STATE UNIVERSITY OF NEW YORK AT

STONY BROOK

CEAS TECHNICAL REPORT 616

Processor Equivalence for a Linear Daisy Chain of Load Sharing Processors

T.G. Robertazzi

January 22, 1992

Processor Equivalence for a Linear Daisy Chain of Load Sharing Processors

Thomas G. Robertazzi

Dept. of Electrical Engineering,
SUNY at Stony Brook,
Stony Brook N.Y. 11794

ABSTRACT

A linear daisy chain of processors where processor load is shared among the processors is examined. It is shown that two or more processors can be collapsed into a single equivalent processor. This equivalence allows a characterization of the nature of the minimal time solution, a simple method to determine when to distribute load for linear daisy chain networks of processors without front end communication sub-processors and closed form expressions for the equivalent processing speed of infinitely large daisy chains of processors.

I. Introduction

Parallel computation has been of great interest in recent years. A parallel machine consists of a number of processors and an interconnection network to tie them together. This paper examines a specific parallel processing problem on a specific architecture that allows the study of the integration of communication and computation. While these two issues are often studied separately, a combined study is rare.

The situation to be considered involves a linear daisy chain of processors, as is illustrated in Figure 1. A single "problem" is solved on the network at one time. It takes time $w_i T_{cp}$ to solve the entire problem on processor i. Here w_i is inversely proportional to the speed of the ith processor and T_{cp} is the normalized solution time when w_i=1. It takes time $z_i T_{cm}$ to transmit the entire problem problem representation (data) over the ith link. Here z_i is inversely proportional to the channel speed of the ith link and T_{cm} is the normalized transmission time when z_i=1.

It is assumed that the problem representation can be divided amongst the processors. That is, fraction α_i of the total problem is assigned to the ith processor so that it's computing time becomes $\alpha_i w_i T_{cp}$. It is desired to determine the optimal values of the α_i's so that the problem is solved in the minimum amount of time. The situation is non-trivial as there are communication delays incurred in transmitting fractional parts of the problem representation to each processor from the originating processor.

This framework is different from the usual multiprocessor scheduling problem [5,6,7] where a number of jobs of known fixed solution time are assigned to processors with each job assigned to only one processor. In this paper one has a single job that can be arbitrarily partitioned among a number of processors. The framework being described is particularly germane to processing involving large data files (so that communication delay is non-negligible), such as sensor data processing, signal processing, image processing and Kalman filtering, where the data can be divided among multiple processors.

Two cases will be considered: processors that have front end communications sub-processors for communications off-loading so that communication and computation may proceed simultaneously and processors without front end communications sub-processors so that communication and computation must be performed at separate times.

A timing diagram for a linear daisy chain network of four processors with front end communications sub-processor (as in Figure 1) is illustrated in Figure 2. There is one graph for each processor. The horizontal axis is time. The upper half of each graph indicates communication time and the lower half indicates computation time. It is assumed that the problem (load) originates at the left, first, processor.

At time 0, processor 1 can start working on its fraction, α_1, of the problem in time $\alpha_1 w_1 T_{cp}$. It also simultaneously communicates the remaining fraction of the problem to processor 2 in time $(\alpha_2+\alpha_3+\alpha_4)z_1 T_{cm}$. Processor 2 can then begin computation on its fraction of the problem (in time $\alpha_2 w_2 T_{cp}$) and communicates the remaining load to processor 3 in time $(\alpha_3+\alpha_4)z_2 T_{cm}$. The process continues until all processors are working on the problem.

A similar, but not identical, situation for a linear daisy chain network with processors that do not have front-end communication sub-processors is illustrated in Figure 3. Here each processor must communicate the remaining load to its right neighbor before it can begin computation on its own fraction.

In [1] recursive expressions for calculating the optimal α_i's were presented. These are based on the simplifying premise that for an optimal allocation of load, all processors must stop processing at the same time. Intuitively this is because otherwise some processors would be idle while others were still busy. Studies of such load sharing for other interconnection topologies appear in [2,3,4].

In this paper the concept of collapsing two or more processors and associated links into a single processor with equivalent processing speed is presented. This allows a complete proof (an abridged one appears in [1]) that for the optimal, minimal time solution all processors must stop at the same time. Moreover, for the case without front end communications sub-processors it allows a simple algorithm, described in section III, to determine when it is economical to distribute load amongst multiple processors. Finally, in section IV, the notion of equivalent processors will enable the derivation of simple closed form expressions for the equivalent speed of a linear daisy chain network containing an infinite number of processors. This provides a limiting value for the performance of this network architecture.

II. Equivalent Processors

Consider a linear daisy chain network of N processors as in Figure 1. Two adjacent processors may be combined into a single "equivalent" processor that presents operating characteristics to the rest of the network that are identical to those of the original two processors. Two cases, processors with and without front end communication sub-processors, will be considered.

In both cases it is assumed that the load originates at the left most processor (processor 1). If the load originates at an interior processor one can use the same methodology to collapse the processors to the left and the right of the originating processor into equivalent processors and then collapse the remaining three processors into a single equivalent processor.

Front End Communications Sub-Processors

We will start with the N-1st and Nth processors, as illustrated in Figure 4. The figure begins at the moment when load has finished being transmitted to the N-1st processor from the N-2nd processor. As in [1], the N-1st processor keeps $\hat{\alpha}_{N-1}$ fraction of what it receives and transmits the remaining $1-\hat{\alpha}_{N-1}$ fraction to the Nth processor. The total load received by the N-1st processor from the N-2nd is $(\alpha_{N-1}+\alpha_N)$. The time each is active, from the figure, is:

$$T_{N-1} = \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp} \tag{2.1}$$

$$T_N = \tag{2.2}$$
$$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_N T_{cp}$$

To prove that the minimal time solution occurs when both processors stop at the same time, the possibilities $T_{N-1} \geq T_N$ and $T_{N-1} \leq T_N$ must be examined. If $T_{N-1} \geq T_N$, simple algebra results in

$$\hat{\alpha}_{N-1} \geq \frac{z_{N-1}T_{cm} + w_N T_{cp}}{w_{N-1}T_{cp} + z_{N-1}T_{cm} + w_N T_{cp}} \tag{2.3}$$

with equality occurring when both processors stop at the same time. Minimizing the solution time, $T_{sol} = T_{N-1}$, clearly requires

$$\min T_{sol} = (\min \hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp} \tag{2.4}$$

so that the optimal value of $\hat{\alpha}_{N-1}$ occurs for equality in (2.3). The quantity $(\alpha_{N-1}+\alpha_N)$ is not involved in the minimization since the value of $\hat{\alpha}_{N-1}$ is unaffected by the total load, $(\alpha_{N-1}+\alpha_N)$, delivered to the N-1st processor. Put another way, the optimization involves the fraction of load being allocated between P_{N-1} and P_N, not the total load allocated to these two processors. The other half of the proof, for $T_{N-1} \leq T_N$, is similar.

The two processors with front end (fe) communications sub-processors can be replaced by a single processor with equivalent speed constant:

$$w_{eq}^{fe} = \frac{\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N) w_{N-1} T_{cp}}{(\alpha_{N-1}+\alpha_N) T_{cp}} = \hat{\alpha}_{N-1} w_{N-1} \tag{2.5}$$

Here $\hat{\alpha}_{N-1}$ is given by (2.3) with equality. The solution time is divided by the normalized computation time to yield the equivalent speed constant. Thus, starting with the N-1st and Nth processors, the entire linear chain of processors can be collapsed, two at a time, into a single equivalent processor. Thus one can recursively show that for a network of N processors the optimal solution occurs when all processors stop at the same time.

No Front End Communications Sub-Processor

Again, consider the N-1st and Nth processors in a linear chain. Figure 5 starts from the moment when load has finished being transmitted from the N-2nd to the N-1st processor. Again, as in [1], the N-1st processor keeps $\hat{\alpha}_{N-1}$ fraction of what it receives and transmits the remaining $1-\hat{\alpha}_{N-1}$ fraction to the Nth processor. From Figure 5, the time each is active is:

$$T_{N-1} = \tag{2.6}$$
$$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N) z_{N-1} T_{cm} + \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N) w_{N-1} T_{cp}$$

$$T_N = \tag{2.7}$$
$$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N) z_{N-1} T_{cm} + (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N) w_N T_{cp}$$

Once again, to prove that the minimal time solution requires both processors to stop at the same time, the cases $T_{N-1} \geq T_N$ and $T_{N-1} \leq T_N$ can be considered. For $T_{N-1} \geq T_N$, simple algebra results in

$$\hat{\alpha}_{N-1} \geq \frac{w_N}{w_{N-1} + w_N} \tag{2.8}$$

with equality occurring when both processors stop at the same time. From (2.6) the solution time can be rewritten as:

$$T_{sol} = T_{N-1} = \tag{2.9}$$
$$(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)(w_{N-1}T_{cp} - z_{N-1}T_{cm})$$

The sign of the term $(w_{N-1}T_{cp}-z_{N-1}T_{cm})$ now becomes important. If it is positive, minimizing T_{sol} is equivalent to minimizing $\hat{\alpha}_{N-1}$ and the optimal solution occurs at equality for (2.8). In other words, if $w_{N-1}T_{cp} > z_{N-1}T_{cm}$, communication is fast enough relative to computation that the distribution of load is economical. Again, $(\alpha_{N-1}+\alpha_N)$ is not involved in the minimization.

On the other hand, if $(w_{N-1}T_{cp}-z_{N-1}T_{cm})$ is negative, then minimizing T_{sol} is equivalent to maximizing $\hat{\alpha}_{N-1}$ at $\hat{\alpha}_{N-1}=1$. That is, communication speeds are slow relative to computation speed so that it is more economical for processor N-1 to process the entire load itself rather than to distribute part of it to processor N.

The case where $T_{N-1} \leq T_N$ proceeds along similar lines. Again, the ability to collapse processors into equivalent processors allows one to extend the proof that two processors must stop at the same time for a minimal time solution to N processors.

III. When to Distribute Load

A practical problem for the case without front end communications sub-processor is to compute the equivalent computation speed of a linear daisy chain network when, in fact, the optimal solution may not make use of all processors, because of too slow communication speeds. Again, if the load originates at the left most processor, this can be done by collapsing the processors, two at a time, from right to left in Figure 1, into a single equivalent processor. However, when looking at two adjacent processors, say the i-1st and the ith (where the ith is an equivalent processor for processors i, i+1, ...), one must determine whether or not it is economical to distribute load. That is, one seeks the faster of either the solution with both processors, T_{both}, or with just the single i-1st processor, T_{single}:

$$T_{both} = \tag{3.1}$$
$$(1-\hat{\alpha}_{i-1})(\alpha_{i-1}+\alpha_i)z_{i-1}T_{cm} + \hat{\alpha}_{i-1}(\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp}$$

$$T_{single} = (\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp} \tag{3.2}$$

Here fraction $\hat{\alpha}_{i-1}$ of the total load, $(\alpha_{i-1}+\alpha_i)$, is assigned to processor i-1 and fraction $1-\hat{\alpha}_i$ is assigned to processor i. If $T_{single} < T_{both}$ then the ith processor is removed from consideration and the equivalent processing speed constant, with no front end (nfe) communication sub-processor, is:

$$w_{eq}^{nfe} = \frac{(\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp}}{(\alpha_{i-1}+\alpha_i)T_{cp}} = w_{i-1} \tag{3.3}$$

If $T_{single} > T_{both}$ then load distribution is economical and the two processors are collapsed into a single equivalent processor with speed constant:

$$w_{eq}^{nfe} = \frac{(1-\hat{\alpha}_{i-1})z_{i-1}T_{cm}+\hat{\alpha}_{i-1}w_{i-1}T_{cp}}{T_{cp}} \tag{3.4}$$

From (2.8):

$$\hat{\alpha}_{i-1} = \frac{w_i}{w_{i-1}+w_i} \tag{3.5}$$

Note that in (3.4) factors of $(\alpha_{i-1}+\alpha_i)$ cancel in the numerator and denominator.

By keeping track of which of (3.1) and (3.2) is smaller, it is possible to determine which processors to remove from the final network.

Note that the above procedure can also be applied to the situation when the load originates at a processor which is located in the interior of the network. The parts of the network to the left and to the right of the originating processor can be collapsed, into equivalent processors, following the previous procedure. The remaining three processors (left, originating, right) can then be further collapsed into a single equivalent processor. Naturally, it must be checked whether the inclusion of the left and/or right equivalent processor leads to a faster solution.

IV. Infinite Number of Processors

A difficulty with the linear network daisy chained architecture is that as more and more processors are added to the network, the amount of improvement in the network's equivalent speed approaches a saturation limit. Intuitively, this is because of the overhead in communicating the problem representation down the linear daisy chain in what is essentially a store and forward mode of operation.

It is possible to develop simple expressions for the equivalent processing speed of an infinite number of homogeneous processors and links. These provide a limiting value on the performance of this architecture. The technique is similar to that used for infinitely sized electrical networks to determine equivalent impedance.

Let the load originate at a processor at the left boundary of the network (processor 1). The basic idea is to write an expression for the speed of the single equivalent processor for processors 1,2...N. This is a function of the speed of the single equivalent processor for processors 2,3...N. However these two speeds should be equal since both involve an infinite number of processors. One can simply solve for this speed.

Consider, first, the case where each processor has a front-end communication sub-processor. Let $w_i = w$ and $z_i = z$. Let the network consist of P_1 and an equivalent processor for processors 2,3...N. Then:

$$w_{eq}^{fe} = \hat{\alpha}_1 w \tag{4.1}$$

But from (2.3) with equality, and making the above assumption,

$$w_{eq}^{fe} = \frac{z\rho + w_{eq}^{fe}}{w + z\rho + w_{eq}^{fe}}\, w \tag{4.2}$$

where $\rho = T_{cm}/T_{cp}$. Solving for w_{eq}^{fe} results in:

$$\boxed{w_{eq}^{fe} = \frac{-z\rho + \sqrt{(z\rho)^2 + 4wz\rho}}{2} \qquad (4.3)}$$

The solution time for such an infinite network is simply given by $T_{sol} = w_{eq}^{fe} T_{cp}$.

In a similar manner, an expression for the equivalent processing speed of a linear daisy chain network with an infinite number of processors with no front end (nfe) communication sub-processors can be determined. Again, the load originates at processor 1 at the left boundary of the daisy chain.

$$\boxed{w_{eq}^{nfe} = \sqrt{wz\rho} \qquad (4.4)}$$

The solution time for this infinite network is simply given by $T_{sol} = w_{eq}^{nfe} T_{cp}$.

This last expression is quite intuitive. Doubling w and z doubles w_{eq}^{nfe}. Doubling either w or z alone increases w_{eq}^{nfe} by a factor of $\sqrt{2}$. These results agree very closely with numerical results presented in [1]. It is straight forward to show that $w_{eq}^{fe} < w_{eq}^{nfe}$. Thus, in this limiting case, solution time is always reduced through the use of front end processors.

It is also possible to use the above results to calculate the limiting performance of an infinite sized daisy chain when the load originates at a processor at the interior of the network (with the network having infinite extent to the left and the right). Expressions (4.3) or (4.4) can be used to construct equivalent processors for the parts of the network to the left and right of the originating processor. The resulting three processor system then can be simply solved [1].

V. Conclusion

The concept of collapsing two or more processors into an equivalent processor has been shown to be useful in examining a variety of aspects related to these linear daisy chain networks of load sharing processors. Expressions for the performance of infinite chains of processors are particularly useful as if one can construct a finite sized system that approaches the performance of a hypothetical infinite system, one can feel comfortable that performance can not be improved further.

Acknowledgements

The research in this paper was supported by the National Science Foundation under grant no. NCR-8703689 and by the SDIO/IST and managed by the U.S. Office of Naval Research under grant no. N00014-85-K0610.

References

[1] Cheng, Y.-C. and Robertazzi T.G. (1988) Distributed Computation with Communication Delays. *IEEE Transactions on Aerospace and Electronic Systems*, **AES-24** (Nov. 1988) 700-712.

[2] Cheng Y.-C. and Robertazzi, T.G. (1990), Distributed Computation for a Tree Network with Communication Delays. *IEEE Transactions on Aerospace and Electronic Systems*, **AES-26** (May 1990) 511-516.

[3] Bataineh, S. and Robertazzi, T.G. (1991) Distributed Computation for a Bus Network with Communication Delays. *Proceedings of the 1991 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore MD (March 1991) 709-714.

[4] Bataineh, S. and Robertazzi, T.G. (1991) Bus Oriented Load Sharing for a Network of Sensor Driven Processors. *IEEE Transactions on Systems Man & Cybernetics special issue on distributed sensor nets*, Vol. 21, No. 5 (Sept. 1991).

[5] Leung, J. Y.-T. and Young, G.H., (1989) Minimizing Schedule Length Subject to Minimum Flow Time. *SIAM Journal on Computing*, Vol. 18, No. 2 (April 1989) 314-326.

[6] Coffman Jr., E.G. and Gavey, M.R. and Johnson, D.S. (1978) An Application of Bin Packing to Multiprocessor Scheduling. *SIAM Journal on Computing*, Vol. 7, No. 1, (Feb. 1978).

[7] Coffman Jr., E.G. and Sethi, R. (1976) Algorithms Minimizing Mean Flow Time: Schedule Length Properties. *Acta Informatica*, Vol, 6, (1976) 1-14.

Figure Captions

Fig. 1: Linear Chain of Processors

Fig. 2: Timing Diagram; Network with Front End Communications Sub-Processors

Fig. 3: Timing Diagram; Network without Front End Communications Sub-Processors

Fig. 4: Timing Diagram for N-1st and Nth processors with Front End Communications Sub-Processors

Fig. 5: Timing Diagram for N-1st and Nth processors without Front End Communications Sub-Processors

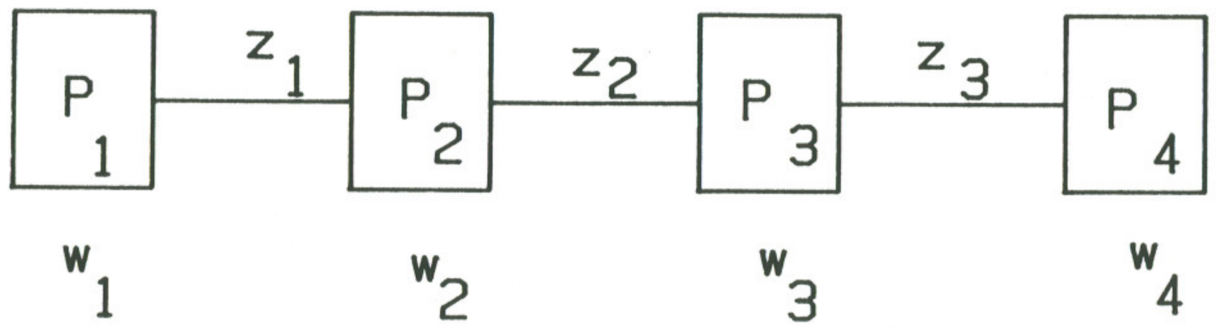
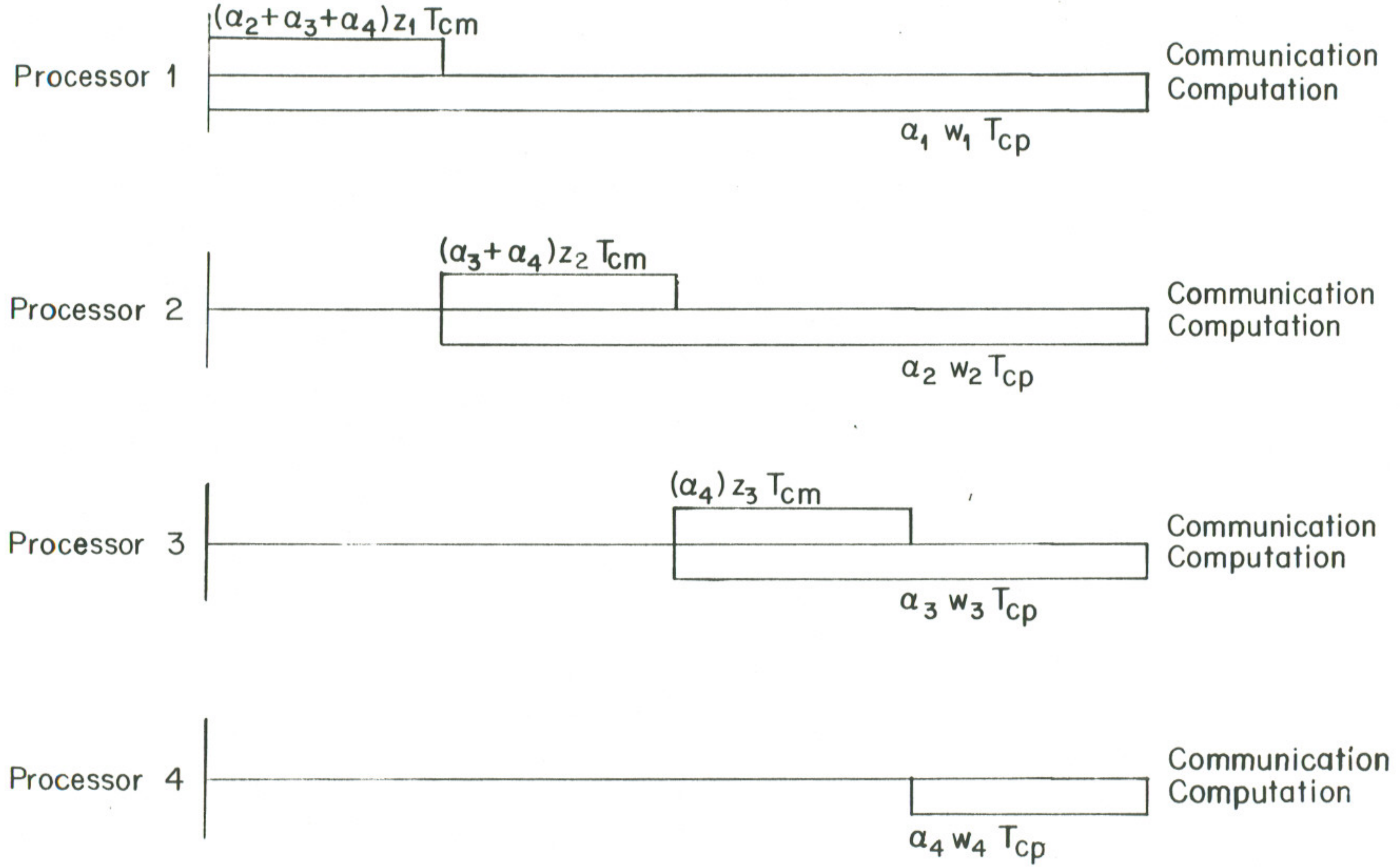
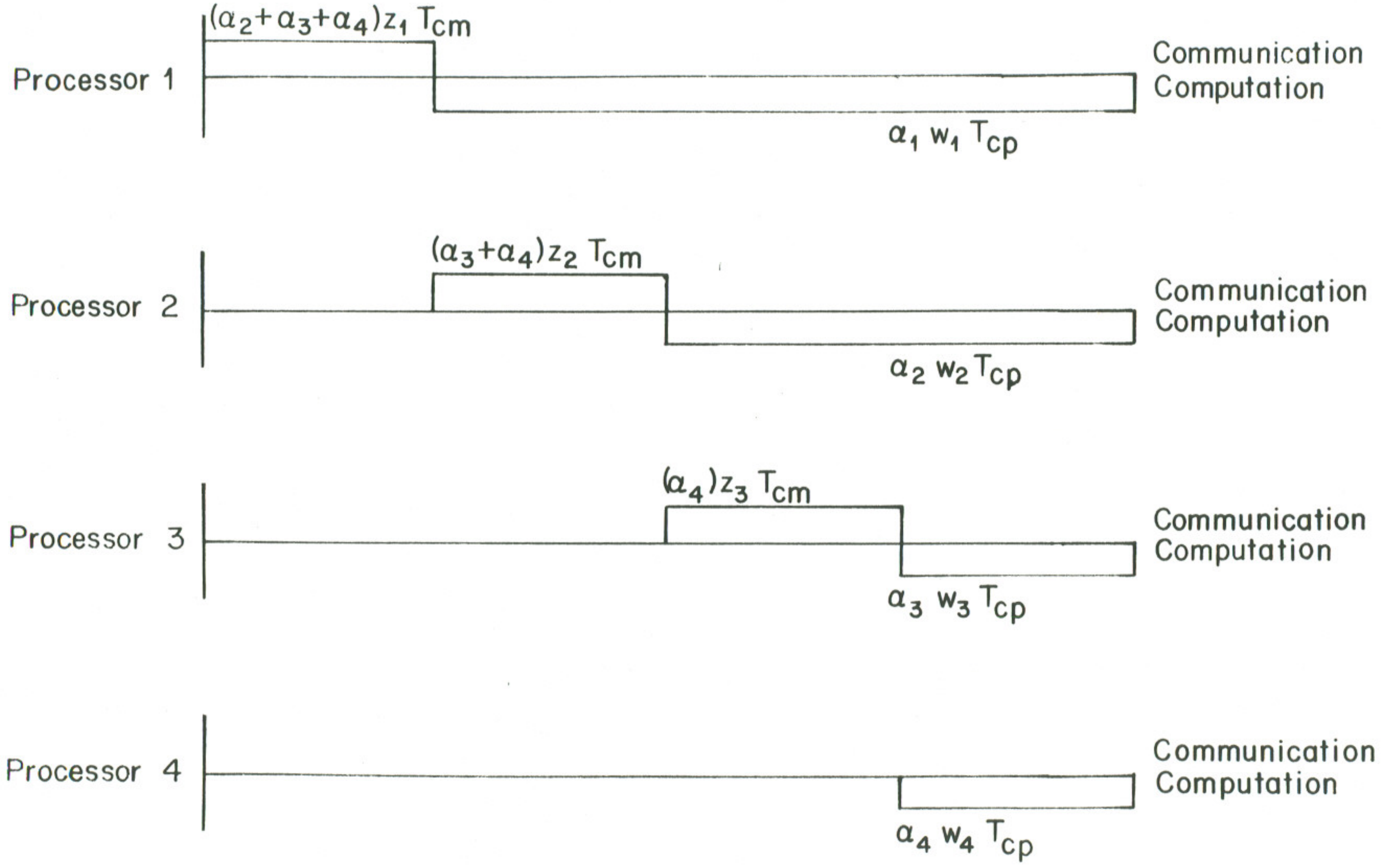


Figure 1



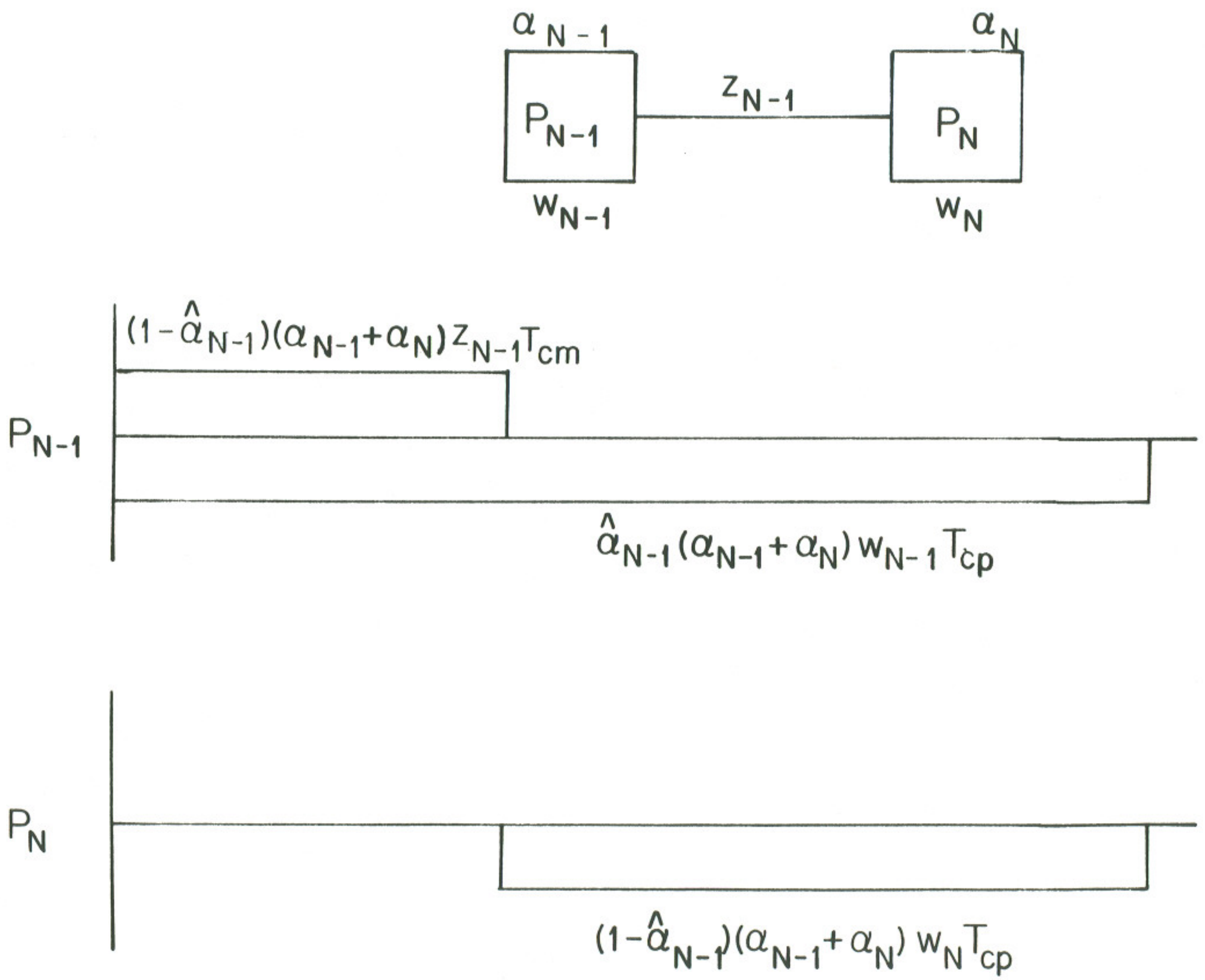
Processor 1
$(\alpha_2+\alpha_3+\alpha_4)z_1 T_{cm}$
Communication
Computation
$\alpha_1 w_1 T_{cp}$
Processor 2
$(\alpha_3+\alpha_4)z_2 T_{cm}$
Communication
Computation
$\alpha_2 w_2 T_{cp}$
Processor 3
$(\alpha_4)z_3 T_{cm}$
Communication
Computation
$\alpha_3 w_3 T_{cp}$
Processor 4
Communication
Computation
$\alpha_4 w_4 T_{cp}$

Figure 2



$(\alpha_2+\alpha_3+\alpha_4)z_1 T_{cm}$
Processor 1
Communication
Computation
$\alpha_1 w_1 T_{cp}$
$(\alpha_3+\alpha_4)z_2 T_{cm}$
Processor 2
Communication
Computation
$\alpha_2 w_2 T_{cp}$
$(\alpha_4)z_3 T_{cm}$
Processor 3
Communication
Computation
$\alpha_3 w_3 T_{cp}$
Processor 4
Communication
Computation
$\alpha_4 w_4 T_{cp}$

Figure 3



α_{N-1}
α_N
P_{N-1}
z_{N-1}
P_N
w_{N-1}
w_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm}$
P_{N-1}
$\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp}$
P_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_NT_{cp}$

Figure 4

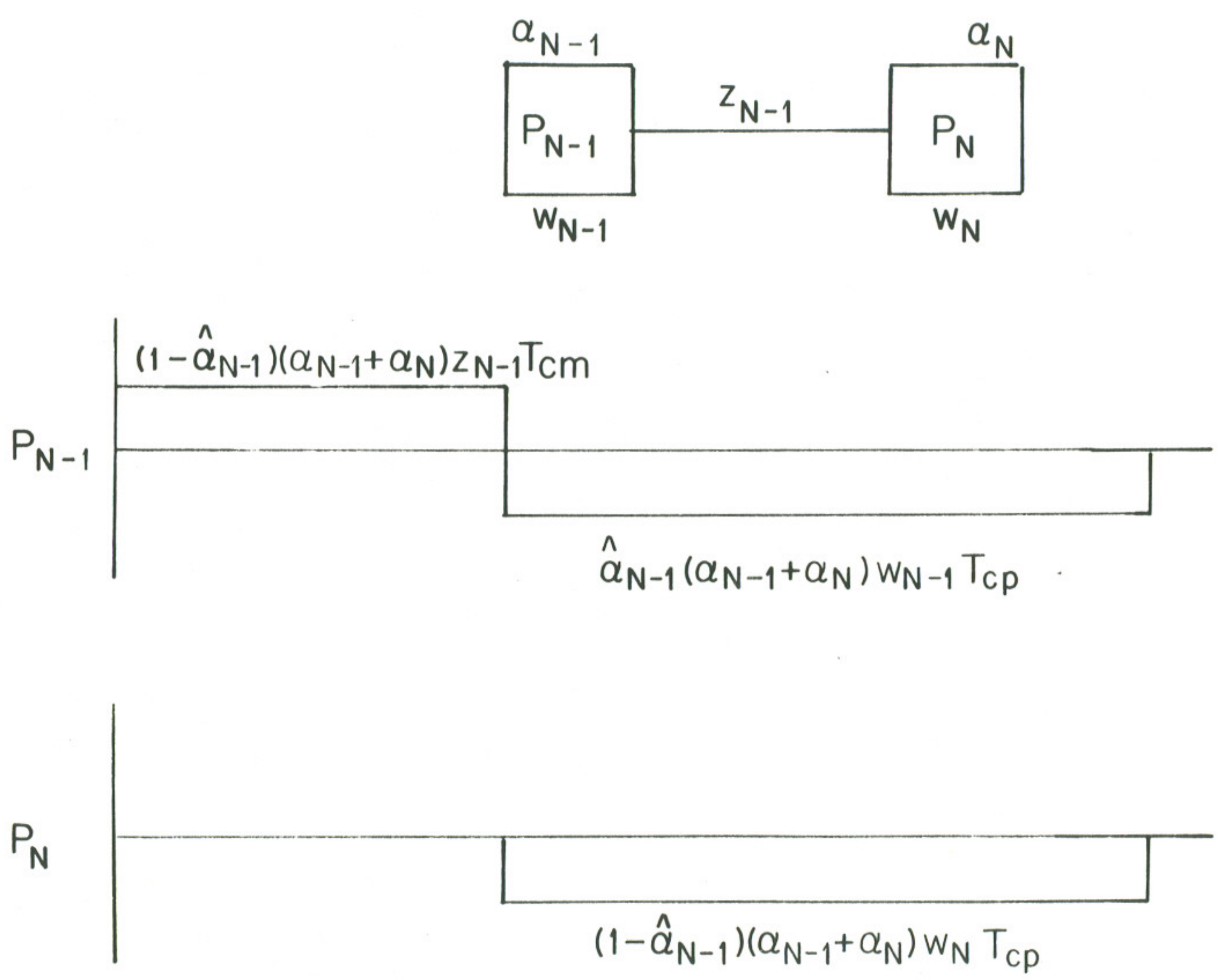
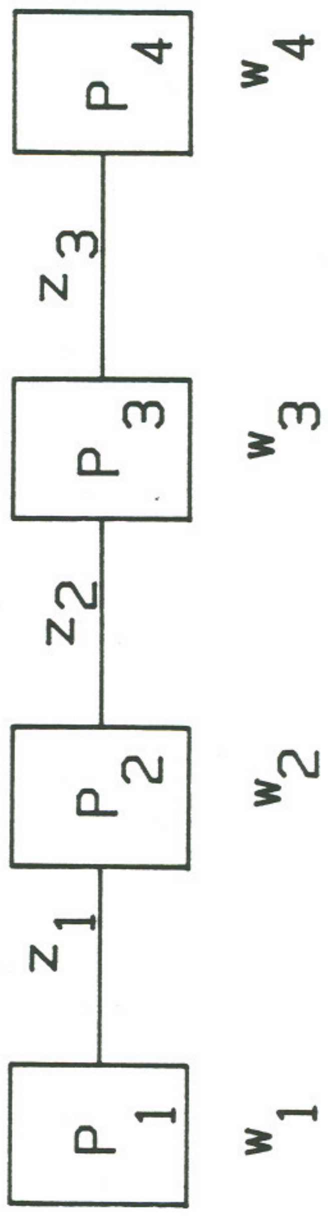
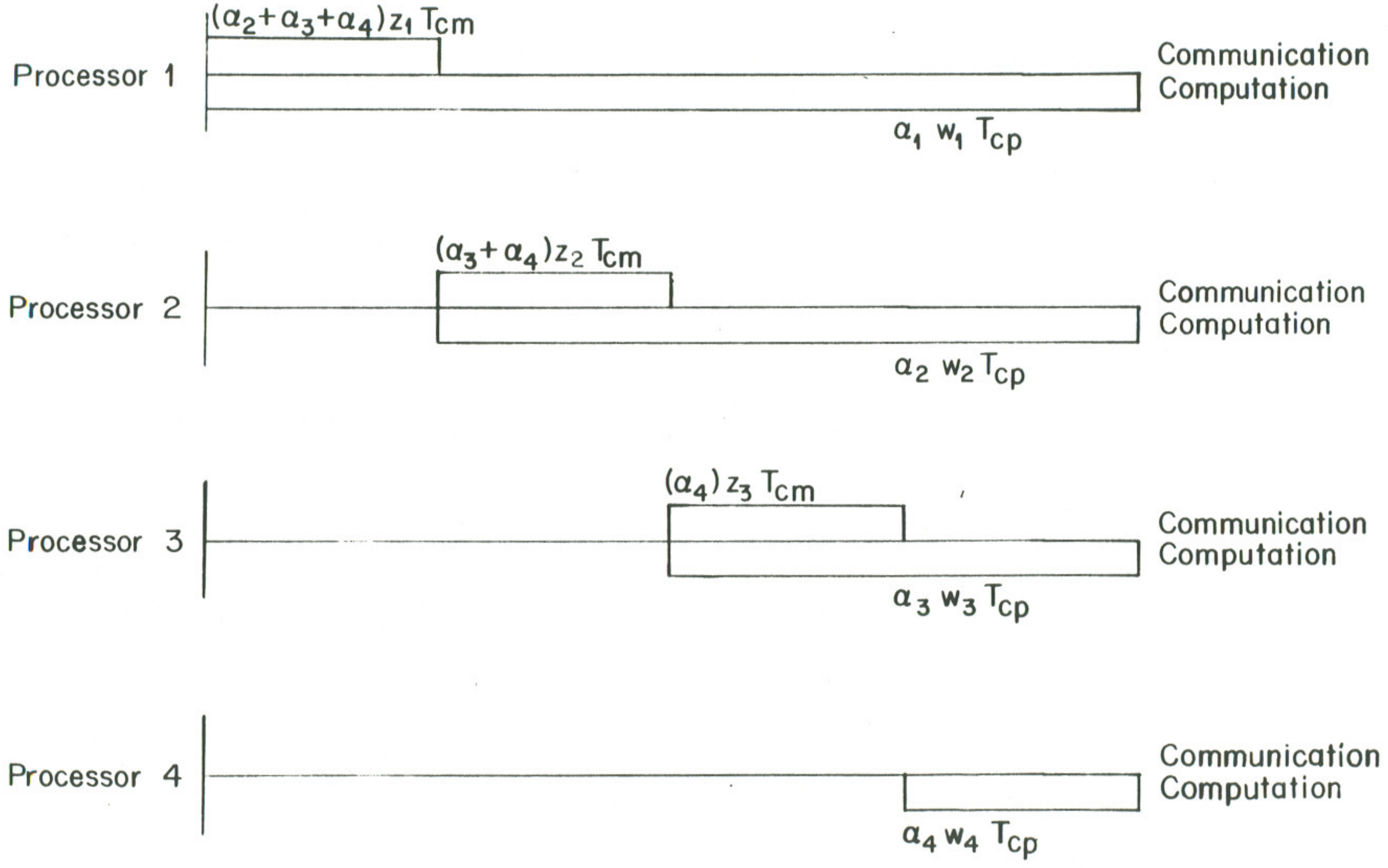

α_{N-1}
α_N
P_{N-1}
z_{N-1}
P_N
w_{N-1}
w_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm}$
P_{N-1}
$\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp}$
P_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_N T_{cp}$

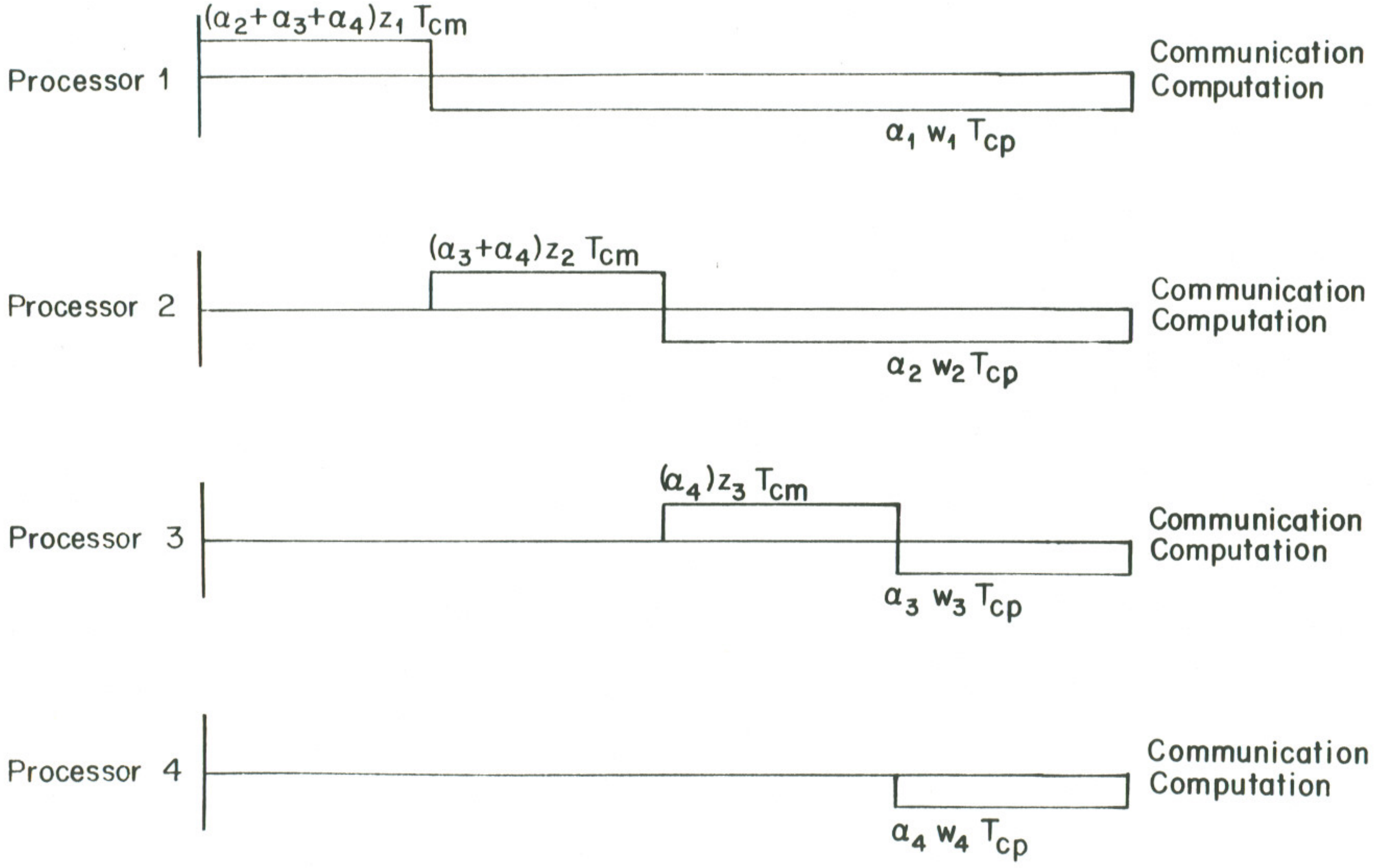
Figure 5

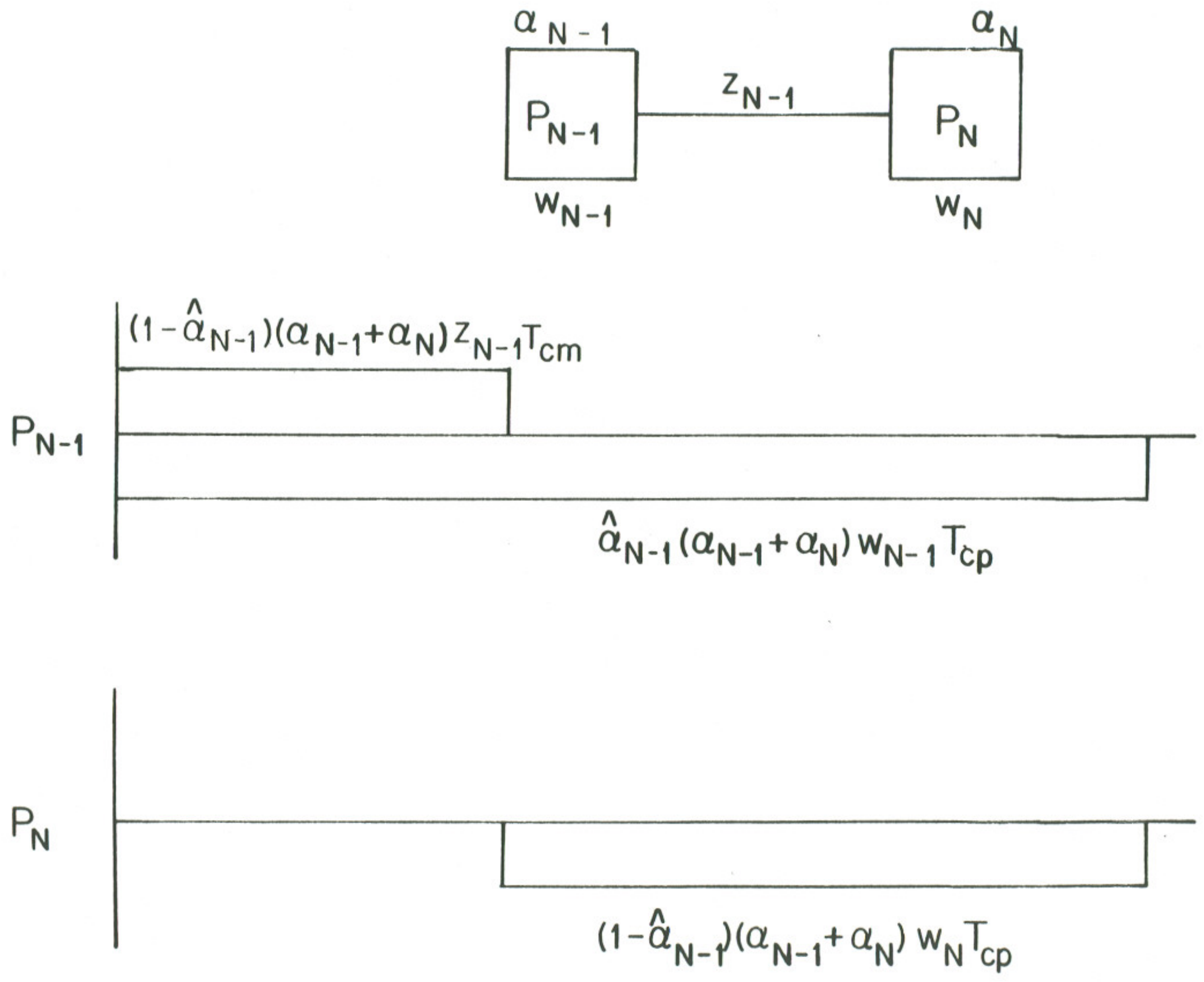
Key Words

Parallel Processing
Multiprocessor
Load Balancing
Scheduling
Daisy Chain
Equivalent Processor
Asymptotic
Front End Processor
Sensor Data

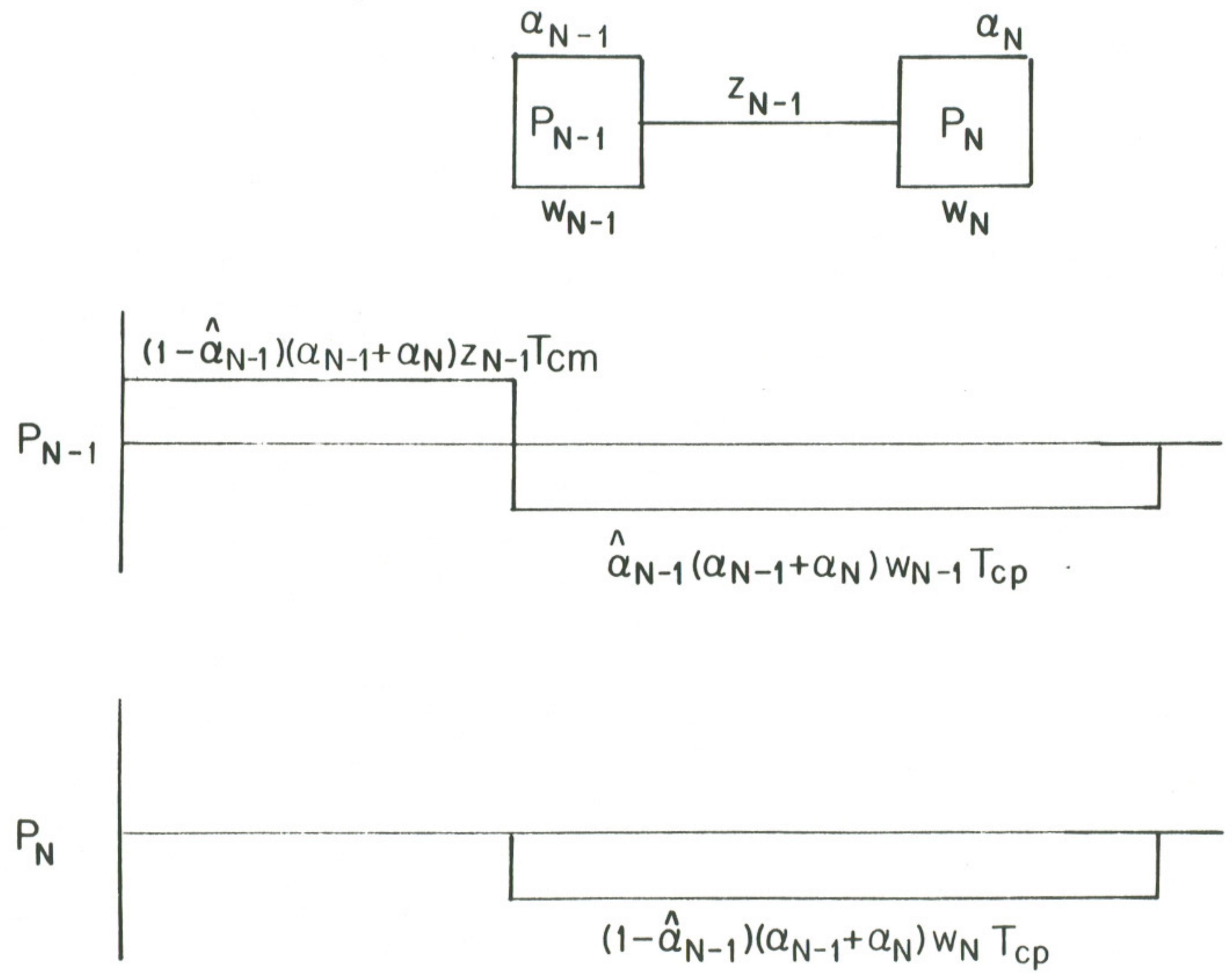

P_1
z_1
P_2
z_2
P_3
z_3
P_4
w_1
w_2
w_3
w_4


$(\alpha_2+\alpha_3+\alpha_4)z_1T_{cm}$
Processor 1
Communication
Computation
$\alpha_1 w_1 T_{cp}$
$(\alpha_3+\alpha_4)z_2T_{cm}$
Processor 2
Communication
Computation
$\alpha_2 w_2 T_{cp}$
$(\alpha_4)z_3T_{cm}$
Processor 3
Communication
Computation
$\alpha_3 w_3 T_{cp}$
Processor 4
Communication
Computation
$\alpha_4 w_4 T_{cp}$


$(\alpha_2+\alpha_3+\alpha_4)z_1 T_{cm}$
Processor 1
Communication
Computation
$\alpha_1 w_1 T_{cp}$
$(\alpha_3+\alpha_4)z_2 T_{cm}$
Processor 2
Communication
Computation
$\alpha_2 w_2 T_{cp}$
$(\alpha_4)z_3 T_{cm}$
Processor 3
Communication
Computation
$\alpha_3 w_3 T_{cp}$
Processor 4
Communication
Computation
$\alpha_4 w_4 T_{cp}$



α_{N-1}
α_N
P_{N-1}
z_{N-1}
P_N
w_{N-1}
w_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm}$
P_{N-1}
$\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp}$
P_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_NT_{cp}$


α_{N-1}
α_N
P_{N-1}
z_{N-1}
P_N
w_{N-1}
w_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm}$
P_{N-1}
$\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp}$
P_N
$(1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_N T_{cp}$