NASA Technical Memorandum 100814
ICOMP-88-3

Distributed Computation of Graphics Primitives on a Transputer Network

Graham K. Ellis
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio

Prepared for the
1988 Summer Computer Simulation Conference
sponsored by the Society for Computer Simulation
Seattle, Washington, July 25-28, 1988

NASA

DISTRIBUTED COMPUTATION OF GRAPHICS PRIMITIVES ON A TRANSPUTER NETWORK

Graham K. Ellis*
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

A method is developed for distributing the computation of graphics primitives on a parallel processing network. Off-the-shelf transputer boards are used to perform the graphics transformations and scan-coversion tasks that would normally be assigned to a single transputer based display processor. Each node in the network performs a single graphics primitive computation. Frequently requested tasks can be duplicated on several nodes.

The results indicate that the current distribution of commands on the graphics network shows a performance degradation when compared to the graphics display board alone. A change to more computation per node for every communication (perform more complex tasks on each node) may cause the desired increase in throughput.

INTRODUCTION

In an effort to increase the graphics rendering speed on a transputer based display board, a method has been developed that off loads the scan-conversion tasks to a network of other processors and frees the display board for performing only display tasks.

NETWORK ARCHITECTURE

The network architecture of the graphics computation network is shown in Figure 1. The input control master node reads in the drawing command from an application program. Since the configuration of the network and the primitives present on each node of the network are known, a decision can be made on the correct routing path for the command. For every command received by the input master node, a copy is sent to the output master node. These commands are queued up in a first-in-first-out (FIFO) buffer on the output master control node for later processing.

For the commands that do not get processed on the graphics network such as display board hardware commands, the command is sent directly to the output control master FIFO buffer and no command is sent to the graphics network.

NETWORK COMMUNICATION

The graphics commands are distributed through the network by using smart buffer processes that run concurrently (multi-tasked) on each processor in the network. The configuration of the buffers contained on each processor in the network is shown in Figure 2.

Commands sent anywhere on the network are prefixed with a command byte that specifies the type of data to expect next. There are four tags that each input buffer uses to make network routing decisions. They are as follows:

tag.global
tag.work
tag.dump
tag.result

The input buffer process controls commands sent from both the input and output master control nodes. The input buffer reads a command from any one of three input links. These are multiplexed inputs that use the occam ALT construct (Pountain 1987).

Because the transputer graphics network used in this example is a multiple instruction multiple data stream (MIMD) parallel processor with no shared memory, a method was devised to distribute the graphics data that is normally globally scoped to the entire network of processors. This was required to keep track of which window or screen coordinate system was active so the scan-conversion tasks would be performed properly.

When the input buffer process gets a tag.global command, the command is sent to both the staging buffers on the current processor and the adjacent processor. Routing decisions are made so that each processor only gets one copy of the tag.global command and its respective data packet. The data flow for the tag.global command is shown in Figure 3.

If the tag is tag.work, the next packet of data is read from the input channel to determine the destination of the work packet. The destination is determined only by the processor number, see Figure 1. Routing decisions are made by comparing the current node number with the destination node. If the work packet is for the current node, the data is sent to the first of several staging buffers. If the work packet is for another node, the data is sent to an adjacent processor based on the routing algorithm.

The return buffer is also used when routing the tag.work commands, but only is used by processors 0 and 8. This allows data to be sent to the processing node below for the tag.work or tag.global commands. See Figure 3 for the network data flow for the tag.global command.

The staging buffer processes are used to buffer pending graphics command requests as well as to keep the network from deadlocking. It is the input control master nodes responsibility to keep track of how many of each command are pending on the network. The number of commands for each node cannot exceed the number of staging buffers in the network or the network will deadlock. When the output master control buffer receives data off of the network (this is discussed below), a command is sent to the input master

*Senior Research Associate (work funded under Space Act Agreement C99066G).

control node to decrement the command counter for the appropriate graphics command.

The staging buffers act as FIFO queues and the requested graphics commands step through the staging buffers outputting to the graphics computation process. It is the graphics computation process that performs the scan-conversion task for the requested command. The computed data is packed into an array. The process then waits until a tag.dump command is received before dumping the computed data to the return buffer process.

If the tag is tag.dump, the input buffer sends a message directly to the graphics computation process and bypasses the staging buffers completely. When the graphics computation process receives the tag.dump command, it sends its computed data to the return buffer process where it is routed to the output master node. Figure 4 shows the routing from the output master control node for the tag.dump command.

Once the computed data are received by the return buffer process, the tag, tag.result, is attached to the beginning of the data packet to identify it as computed data. Based on the current processor number, routing decisions are made to send the data to the output master control node. The data flow to the output master control node for tag.result is shown in Figure 5.

SYNCRONIZATION

On a network of processors such as the one used for the graphics engine, it is not possible to determine the exact order of completion of any of the computations distributed on the network. Initially, it may not seem important to maintain the correct sequence of drawing commands that get sent to the display processor; however, when dealing with multiple windows, multiple drawing colors, and screen double-buffering, the order the display board receives the commands is critical. For example, if the application sends the following sequence of commands:

```
draw.line()
select.window()
draw.line()
```

and the graphics network completes the computations in a different order such as:

```
select.window()
draw.line()
draw.line()
```

the desired result will not be achieved.

The output master control node controls the synchronization of the graphics network. It does this by reading the commands stored in its FIFO queue and then sending a tag.dump command to the graphics network that gets routed to the appropriate primitive computation node. The command signals the primitive computation node to send its result back to the output master control node. Regardless of the completion order of the computations distributed on the network, the output node controls the data flow to the graphics display board. This maintains the correct FIFO ordering of the requested graphics display commands.

PERFORMANCE

The transputer serial links used on the graphics network are set to transfer data at 10 Mbits/sec (INMOS 1986). To check to see if the devised network did indeed increase the graphics throughput rate, several test cases were run to determine the scan-conversion, data transfer, and data display times. The tests were performed on a subset of the full network. Two processors, a compute node and the display board were used and all timings were taken using the transputer's high-priority microsecond resolution timer.

The results for two test cases are shown below in Tables 1 and 2. The distributed results are compared to the computation/display times for the display board alone. All computations were performed in integer device coordinates.

The first case tested was a line scan-conversion using Bresenham's integer line algorithm (Foley and Van Dam 1982). Times for scan-conversion, data transfer, and display are given in Table 1.

The second case tested was a circle scan-conversion using Bresenham's integer circle algorithm. These results are shown in Table 2.

Unfortunately, the results presented above do not shown any advantage to performing the distributed computation of graphics primitives. With the current data transfer rates, the data transfer time dominates over the computational time.

A more advantageous computation scheme would be to perform more computation on each node in the network. For example, the mapping from three-dimensional world coordinates to two-dimensional integer device coordinates could be performed on the graphics computation network and then only the integer draw commands would have to be sent to the display board.

SUMMARY

An array of transputers was designed to speed graphics computations by off loading scan-conversion tasks and freeing the display processor for display only. Because of the current bandwidth of the transputer serial links, and the small computation time for the scan conversions tested, a performance degradation was observed on the network when compared to the display board by itself. Until higher bandwidth serial links are available on the transputers or more computation is performed on each node of the graphics network for every communication, this method of distributing the graphics workload does not offer any performance gains.

REFERENCES

Foley, J.D.; and Van Dam, A. 1982. Fundamentals of Interactive Computer Graphics. Addison-Wesley, Reading, MA.

INMOS Corp. 1986. Transputer Reference Manaul. INMOS Corp., Colorado Springs, CO.

Pountain, D. 1987. A Tutorial Introduction to OCCAM Programming. INMOS Corp., Colorado Springs, CO.

TABLE 1. - COMPARISON OF LINE COMPUTE/DISPLAY TIMES

Operations performed	Time, μs
Scan convert line (0,0) to (511, 511)	7933
Transmit computed data to display board	12 512
Transmit data and display line	28 400
Scan convert, transmit, and display	36 399
Graphics board draw line command	14 887
Graphics board fast draw line command	3542

TABLE 2. - COMPARISON OF CIRCLE COMPUTE/DISPLAY TIMES

Operations performed, radius = 100, center = (256, 256)	Time, μs
Scan convert circle	2338
Transmit computed data to display board	37 362
Transmit and display circle	54 685
Scan-convert, transmit, display	57 064
Graphics board circle draw command	37 349

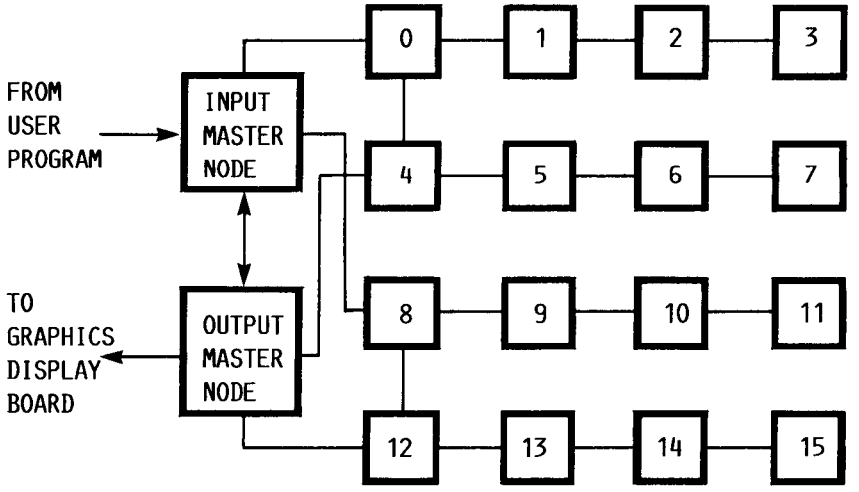

FIGURE 1. - MULTIPLE PROCESSOR GRAPHICS DISPLAY ENGINE SHOWING PROCESSOR NUMBERS FOR ROUTING COMPUTATIONS.

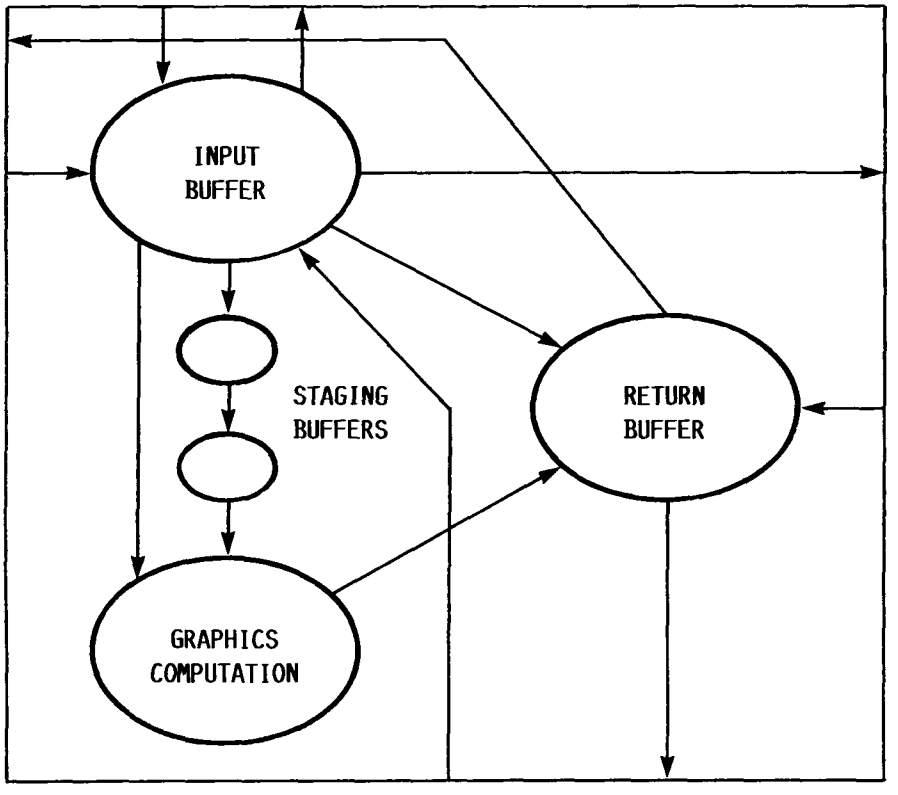

FIGURE 2. - COMPUTE NODE BUFFER PROCESSES AND COMMUNICATION ROUTING.

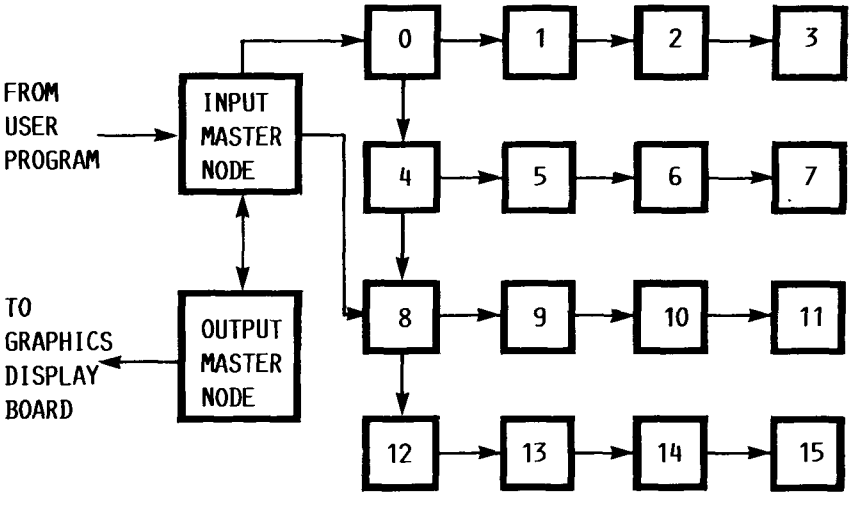

FIGURE 3. - DATA FLOW USED FOR DISTRIBUTING WORK OR GLOBAL NETWORK COMMANDS.

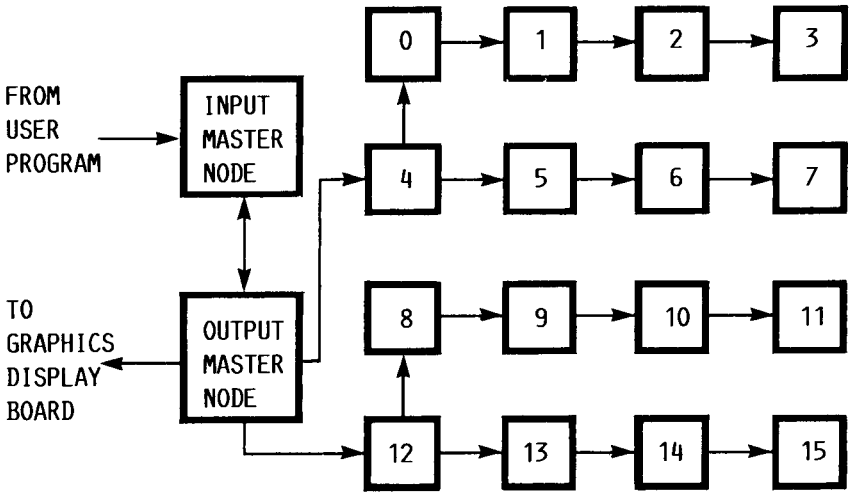

FIGURE 4. - DATA FLOW FOR REQUESTING INFORMATION FROM NETWORK.

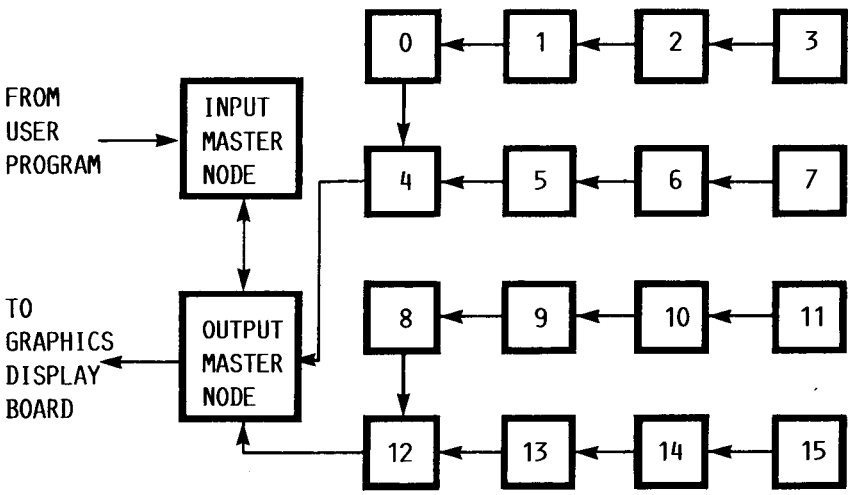

FIGURE 5. - DATA FLOW FOR SENDING COMPUTED DATA TO OUTPUT MASTER NODE.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100814 ICOMP-88-3		

4. Title and Subtitle

Distributed Computation of Graphics Primitives on a Transputer Network

5. Report Date

6. Performing Organization Code

7. Author(s)

Graham K. Ellis

8. Performing Organization Report No.

E-3999

10. Work Unit No.

505-63-1B

9. Performing Organization Name and Address

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

Prepared for the 1988 Summer Computer Simulation Conference sponsored by The Society for Computer Simulation, Seattle, Washington, July 25-28, 1988. Graham K. Ellis, Institute for Computational Mechanics in Propulsion, NASA Lewis Research Center (work funded under Space Act Agreement C99066G).

16. Abstract

A method is developed for distributing the computation of graphics primitives on a parallel processing network. Off-the-shelf transputer boards are used to perform the graphics transformations and scan-conversion tasks that would normally be assigned to a single transputer based display processor. Each node in the network performs a single graphics primitive computation. Frequently requested tasks can be duplicated on several nodes. The results indicate that the current distribution of commands on the graphics network shows a performance degradation when compared to the graphics display board alone. A change to more computation per node for every communication (perform more complex tasks on each node) may cause the desired increase in throughput.

17. Key Words (Suggested by Author(s))

Parallel processing; Transputer; Computer graphics; Multiple instruction; Multiple data stream

18. Distribution Statement

Unclassified - Unlimited
Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	7	A02