A NOVEL SLOTTED-RING ARCHITECTURE FOR PARALLEL PROCESSING: AN APPLICATION

George Yaremchuk, Carlos R. Pon, Tad Kwasniewski *, Rafik Goubran**

Department of Electronics
**Department of System and Computer Engineering
Carleton University, Ottawa, Ontario, K1S 5B6 Canada, * tak@doe.carleton.ca

1.0 Abstract

A novel efficient bus architecture is presented together with an application. The bus architecture belongs to a slotted-ring class. 32-bits of data, 14-bits address, and signalling buses span across a maximum of sixteen processors configured in a ring. The bus information arriving at each processing element can be either: passed without change, captured by the processing element (PE) and/or overwritten by the PE. The delay through each PE is 30 ns when using 1989 IC technology. Through the use of newer IC technology and due to unique physical arrangement of the bus the delay time can be reduced to approximately 15 ns. Through the use of time slot arrangements and/or signalling lines the data can reach any of the other processors in the system. Logically each processor sees the memory of the other as part of a global write-only memory. The unique hardware processor internal synchronization mechanism reduces the synchronization overhead. This paper presents implementation details of the hardware as well an application in the iterative solution of dense linear equations as the test-bed multiprocessor.

2.0 Introduction

A multiprocessor system interconnected as slotted-ring is the proposed programmable hardware tool that is best suited to modelling DSP algorithms. From the computational requirements versus best computational power available, a multiprocessor architecture was selected based on some of the following inter-communications characteristics and properties of DSP algorithms:

- The time at which data transfer occurs is very deterministic. i.e. Synchronous Data Flow [1].

- The data transfer quantity is known between producers and consumers and most the time is unidirectional.

- The transfer in a multiple PE system observed the principle of locality (i.e. most transfer occurs between those PEs which are physically closest when the PEs are arranged linearly).

- The inter-communication is burst-like (i.e. data is generally produced at the end of a computational cycle, and consumed at the start of the cycle).

With these factors being considered, the simple ring architecture was selected as best solution to the problem. However, the main disadvantage is the latency introduced by processor intervention when data is passed between non-adjacent PEs. The processor must check the incoming data to determine if it is for itself, and if not then pass it on (data routing). This process consumes valuable processor computational resources that are required if the algorithm simulations are to be performed at or near real-time. The latency introduced can be especially detrimental if overall feedback loops are present in the application. Thus an I/O processor must be added to off-load the data intercommunication routing task from the main processor. Ideally the I/O processor would allow other PEs to be mapped in the address space of the main processor, so that writing to them appears and takes the same amount of time as writing to local memory.

To solve this problem to a great extent, this system will utilize a variant of the ring architecture, coined here as the “slotted-ring”. Essentially a separate Input-Output Processor(IOP) is introduced to handle intercommunication between PEs in a pipelined “registered” ring structure [2].

Other multiprocessor architectures with similar algorithm characteristics are under investigation [3][4][5].

3.0 Slotted-Ring Bus Description

Each PE incorporates an I/O processor that operates asynchronously to the main processor. The processor
writes to other processors in a memory-mapped address space, and reads incoming data from other processors from a local dual-port memory (DPRAM). The transport of data via the ring bus is totally transparent to the processor. This concept is illustrated in Figure 1.

![Figure 1. Slotted-Ring Structure for Three Processing Elements.](image)

The mechanism for setting up a protocol to allow orderly passage of data is programmable at run time, allowing flexibility. The structure of the IOP is shown in Figure 2. The processor can transfer data to another processor by writing to the memory-mapped address block in the processor's local address space. The packet passes through a FIFO due to the asynchronous operating of the processor and bus, and is transported onto the bus by the IOP. This is transparent to the processor. The address of the packet is monitored by each IOP as the packet is shifted along the bus. If the address is for that PE or is a special broadcast address, then the IOP takes a copy of the data and places it into the dual port RAM at the address specified by the packet. Thus the packet address can be divided into two fields:

- **PE address specifying a destination PE(s).** The destination can be all PEs (i.e. a broadcast mechanism).
- **The DPRAM memory address within the destination PE.**

The division between the two fields is programmable by the user.

The proposed slotted bus architecture overcomes the major disadvantage of the "standard" ring bus - latency. This is because there is no processor intervention in moving a data packet from its source to destination - the IOP performs this transparently. The short interconnect path allows the bus clock to operate at high speed, which further reduces latency. For an N processor system the maximum bus transport latency is N-1 bus clock periods.

![Figure 2. Bus Input/Output Block Diagram](image)

4.0 **Bus Performance**

The bus is designed to operate with a minimum clock period of 30 nS (33.3 MHz maximum). During this time period in the first time slot during the bus frame sync, each slot can be filled with a data packet from a different PE, and then shifted along the bus on subsequent periods of the bus clock. The bus is reloaded with new data upon the next frame sync and the shifting repeated. Due to the shifting, a bus communication latency is introduced. The actual bus bandwidth is dependant upon how the bus is configured in terms of its frame sync length. In the general case (same as the worst case) it frame sync and bus are set up to allow communication between any PEs. A best case peak data transfer rate can be obtained when the data flow is unidirectional to the adjacent PE only. Then the bus can be configured for a unity frame sync period, and new data can be transferred to the adjacent PE every bus slot (but not further than the adjacent PE).

In the general case, the bus frame length is the same as the number of PEs in the system i.e. there are N PEs and M bus time slots, where N=M. Each PE may write a data/
address packet to the bus during the first time slot of a frame and then it is shifted M-1 times on the bus so it passes by each PE. During the first time slot, N packets are written, and then they are shifted for N-1 slots for a total of M=N slots. Thus data is transferred at a rate of N/M = N/N = 1 packets per time slot. The data transfer rate can be stated in a variety of ways, depending upon what type of system it is being compared to. For comparison to parallel data buses, the transfer rate is usually stated in data words or bytes per second. The transfer rate of this bus would be:

1 word/30 ns * 4 bytes/word = 33.3 Mwords/s = 133.3 Mbyte/s

When considering serial buses, the rate is usually stated in bits per second. This is the raw bit speed of the bus. For fair comparison between parallel and serial buses, the address bits should be included in the calculation when converting from a parallel bus speed to an equivalent serial bus speed. Accordingly, the transfer rate of the bus is:

(32 + 13) bits/30ns = 45 x 33.3 Mpackets/s = 1.4985 Gigabit/s

The inclusion of the address bits in this calculation is fair because serial links must include address information pertaining to the destination of the packet of information.

Table 1 shows the latency of communication between PEs for the minimum, average, and maximum cases. This table states the elapsed time from when a source processor writes a data item to its bus output FIFO, until the time that data is received in the local DPRAM of the destination processor D logical positions away on the ring bus, in a system with N processing elements (N PE).

<table>
<thead>
<tr>
<th>Delay for:</th>
<th>Minimum</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waiting for output slot</td>
<td>0</td>
<td>N/2</td>
<td>N</td>
</tr>
<tr>
<td>Time through bus output FIFO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Transport via bus</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Time through bus input FIFO, write to DPRAM</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Total (Bus cycles)</td>
<td>D + 3</td>
<td>N/2 + D + 3</td>
<td>N + D + 3</td>
</tr>
</tbody>
</table>

In Table 1, D is the distance along the bus of the destination PE (in the direction of the bus shifting) and N is the number of PEs installed in the system. The minimum distance is to the next PE in the direction of the bus data flow. The maximum distance is to the adjacent PE in the direction opposite the bus data flow, and the average is for communication to the PE N/2 positions away.

5.0 An Application: The Iterative Solution of Dense Linear Equation in the Test Bed Multiprocessor

The Ring Multiprocessor Architecture was tested by solving a set of linear equations of the form \( Ax = b \), where \( A \) is an \( N \times N \) (known) matrix, and \( x \) (unknown) and \( b \) (known) are (column) \( N \)-vectors. The iterative method used to solve the above system is:

\[
x_{i+1} = \frac{b_i - \sum_{i\neq j} a_{ij} x_j}{a_{ii}} \quad \text{for } i, j = 1, 2, ..., N \quad (EQ \ 1)
\]

where the superscripts denote the iteration numbers. The \( N \) equations are assumed to be diagonally dominant for the solution to be convergent.

The iteration is started by assigning an initial value to each of the components of \( x \) (say, \( x_i = 0, i = 1, 2, ..., N \)) and is repeated until the absolute value of the difference for two consecutive approximations stays below a predetermined bound \( \epsilon \) for all components of \( x \).

Because each PE in the ring multiprocessor system has its own local private memory, the problem is partitioned into tasks whose number correspond to the available number of PEs in the system (\( n \) PE's). Therefore, each PE in the ring is assigned a task with two sets of inputs: (i) slices of the matrix \( A \) and (ii) corresponding slices of the vector \( b \), consisting of the elements \( A_{ij} \) and \( b_i \) respectively, where

\[ j = 1, 2, ..., N \]
\[ i = (r-1) \times (m+1), ..., r \times m \]

\( r \): the number of the particular PE (\( r = 1, 2, ..., n \))

\( m \): the number of rows in a slice (\( = \frac{N}{n} \)).

Assuming that \( N/n \) is an integer, the problems consist of a set of \( n \) cooperating tasks in which each task commu-
nicates with all other tasks. Because all the tasks are of the same computational size, an even load distribution results for all processors, making the computation and the communication steps for all the tasks the same. Hence any arbitrary assignment of the problem tasks to the PEs in the ring would result in the same execution time for the problem.

After each iteration, each task consist of computing
\[ R = \| x_i^{k+1} - x_i^k \|, \]
for all \( i \) to check for convergence \( R \leq \epsilon \). If convergence has occurred or the maximum allowable number of iterations has been exceeded, the iteration process stops; otherwise \( x_i^k \) is replaced by \( x_i^{k+1} \) and the iterative step is repeated. In each iteration, each PE broadcasts (distribution of the data to all other PEs) the new computed \( x_i^{k+1} \) values and the \( R_{\text{max}} \) for testing against tolerance.

The theoretical speedup of the algorithm described above is given by:
\[
S(n) = \frac{N \times (N-1) \times (t_a + t_m)}{N \times (N-1) \times (t_a + t_m) + N \times t_b} \tag{EQ 2}
\]
where the numerator of eq. 2 represents the execution time on a single processor. The denominator represents the execution time on \( n \) PEs, \( t_a + t_m \) are the times to perform and addition and a multiplication respectively, and \( t_b \) is the time for a PE to broadcast a data value to all the processors in a ring topology. After the derivation of the linear speedup, a parametric study of the algorithm on the ring architecture is undertaken. The results are taken for different values of \( N \), with \( n = 4 \), \( t_b = 200 \text{ nsec} \) and \( (t_a + t_m) = 2 \mu \text{sec} \). Table 2 shows the speedup as a function of \( N \).

<p>| TABLE 2. Speedup performance for a 4 PE |</p>
<table>
<thead>
<tr>
<th>N</th>
<th>16</th>
<th>40</th>
<th>80</th>
<th>120</th>
<th>160</th>
<th>200</th>
<th>240</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(n)</td>
<td>3.75</td>
<td>3.9</td>
<td>3.95</td>
<td>3.96</td>
<td>3.98</td>
<td>3.98</td>
<td>3.99</td>
</tr>
</tbody>
</table>

The actual simulations on the ring architecture were performed with 4 PEs, whose run times are shown in Figure 3 in dotted lines. The theoretical performance are also shown in the same figure with solid lines. Both cases are displayed for 1PE and 4PE. The segmented line displays the actual simulation in a Sun Sparc-2 station, which are comparable to one of a single PE. The speedup of the theoretical iteration (solid line) and actual runs (dotted line) are displayed in Figure 3.

The speedup eq. 2 indicates that the communication time is \( O(N) \) while the computation time is \( O(N^2) \). Thus

6.0 Conclusions

The ability of the slotted-ring architecture to solve large dense matrix with linear speedup, represents a good benchmark for the solution of many general purpose scientific problems as well. Meanwhile the slotted-ring architecture is being tested as main vehicle to be used in the area of circuit simulation at the transistor level.

7.0 References


