World Academy of Science, Engineering and Technology
International Journal of Aerospace and Mechanical Engineering
Vol:1, No:3, 2007

Parallel Computation of Data Summation for Multiple Problem Spaces on Partitioned Optical Passive Stars Network

Khin Thida Latt, Mineo Kaneko, and Yoichi Shinoda

Abstract**—** In Partitioned Optical Passive Stars POPS network, nodes and couplers become free after slot to slot in some computation. It is necessary to efficiently utilize free couplers and nodes to be cost effective. Improving parallelism, we present the fast data summation algorithm for multiple problem spaces on $POPS(g, g)$ with smaller number of nodes for the case of $d = \sqrt{n} = g$. For the case of $d > \sqrt{n} > g$, we simulate the calculation of large number of data items dedicated to larger system with many nodes on smaller system with smaller number of nodes. The algorithm is faster than the best know algorithm and using smaller number of nodes and groups make the system low cost and practical.

Keywords**—** Partitioned optical passive stars network, parallel computing, optical computing, data sum

I. Introduction

THE optical Technology offers simple interconnection schemes with straightforward layouts providing with complex logical interconnection patterns. The optical passive star (OPS) is often suggested as a platform for implementing the optical network. The Partitioned Optical Passive Stars (POPS) network was proposed in [3], [4], [5], [6], [7], [8], and [9] as a fast optical interconnection network for multiprocessor systems. The POPS network uses multiple optical passive star (OPS) couplers to construct a flexible interconnection topology.

Gravenstreter and Melhem [4] have shown the embedding of rings and tori on POPS networks when the number of processors n and the degree d of an OPS coupler are powers of 2. Data broadcasting algorithms are also developed in [4] and [6]. Berthome and Ferreira [1] have shown that POPS networks can be modeled by directed stack-complete graphs with loops. This is used to obtain optimal embedding of rings and de Bruijn graphs into POPS networks. Berthome and Ferreira [1] and Berthome et al. [2] have generalized the results obtained by Gravenstreter and Melhem [4] for embedding of rings [1] and tori [2] in POPS networks with arbitrary values of n and d. Sahni [7] has shown that an n processor $POPS(d, g)$ can simulate every move of an n processor SIMD hypercube. Sahni [7] has presented algorithms for several fundamental operations like data sum, prefix sum, rank, adjacent sum, consecutive sum, concentrate, distribute, and generalize. In another paper, Sahni [8] has presented fast algorithms for matrix multiplication, data permutations, and BPC permutations on the POPS network. Datta and Soundaralakshmi [10] presented fast algorithms for data sum, prefix sum and permutation routing on $POPS(d, g)$ such that $d > \sqrt{n} > g$.

The rest of this paper is organized as follows: in section 2, we discuss some details of the POPS network topology. Data sum algorithm is presented in section 3 and finally, we conclude with some comments in section 4.

Authors are with Japan Advanced Institute of Science and Technology 1-1, Asahidai, Nomi-shi, Ishikawa, 923-1292 Japan. (e-mail: kt-latt@jaist.ac.jp)

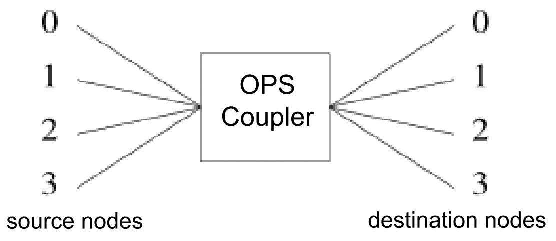

Fig. 1. An optical passive star coupler with four source and four destination nodes.

II. POPS Topology

This section describes the topological approach to providing multiple physical data channels. POPS networks use a multiple passive star topology. POPS networks are distinguished from other types of multiple passive star topologies in [11] and [12] as follows. First, all couplers have equal fanout and are symmetric in the degree of fanin and fanout. Second, the nodes are completely connected with couplers arranged in parallel and without hierarchical interconnections. In POPS network, n nodes are partitioned into g groups of d processors each. We denote such a POPS network as a $POPS(d, g)$. n is the number of nodes and d is the partition size. It is also the degree of each coupler. g refers to the number of groups in the system and it determines the number of transmitter/receiver channels per node as well as the total number of couplers in the system. There is OPS couler between every pair of groups and hence, overall, g^2 couplers are needed. Each processor is connected to the inputs of g couplers and thus every processor must have g optical transmitters for transmitting data to any coupler. Also, each processor is connected to the outputs of g couplers and thus every processor must have g optical receivers for receiving data from each of the g coupler. Each OPS receives optical signal from any one of its source nodes through d input channels and broadcasts the received signal to all of its

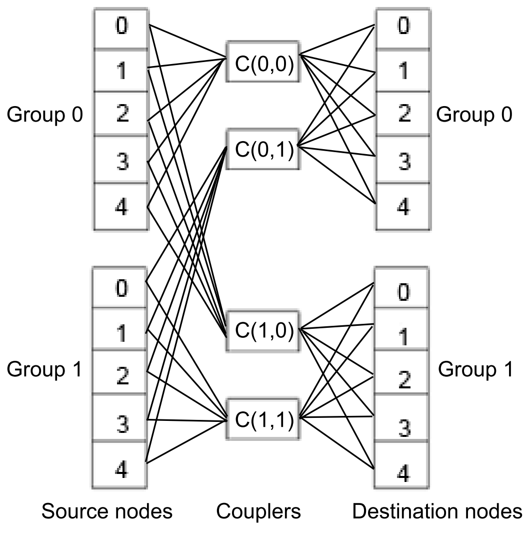

Fig. 2. A 10-processor computer connected via *POPS(5,2)* network

destination nodes through d output channels. Hence, each OPS in a $POPS(d,g)$ network has degree d.

In a single slot, a processor can send the same message to all of the OPSs for which it is a source node. If more than one coupler attempts to send the message to same coupler at the same time, sending conflict will occur at the coupler. In a single slot, a processor can receive a message from only one of the OPSs for which it is a destination node. Melhem et al. [6] observe that faster all-to-all broadcasts can be implemented by allowing a processor to receive different messages from different OPSs in the same slot. Figure 1 shows that each OPS coupler can receive an optical signal from any one of its source nodes and broadcasts the received signal to all of its destination nodes. The time needed to perform this receive and broadcast is referred to as a slot. The complexity of an algorithm designed for the POPS network is measured in terms of the number of slots taken.

The choice of partition size affects both the interconnection cost as well as bandwidth. Single OPS can be used to interconnect n processors and such a network is denoted by $POPS(n,1)$. In such kind of network, only one processor can send a message through the single coupler per slot since at most one input link to a single coupler should be active at a given slot to prevent collisions. As the coupler degree d approaches n and the system consists of a single passive star, this kind of network has low system cost, low complexity, restricted throughput, low bandwidth and increased power dissipation. Whereas, coupler degree d approaches 1, the system consists of n^2 couplers and such network is denoted by $POPS(1,n)$. In such kind of network, the system is fully connected and network bandwidth also becomes n^2. This kind of POPS network is highly expensive with optimal throughput and high bandwidth.

A 10-processor computer connected via a $POPS(5,2)$ network is shown in figure 2. Typically, a source node and the corresponding destination node are the same processing element. There are source nodes in the left of couplers and receiving nodes in the right indicating that processors in each group are both receivers and senders. A pair of groups is connected by an OPS coupler. A major advantage of the POPS network is that its diameter is one. To show how to route a message from one node to another node, we use the following notations. There are n processors and they are numbered from 0 to $n-1$. $P(k)$ such that $0 \leq k < n$ is used to describe the processor. There are g groups and $n/g = d$ processors are in each group. The groups are numbered from 0 to $g-1$ and G_i is used to indicate i_{th} group of processors for $0 \leq i < g$. $P(i,j)$ such that $0 \leq i < g$ and $0 \leq j < d$ is used to denote j_{th} node in i_{th} group. We can calculate the actual node number $P(k)$by means of $id+j$. OPS couplers are numbered as a pair of group numbers denoted by $C(i,j)$. The source nodes are the processors in group j and the destination nodes are the processors in group i such that $0 \leq (i,j) < g$. For example, in figure 2, if a node from group 0 sends a message to any node of group 1, the message will be sent through the coupler $C(1,0)$.

The route of any message has a unique path composed of a transmitter, a coupler, and a receiver, on the other hand, this path is defined by a triple parameters (source, coupler and receiver). Thus, a path exists between every pair of nodes and each path traverses exactly one coupler. If more than one message is sent using the same coupler during the same slot, the conflict occurs. To send a message from processor $P(i,j)$ to processor $P(x,y)$, processor $P(i,j)$ uses the coupler $C(x,i)$, x is the group number of destination node and i is the group number of source node. At first, $P(i,j)$ sends the message to $C(x,i)$ and then $C(x,i)$ broadcasts that message to all processors in G_x and $P(x,y)$ receives it. Similarly, a one-to-all broadcast can also be implemented in one slot [4] and [6]. Also, [4] and [6] give an algorithm for all-to-all personalized communication.

III. Data Sum

A. Parallel computation of data summation on one problem space

In this section, we present data summation algorithm for single problem space. We need to ensure that the following conditions are true for each of the routings:

(1) There is no simultaneous attempt to send the messages from more than one processor to same coupler in the same slot.

(2) Each processor has g receivers for receiving packets from g couplers which it is connected to. A processor needs to know exactly which receiver it should listen to since a processor can listen to only one of its receivers in a single slot.

(3) For simplicity, we assume that g is even and if it is odd, the algorithm may use $g+1$ to become even ignoring

operations of processors that do not exist.

There can be three cases for $POPS$ network: (1) $d = \sqrt{n} = g$, (2) $d < \sqrt{n} < g$ and (3) $d > \sqrt{n} > g$. At first, we consider the algorithm for the case (1) and it is designed to efficiently utilize the resources. We do not consider the case (2) because it is highly expensive and not realistic for most practical systems. At last, we simulate case (3).

At first, we consider the case $d = \sqrt{n} = g$. Using matrix transpose function, we show how to reduce the original problem size of $POPS(g, g)$ until we get $POPS(1, 1)$ with one node left. Initially, n data items are distributed among n processors, one item per processor on $POPS(g, g)$. We reduce the original problem size n to $n/2$ and again to $n/4$ and repeating this process until $POPS(1, 1)$ with one node left. Finally, processor $P(0, 0)$ holds the final data summation. Our algorithm is based on the following strategy. The processors are placed in a two-dimensional array on $POPS(g, g)$ network. Each row represents the processors in the same group and first row is in group 0 and the last row is in group $g - 1$. The processor in the top left corner is $P(0, 0)$. As shown in figure 3.a, horizontal line and vertical line are logically drawn to become four $P(g/2, g/2)$ networks. Let us call that logically divided $POPS(g, g)$ as $POPS(d, d)$ network. We apply matrix transpose function to map the nodes of each side to those of another side in order to send the data values. Once a receiver of destinatation node receives a data item, it adds the received data with its accumulated sum.

In order to do first stage routing as shown in figure 3.b, the data values on the nodes of lower-left side are routed to those of upper-left side to make partial summation. To indentify the destination nodes to be mapped, swap the row and column indices of source nodes. Then (i, j) becomes (j, i). Since the group numbers of destination nodes are less than those of the source nodes, i indices are reduced by $d/2$ since we horizontally divided $POPS(g, g)$ network by 2. Formally $P(i, j)$ maps to $P(j, i - d/2)$ for all i and j such that $g/2 \le i < g$ and $0 \le j < g/2$. The coupler indices through which the routing should be carried are $c(destination\ group\ number, source\ group\ number)$. This routing is accomplished by the following routing.

$$p(i, j) \rightarrow c(j, i) \rightarrow p(j, i - d/2) \tag{1}$$

Similarly, at the same time, the data values on the nodes of upper-right side are routed to those of lower-right side and it is opposite to line 1. After swapping the row and column indices of source nodes, as the group numbers of destination nodes are greater than those of the source nodes, i indices are added by $d/2$. $P(i, j)$ maps to $P(j, i + d/2)$ for all i and j such that $0 \le i < g/2$ and $g/2 \le j < g$. This routing is carried out as belows.

$$p(i, j) \rightarrow c(j, i) \rightarrow p(j, i + d/2) \tag{2}$$

In a single slot, above line 1 and line 2 routings are done in parallel without any coupler conflict. We will verify in order to ensure that no two processors in both routing use the same coupler. Line 1 routing is done for all i such that $g/2 \le i < g$ whereas line 2 routing is done for all i such that $0 \le i < g/2$. Similarly, line 1 routing is done for all j such that $0 \le j < g/2$ while line 2 routing is done for j such that $g/2 \le j < g$. It is clear that i and j indices of line 1 are different from those of line 2. On the other hand, the destination group number of line 1 routing is less than that of its source group while the destination group number of line 2 routing is greater than that of its source group. Hence, the couplers $c(destination\ group\ number, source\ group\ number)$ used by both routing are different in first index. Again, the source nodes of both routing are from different groups and there is no chance to get the same number for second index of $c(destination\ group\ number, source\ group\ number)$. Therefore, line 1 and line 2 routings can be carried out in parallel without any coupler conflict and it takes one slot.

After the first stage routing, n nodes becomes $n/2$ reducing to half of its original size. As shown in figure 3.c, there are $n/2$ nodes left and those nodes hold partial summation results. In order to carry out the second stage routing in figure 3.c, the nodes on the lower side are mapped to those on the upper side. This routing is a little similar with line 1 but not identical. Swap the row and column indices of source nodes. Since the group numbers of destination nodes are less than those of the source nodes as in line 1, i indices are reduced by $d/2$. Moreover, as they are in different sides i.e. left and right, their column indices are also different. This routing is carried as below and it also takes one slot.

$$p(i, j) \rightarrow c(j - d/2, i) \rightarrow p(j - d/2, i - d/2) \tag{3}$$

After the second stage routing, $n/2$ nodes becomes $n/4$ and the resulting $POPS$ network becomes as shown in figure 3.d. The original $POPS(g, g)$ network becomes $POPS(g/2, g/2)$ network and thus, we need to update $d = d/2$ before starting third stage routing. In this case, the nodes, which hold the partial results, are in one square and this condition is similar to the original stage. Therefore, logically divide new $POPS(d, d)$ network to get four squares as stated in figure 3.a. Following routings are done in parallel without any coupler conflict in a single slot and its correctness proof is similar with line 1 and line 2 routings.

$$p(i, j) \rightarrow c(j - d/2, i) \rightarrow p(j - d/2, i) \tag{4}$$

$$p(i, j) \rightarrow c(j + d/2, i) \rightarrow p(j + d/2, i) \tag{5}$$

We can also make above routings in different way. As the source and destination nodes are in same column, we can skip swapping of two indices and alternative routings can be done as follows:

$$p(i, j) \rightarrow c(i - d/2, i) \rightarrow p(i - d/2, j) \tag{6}$$

$$p(i, j) \rightarrow c(i + d/2, i) \rightarrow p(i + d/2, j) \tag{7}$$

After third stage routing, there are only $n/8$ nodes left as shown in figure 3.e. This fourth stage is similar with figure 3.c

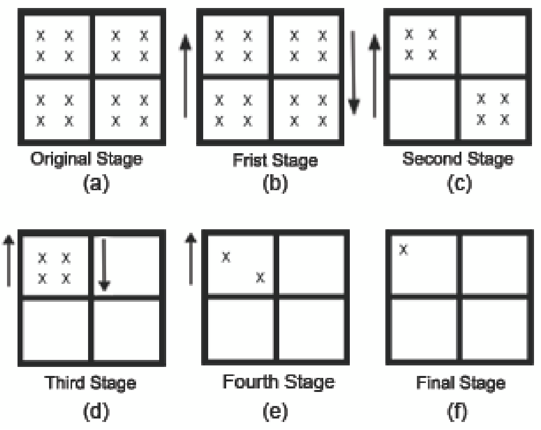

Fig. 3. data summation of *n* data items on *n* processors of *POPS(g,g)*

and we can apply line 3 routing as follows and it also takes one slot.

$$p(i,j) \rightarrow c(j-d/2, i) \rightarrow p(j-d/2, i-d/2) \qquad (8)$$

After fourth stage routing, $n/8$ nodes becomes $n/16$ and there is only one node left as shown in figure 3.f. Now the final data summation is held in the processor $P(0,0)$.

After every two slots, as the problem is back to original state, we can repeat the routing until one node left. Therefore, the algorithm is much simpler. n becomes $n/2$, $n/2^2$ and $n/2^3$ etc. after each slot. Generally, there are $n/2^k$ left after k_{th} slots. On the other hand, the algorithm takes one slot to reduce n nodes to half of its size and total time slots taken until one node left for n data items is log n. The data summation algorithms designed by Sahni [7] have different complexities for different group size. Among them, the complexity for $d = \sqrt{n} = g$ is $\log n$. The idea behind that algorithm is to fold the nodes residing on two-dimensional array along the anti diagonal line. But there is coupler conflict in first folding stage. Although the time complexity is the same, our algorithm is much simpler and free from coupler conflict.

B. Parallel computation of data summation on multiple problem spaces

We showed how data summation is done for n data items distributed on n nodes of $POPS(g,g)$ network. The algorithm uses at most half of couplers in each stage of parallel routing and at least, other half of the couplers are free. Moreover, in figure 3, we can easily see that some of nodes and couplers become free after each stage. To efficiently utilize the resources, we design the algorithm, which can calculate data items for multiple problem spaces in parallel. Improving parallelism makes the computation faster and it is also necessary to ensure that

(1) Each processor in a $POPS$ network consists of the processing element with a distributed memory unit.

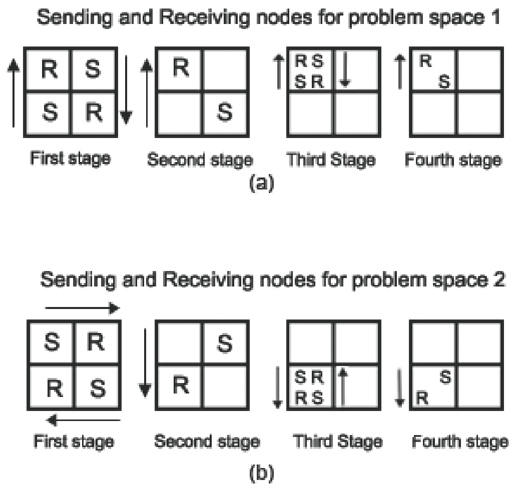

Fig. 4. parallel data summation of *2n* data items on *n* processors of *POPS(g,g)*

In some $POPS$ network, in order to transmit two different messages simultaneously to two different couplers, the transmitters of the node should be tunable to different wavelengths simultaneously and independently. Similarly, if the receivers of the node are active at the same time, it can receive the messages in parallel. However, in order to ensure that the conditions of the new proposed algorithm are same with those of the algorithm for the system mentioned in section 3.A, we need to make sure the following conditions are true:

(2) There are g transmitters in each node and these transmitters cannot tune to different wavelengths simultaneously. It defines that in a single slot, a processor can send the same message to all of the OPSs for which it is a source node.

(3) There are g receivers in each node and only one of the receiver can be active at a time. It means that, in a single slot, a processor can receive a message from only one of the OPSs for which it is a destination node.

In order to do multiple parallel computations, we need to develop the algorithm not only free from coupler conflict, but also satisfying the underlying hardware mentioned above.

The transmitter and receiver of a certain node are independent i.e. the processor can send the data using its transmitter while receiving the data from its receiver. The algorithm for multiple problem space is as simple as that of single problem space. The idea behind is to take advantage of the independent nature of transmitter and receiver. As shown in figure 4, there are two problems of n data items each and hence altogether $2n$ data items. These $2n$ data items are distributed over n nodes with one data item each from two problems per node. A node holds one data item from first problem space as shown in figure 4.a and that node holds the another data item from second problem space as shown in figure 4.b. While the node acts as the receiver for the first problem, that node acts as sender for the second problem as shown in the first stages of figure 4.a and 4.b.

Figure 4.a is the same as figure 3 and thus, the algorithm for problem space 1 is also same as that described in section 3.A. The algorithm of problem space 2 is not very much different from the first one. We need to compute problem space 1 and 2 simultaneously carrying out the data routing in parallel. For the first stage routing of figure 4.a, the following routings are done in parallel as described in section 3.A.

$$p(i,j) \to c(j,i) \to p(j,i-d/2) \quad (9)$$

$$p(i,j) \to c(j,i) \to p(j,i+d/2) \quad (10)$$

At the same time, for the first stage of figure 4.b, data values on the nodes of upper left side are routed to those of upper right side. Similarly, those of lower right side are routed to those of lower left side. These routings are carried out as follows.

$$p(i,j) \to c(j,i) \to p(j,i+d/2) \quad (11)$$

$$p(i,j) \to c(j,i) \to p(j,i-d/2) \quad (12)$$

Above four routings are done in parallel and it only takes single slot. In section 3.A, it is already shown that line 9 and 10 routing are free from coupler conflict and can be done in parallel in a single slot. For the case of line 11 and 12, it is also clear that routings are opposite to each other. Therefore, we will prove that there is no sending and receiving conflict between two problem spaces. In the first stages of figure 4.a and 4.b, it is shown that the receiving nodes for problem space 1 become sending nodes for problem space 2 and vice versa. Since the transmitter and receiver of a certain node are independent from each other, it is clear that there is no transmitting and listening conflict. Then, we will show that there is no coupler conflict between two problem spaces as below.

For problem space 1, in the first stage of figure 4.a, the nodes on lower left side send data to those on upper left side and generally, the routing is denoted by

$$p(i_1,j_1) \to c(i_2,i_1) \to p(i_2,j_2) \quad (13)$$

The receiving nodes on upper left side of problem space 1 becomes sending nodes in problem space 2 as shown in first stage of figure 4.b. In addition, these nodes send data to those on upper right side and generally, the routing is denoted by

$$p(i_2,j_2) \to c(i_3,i_2) \to p(i_3,j_3) \quad (14)$$

The coupler numbers used by both routings are different. Therefore, it is clear that there is no coupler conflict among the routings of two problems. The reason why the first index of first coupler is the same as second index of the second coupler is that the nodes on upper left side acts as senders and receivers respectively in both problems. The other stages of problem space 2 are similar with the algorithm described in section 3.A. It is necessary to update d after reducing $POPS(g,g)$ to $POPS(g/2,g/2)$ and again to $POPS(g/4,g/4)$ etc. On the other hand, after every two slots, the problem is back to original state and routing can be repeated using updated d. At last, the final data sum of first problem space is held in the first processor on first row i.e. $P(0,0)$ and that of second problem space is held in the first processor on the last row i.e $P(g-1,0)$. Although there are two problems, the algorithm takes the time slots needed for one problem because of the parallelisms. For the case of $d=\sqrt{n}=g$, the algorithm designed by Sahni [7] takes $\log n$ slots for n data items on $POPS(g,g)$ network. If the number of data items is the same as number of nodes available, it takes $\log n$. For multiple m problems with at most n data items each, it takes at least $m \log n$. Whereas our algorithm takes only $\lceil m/2 \rceil \log n$ and it is at most two times faster than previous best algorithm. Moreover, our algorithm is much simpler since it uses only matrix transpose function for all routing. To avoid sending, receiving and coupler conflict, we just change the direction of routing from up to down and left to right etc. Figure 5 shows the comparison of time slots taken between previous best algorithm and our algorithm.

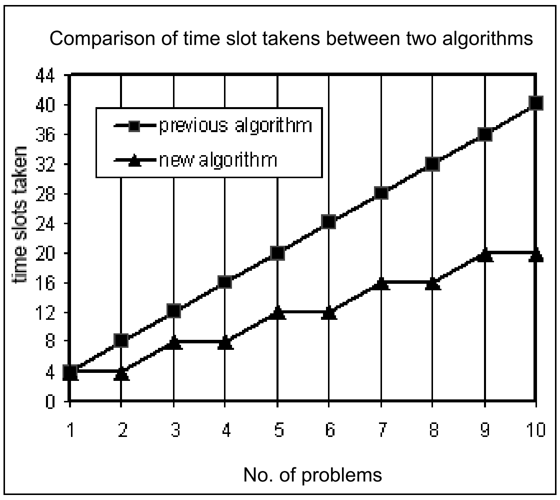

Fig. 5. Comparison between previous best algorithm and our proposed algorithm

For the case of $d < \sqrt{n} < g$, as $g > \sqrt{n}$, we need $g^2 > n$ couplers to be interconnected with the nodes. Gravenstreter and Melhem [4] mention that, the number of couplers will be less than the number of processors for most practical systems. It is not likely that a large number of couplers will be used in a practical system. Since the number of groups is larger than the partition size and such kind of system is highly expensive, we do not consider the case for $d < \sqrt{n} < g$.

In order to achieve certain speedup to compute the large data items, a large system with many nodes is needed, for instance, $POPS(d,g)$ with $d > \sqrt{n} > g$. If the system is

$POPS(32,2)$, 64 nodes and 4 couplers are needed for such kind of system. Since $g < \sqrt{n}$, only $g^2 < n$ couplers are needed and therefore, the system is low cost and practical. However, we can also achieve a similar speedup on smaller system with smaller number nodes. For instance, the system size we need is $POPS(2,2)$ with only 4 nodes and 4 couplers. As only the smaller number of nodes is needed and the number of couplers is the same as that of former system, the proposed system is also low cost. We simulate the system to handle the large number of data items dedicated for large system by breaking up the problem sizes to the system size. If we extend the algorithm mentioned in section 3, we do not need very new algorithm for this simulation.

Chiarulli et al. [3] mention that computation achieves certain speedup on a large system with many nodes. Then it can also achieve a similar speedup on smaller system. We simulate the $POPS(g,g)$ with smaller nodes for the case of $d > \sqrt{n} > g$. If the algorithm computes multiple m problems of at most n data items each on $n = g^2$ nodes, it takes $\lceil m/2 \rceil \log n$. If the number of data items n is larger than the number of nodes available in the system such that $n > g^2$, the system takes $2g^2$ data items for each time. Hence, after $n/2g^2$ times, $P(0,0)$ and $P(g-1,0)$ hold the partial summation. Thus, extra one slot is needed to sum up the data value on these nodes. By this way, for the case of $POPS(d,g)$, with $d > \sqrt{n} > g$, we simulate $POPS(g,g)$ with smaller nodes in order to compute the large data items. The complexity for the case of $POPS(d,g)$ with $d > \sqrt{n} > g$ is $\lceil n/g^2 \rceil \log g + 1$. The correctness proof for this algorithm is similar to that for the case when $d = g$ mentioned in section 3.B. It is also faster than the algorithm designed by Sahni [7]. Datta and Soundaralakshmi [10] presented fast algorithms for data sum on $POPS(d,g)$ such that $d > \sqrt{n} > g$. Their data sum algorithm improves upon the best known algorithm designed by Sahni [7] reducing the number of data items rapidly until there are only g data items in each group and then the rest of calculation is done using Sahni's algorithm [7]. Their algorithm is faster than that designed by Sahni [7] and the time complexity is $d/g + 2\log g - 1$. Although it is difficult to compare their algorithm with our proposed algorithm, except that both of them takes same slots for $g = 2$, sometimes one of the algorithm is faster than each other respectively depending on different g values. However, as the system simulated by our algorithm needs smaller number of nodes, it is low cost and more practical than the system with many nodes.

IV. Conclusion

We have presented the algorithms for data summation for the $POPS$ network. Although the time complexity of algorithm for single problem space is the same as previous best known algorithm, we have shown that there is no coupler conflict compared to [7]. The algorithm for multiple problem spaces is designed for the reasons: (1) to utilize free resources (2) to compute multiple problems using only half of the time slots that actually needed and (3) to simulate the calculation of large number of data items on smaller system to achieve the similar speed of large system. It takes $\lceil m/2 \rceil \log n$ for multiple problems and $\lceil n/g^2 \rceil \log g + 1$ for simulation. Compared to the best known algorithm designed by Sahni [7], the algorithm for multiple problems is at most two times faster and the simulation is considerably faster. Moreover, all algorithms are much simpler and dedicated to low cost and practical systems. It also confirms that, POPS network data channels, couplers, and nodes can be efficiently utilized to be cost effective.

References

[1] P. Berthome and A. Ferreira, "Improved Embeddings in POPS Networks through Stack-Graph Models", *Proc. Third Int'l Workshop Massively Parallel Processing Using Optical Interconnections*, pp. 130-135, 1996.

[2] P. Berthome, J. Cohen, and A. Ferreira, "Embedding Tori in Partitioned Optical Passive Stars Networks," *Proc. Fourth Int'l Colloquium on Structural Information and Comm. Complexity*, pp. 40-52, 1997.

[3] D. Chiarulli, S. Levitan, R. Melhem, J. Teza and G. Gravenstreter, "Partitioned Optical Passive Star (POPS) Multiprocessor Interconnection Networks with Distributed Control," *Proc. First Int'l Workshop on Massively Parallel Processing Using Optical Interconnections*, pp. 70-80, 1994.

[4] G. Gravenstreter and R. Melhem, "Realizing Common Communication Patterns in Partitioned Optical Passive Stars (POPS) Networks", *IEEE Trans. Computers*, vol. 47, no. 9, pp. 998-1013, September 1998.

[5] G. Gravenstreter, R. Melhem, D. Chiarulli, S. Levitan, and J. Teza, "The Partitioned Optical Passive Stars (POPS) Topology", *Proc. Ninth Int'l Parallel Processing Symp.*, pp. 4-10, 1995.

[6] R. Melhem, G. Gravenstrater, D. Chiarulli and S. Levitan, "The Communication Capabilities of Partitioned Optical Passive Stars Networks", *Parallel Computing Using Optical Interconnections, K. Li, Y. Pan, and S. Zheng, eds. Kluwer Academic*, pp. 77-98, 1998.

[7] S. Sahni, "The Partitioned Optical Passive Stars Network: Simulations and Fundamental Operations", *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 7, pp. 739-748, July 2000.

[8] S. Sahni, "Matrix Multiplication and Data Routing Using a Partitioned Optical Passive Stars Network", *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 7, pp. 720-728, July 2000.

[9] S. Sahni, "Models and Algorithms for Optical and Optoelectronic Parallel Computers", *Int'l J. Foundations of Computer Science*, vol. 12, no. 3, pp. 249-264, 2001.

[10] A. Datta and S. Soundaralakshmi, "Summation and Routing on a Partitioned Optical Passive Stars Network with Large Group Size", *IEEE Trans. ON Parallel and Distributed Systems*, vol. 14, no. 12, December 2003.

[11] Y. Birk, "Power-optimal layout of passive, single-hop, fiber-optic interconnection whose capacity increases with the number of stations", *Proc. IEEE INFOCOM*, 1993.

[12] A. Ganz, B. Li, and L. Zenou, "Reconfigurability of multi-star based lightwave LANs", *Proc. IEEE GLOBE-COM*, 1992.