PERIODICA POLYTECHNICA SER. EL. ENG. 33/3, pp. 151-160 (1989)

A DESIGN METHOD OF LOGICAL SYSTEMS BASED ON PARALLEL PROCESSING

E. BÁNYAI

Department of Process Control
Technical University of Budapest

Received May 19, 1988

Abstract

The paper outlines a design method of digital systems. The effect of problem partitioning can be taken into consideration in the specification of the control procedure. The event-sequence represented by the operation order of the functional units can be handled and so, maximally parallel structures can be formed in systematic steps. The functional units are considered as "black boxes" and there is no need to make any difference between hardware and software realisations during the design procedure of the architecture. For that reason, the method can be used for redesigning e.g. a software problem, realised by a single source, in order to implement it in a parallel structure, which may consist of either software or hardware units.

Keywords: logic design, parallel processing, problem partitioning, control unit.

Introduction

The aim of the outlined in this paper method is to design digital systems to be decomposed for parallel processing (for details see BÁNYAI (1987)).

A digital system consists of a control unit and some functional units. The functional units perform separate subtasks; the control unit co-ordinates the functional units and organizes the internal and external procedures. Most methods provide systematic tools only for designing the control unit and the functional units are assumed to be given or chosen in advance. In this way, the design procedure is separated into at least two parts. The first part consists of intuitive steps from the initial drafting of the problem till specifying the functional units and the control procedure. The second part is a mainly systematical design of the control unit.

For that reason the control unit and the functional units to be controlled are designed separately and thus communication between them is determined by the formal rules of the design methods. In this way, many special properties of the functional units are hidden by the formal tools of

describing (e.g. flow graph). This separated design increases the number of heuristic steps, because:

- the effect of the heuristic problem decomposition can be evaluated only by trial;
- checking the possibility of parallel processing cannot be made systematically;
- the availability of the input values for a functional unit cannot be checked systematically;
- a correspondence between the functional units and the available resources are determined intuitively by the designer;

So, optimality cannot be ensured for the further design steps.

The New Method

For avoiding the disadvantages mentioned above, a new structure is to be applied. In this, the control unit does not exist separately. The efficiency of a heuristic problem-partitioning can be examined by introducing extra symbols in addition to the usual input and output ones. The main new symbols are as follows:

1. Functional elements (FE) which are abstract resources performing the subtasks of the problem. Each FE may have arbitrary properties.
2. Processor elements (PE) which are concrete realisations of FE-s.

The FE-s are handled as black boxes specified by their properties. Thus, any set of building blocks can define the FE-s. After having realized the FE-s by concrete PE-s, the extra properties of the PE-s can be exhausted by updating the knowledge-base of the conditions and new design rules can be formed.

A Simple Example for Demonstrating the Method

Let a simple example illustrate the most important results of the method.

The problem to be solved is the scalar product of two vectors, expressed as

$$x := x_{i-1} + a_i \cdot b_i \,.$$

Problem Partitioning

Let the problem be partitioned as follows:

- The product of the two external input variables (a_i, b_i) is produced by FE_1. It is denoted by $FE_1(a_i, b_i)$.
- Summation the product and the result of the last calculation step is performed by FE_2. It is denoted by $FE_2(FE_1, x_{i-1})$.
- Storing the result for one cycle of the calculation is executed by FE_3. It is denoted by $FE_3(FE_2)$, thus, x_{i-1} can be denoted by FE_3.

In a short form:

$$FE_1\,(a_i, b_i)\,,$$
$$FE_2\,(FE_1, FE_3)\,,$$
$$FE_3\,(FE_2)\,.$$

As mentioned earlier, FE-s are handled as black boxes specified by their connections. For example:

- As the above expressions indicate, the variables within the brackets represent inputs which must be ready for starting the given FE.
- The time required by FE from starting till producing a result, can also be given as a specification. The method can also handle unknown times. In the above example, let the times be given as follows:

$$\begin{aligned} T_{FE_1} &= 10 \\ T_{FE_2} &= 5 \\ T_{FE_3} &= 1\,, \end{aligned}$$

  where the values denote the numbers of the clock cycles required for producing the results.
- The FE-s are supposed not to accept a new start, while their results are being used by any other FE-s. Thus the Operation Cycle of a FE (OC_{FE}) consists of two parts. The first part is the time of producing the result; and the second part is the time, while the result is being used.

Event Sequence and Parallel Processing

Let the possibility of parallel processing and the sequence of events be examined in two phases:

- It is obvious that two FE-s cannot work in a parallel way, if at least one of them requires the result of the other. It can be expressed as a "compatibility relation", so the maximal compatibility classes and optimal closed covers can be determined by the well-known steps. In our example, the optimal cover is:

$$(FE_1, FE_3)(FE_2).$$

- The FE's contained by a common class of the cover can work in a parallel way if the input variables required by them are available. The input variables of a FE are to be distinguished according to their sources, because they may be supplied from outside or produced inside the system. Both types of the variables can be observed in *Fig. 1*, which is obtainable directly from the above cover. The lines with arrows represent the ways of data transfer. In other words, the existence of a cover provides only a necessary condition for parallel processing. For the cause of sufficiency, the input variables must also be available at the proper time.

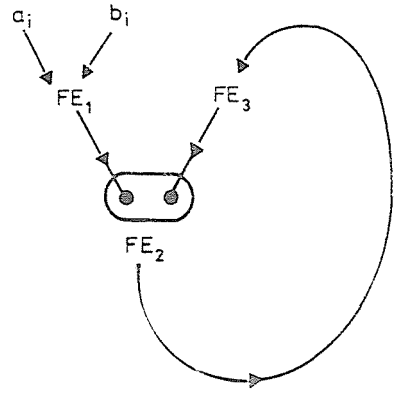

Fig. 1. Initial drafting of the problem

- The availability of the internal input variables can be examined easily on the basis of the routes in *Fig. 1.*

- The availability of the external input variables can be ensured by a proper dialogue with the environment.

If the availability cannot be ensured in any of the cases mentioned above, then another closed cover may be attempted. In this way, the solution

can be fitted and optimized to the preconditions set by the environment of the system.

Possible Goals of Optimization

Let the effect of some aims of optimization be demonstrated as follows.

- Minimizing the time required by the complete procedure. The operation cycles specified by *Fig. 1* and by the values of T_{FE}'s listed above, are to be calculated as sums of the values of times required for producing the results and for the necessary data availability:

$$\begin{aligned} OC_{FE_1} &= 10+5 = 15 \\ OC_{FE_2} &= 5+1 = 6 \\ OC_{FE_3} &= 1+5 = 6. \end{aligned}$$

  Obviously, the longest time ($OC_{max} = OC_{FE}$) can be considered as a bottleneck, which determines the minimal time required for executing the complete procedure. In our example: $OC_{max} = OC_{FE_1}$.

  If a Local Memory is introduced for storing the result of FE_1 during one operation-cycle, then the value of OC_{max} can be reduced. The Local Memory can be considered as a new FE, which changes the connections. As a consequence, FE's may require a new interpretation and thus the design procedure must be repeated for the new set of FE's. In our example:

$$\begin{aligned} &FE_1\,(a_i, b_i) \\ &FE_2\,(FE_3, FE_4) \\ &FE_3\,(FE_2) \\ &FE_4\,(FE_1), \end{aligned}$$

  where FE_4 represents the Local Memory.

  Based on this new description, a new structure and new values of the operation-cycles are obtained as shown in *Fig. 2.* In this case, the bottleneck remained at FE_1. The operation cycle of FE_1 is almost twice as long as of the other FE's. If FE_1 is duplicated as shown in *Fig. 3.* then both copies of $FE_1 \longrightarrow (FE_{11}, FE_{12})$ may be started in an overlap. So, the complete procedure will be executed in a time determined by the fastest elements. Obviously, the duplication of FE_1 requires the duplications of other FE's, too (*Fig. 3*), because the meaning of the complete procedure must remain unchanged. The duplication of all FE's, does not necessarily involve the duplication of all PE's required for the realization.

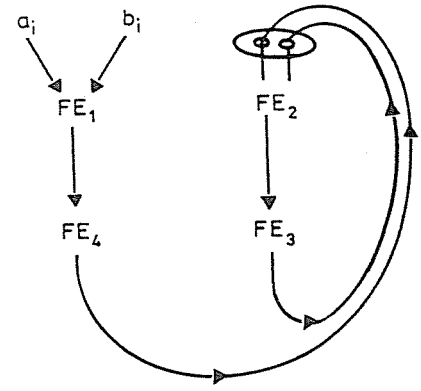

Fig. 2. Introduction of a local memory

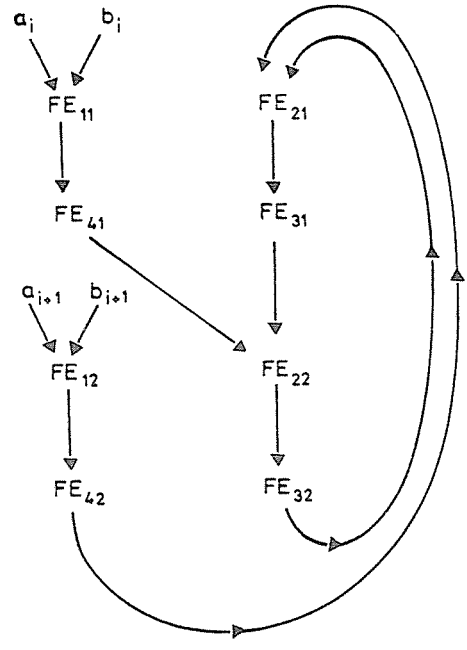

Fig. 3. Duplication of functional elements

- Minimizing the number of the processor elements (PE's) required for the realisation.

Let it be assumed that a set of processor elements is to be applied, which can realise the set of FE's as follows:

$$PE_1 : FE_{11}, FE_{12}$$
$$PE_2 : FE_{21}, FE_{22}$$
$$PE_3 : FE_{31}, FE_{32}, FE_{41}, FE_{42}.$$

The covering requirement determines the necessary numbers of copies for each PE as follows:

for PE_1 :	PE_{11}	and	PE_{12}	(two elements)
for PE_2 :	PE_2			(one element)
for PE_3 :	PE_3	and	PE_{32}	(two elements).

Fig. 4 shows the PE-structure obtained by the above steps.

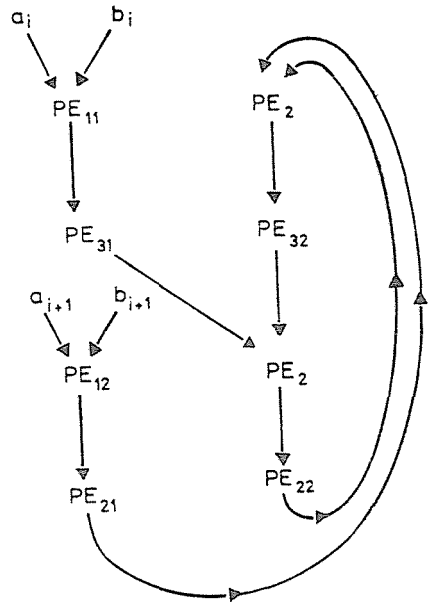

Fig. 4. Assignment of processor elements

Based on *Fig. 4*, the flow graph of the control procedure can be obtained realising the parallel processing shown in *Fig. 5*. On the top of the PE blocks, the input variables are listed which are required by the given PE.

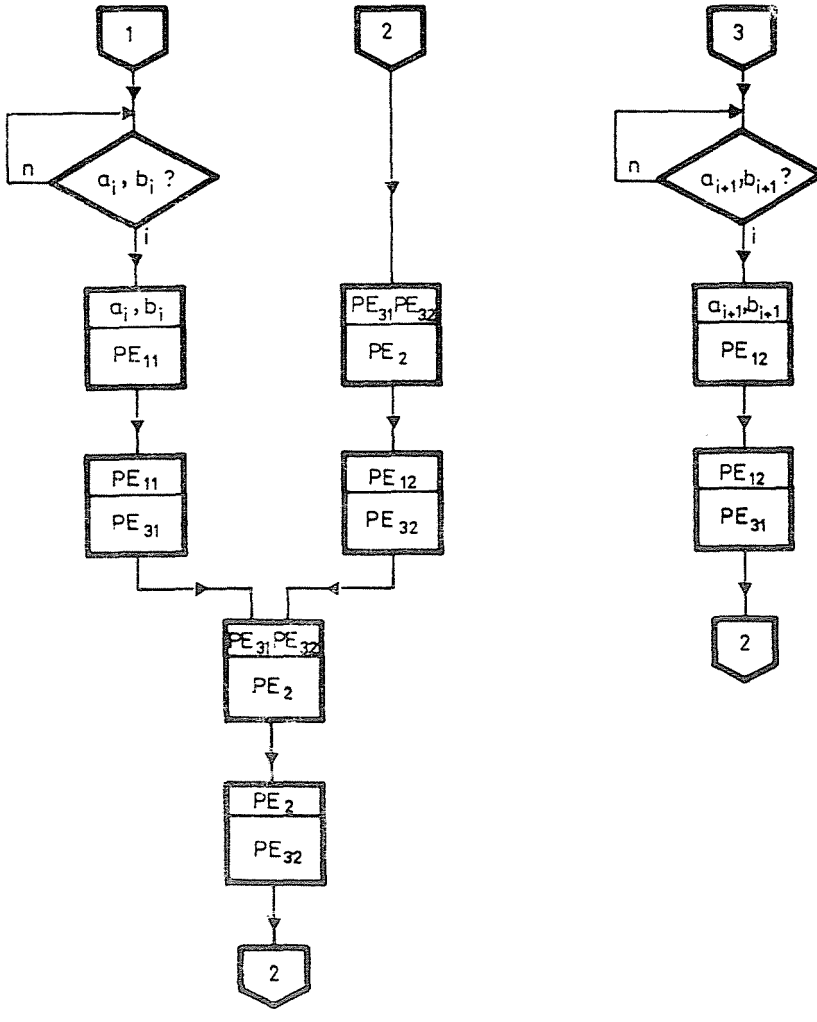

Fig. 5. Completing the control procedure

This flow graph can be considered as the initial specification for a control unit design procedure (BÁNYAI (1987)).

Conclusions

The design method demonstrated by the above simple example is based on the following results:

- A simultaneous handling of the problem partitioning and the control procedure.
- Expanding of the control unit design methods for the systematic system design.
- The realization of the parallel processing based on a selection algorithm which ensures the optimal parallel structure for conditions given in advance.
- The interpretation of the state-reduction for simultaneous control procedures ensuring the maximal speed.

Acknowledgments

The author wishes to thank Professor Dr. Péter Arató for his help and his support to author's research work.

References

ARATÓ, P. (1987): Specification and Realization of Logic Control Procedures on the Basis of Prescribed Input-output Changes. *Periodica Polytechnica Ser. El. Eng.* Vol. *31.*, No. 3–4., pp. 99–153.

BÁNYAI, E. (1987): A Design Method of Logical Systems Based on Parallel Processing. *Scientific Students' Seminar.* Faculty of Electrical Engineering, Technical University of Budapest, pp. 121–183 (in Hungarian)

BÁNYAI, E. (1988): A New Method for Designing Control Units Based on Synchronous Phase Register. *Híradástechnika.* Vol. *39.*, pp. 19–22 (In Hungarian) Jan. 1988.

DAVIS, A.L. (1978): The Architecture and System Method of DDMI: A Recursively Structured Data Driven Machine. *SIGARCH Newsletter,* Vol. *6.*, No. 2., pp. 127-143.

DENNIS, J.B. (1975): First Version of a Data Flow Procedure Language. *MIT MAC Project Technical Memo 61.*

GAJSKI, D.D. – KUHN, R.H. (1983): Guest Editor's Introduction: New VLSI Tools. *Computer* Vol. *16.*, No. 1, December 1983., pp. 178–182.

KUNG, H.T. (1980): The Structure of Parallel Algorithms. *Advances in Computers* Vol. *19.*, No. 3., pp. 65–112.

MCFARLAND, S.J. (1983): Computer Aided Partitioning of Behavioral Hardware Descriptions. *Design Automation Conference Proceedings, ACM IEEE* June, 1983 Miami Beach, Florida, pp. 472–476.

THOMAS, D.E. (1986): Automatic Data Path Synthesis. *In Goto, S. (ed.): Design Methodologies.* North-Holland—Elsevier Science Publishers B.V., Amsterdam/New York, pp. 401–439.

Address:

Ervin BÁNYAI
Department of Process Control
Technical University of Budapest
Budapest, Hungary, H-1521