BUS PARTITIONABILITY AND PARALLEL PERMUTATIONS

Bruce W. Arden
College of Engineering and Applied Science
University of Rochester

Toshio Nakatani
Department of Computer Science
Princeton University

CS-TR-041-86

May, 1986

Bus Partitionability and Parallel Permutations

Bruce W. Arden

College of Engineering and Applied Science
University of Rochester
Rochester, N.Y. 14627

Toshio Nakatani

Department of Computer Science
Princeton University
Princeton, N.J. 08544

ABSTRACT

This memo provides an answer to the question whether bus-partitionability can improve the performance of parallel permutations. With permutations, no further reduction in time, due to increased parallelism, is possible through the use of partitionable buses.

May, 1986

Bus Partitionability and Parallel Permutations

Bruce W. Arden

College of Engineering and Applied Science
University of Rochester
Rochester, N.Y. 14627

Toshio Nakatani

Department of Computer Science
Princeton University
Princeton, N.J. 08544

1. Introduction

We have shown that a square grid of superposed parallel buses (Arden and Nakatani[1986a, 1986b]) has the best time performance for permutations among the linear-area interconnection networks that have been reported so far. Naturally, for the next step, the question arises whether the performance of permutations can be improved by bus-partitionability. By partitionability, we mean that the bus can be separated into arbitrary, contiguous segments. If the system can be partitioned into many subsystems that can work independently in parallel, then the time to perform an arbitrary permutation may be improved. This is the case for the divide-and-conquer algorithms (Arden and Ginosar [1982]).

In this memo, we prove that bus-partitionability cannot improve the performance of permutations. That is, it is shown that some permutations, such as a bit-reversal permutation, require the same number of steps whether the buses involved are partitioned or non-partitioned. This is true for both linear and superposed parallel buses.

2. Permutations on a partitionable linear bus

An arbitrary permutation can be performed in N cycles by N processors connected by a linear bus (Figure 2.1). A cycle is the time to make one packet transfer on a bus. Here, we assume that only one processor can send a packet on a linear bus, while all the other processors monitor the bus. If a bus can be partitioned into arbitrary, contiguous segments, then the processors in each partition can communicate with each other independently of the processors in other partitions. That is, bus partitionability induces parallelism. Divide-and-conquer algorithms nicely fit this structure. However, we prove by the following theorem that this is not the case for permutations:

Theorem 2.1: There exist some permutations that take the same number of cycles on a partitionable, linear bus as a non-partitionable one.

Proof: We assume direct transfers of packets, that is, no intermediate storage of packets, but it is easily shown that no improvement can be made even with intermediate storage. Consider a bit-reversal permutation (Figure 2.2). First, we partition the bus into two halves of equal size. Since every packet must cross the boundary, we cannot partition the bus until every processor finishes sending and receiving a packet. When the bus is ready to be partitioned, no processor has a packet to send and receive any more. Similarly, wherever a partition is made, the bus cannot be partitioned until all the processors in one partition finish sending and receiving a packet. When the bus is ready to be partitioned, there is no packet left to send and receive in that partition. Therefore, bus-partitionability does not decrease the total time.□

3. Permutations on a Partitionable Superposed Parallel Buses

$N=n^2$ processors can be organized as a $n \times n$ square grid and the processors on the same row or the same column are connected by a linear bus, where a set of n horizontal buses and a set of n vertical buses are superposed (Figure 3.1). An arbitrary permutation can be performed in $n+1$ cycles, if it is pre-scheduled. An arbitrary permutation can be performed dynamically in $2n$ cycles without pre-scheduling. Similar to the linear bus, we can assume partitionable buses for the grid of superposed parallel buses so that a $n \times n$ square grid of processors can be partitioned into arbitrary, rectangular or square segments. However, as in the case of the linear bus, the performance of permutations cannot be improved:

Theorem 3.2: There exist some permutations that take the same number of cycles on a square grid of partitionable, superposed parallel buses as on a non-partitionable one.

Proof: We again assume direct transfers of packets, that is, no intermediate storage of packets, but it is easily shown that no improvement can be made even with intermediate storage. Consider again a bit-reversal permutation (Figure 3.2). First, we partition a square grid into two halves; the upper half and the lower half. Since every packet must cross the boundary, we cannot partition the buses until every processor finishes sending and receiving a packet. When the buses are ready to be partitioned, no processor has a packet to send. Second, we partition a square grid into four quarters; the upper left, the upper right, the lower left, and the lower right. Every processor in the upper left quarter must send and receive a packet to and from the lower right quarter. Similarly, every processor in the upper right quarter must send and receive a packet to and from the lower left quarter. Therefore, we cannot partition the buses until all the processors finish sending and receiving a packet. In general,

wherever partitions are made, we cannot partition the buses until all the processors in one partition finish sending and receiving a packet. When the buses are ready to be partitioned, there is no packet left to send and receive in that partition. Therefore, bus-partitionability does not reduce the time required for the permutation.□

4. Conclusions

In this memo, it is proved by example that bus-partitionabilty does not improve the performance of permutations on a linear bus or a square grid of superposed parallel buses. Intuitively, there are permutations, for example, a bit-reversal permutation, in which all transmission path lengths are "long". Partitionability introduces parallelism when there are short paths that can be used concurrently on the same bus.

References

Arden, B. and R. Ginosar [1982]. "MP/C: A multiprocessor/computer architecture," *IEEE Trans. on Computer* C-31:5, pp. 455-473.

Arden, B. W. and T. Nakatani [1986a]. "Permutations on superposed parallel buses," Technical Report CS-24, Department of Computer Science, Princeton University.

Arden, B. W. and T. Nakatani [1986b]. "The optimal uniform schedules of arbitrary static permutations on superposed parallel buses," Technical Report CS-33, Department of Computer Science, Princeton University.

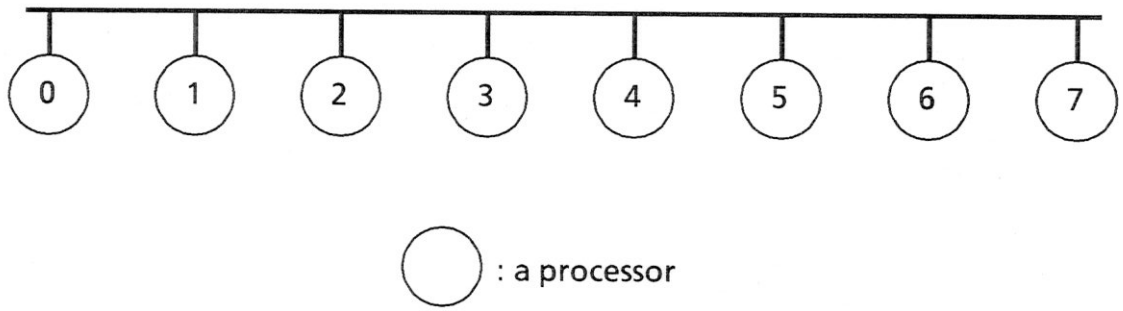

Figure 2.1a: A linear bus

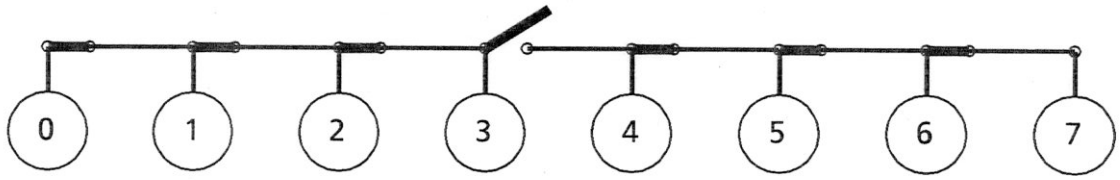

Figure 2.1b: A partitionable linear bus

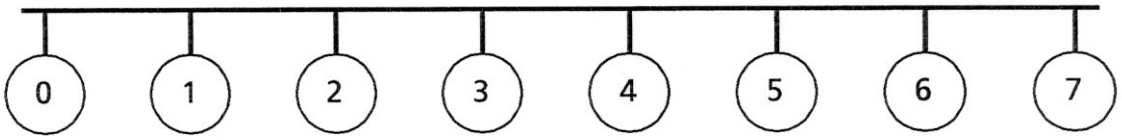

Figure 2.2a: Source addresses

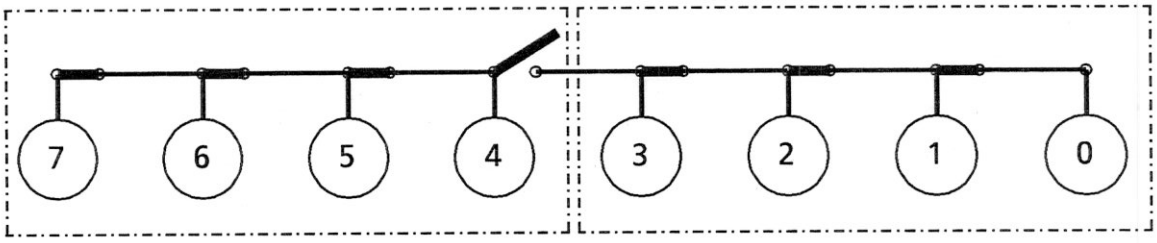

Figure 2.2b: Destination addresses (a bit-reversal permutation)

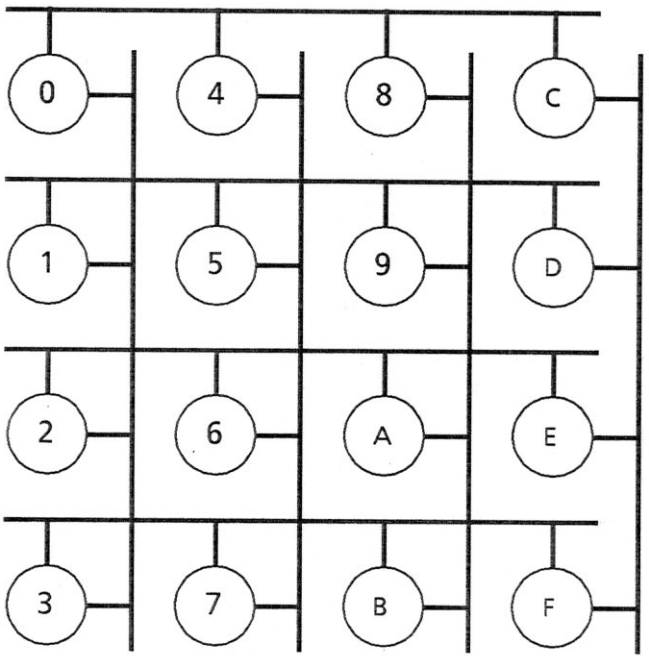

Figure 3.1a: A square grid of superposed parallel buses

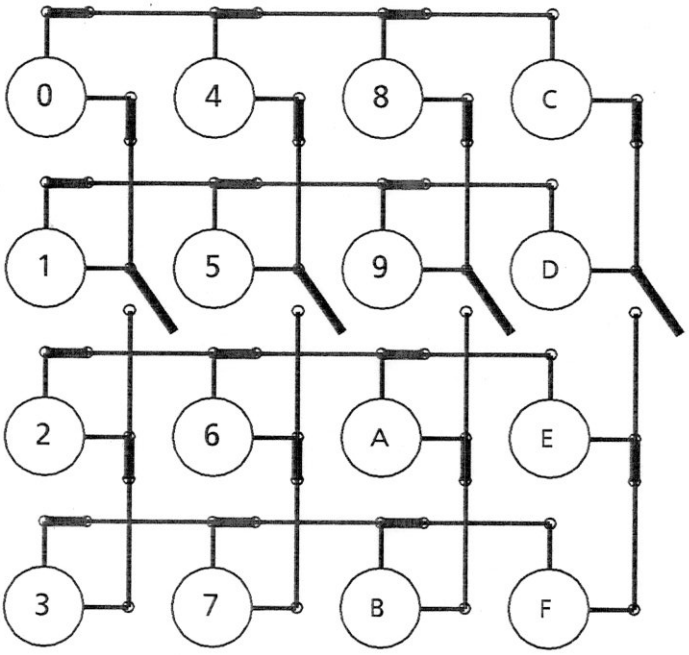

Figure 3.1b: A square grid of partitionable superposed parallel buses

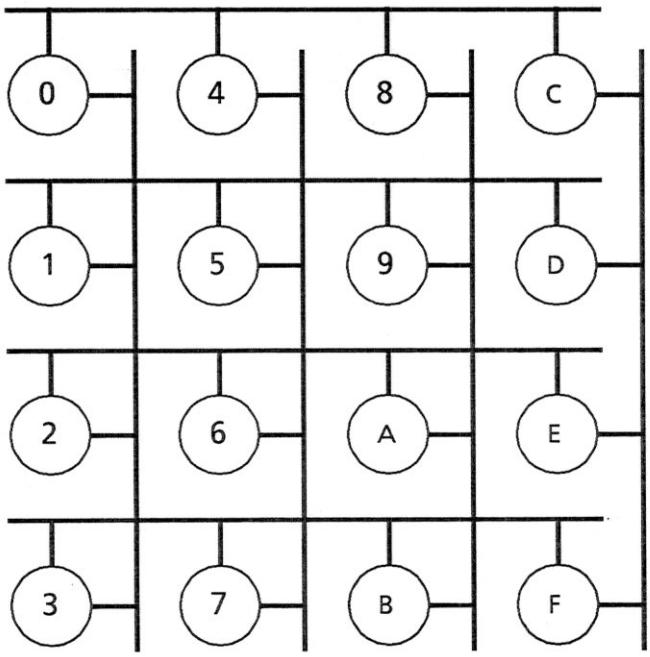

Figure 3.2a: Source addresses

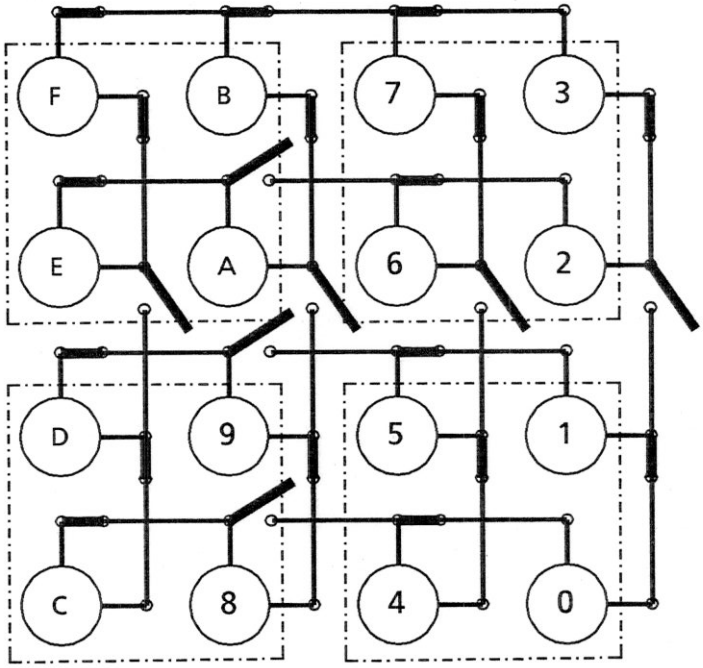

Figure 3.2b: Destination addresses (a bit-reversal permutation)