16	6886.36	1.84
diff.			-2.5%	-4.7%

V. CONCLUSION

We have presented two bufferless on-chip routers which guarantee in-order packet delivery. We modeled both routers in Verilog for simulation and synthesized them targeting standard cells for comparison. When running at the same clock rate with a fixed packet length of 5 flits, bufferlessEFC routers are 31% less latency and 90% higher throughput compared to bufferless routers. When running at the same clock rate with variable packet lengths, bufferlessEFC router is 23% less latency and 60% greater throughput than bufferless router. If running at the maximum clock rate, they have the same latency while bufferless router achieves 7% higher throughput than bufferlessEFC router. When utilizing wire shielding for EFC_en bit signals to protect them from the crosstalk effect, bufferlessEFC routers can achieve 6% greater throughput than bufferless routers. The synthesis results show bufferlessEFC router has 2.5% smaller area and 4.7% lower power compared to bufferless router.

ACKNOWLEDGMENTS

This work was supported by a VEF Fellowship, SRC CSR Grant 1659 and SRC GRC Grant 1971, ST Microelectronics, UC Micro, NSF Grants 0903549, 1018972 and CAREER Award 0546907.

REFERENCES

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Francisco, USA, 2007.

[2] S. Borkar, “Thousand core chips: a technology perspective,” in *Design Automation Conference (DAC)*, June 2007, pp. 746–749.

[3] C. H. V. Berkel, “Multi-core for mobile phones.,” in *Design, Automation and Test in Europe (DATE)*, 2009, pp. 1260–1265.

[4] W. J. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Design Automation Conference (DAC)*, 2001, pp. 684–689.

[5] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, USA, 2004.

[6] Y. Hoskote et al., “A 5-GHz mesh interconnect for a teraflops processor,” *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sept. 2007.

[7] M. Hayenga et al., “SCARAB: A single cycle adaptive routing and bufferless network,” in *IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*, 2009, pp. 244–254.

[8] T. Moscibroda and O. Mutlu, “A case for bufferless routing in on-chip networks,” in *Int. Symp. on Computer Architecture (ISCA)*, 2009, pp. 196–207.

[9] L. Peh and W. J. Dally, “A delay model and speculative architecture for pipelined routers,” in *Intl. Symp. on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 255–266.

[10] A. T. Tran, D. N. Truong, and B. M. Baas, “A reconfigurable source-synchronous on-chip network for GALS many-core platforms,” *IEEE TCAD*, vol. 29, pp. 897–910, Jun. 2010.

[11] W. Zhao and Y. Cao, “New generation of predictive technology model for sub-45nm early design exploration,” *IEEE TED*, vol. 53, pp. 2816–2823, Nov. 2006.