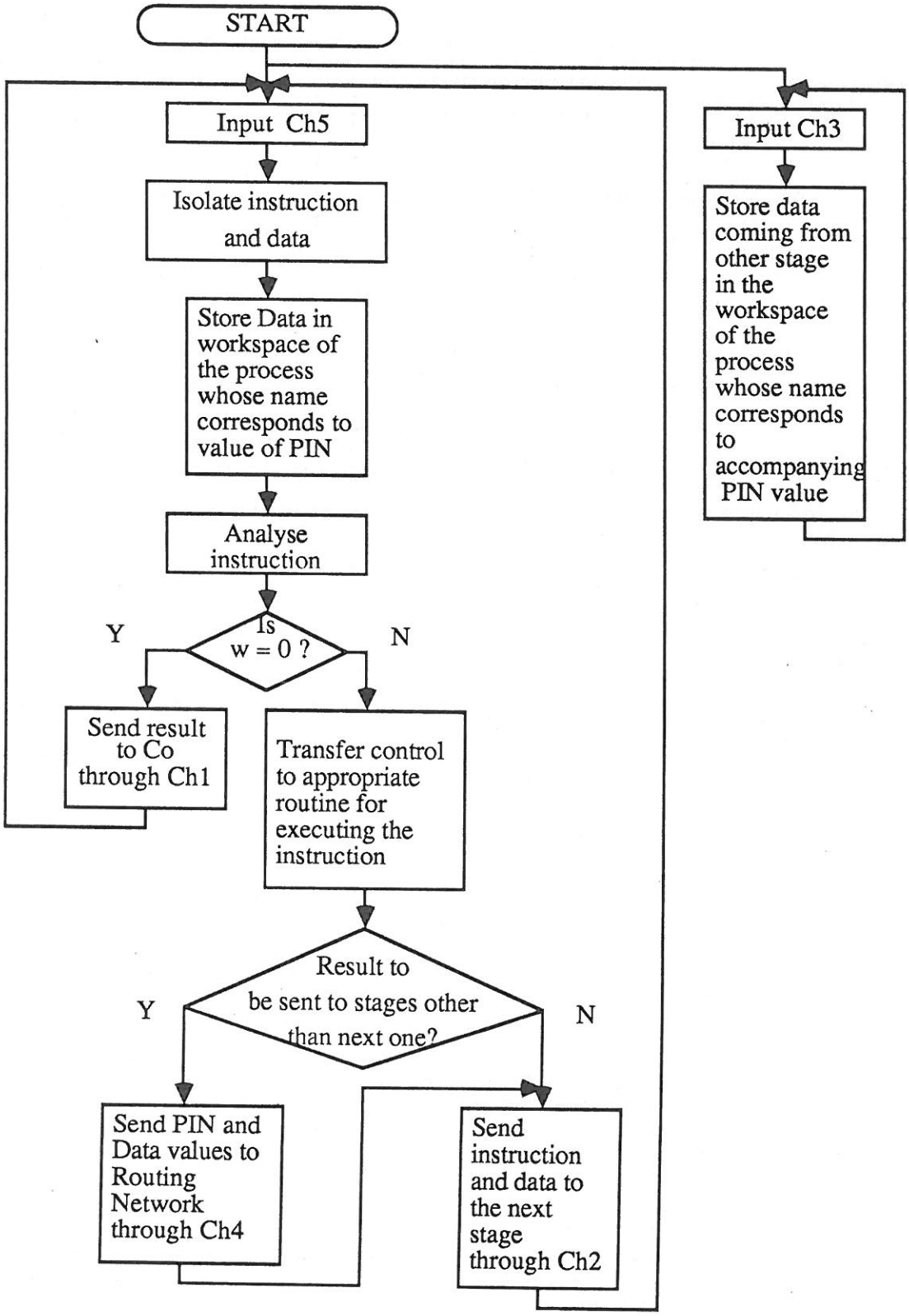

Figure 6 Flow of operations in a Pipeline Stage

processes by replacing small inner loops with one outer loop reduces non-linearities in the flow of data. This simplifies the reservation tables of computational processes. For example, the reservation table for the computational process of Figure 4(a) when executed in the Adaptable Pipeline is as given in Figure 4(c). From the reservation table, it is found that the forbidden list [14] of all computational processes contain only one number which is equal to the number of stages in the pipeline. This makes it easier for Co to analyse the reservation tables of different computational processes and to determine the time of initiation of the different instructions.

It is assumed that after analysis of the reservation tables of different computational processes, Co determines a value 'm' which specifies the number of instructions which must be executed by the first stage before a new instruction can be initiated. As shown in the flow chart of Fig. 7, after the execution of every instruction, the value of m is decremented. If on testing it is found that the value of 'm' is not 0, then the first stage inputs from Ch3, i.e., it inputs the instruction and data coming from the last stage of the pipeline. Otherwise it inputs the contents of Ch0 which is a new instruction sent by Co.

It should be noted that the system proposed here is capable of forming many parallel pipelines. Therefore any stage may become the first stage of a pipeline. This necessitates that all the stages be equipped with programs to implement the steps in both of the flow charts given in Figures 6 and 7.

7. Conclusion

The primary objective of this work has been to design an Adaptable Pipeline which can be used for the processing of different types of computional processes concurrently and which can be easily implemented with commercially available VLSI chips. Both the hardware and the software have been designed with this in mind. The software organisation of a pipeline stage has been described in terms of OCCAM model of processes and channels, thereby facilitating writing of OCCAM programs for the purpose.

It has been stated in section 3, that the Adaptable Pipeline system described in this report has multifarious processing capabilities. While its operation in processing complex arithmetic functions has been illustrated, details of its application in other areas are not discussed. However, it is worth mentioning that besides executing vector instructions, the system is also capable of implementing vector chaining easily. Since the system proposed here can form parallel

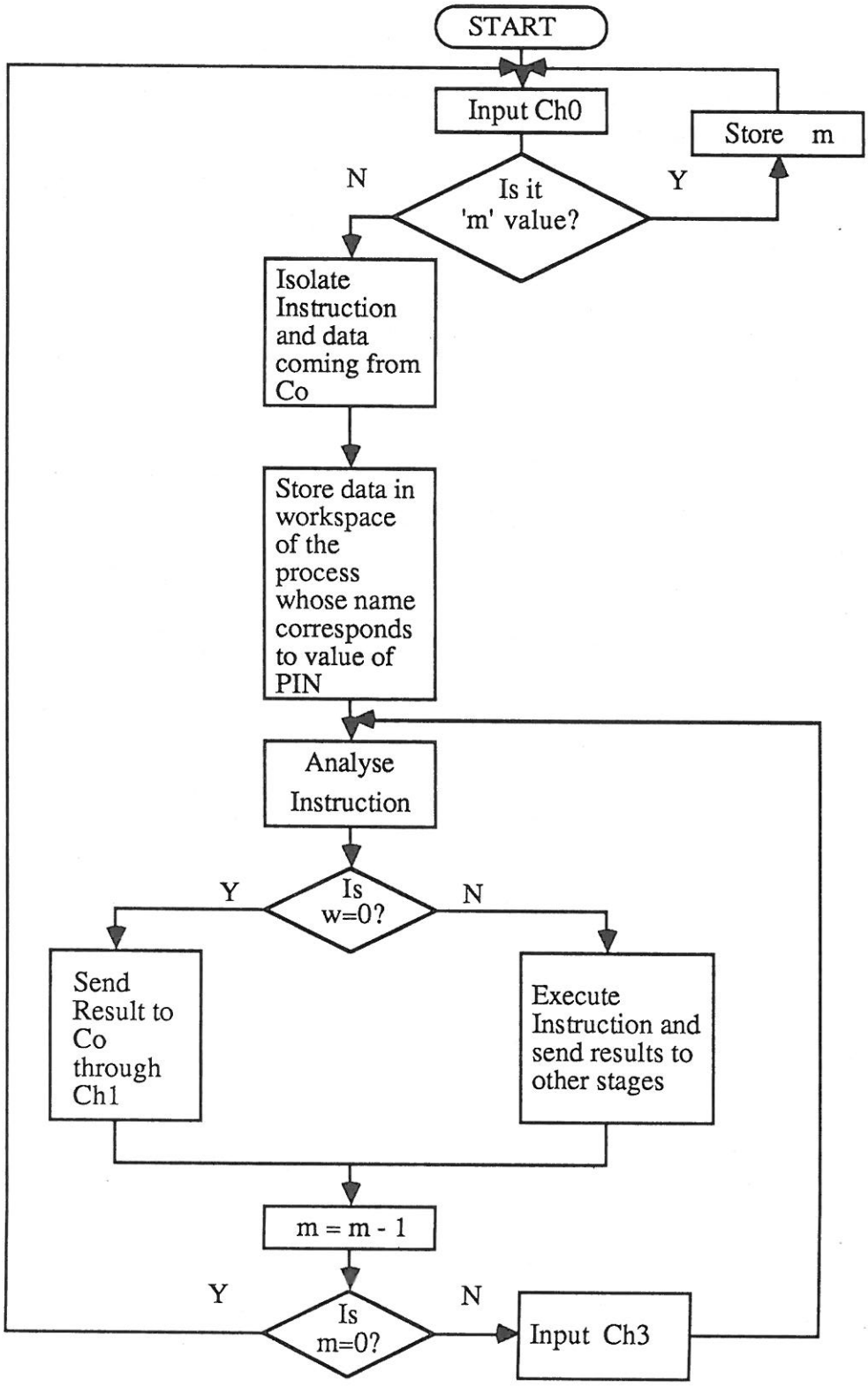

Figure 7 Flow of operations in the First Stage

pipelines, in case of vector chaining, the results formed in the last stage of the first pipeline are not sent back to Co, but is passed on to the first stage of the next pipeline for being used as operands in the next vector instruction.

Further, the Adaptable Pipeline described here is very flexible. As the entire operation of the system is based on software, one can change its area of application by simply altering the OCCAM programs in the Transputers according to the requirements.

Acknowledgement

The author wishes to thank Prof. Nicola Santoro of the Distributed Computing Group, School of Computer Science for his valuable suggestions.

References

[1] N. A. Alexandridis, 'Adaptable Hardware and Software: Problems and Solutions', IEEE Computer, Vol. 19, No.2, February 1986, pp. 29-39

[2] C. Whitby-Strevens, 'The Transputer', Proc. 12 th Annual International Symposium on Computer Architecture, June 1985, pp. 292-300

[3] S.P. Kartashev and S.I. Kartashev, 'Adaptable Pipeline System with Dynamic Architecture', Proc. 1979 International Conference on Parallel Processing, August 1979, pp. 222-230

[4] S.I.Kartashev , S.P.Kartashev and C.V. Ramamoorthy, 'Adaptation Properties for Dynamic Architectures ', Proc. 1979 AFIPS National Computer Conference, Vol. 48, AFIPS Press, Montvale, N.J., pp. 543-556

[5] S.I.Kartashev and S.P.Kartashev, 'Dynamic Architectures: Problems and Solutions', IEEE Computer, Vol. 11, No.7, July 1978, pp. 26-40

[6] INMOS Ltd., OCCAM Programming Manual, Prentice Hall International, 1984

[7] S.Kartashev and S. Karatashev, Guest Editors' Introduction, IEEE Computer, Vol. 19, No.2, February 1986, pp. 9-13

[8] K. Hwang and F. A. Briggs, Computer Architecture and Parallel Processing, McGraw Hill Book Company, 1984

[9] C.V. Ramamoorthy and H.F.Li, 'Pipeline Architecture', ACM Computing Surveys, Vol. 9, No. 1, March 1977, pp. 61-102

[10] W.J.Watson, 'The TI ASC - A Highly Modular and Flexible Supercomputer Architecture' , Proc. AFIPS 1972 Fall Joint Computer Conference, AFIPS Press, 1972, pp. 221-228

[11] R.N. Ibbett and P.C.Capon, 'The Development of the MU5 Computer System', Communications of the ACM, Vol. 21, No.1, January 1978, pp. 13-24

[12] R.M.Russell, 'The CRAY-1 Computer System', Communications of the ACM, Vol. 21, No.1, January 1978, pp. 63-72

[13] INMOS Limited, IMS T424 Transputer Data Sheet

[14] P.M. Kogge, The Architecture of Pipelined Computers, McGraw Hill Book Company, 1981