Combining Switches for the NYU Ultracomputer

Susan R. Dickey
Richard Kenner
New York University

Abstract

A pairwise combining switch has been implemented for use in the 16 × 16 *processor/memory interconnection network of the NYU Ultracomputer prototype. The switch design may be extended for use in very large systems by providing greater combining capability; methods for doing so are discussed.*

1 Combining switch architecture

The 2 × 2 combining switch node that has been fabricated for use in the Ω network of the NYU Ultracomputer prototype [2] is composed of four each of two types of custom VLSI chips: *forward path* and *return path* components. Control for the switch is distributed as tri-state selection logic in each chip. Routing information is included in each message. Flow control avoids the necessity of handshaking when transmitting messages and allows pre-computation of signals that control data movement from stage to stage. The components accept two and four packet messages, 32 bits per data packet, and allow the first packet of a message arriving when the queue is empty to exit at the next cycle. Details about packaging, message types and message formats and the internal logic of the two component types used in the network are given in [1].

The **forward path component** (FPC) is essentially a single combining queue (Figure 1), based on the systolic queue designs in [3] and [7]. The ALU is embedded in the first packet location of the queue, rather than on the critical off-chip path, so that message transmission is not delayed by the combining logic. Separate control logic decodes the op code to determine the ALU operation and the length of the message and handles the flow control.

The **return path component** (RPC) contains a wait buffer (associative memory array holding the data from a pair of combined messages), an ALU, and two non-combining queues (Figure 2). A wait buffer is associated with each input/output pair, but the wait

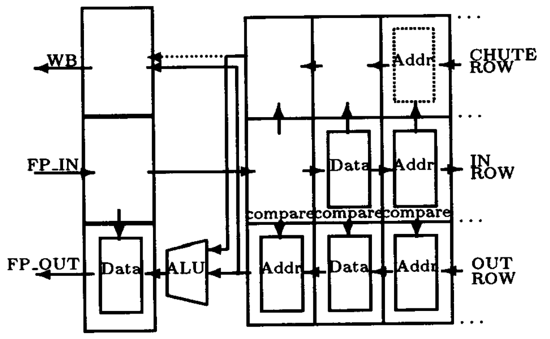

Figure 1: Design of systolic combining queue.

buffer input buses associated with a forward path output port can be tied together since they will never both be driven at the same time. A message received on RP_IN starting at cycle $2t$ is sent to the main queue and also to the wait buffer where its address packet is compared with the messages currently in the wait buffer. If a match is found, the wait buffer asserts its *match* line during cycle $2t + 1$. The output of the ALU is sent to the decombined queue at cycle $2t + 2$ so that the two queues receive the first packets of their messages at cycles of the same parity.

Both the FPC and the RPC have been fabricated using the MOSIS service in 2 μ CMOS with 132 pin packages. Both parts run solidly at 10 MHz, the upper limit of performance which can be measured in our test rig. A single-board, 2-input/2-output non-combining network composed of eight of the forward path components with their "don't combine" signal held asserted has been used in a 2-PE/2-MM system for over a year, functioning correctly at all speeds at which the memory and processors work reliably (up to 15MHz). A 4 × 4 combining switch board runs at 12.5 MHz and should be in use in a 16 PE prototype in the fall of 1992.

With higher pin count packages, a switch can be im-

0-8186-2772-7/92 $3.00 © 1992 IEEE

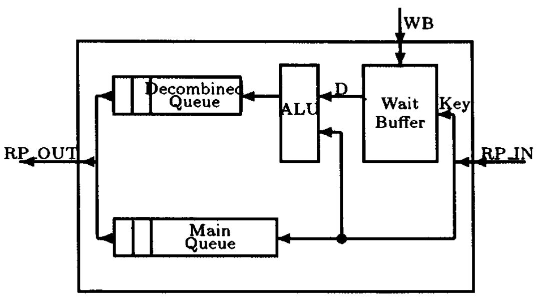

Figure 2: Block diagram of a return path component.

plemented with fewer chips; the next level of integration is achieved by packaging components which share output ports together. Pin counts, area estimates and transistor counts for these packaging alternatives are shown in Table 1.

The area estimates include routing for internal buses, but do not include pads. The transistor count is slightly overestimated for the higher levels of integration because some of the control logic is duplicated in each part. The switch design is pin-limited rather than area-limited. Even with all the logic on a single chip, the area for internal logic would only be approximately 6 mm × 6 mm in a process with 0.8 micron feature size.

2 Queues with greater combining capability

In our design, a combining queue with 2 inputs and 1 output is implemented by multiplexing the outputs of two combining queues each having 1 input and 1 output. Let the abbreviation a, b queue stand for a combining queue having a-inputs and b-outputs. A disadvantage of implementing a 2,1 queue as two 1,1 queues is that requests in separate 1,1 queues cannot be combined. A further limitation of the current design is that it combines only pairs of requests so that if N requests to a single combining queue are all directed at the same address, at least $\lceil N/2 \rceil$ will emerge. Requests to the same location that enter via different ports will not be combined but will exit via the same output port and hence will be eligible for combination at the next stage of the network. In particular, unless a *single* processor has multiple outstanding requests for the same memory location, the first stage of the network will perform no combining.

Three-way combining can be supported in a 1,1 queue by adding a second CHUTE to the design shown in Figure 1. When an entry in the IN row matches one in OUT, the first entry moves to a CHUTE unless the corresponding entries in both CHUTEs are full, which means that three items have already combined. This design presents several difficulties.

- Logic is needed to decide which CHUTE is to obtain a matched entry.
- Two messages must be sent to the wait buffer. If they are sent on the same cycle, more pins will be needed. If they are sent serially, there is a problem with flow control for consecutive triple combines.
- The ALU in the FPC must now accept three inputs. For correct decombining, the ALU in the RPC must either accept three inputs, to combine the Out data packet and the Chute0 data packet with the the data packet from memory, or the Out and Chute0 data packet must be added together before being stored in the wait buffer.
- The wait buffer must be able to match two messages at once.
- Bursts of three messages may increase delays on the return path [5].

One can use still more CHUTEs to support higher levels of combining but the VLSI layout seems likely to be poor, the multi-input adder is likely to be slow, and results in [4,6] indicate that 3-way combining suffices for two-way switches.

By adding a second IN row as well as a second CHUTE, a 2,1 queue can be implemented directly. Three-way combining could be performed by having each IN row connected to both CHUTE rows, but this implementaiton has poor layout and requires separate arbitration for OUT and each CHUTE [1]. "Two-and-a-half-way" combining can be performed more cheaply by connecting the IN row only to the adjacent CHUTE. Three messages can combine as long as the second and third messages come from different inputs. Comparison of three-way with two-and-a-half-way in [6] show no significant performance difference between the two for systems with 1024 PEs and MMs.

The logic to do two-and-a-half way combining requires

- No additional storage or data paths in the main part of the two-input queue over that for the current implementation of two one-input queues.

	Transistors	Area	Pins				
	10^3	$10^6\lambda^2$	Input	Output	Tri-state	Clocks	Total
8 chips per switch							
FPC	29	18	43	2	74	2	121
RPC	50	32	70	4	35	2	111
4 chips per switch							
FPC	58	36	82	76	0	2	160
RPC	100	64	134	38	0	2	174
2 chips per switch							
FPC	116	72	84	150	0	2	236
RPC	200	128	136	74	0	2	210
1 chip per switch	316	200	152	150	0	2	304

Table 1: Signal pin count, area and transistors per chip for 2 x 2 combining switches to be used in a 256-PE system with a 4 gigabyte address space

One fewer row is required, though arbitration must be carried out for each slot in the OUT row, not just at the output.

- The same number of wait buffer input ports as in eight chip implementation with 1,1 queues, though the buses cannot be tied together and thus more pins would be required for the 4 chip and 2 chip designs in Table 1.
- Additional ALU operations on the Out and two Chute outputs, either with a tree of 2-way ALUs or with special 3-way carry logic, as for the 1,1 queue with three-way combining.

Since the combined messages came in on different input ports on the forward path, they are guaranteed to exit from different wait buffers on the return path. Therefore, decombining is unlikely to create a problem with bursty traffic. Furthermore, if the combination of the data packets from the OUT row and one of the CHUTE rows is done on the forward path, the wait buffer design can remain simple. With one wait buffer for each input-output pair, the two combined messages will be in different wait buffers, and can be matched and queued in parallel.

References

[1] Susan R. Dickey and Richard Kenner. A combining switch for the NYU Ultracomputer. Ultracomputer Note No. 178, New York University, February 1992.

[2] Allan Gottlieb. An overview of the Ultracomputer project. In J. J. Dongarra, editor, *Experimental Parallel Computing Architectures*, pages 25–95, New York, 1987. Halstead Press.

[3] Leo J. Guibas and Frank M. Liang. Systolic stacks, queues and counters. *MIT Conference on Advanced Research in VLSI*, pages 155–164, 1982.

[4] Gyung Ho Lee, Clyde P. Kruskal, and David J. Kuck. The effectiveness of combining in shared memory parallel computers in the presence of 'hot spots'. In *Proceedings of the 1986 International Conference on Parallel Processing*, pages 35–41, August 1986.

[5] Gyungho Lee. Another combining scheme to reduce hot spot contention in large-scale shared memory parallel computers. In *Lecture Notes in Computer Science, Vol 297: Supercomputing*, pages 68–79. Springer-Verlag, 1988.

[6] Yue-sheng Liu. *Architecture and Performance of Processor-Memory Interconnection Networks for MIMD Shared Memory Parallel Processing Systems*. Ph. D. Dissertation, New York University, Department of Computer Science, September 1990.

[7] Marc Snir and Jon Solworth. The Ultraswitch – a VLSI network node for parallel processing. Ultracomputer Note No. 39, New York University, January 1984.