CHED National Center of Excellence in Information Technology Education
INSTITUTE OF COMPUTER SCIENCE
College of Arts and Sciences
University of the Philippines Los Baños
College 4031, Laguna, PHILIPPINES
Phone (63-49) 536-2313 Fax (63-49) 536-2302 Campus VoIP 6123
E-mail jppabico@uplb.edu.ph Web www.ics.uplb.edu.ph/jppabico

CMSC 180
Introduction to Parallel Computing
Second Semester AY 2023-2024

Final Examination

I. DIRECTIONS

- Read and understand Section II (Preliminaries) of this examination. The section has three subsections: Basic Communication, Prefix Sum, and Prefix Sum on a Hypercube.
- Answer all the questions (or problems) in Section C (Examination). The section is the examination proper and has three subsections: Basic Communication, Prefix Sum on a Hypercube, and 32-element Bitonic Sorter, with a bonus question.
- Write your name in the bottom of every page of this examination (6 pages). In the last section, Section IV (certification), you need to sign in the appropriate blank to signify your agreement to the certification stated.
- You are free to use any software to annotate the PDF format of this exam questionaire so that your answer can be clearly read. Write legibly if you choose to use a pen instead.

II. PRELIMINARIES

A. Basic Communication: We have learned from our lecture the seven basic communication protocols that may be used to distribute data to p processors connected via some interconnect networks. The basic prerequisite of these protocols is the one out-port, one in-port constraint wherein a processor p_i may only open two ports at a time. Strictly, one port must be used for sending data, and another port must be used for receiving data. These open ports may be connected to one processor p_j or to two different processors p_k and p_l. When p_i is connected to p_j via a two-port connection, then the two are swapping data: p_i is both sending to and receiving from p_j and vice versa. When p_i is connected to p_k via its in-port and concurrently connected to p_l via its out-port, then p_i is receiving data from p_k while it is sending data to p_l (Figure II-A-1).

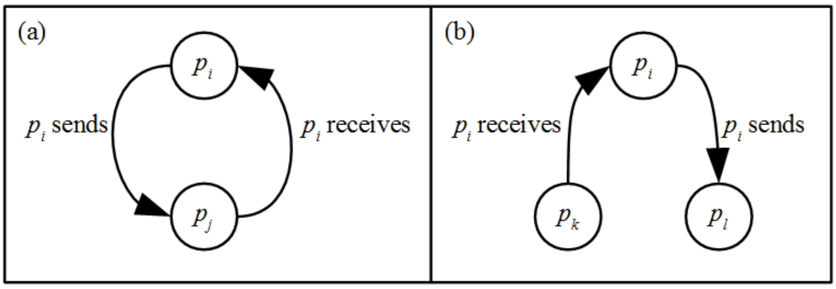


Figure II-A-1. A processor p_i is concurrently sending and receiving data (a) to other processor p_j, and (b) respectively to other processors p_k and p_l.

In our lecture discussion, we have talked about the seven basic data communication protocols that obey the one in-port, one out-port constraints over p processors $p_0, p_1, \ldots, p_{p-1}$ interconnected in a completely-connected network. One of these protocols is the:

One-to-many Broadcast (`1MB`) where, without loss of generality, a processor p_0 sends a data D to all other processors $p_1, p_2, \ldots, p_{p-1}$. The broadcast can be seen as a function that accepts a $1 \times p$ vector of the form $[D, 0, 0, \ldots, 0]$ and returns a $1 \times p$ vector of the form $[D, D, D, \ldots, D]$. Thus, mathematically, the protocol can be seen as:

$$[D, D, D, \ldots, D] = \texttt{1MB}([D, 0, 0, \ldots, 0]) \quad (1)$$

`1MB()` takes $\lg(p)$ steps to complete as illustrated in Figure A-2 for $p = 8$, where $\lg(x)$ is the base two logarithm of a number x.

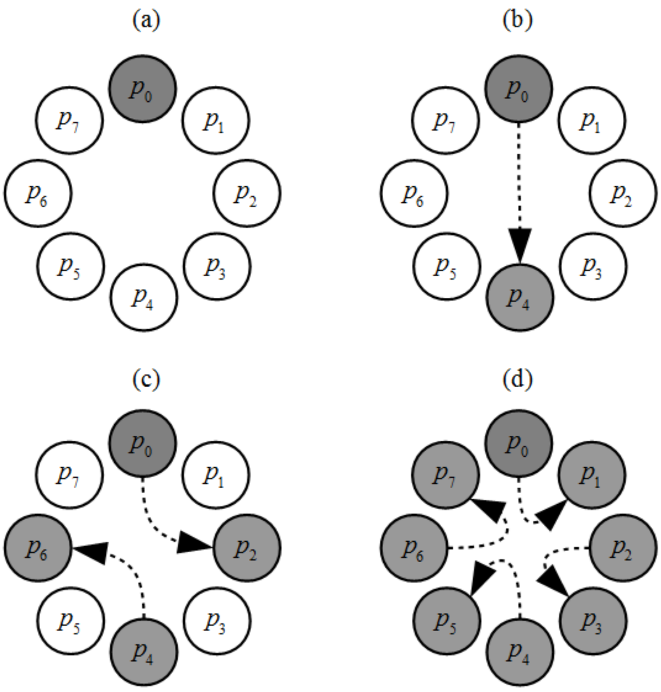


Figure II-A-2. A completely-connected interconnection network with eight processors implementing a `1MB()` with p_0 as the origin of data D. The network links are not shown to unclutter the figure but instead, dashed arrows show the direction of data transfer: (a) The state before `1MB()` where a gray processor means that the processor has the data D; (b) The state after the first communication step; (c) The state after the second communication step; and (d) The state after the third and last communication step.

B. Prefix Sum: Computing the sum of n numbers over n processing elements connected via a fully-connected interconnect network

Initially, each processing element is assigned one of the numbers to be added and, at the end of the computation, one of the processing elements stores the sum of all the numbers. Assuming that n is a power of two, we can perform this operation in log n steps by propagating partial sums up a logical binary tree of processing elements. Figure II-B-1 below illustrates the procedure for $n = 8$. The processing elements are labeled from p_0 to p_7. Similarly, the eight numbers to be added are labeled from s_0 to s_7. The sum of the numbers with consecutive labels from i to j is denoted by $\Sigma_{i.j}\, s_k$

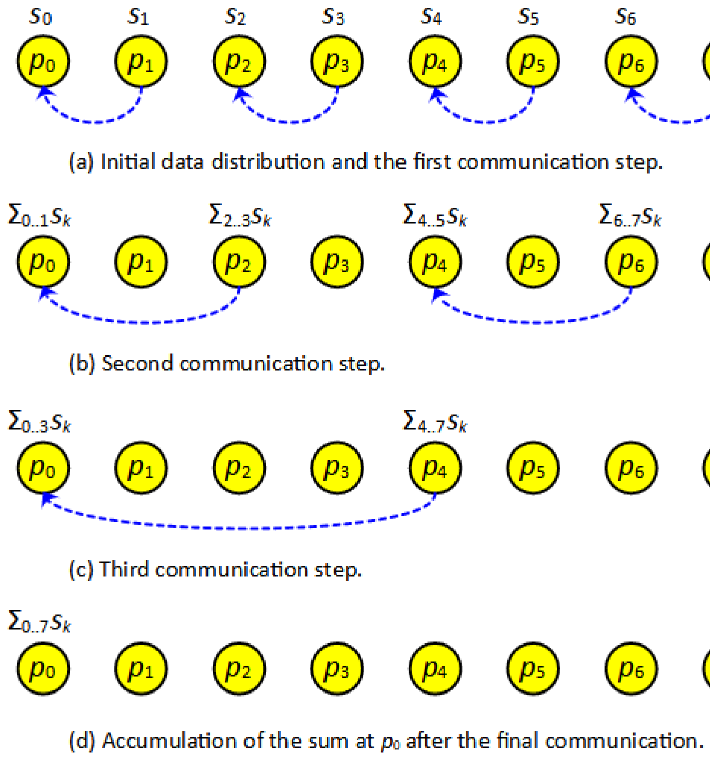


Figure II-B-1. Computing the global sum of eight partial sums using eight processing elements. The symbol $\Sigma_{i..j}\ s_k$ denotes the sum of numbers with consecutive labels from i to j.

C. Prefix Sum on a Hypercube: Computing the sum of n numbers over n processing elements connected via a hypercube. Figure II-C-1 illustrates the same process described in subsection II-B for n=8 but instead of over a fully-connected network, the process is conducted over a hypercube.

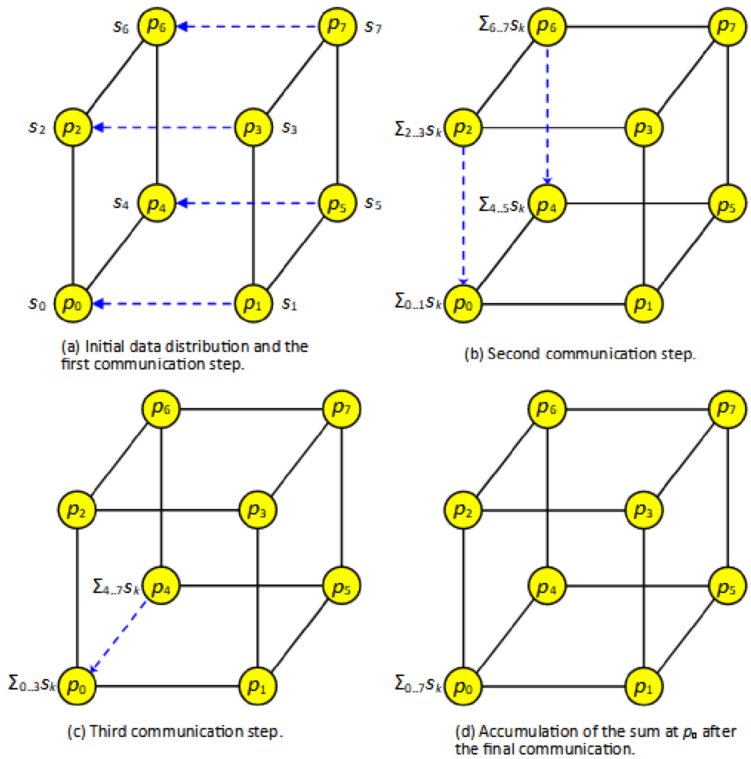


Figure II-C-1. The version of prefix sum implemented over a hypercube for n = 8.

III. EXAMINATION

A. Basic Communication (30%)

In some of our video discussions, I've asked that you analyze the overall number of steps required for <u>optimally completing</u> `1MB()`, Many-to-one Reduction (`M1R`), Many-to-many Broadcast (`MMB`), Many-to-many Reduction (`MMR`), One-to-many Personalized Broadcast (`1MPB`), Many-to-one Personalized Reduction (`M1PR`), and Many-to-many Personalized broadcast (`MMPB`) over a fully-connected network. When we say "*optimally completing*," we mean the minimum number of steps required to complete the process. We also mean the minimum memory space required to realized the minimum number of steps, where one unique data D means one memory space. Summarize the respective results of your analysis in Table III-A-1 utilizing the convention in this questionaire. Consider all source processors to be p_0, where p_0 is located as illustrated in our video lectures.

Table III-A-1. Exact number of steps, the memory space required for the source processor p_0, the memory space required per processor to <u>optimally finish</u> the basic communication protocols over the fully-connected network with p processors.

Measurement	Basic Communication Protocol						
	`1MB()`	`M1R()`	`MMB()`	`MMR()`	`1MPB()`	`M1PR()`	`MMPB()`
Exact Number of Steps	$\lceil \lg(p) \rceil$	log(p)	log(p)	log(p)	log(p)	log(p)	log(p)
Exact memory space required for the source processor	1	1	1	p	p	1	p
Exact memory space required per processor (other than the source)	1	1	p	1	1	p	p

B. Prefix Sum on a Hypercube (20%)

Consider the problem of computing the prefix sums of n numbers on n processing elements connected via a hypercube. What is the parallel runtime T_p, speedup S, and efficiency E of this algorithm? Assume that adding two numbers takes one unit of time and that communicating one number between two processing elements takes 10 units of time. Is the algorithm cost-optimal?

1. (5%) T_p = O(11log n) = O(log n)

2. (5%) S = Ts / Tp = O(n) / O(log n) = O(n/log n)

3. (5%) E = S / n = (n/log n) / n = 1/log n

4. (5%) Is this cost optimal? Why? No, because the efficiency decreases as n increases

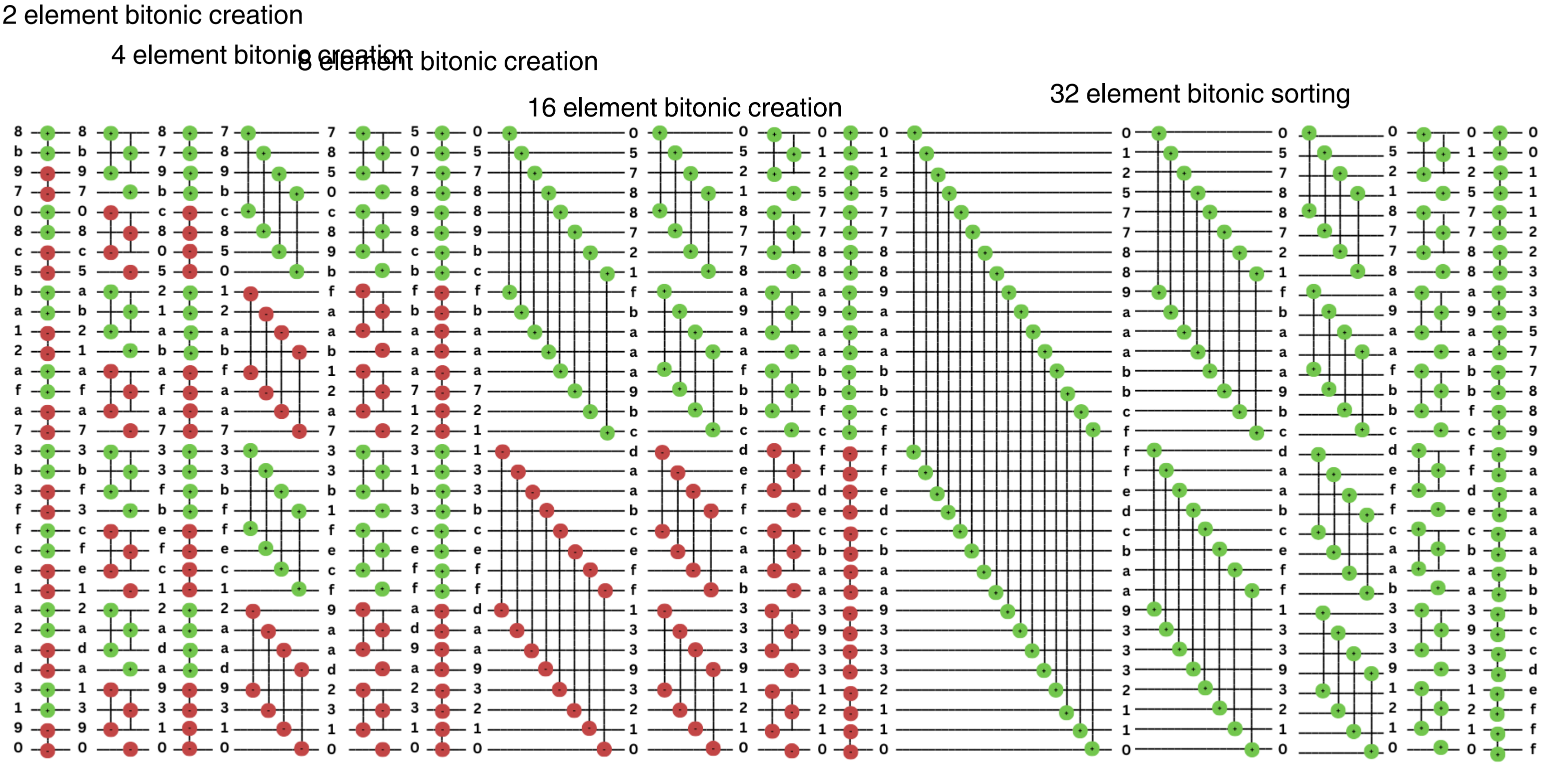
C. 32-element Bitonic Sorter (50%)

Write your student number here in this format YYYYNNNNN, where YYYY is a 4-digit year and NNNNN is a 5-digit number: 2021-00075

Obtain the MD5SUM of your entry above using the online tool for MD5SUM hash (https://codebeautify.org/md5-hash-generator). Make sure that the CRLF (carriage return and/or line feed) and other white characters is/are not part of your student number.
Write the hash here: 8b9708c5ba12afa73b3ffce1a2ad3190

The MD5SUM hash is a 32-character one-way hash of any text. One can use the case-sensitive characters of the MD5SUM as unsorted inputs to a 32-element bitonic sorting network.

5. (25%) Create a 32-element non-decreasing bitonic sorting network and draw it here. Clearly label each object in your drawing.



6. (25%) Using each character of the MD5SUM of your student number as individual inputs to the 32-element bitonic sorting network, with the *i*th character of the MD5SUM as the *i*th input of the network, what will be the sequence after the first column of the bitonic merge (hint: this is the column after the input has been converted into a bitonic sequence)? Write the sequence in the blank below:

012577889aaabbcfffedcbaa93332110

7. (Bonus 20%) Explain in your own words (preferably aided with drawings) how can you run the 32-element bitonic sorting network over a hypercube. You may use extra sheet when necessary.

I can run the run 32 element bitonic sorting over a hypercube by first creating a bitonic sequence by2 elements, followed by 4 elements, by 8 elements, and by 16 elements. This will create a bitonic sequence that can be sorted by a bitonic network. Then 32 bitonic sorting will be executed to sort the bitonic sequence.

IV. CERTIFICATION

I certify in my honor that the academic work I am submitting here is fully on my own and that I fully understand the Academic Rules governing examinations. My signature below proves the truthfulness of this certification. Not signing this certification means that I am not taking this examination even if I have visible answers in Section III of this examination.

JERICO SABILE	2021-00075	May 21, 2024
Signature over printed name	Student Number	Date

END OF FINAL EXAMINATION