An Associative Self-Compacting Buffer for Communication Switches

José G. Delgado-Frias and Asif Jafri
Department of Electrical Engineering
State University of New York
Binghamton, NY 13902-6000

Abstract

This paper describes a novel VLSI CMOS implementation of a self-compacting buffer (SCB) for the dynamically allocated multi-queue (DAMQ) switch architecture. The SCB scheme dynamically allocates data regions within the input buffer for each output channel. The proposed scheme provides a high-performance solution to buffered communication switches that are required in interconnection networks. This performance comes from the DAMQ approach as well as the Associative Channel Pointer implementation and novel circuitry. The major components of the SCB are described in detail in this paper. The system has the capability of performing a read, a write, or a simultaneous read/write operation per cycle.

1 Introduction

An n by m buffered switch is a critical component in many interconnection networks. The performance of these networks is closely related to the architecture of the buffered switch. This paper describes a VLSI design and implementation of a switch architecture that uses a self-compacting buffer [1].

A router is composed of input controllers, a (n by n) switch, and output controllers. The input controller receives incoming packets and determines the appropriate output channel number according to a routing algorithm. The (n by n) switch delivers the packets from n input controllers to the n output controllers and the output controller sends the packet to the neighboring node. The input controller has three major functions. First, the input controller is responsible for receiving the packet and distributing the header part of the packet to the routing algorithm handler and to the packet flow controller. Second, it determines the output channel number based on the header information which is received from the input controller [4, 5]. This task is carried out by the routing algorithm handler. Third, it allocates and manages the buffer space for incoming and outgoing packets [6]. This function is performed by the packet flow controller.

Tamir and Frazier [2] developed a scheme to dynamically allocate buffer space using a dynamically allocated multi-queue (DAMQ). Space allocated for each buffer changes dynamically to fulfill the buffer space demands at a particular time. This is achieved using a linked list. It has been reported that the DAMQ switch achieves the best performance among switches of this type [1, 2, 3]. The self-compacting buffer implements the DAMQ using a small amount of hardware and taking advantage of VLSI technology.

2 Self-Compacting Buffer

The self-compacting buffer (SCB) architecture has been organized to implement the DAMQ scheme for buffer management. The SCB consists of a data buffer, buffer/pointer controller and channel pointers. Figure 1 shows the SCB organization.

The function of the SCB is to store incoming packets from the input port and transfer outgoing packets to the switching network. An output channel number, received from the routing algorithm decoder, points to an address in the buffer where data is stored. The channel pointer determines the buffer address for that channel number and accordingly updates the pending actions (read or write). The F (first) and E (empty) section of the channel pointers determines which row is the top/bottom of a the particular channel block. The Channel pointers pass information in the form of read/write to the data buffer and shift up/down to the buffer/pointer controller. The actual shift signals are generated here which control data movement in the data buffer and

0-7803-5588-1/99/$10.00 ©1999 IEEE.

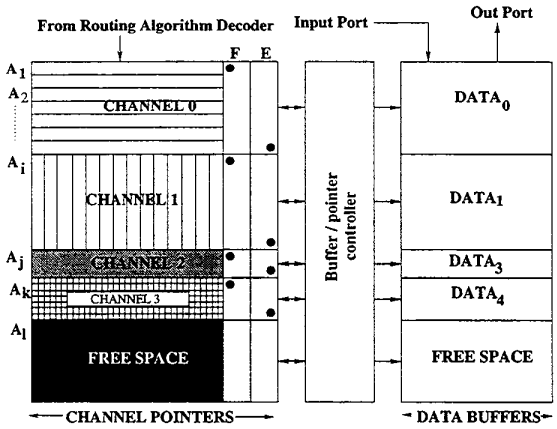

Figure 1: Self-compacting buffer organization

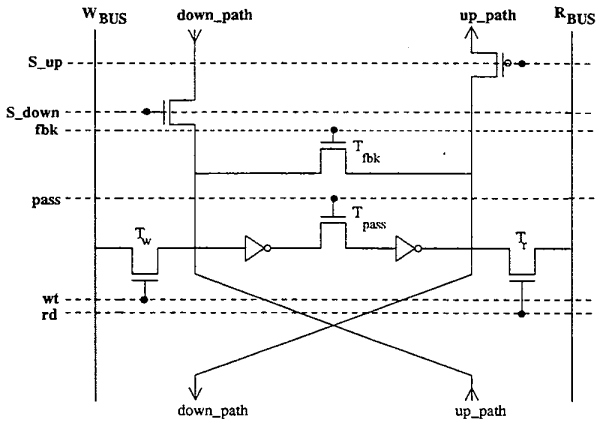

Figure 2: Buffer cell

the channel pointers.

The self-compacting buffer is divided dynamically into regions with every region containing the data associated with a single output channel. This scheme supports the dynamically allocated multi-queue (DAMQ) buffer management method introduced by Tamir and Frazier [2]. The self-compacting buffer scheme has the following properties:

Property 1. If two channels are denoted as i, k with $i < k$, then the dynamically allocated region for channel i and k always resides in a space with addresses A_i and A_k respectively where $A_i < A_k$.

Property 2. Within the space for each channel, the data are stored in a FIFO manner.

Property 3. For every output channel i, there is a number δ_i, denoting the number of entries present in the region reserved for that output channel.

The set of properties of the buffer suggests that when an insertion/deletion in the buffer occurs via a write/read operation, there should be a mechanism to access arbitrarily the region associated with a channel. In particular, if the insertion of the packet requires space somewhere in the middle of the buffer, the required space must be created by moving all the data which reside below the insertion address. Furthermore, a reading from the top of the region for output channel data may create empty spaces in the middle of the buffer. The data below the read address is shifted up to fill the empty spaces. The buffer space is maintained under the self-compacting buffer scheme.

3 Buffer Implementation

This section describes the VLSI implementation of the buffer architecture. The requirements and circuitry of the buffer is described in detail here.

For a storage location k, the following actions can occur:

- Shift up: row k moves its data to row $k-1$,
- Shift down: row k moves its data to row $k+1$,
- Hold (no action): row k holds its data,
- Write: row k moves the write bus contents onto the cell, and
- Read: row k moves its contents to the read bus.

These actions have to be performed by the buffer once the proper paths are set up. The buffer cell has to implement shift up, shift down, hold, write, and read actions as described above. A buffer cell in this buffer organization shares the up, down, read and write signals with cells in the same row. Figure 2 shows a CMOS circuit that implements the proposed buffer. When shift down (or up) needs to be performed, a communication path between the buffer cells is established to move data. When read (or write) occurs the path between R_{BUS} (or W_{BUS}) is set to transfer this data. It should be pointed out that the internal feedback in this cell is canceled while data is moved (i.e. T_{fbk} and T_{pass} are set off).

4 Channel Pointers

The channel pointers are implemented using a content addressable memory (CAM). As shown in Figure 1, these channel pointers have three different fields: Channel number, The top of a channel block, F (1-top,0-not top), and The end of a channel block, E (0-empty/end,1-not empty).

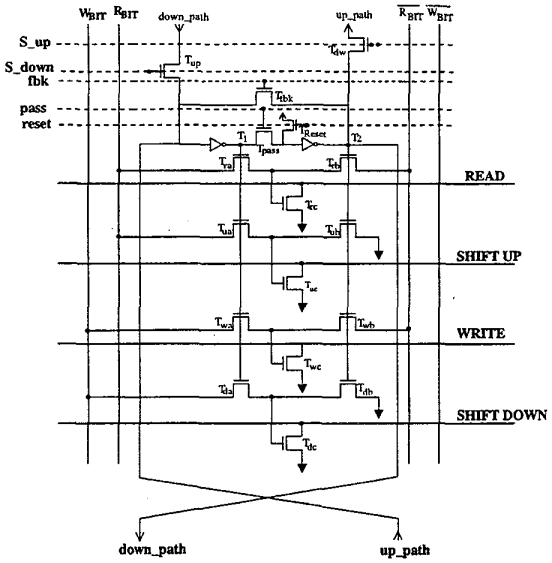

Figure 3: Channel number CAM cell.

4.1 Channel Number CAM

Figure 3 shows the CMOS circuit diagram for a content addressable memory (CAM) used to generate read, write, shift_up and shift_down signals to the buffer. The appropriate channel address bit is stored in the feedback circuit during reset through T_{reset}. The R_{bit} and W_{bit} buses hold the address bit to Write/Read and the $\overline{W_{bit}}/\overline{R_{bit}}$ hold their inverse respectively. The data in the feedback circuit is compared with the channel number to generate the read, write, shift_up or shift_down.

4.1.1 Generating read/write signals

Signal at point T_1 and T_2 in the feedback circuit are inverse of each other and allow either the R_{bit}/W_{bit} or $\overline{W_{bit}}/\overline{R_{bit}}$ to pass through T_{ra}/T_{wa} or T_{rb}/T_{wb} to control T_{rc}/T_{wc}. With these signals and transistors a comparison (at each entry in the CAM) is performed. When there is no match the precharged lines(Read and Write) are discharged through T_{rc}/T_{wc}.

4.1.2 Generating shift up and down signals

In the case of Shift_up/Shift_down signal generation there are four cases to consider.

Case1: If R_{bit}/W_{bit} is zero and channel address bit is zero a match is made keeping the respective Read/Write lines high through T_{uc}/T_{dc} which is switched off by the R_{bit}/W_{bit}. Though a match is made shift is not sure as only a part of the address is matched.

Case2: If R_{bit}/W_{bit} is zero and channel address bit is one the match is made assuming the address needed to read/write is available above in the channel pointers. The shift_up/Shift_down lines remain high through T_{uc}/T_{dc} whose gates are connected to ground via T_{ub}/T_{db}.

Case3: If R_{bit}/W_{bit} is one and channel address is zero, no match is made assuming the address needed is below in the channel pointers and the Shift_up/Shift_down lines are discharged through by the R/W bit through T_{ua}/T_{da}.

Case4: If R_{bit}/W_{bit} is one and channel address bit is one, match is made assuming we have the right read/write address or it is available above.

5 System Simulation

Figure 4 shows an IRSIM simulation of the Buffer organization. Here we are going to look at a specific case when we have a simultaneous write and read operation.

The read_addr/write_addr represent the channel address to (read from)/(write into). CAM_row$i$ represents the eight rows of the CAM pointing to the buffer shown as BUFFER_row$i$.

In Cycle1 we write into CAM_row0. The four match lines read, Shift_up, write and Shift_down display the matched data. The actual S_down signal going to the buffer is generated at the trailing edge of the clock ϕ_2 and is available until the next ϕ_1. As can be seen from simulation CAM_row0 becomes full, CAM_row1 becomes a part of the same channel but empty. The rest of the channel addresses move down one place shown by the first set of arrows. On the Buffer side Data held on the write_bus is written into BUFFER_row0 which is 0000. During Cycle2 we write into CAM_row2. The data 1010 present on the write_bus is written into BUFFER_row2.

In Cycle3 a simultaneous write and read is performed (Write address < Read address). We write into CAM_row0 again and read from CAM_row2. The Buffer/pointer controller generates the proper S_up or S_down signals. The S_up is cancelled and proper S_down signal is generated and passed on to the buffer. CAM_row0 continues to be full, CAM_row1 changes to 01 representing that it is full after the write and CAM_row2 changes to 00 showing it to be part of the same channel but empty to distinguish it from the other channels. Data 1111 is written onto Buffer_row1. At the same time Data is read from CAM_row2, therefore 07 goes back to its original state 06 but is placed at CAM_row3 as the

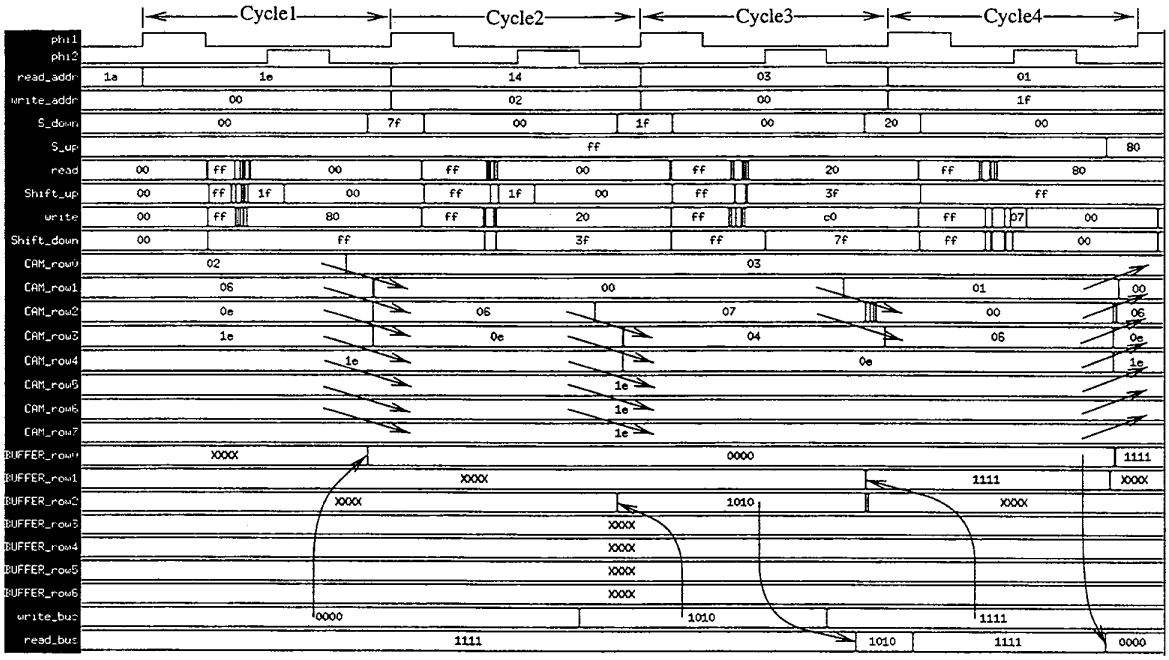

Figure 4: Simulation

write was above it. Data pointed to by CAM_row2 in the buffer is now placed onto the read_bus as can be seen is 1010.

In Cycle4, we read from CAM_row0. Appropriate matches are made and the actual S_up is generated. Data in all the rows moves up one place shown by the last set of arrows. On the buffer side data held in BUFFER_row0 is placed on the read_bus (0000).

6 Concluding Remarks

A novel VLSI CMOS implementation of a self-compacting buffer (SCB) for the dynamically allocated multi-queue (DAMQ) switch architecture has been presented in this paper. The DAMQ switch has been shown to provide the best performance among the buffered switch architectures [2]. The SCB allocates only the required buffer space per channel allowing data expansion as needed to accommodate data storage demands. We have presented the SCB architecture major components and their VLSI CMOS circuitry. The components of the SCB are capable of performing a read, a write, or simultaneous read/write operations. For each of these components we have developed novel circuitry that accomplishes the required functions.

References

[1] J. Park, B. W. O'Krafka, S. Vassiliadis, J. G. Delgado-Frias, "Design and Evaluation of a DAMQ Multiprocessor Network With Self-Compacting Buffers," *IEEE Supercomputing '94*, pp. 713-722, Washington D.C., November 1994.

[2] Y. Tamir and G. L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Trans. on Computers*, Vol. 14, No. 6, pp. 725-737, 1992.

[3] J. G. Delgado-Frias, and R. Diaz, "A VLSI Self-Compacting Buffer for DAMQ Communication Switches," *IEEE 8th Great Lakes Symp. on VLSI*, pp. 128-133, Feb. 1998.

[4] J. Park, S. Vassiliadis, and J. G. Delgado-Frias, "Flexible Oblivious Router Architecture," *IBM Journal of Research and Development*, Vol. 39, No. 3, pp. 315-334, 1995.

[5] A. A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers," *Hot Interconnects '93*, Palo Alto, California, August 1993.

[6] W. J. Dally, "Virtual-channel Flow Control," *17th Int. Symp. on Computer Architecture*, pp. 60-68, May 1990.