The full-decomposition of sequential machines with the separate realization of the next-state and output functions

Research Report
ISSN 0167-9708
Coden: TEUEDE

Eindhoven
University of Technology
Netherlands

Faculty of Electrical Engineering

The Full-Decomposition of Sequential Machines with the Separate Realization of the Next-State and Output Functions

by
L. Jóźwiak

EUT Report 89-E-222
ISBN 90-6144-222-2

March 1989

Eindhoven University of Technology Research Reports

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering

Eindhoven The Netherlands

ISSN 0167-9708 Coden: TEUEDE

THE FULL-DECOMPOSITION OF SEQUENTIAL MACHINES WITH THE SEPARATE REALIZATION OF THE NEXT-STATE AND OUTPUT FUNCTIONS

by

L. Jóźwiak

EUT Report 89-E-222

ISBN 90-6144-222-2

Eindhoven

March 1989

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Jóźwiak, L.

The full-decomposition of sequential machines with the separate realization of the next-state and output functions / by L. Jóźwiak. - Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering. - Fig. - (EUT report, ISSN 0167-9708; 89-E-222)
Met lit. opg., reg.
ISBN 90-6144-222-2
SISO 664 UDC 681.325.65:519.6 NUGI 832
Trefw.: automatentheorie.

THE FULL-DECOMPOSITION OF SEQUENTIAL MACHINES WITH THE SEPARATE REALIZATION OF THE NEXT-STATE AND OUTPUT FUNCTIONS

L. Jóźwiak

ABSTRACT - The decomposition theory of sequential machines aims to find answers to the following important practical problem: *how to decompose a complex sequential machine into a number of simpler partial machines in order to: simplify the design, implementation and verification process; make it possible to process (to optimize, to implement, to test,...) the separate partial machines although it may be impossible to process the whole machine with existing tools; make it possible to implement the machine with existing building blocks or inside of a limited silicon area.*

For many years, decomposition of the internal states of sequential machines has been investigated. Here, decomposition of the states, as well as, the inputs and outputs of sequential machines is considered, i.e. full-decomposition.

In [16], classification of full-decompositions is presented and theorems about the existence of different full-decompositions are provided. In this report a special full-decomposition strategy is investigated - the full-decomposition of sequential machines with the separate realization of the next-state and output functions. This strategy has several advantages comparing to the case where a sequential machine is considered as a unit. In the report, the results of theoretical investigations are presented; however, the notions and theorems provided here have straightforward practical interpretations and they can be directly used in order to develope programs computing different sorts of decompositions for sequential machines.

INDEX TERMS - Automata theory, decomposition, logic system design, sequential machines.

ACKNOWLEDGEMENTS - The author is indebted to Prof.ir. A. Heetman and Prof.ir. M.P.J. Stevens for making it possible to perform this work, to Dr. P.R. Attwood for making corrections to the English text and to mr. C. van de Watering for typing the text.

CONTENTS

1. Introduction

The decomposition theory of sequential machines aims to find answers to the following question:

How to decompose a complex sequential machine into a number of simpler partial machines in order to: simplify the design, implementation and verification process; make it possible to process (to optimize, to implement, to test,...) the separate partial machines although it may be impossible to process the whole machine with existing tools; make it possible to implement the machine with the existing building blocks or inside of a limited silicon area.

The solution of this problem is very important, because the control units and the serial processing units of today's large information processing systems are often functionally defined in the form of a big sequential machine or of a number of such machines.

For many years, decomposition of the internal states of sequential machines has been investigated [2][3][8][9][11][12][13][17]..[21]; however, together with progress in LSI technology and the introduction of array logic (PAL,PGA,PLA,PLS) into design of sequential circuits, a real need has arisen for decompositions of the states of sequential machines, as well as, inputs and outputs, i.e. for full-decompositions.

An approach to the full-decomposition of sequential machines has been presented in [14] and [15].

In [16], classification of full-decompositions and formal definitions of different types of ful-decompositions for Mealy and Moore machines are presented and theorems about different full-decompositions are provided.

In this report, another type of full-decomposition is considered - the full-decomposition of sequential machines with the separate realization of the next-state and output functions.

2. Two full-decomposition strategies

DEFINITION 2.1 A *sequential machine* M is an algebraic system defined as follows:

$$M = (I,\ S,\ O,\ \delta,\ \lambda)\ ,$$

where:

- I - a finite non-empty set of inputs,
- S - a finite non-empty set of internal states,
- O - a finite set of outputs,
- δ - the next-state function: δ: SxI $\longrightarrow$ S,
- λ - the output function, λ: SxI $\longrightarrow$ O (a *Mealy* machine), or λ: S $\longrightarrow$ O (a *Moore* machine).

When an output set O and the output function λ are not defined, the sequential machine M = (I, S, δ) is called a ***state machine***.

Let M = (I, S, O, δ, λ) be the sequential machine to be decomposed. In [16] such a full-decomposition is presented, that it is necessary to find two partial sequential machines $M_1 = (I_1, S_1, O_1, \delta^1, \lambda^1)$ and $M_2 = (I_2, S_2, O_2, \delta^2, \lambda^2)$ each having fewer states and/or inputs and/or outputs than M. Each calculates its next-states and outputs using only the information about the input of M and, in combination, forming a sequential machine M' that imitates the behaviour of M from the input-output, or state-output and input-output, point of view (Fig. 2.1).

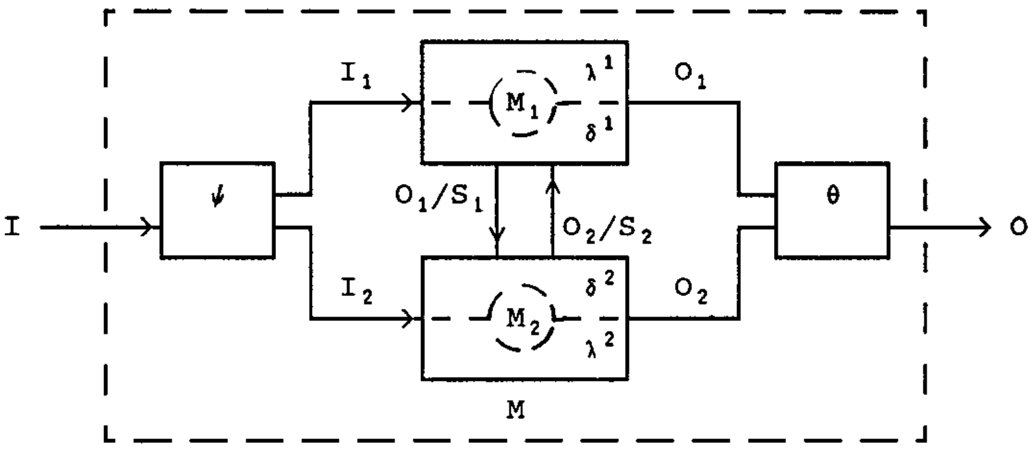

Fig. 2.1 The full-decomposition of a sequntial machine M with two partial sequential machines M_1 and M_2.

Here, another kind of a full-decomposition will be considered. Instead of considering the realization of a machine M as the whole, the realization of the next-state function δ is considered separately from the realization of the output function λ.

It is possible to abstract from the output function λ and first to decompose the state machine defined by I, S and the next-state function δ. Then, it is possible to realize the output function λ, where λ is treated as a function of the primary inputs to a sequential machine M (in the Mealy case), and of the states of partial state machines M_1 and M_2 obtained from a full decomposition of the state machine defined by I, S and δ (Fig. 2.2).

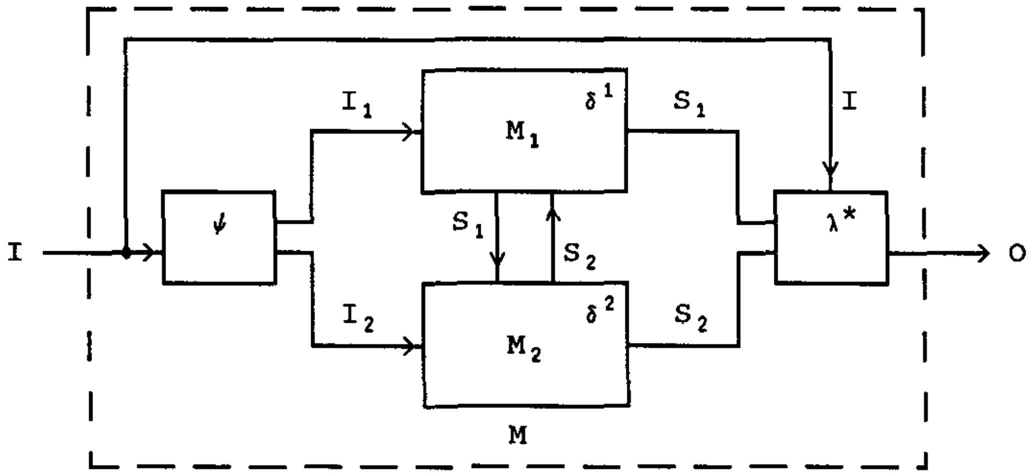

Fig. 2.2 The full-decomposition of a sequential machine M with the separate realization of the next-state and output functions.

3. The full-decomposition of state machines

Let M = (I, S, δ) be the state machine to be decomposed and M_1= (I_1,S_1,δ^1) and M_2= (I_2,S_2,δ^2) be two partial state machines.

In a full-decomposition of a state machine, it is necessary to find the partial state machines M_1 and M_2 each of which having fewer states and/or inputs than the state machine M and together forming a state machine M′ that imitates the behaviour of M from the input-state point of view.

The following types of full-decomposition are feasible for a state machine:

- <u>a</u> <u>parallel</u> <u>full-decomposition</u>, where each of the component state machines calculates its own next-state independently of the other component state-machine, using only information about its own internal state and partial information about the inputs (Fig. 3.1).

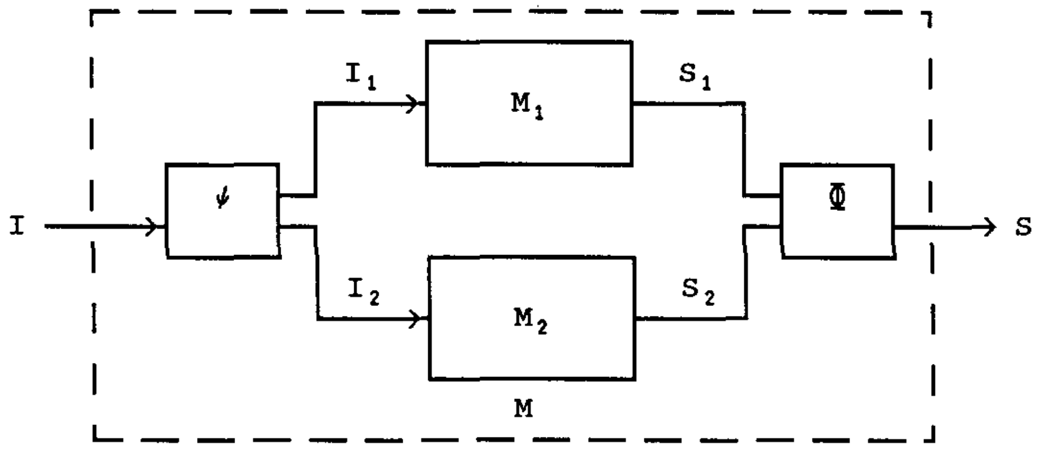

Fig. 3.1 The parallel full-decomposition of a state machine M into component state machines M_1 and M_2.

- <u>a</u> <u>serial</u> <u>full-decomposition</u> <u>of</u> <u>type</u> <u>PS</u> (present-state), where one of the component state machines uses the information about the present-state of the second component state machine and partial information about the inputs in order to calculate its own next-state (Fig. 3.2).

- <u>a</u> <u>serial</u> <u>full-decomposition</u> <u>of</u> <u>type</u> <u>NS</u> (next-state), where one of the component state machines uses the information about the next-state of the second component state machine and partial information about the inputs in order to calculate its own next-state (Fig. 3.2).

- <u>a</u> <u>general</u> <u>full-decomposition</u>, where each of the component state machines uses information about the state of the other component machine and partial information about the primary inputs in order to calculate its own next-state (Fig. 3.3).

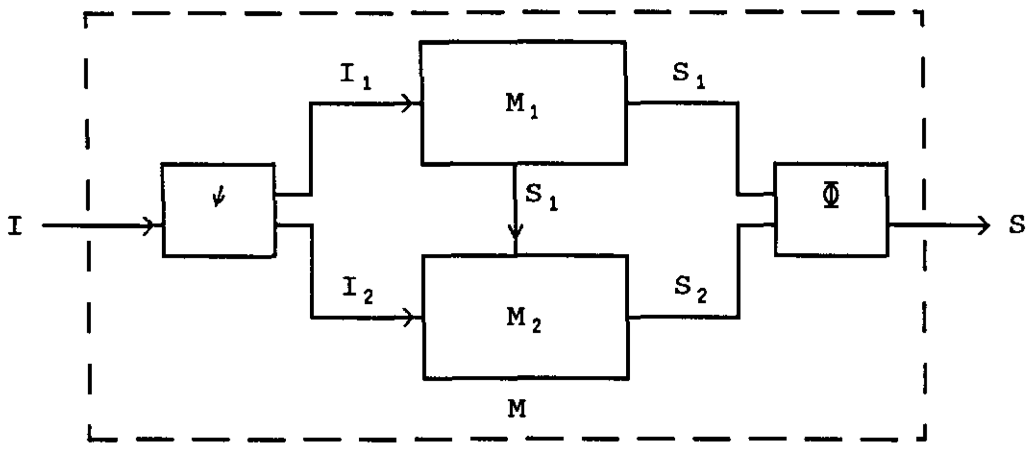

Fig. 3.2 The serial full-decomposition of a state machine M into component state machines M_1 and M_2.

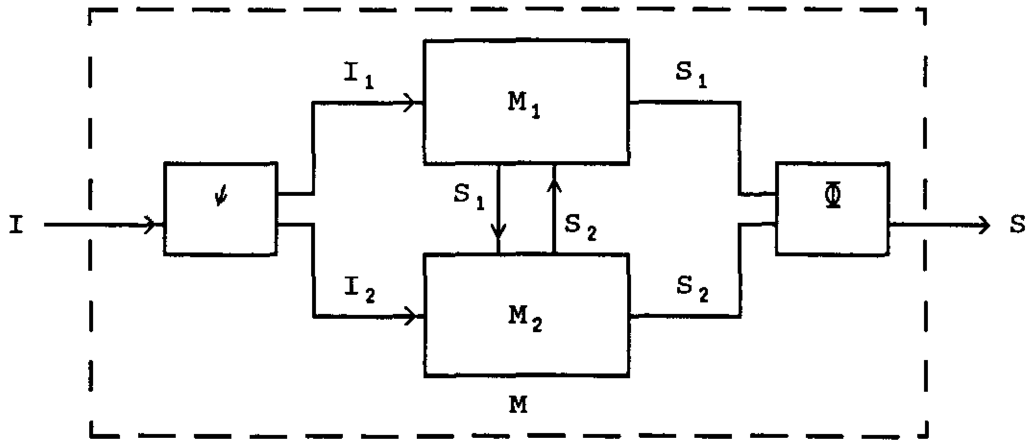

Fig. 3.3 The general full-decomposition of a state machine M into component state machines M_1 and M_2.

For a general full-decomposition, two types are feasible: - type PS (each of the submachines uses information about the present-state of the other submachine); and type **PNS** (one of the submachines uses information about the present-state of the second and the other submachine about the next-state of the first). However, in this paper, only type PS will be considered and the term *"general decomposition"* is assumed to mean *"general decomposition of type PS"*.

Before considering the different types of full-decompositions for state machines, a definition of realization must be formulated.

<u>DEFINITION 3.1</u> The state machine $M'=(I',S',\delta')$ realizes a state machine $M=(I, S, \delta)$ *if, and only if*, the following relations exist:

$\psi: I \longrightarrow I'$ (a function)

and

$\Phi: S' \longrightarrow S$ (a surjective partial function),

such that:

$\Phi(S')\delta_x = \Phi(S'\delta'_{\psi(x)})$.

In a full-decomposition of state machine M, it is necessary to find the partial state machines M_1 and M_2 as well as the mappings: $\psi: I \longrightarrow I_1 x I_2$ and $\Phi: S_1 x S_2 \longrightarrow S$.

The machines M_1 and M_2 together with the mappings ψ and Φ realize the behaviour of the machine M.

A full-decomposition of a state machine M is said to be <u>non-trivial</u> *if, and only if*, the number of inputs to each of the partial state machines is less than the number of inputs to machine M and/or the number of states of each of the partial state machines is less than the number of states of a machine M.

From the considerations above, it is evident that full-decompositions of state machines can be characterized by the type of connection between the component state machines. The formal definitions of all the machine connections considered in this paper and the formal definition of the full-decomposition of a state machine are given below.

Let: $s \in S_1$, $t \in S_2$, $x_1 \in I_1$, $x_2 \in I_2$.

<u>DEFINITION 3.2</u> A *parallel connection* of two state machines:

$M_1 = (I_1, S_1, \delta^1)$

and

$M_2 = (I_2, S_2, \delta^2)$

is the machine:

$M_1 \| M_2 = (I_1 x I_2, S_1 x S_2, \delta^*)$

where:

$\delta^*((s,t),(x_1,x_2)) = (\delta^1(s,x_1), \delta^2(t,x_2))$

/

<u>**DEFINITION 3.3**</u> A *serial connection of type PS* of two state machines:

$$M_1 = (I_1, S_1, \delta^1)$$

and

$$M_2 = (I_2', S_2, \delta^2),$$

for which $I_2' = S_1 x I_2$,

is the machine $M_1 \xrightarrow{PS} M_2 = (I_1 x I_2, S_1 x S_2, \delta^*)$,
where:

$$\delta^*((s,t),(x_1,x_2)) = (\delta^1(s,x_1), \delta^2(t,(s,x_2)))$$

<u>**DEFINITION 3.4**</u> A *serial connection of type NS* of two state machines:

$$M_1 = (I_1, S_1, \delta^1)$$

and

$$M_2 = (I_2', S_2, \delta^2),$$

for which $I_2' = S_1 x I_2$,

is the machine $M_1 \xrightarrow{NS} M_2 = (I_1 x I_2, S_1 x S_2, \delta^*)$,
where:

$$\delta^*((s,t),(x_1,x_2)) = (\delta^1(s,x_1), \delta^2(t,(\delta^1(s,x_1),x_2)).$$

<u>**DEFINITION 3.5**</u> A *general connection of type PS* of two state machines:

$$M_1 = (I_1', S_1, \delta^1)$$

and

$$M_2 = (I_2', S_2, \delta^2),$$

for which $I_2' = S_1 x I_2$ and $I_1' = S_2 x I_1$,
is the machine:

$$M_1 \overset{PS}{\longleftrightarrow} M_2 = (I_1 x I_2, S_1 x S_2, \delta^*),$$

where:

$$\delta^*((s,t),(x_1,x_2)) = (\delta^1(s,(t,x_1)), \delta^2(t,(s,x_2)))$$

DEFINITION 3.6 The state machine $M_1 \lhd\!\rhd M_2$ is a *full decomposition of type* $\lhd\!\rhd$ of state machine M *if, and only if*, the connection of a given type $\lhd\!\rhd$ of the state machines M_1 and M_2 realizes M, where:

$$\lhd\!\rhd = \| \ , \ \xrightarrow{PS} \ , \ \xrightarrow{NS}, \ \xleftrightarrow{PS} \ .$$

In order to analyze the information flow inside and between the state machines, the partition and partition pairs concepts, introduced by Hartmanis [11][12], are used here.

Let: S be any set of elements.

DEFINITION 3.7 *Partition* π on S is defined as follows:

$$\pi = \{B_i \mid B_i \subseteq S \text{ and } B_i \cap B_j = 0 \text{ for } i \neq j \text{ and } \bigcup_i B_i = S\},$$

i.e. a partition π on S is a set of disjointed subsets of S whose set union is S.

For a given $s \in S$, the block of a partition π containing s is denoted by: $[s]\pi$ while $[s]\pi = [t]\pi$ denotes that s and t are in the same block of π. Similarly, the block of a partition π containing S', where $S' \subseteq S$, is denoted by $[S']\pi$.

A partition containing only one element of S in each block is called a *zero partition* and is denoted by $\pi_S(0)$. A partition containing all the elements of S in one block is called an *identity* or *one partition* and is denoted by $\pi_S(I)$.

Let: π_1 and π_2 be two partitions on S.

DEFINITION 3.8 *Partition product* $\pi_1 \cdot \pi_2$ is the partition on S such that $[s]\pi_1 \cdot \pi_2 = [t]\pi_1 \cdot \pi_2$ *if and only if* $[s]\pi_1 = [t]\pi_1$ and $[s]\pi' = [t]\pi_2$.

From this definition, it follows that the blocks of $\pi_1 \cdot \pi_2$ are obtained by intersecting the blocks of π_1 and π_2.

Let: π_S, τ_S, π_I be the partitions on $M = (I, S, \delta)$, in particular: π_S, τ_S on S, π_I on I.

DEFINITION 3.9

(i) (π_S,τ_S) is an *S-S partition pair if and only if* $\forall B\in\pi_S\ \forall x\in I:\ B\overline{\delta}_x\subseteq B',\ B'\in\tau_S$.

(ii) (π_I,π_S) is an *I-S partition pair if and only if* $\forall A\in\pi_I\ \forall s\in S:\ s\overline{\delta}_A\subseteq B\ ,\ B\in\pi_S$.

The practical meaning of the notions introduced above is as follows:

– (π_S,τ_S) is an S-S partition pair *if and only if* the blocks of π_S are mapped by M into the blocks of τ_S. Thus, if the block of π_S which contains the present-state of the machine M is known and the present input of M too, it is possible to compute unambiguously the block of τ_S which contains the next-state of M for the states from a given block of π_S and a given input. The interpretation of the notion of an I-S partition pair is similar.

DEFINITION 3.10 Partition π_S has a *substitution property* (it is an *SP-partition*) *if and only if* (π_S,π_S) is an S-S pair.

Considering a state machine M = (I, S, δ) to be a special case of a Moore machine M'= (I, S, O, δ, λ), where O = S and λ is an identity function or a special case of a Mealy machine M"= (I, S, O, δ, λ), where O = S and $\lambda = \delta$; the definitions for the full-decompositions of state machines are special cases of the appropriate definitions presented in [16] for sequential machines.

Thus, the theorems about the existence of full-decompositions of state machines can be obtained directly from the appropriate theorems proved in [16], therefore, they are given below without proof.

THEOREM 3.1 The state machine M = (I, S, δ) has a non-trivial parallel full-decomposition *if* two partitions π_I and τ_I on I and two partitions π_S and τ_S on S exist, such that the following conditions are satisfied:

(i) (π_S,π_S) is a S-S partition pair,
(ii) (π_I,π_S) is an I-S partition pair,
(iii) (τ_S,τ_S) is a S-S partition pair,
(iv) (τ_I,τ_S) is an I-S partition pair,
(v) $\pi_S\cdot\tau_S = \pi_S(\emptyset)$,
(vi) $|\pi_I|<|I|\wedge|\tau_I|<|I|\vee|\pi_S|<|S|\wedge|\tau_S|<|S|$.

The interpretation of theorem 3.1 is as follows:

Let: $M = (I, S, \delta)$ be the state machine to be decomposed.

Let: $M_1 = (\pi_I, \pi_S, \delta^1)$ and $M_2 = (\tau_I, \tau_S, \delta^2)$ be two state machines for which the partitions π_I, π_S, τ_I and τ_S satisfy the conditions of Theorem 3.1 and let the functions δ^1 and δ^2 be defined as follows:

$$\forall B1 \in \pi_S \ \forall A1 \in \pi_I: \ \delta^1(B1, A1) = [\bar{\delta}(B1, A1)]\pi_S ,$$

and

$$\forall B2 \in \tau_S \ \forall A2 \in \tau_I: \ \delta^2(B2, A2) = [\bar{\delta}(B2, A2)]\tau_S ,$$

where

$$\bar{\delta}: 2^S x 2^I \longrightarrow 2^S$$

and

$$\bar{\delta}(Q, X) = \{\delta(s, x) \mid s \in Q \wedge x \in X\} \text{ for } X \subseteq I \text{ and } Q \subseteq S .$$

Let: $\psi: I \longrightarrow \pi_I x \tau_I$ be an injective function,

$\Phi: \pi_S x \tau_S \longrightarrow S$ be a surjective partial function

and

$$\psi(x) = ([x]\pi_I, [x]\tau_I) ,$$

$$\Phi(B1, B2) = B1 \cap B2 \ \textit{if} \ B1 \cap B2 \neq \emptyset .$$

Since (π_S, π_S) is a S-S partition and (π_I, π_S) is an I-S partition pair, $\bar{\delta}(B1, A1)$ will be included in only one block of π_S. This means that $\delta^1(B1, A1)$ can be defined unambiguously. So, based only on the information about the block of π_I containing the input of M and the block of π_S containing the state of M (i.e. information about the input and present-state of M_1), state machine M_1 can calculate unambiguously the block of π_S in which the next-state of M is contained (i.e. M_1 can calculate its own state).

Similarly, since (τ_S, τ_S) is a S-S partition pair and (τ_I, τ_S) is an I-S partition pair, $\bar{\delta}(B2, A2)$ will be included in only one block of τ_S meaning that $\delta^2(B2, A2)$ is defined unambiguously.

Thus, state machine M_2, based only on the information about its input and state (i.e. knowledge of the adequate block of τ_I and the block of τ_S), can calculate unambiguously its next-state (i.e. the adequate block of τ_S).

Since $\pi_S \cdot \tau_S = \pi_S(\emptyset)$, with information about the blocks of π_S calculated by M_1 and the blocks of τ_S calculated by M_2 (i.e. information about the states of M_1 and M_2), it is possible to calculate unambiguously the state of M.

<u>THEOREM 3.2</u> The state machine $M = (I, S, \delta)$ has a non-trivial type PS serial full decomposition *if* two partitions π_I and τ_I on I and two partitions π_S and τ_S on S exist, such, that the following conditions are satisfied:

(i) (π_S, π_S) is a S-S partition pair,
(ii) (π_I, π_S) is an I-S partition pair,
(iii) (τ_I, τ_S) is an I-S partition pair,
(iv) $\pi_S \cdot \tau_S = \pi_S(\emptyset)$,
(v) $|\pi_I|<|I| \wedge |\pi_S| \cdot |\tau_I|<|I| \vee |\pi_S|<|S| \wedge |\tau_S|<|S|$.

If the partial state machines are defined as follows:
$M_1 = (\pi_I, \pi_S, \delta^1)$ and $M_2 = (\pi_S x \tau_I, \tau_S, \delta^2)$,
the partitions π_S, π_I, τ_I and τ_S will satisfy the conditions of Theorem 3.2 and the functions δ^1 and δ^2 will have the following definitions:

$$\forall B1 \epsilon \pi_S \ \forall A1 \epsilon \pi_I: \ \delta^1(B1,A1) = [\bar{\delta}(B1,A1)]\pi_S$$

$$\forall B1 \epsilon \pi_S \ \forall B2 \epsilon \tau_S \ \forall A2 \epsilon \tau_I: \ \delta^2(B2,(B1,A2)) = [\bar{\delta}((B1 \cap B2),A2)]\tau_S$$

and, if the functions ψ and Φ will be defined in the same way as for Theorem 3,1, then the interpretation of Theorem 3.2 is like that of Theorem 3.1.

<u>THEOREM 3.3</u> The state machine $M = (I, S, \delta)$ has a non-trivial type NS serial full-decomposition, *if* two partitions π_S and τ_S on S and two partitions π_I and τ_I on I exist, such, that the following conditions are satisfied:

(i) (π_S, π_S) is a S-S partition pair,
(ii) (π_I, π_S) is an I-S partition pair,
(iii) $\forall s,t \epsilon S \ \forall x_1,x_2 \epsilon I$:
if $[s]\tau_S=[t]\tau_S \wedge [x_1]\tau_I=[x_2]\tau_I \wedge [s\delta_{x_1}]\pi_S=[t\delta_{x_3}]\pi_S$
then $[s\delta_{x_1}]\tau_S=[t\delta_{x_2}]\tau_S$,
(iv) $\pi_S \cdot \tau_S = \pi_S(\emptyset)$,
(v) $|\pi_I|<|I| \wedge |\pi_S| \cdot |\tau_S|<|I| \vee |\pi_S|<|S| \wedge |\tau_S|<|S|$.

If the partial state machines are defined as follows:
$M_1 = (\pi_I, \pi_S, \delta^1)$ and $M_2 = (\pi_S x \tau_I, \tau_S, \delta^2)$,
the partitions π_I, π_S, τ_I and τ_S will satisfy the conditions of Theorem 3.3 and the functions δ^1 and δ^2 will have the following

definitions:

$\forall B1\epsilon\pi_S\ \forall A1\epsilon\pi_I$: $\delta^1(B1,A1) = [\bar{\delta}(B1,A1)]\pi_S$,

$\forall B1'\epsilon\pi_S\ \forall B2\epsilon\tau_S\ \forall A2\epsilon\tau_I$:

$$\delta^2(B2,(B1',A2)) = [\{\delta(s,x)\,|\,s\epsilon B2, x\epsilon A2, \delta(s,x)\epsilon B1'\}]\tau_S$$

and, if the functions ψ and Φ will be defined in the same way as for Theorem 3.1, then the interpretation of Theorem 3.3 is like that of Theorem 3.1.

<u>**THEOREM 3.4**</u> The state machine $M = (I, S, \delta)$ has a non-trivial type PS general full decomposition, *if, and only if*, two partitions π_I and τ_I on I and two partitions π_S and τ_S on S exist, such, that the following conditions are satisfied:

(i) (π_I, π_S) is an I-S partition pair,

(ii) (τ_I, τ_S) is an I-S partition pair,

(iii) $\pi_S \cdot \tau_S = \pi_S(\emptyset)$,

(iv) $|\tau_S|\cdot|\pi_I|<|I| \wedge |\pi_S|\cdot|\tau_I|<|I| \vee |\pi_S|<|S| \wedge |\tau_S|<|S|$.

If the partial state machines are defined as follows: $M_1 = (\tau_S x\pi_I, \pi_S, \delta^1)$ and $M_2 = (\pi_S x\tau_I, \tau_S, \delta^2)$, the partitions π_I, π_S, τ_I and τ_S will satisfy the conditions of Theorem 3.4 and the functions δ^1 and δ^2 will have the following definitions:

$\forall B1\epsilon\pi_S\ \forall B2\epsilon\tau_S\ \forall A1\epsilon\pi_I$:

$$\delta^1(B1,(B2,A1)) = [\bar{\delta}((B1\cap B2),A1)]\pi_S \quad ,$$

$\forall B1\epsilon\pi_S\ \forall B2\epsilon\tau_S\ \forall A2\epsilon\tau_I$:

$$\delta^2(B2,(B1,A2)) = [\bar{\delta}((B1\cap B2),A2)]\tau_S \quad ,$$

and, if the functions ψ and Φ will be defined in the same way as for Theorem 3.1, then the interpretation of Theorem 3.4 is like that of Theorem 3.1.

In [14], a theorem similar to Theorem 3.2 is proved; however, there are two important differences between Theorem 3.2 and the theorem proved in [14]: Theorem 3.2 is formulated with weaker assumptions (e.g. it is not required to fulfil the condition: $\pi_I \cdot \tau_I = \pi_I(\emptyset)$, but it is required in [14]) and another definition of nontriviality is used. So, Theorem 3.2 is more general than the one proved in [14].

4. The realization of an output function

Let: $M = (I, S, O, \delta, \lambda)$ be the sequential machine to be decomposed.

Let: $M' = (I, S, \delta)$ be the state machine expressing the next-state function of M that is implemented as a given type $\Leftrightarrow$ ($\Leftrightarrow$, $=$, $\parallel$, $\xrightarrow{PS}$, $\xrightarrow{NS}$, $\xleftrightarrow{PS}$) of connection of partial state-machines $M_1 = (I_1, S_1, \delta^1)$ and $M_2 = (I_2, S_2, \delta^2)$.

Let: ψ and Φ be two relations:

$\psi: I \longrightarrow I_1 x I_2$ (a function) ,

$\Phi: S_1 x S_2 \longrightarrow S$ (a surjective partial function) ,

defining mappings from the inputs of M (M') onto inputs of M_1 and M_2 and from states of M_1 and M_2 into states of M (M') ($\psi(x) = ([x]\pi_I, [x]\tau_I)$ where: $x \in I$,

$\Phi(s_1, s_2) = s_1 \cap s_2$ *if* $s_1 \cap s_2 \neq \emptyset$, where: $s_1 \in S_1 = \pi_S, s_2 \in S_2 = \tau_S$).

When the conditions of one of the theorems presented in Paragraph 3 are satisfied and, in particular, the condition $\pi_S \cdot \tau_S = \pi_S(\emptyset)$, then, each state s of M will be defined unambiguously by the states s_1 of M_1 and s_2 of M_2. Now, it is possible to express the output function of M as a function of the states of M_1 and M_2 and, in a Mealy machine, a function of the primary inputs of M:

$\lambda^*: S_1 x S_2 \longrightarrow O \cup \{-\}$

and

$$\lambda^*(s_1, s_2) = \begin{cases} \lambda(s_1 \cap s_2) = \lambda(s) \text{ } if \text{ } s_1 \cap s_2 \neq \emptyset \\ - \text{ } if \text{ } s_1 \cap s_2 = \emptyset \end{cases}$$

(in a Moore machine)

or

$\lambda^*: S_1 x S_2 x I \longrightarrow O \cup \{-\}$

and

$$\lambda^*(s_1, s_2, x) = \begin{cases} \lambda(s_1 \cap s_2, x) = \lambda(s, x) \text{ } if \text{ } s_1 \cap s_2 \neq \emptyset \\ - \text{ } if \text{ } s_1 \cap s_2 = \emptyset \end{cases}$$

(in a Mealy machine),

where "–" means "don't care".

If the resultant function λ^* is not too complicated, then, it can be directly implemented with one matrix-logic building block, otherwise, it must be decomposed before implementation.

Contrary to the states of a sequential machine, inputs and outputs of a sequential machine are pre-assigned in most cases, because the inputs are considered by direct signals from around the machine, while, outputs are direct signals sent by the machine to its surroundings. Therefore, after assigning states of the machines M_1 and M_2, the output fgunction λ^* can be represented by a set of Boolean functions $\{\lambda_i^*\}$ (a multiple output Boolean function) of the input and state variables. So, in order to decompose the function λ^*, the methods for partitioning multiple output Boolean functions for matrix-logic implementation can be used. Describing those methods is beyond the scope of this report.

In the state assignment process for M_1 and M_2, information about the complexity of the resultant function λ^* can be used in order to choose the state assignment that minimizes the complexity of a resultant logic.

5. Conclusion

The full-decomposition of a sequential machine can be done according to two different decomposition strategies. It is possible to consider a sequential machine as a unit and to find the partial sequential machines that realize the behaviour of a given sequential machine, or, the full-decomposition of the state machine, that expresses the next-state function δ of a given sequential machine, can be considered separately from the realization of output function λ.

The first strategy is described in [16] and the second in this report.

In the first case, the output functions λ^1 and λ^2 for the partial sequential machines and the output decoder θ must be implemented. In the second case, instead of λ^1, λ^2 and θ, only the output function λ^* need be implemented. This is especially attractive for Moore machines, where: λ^* is only a function of the states of partial state machines. Additionally, if λ^* need be decomposed prior to implementation, then, the methods for partitioning multiple output Boolean functions can be used for that purpose.

The separate consideration of the next-state and output functions leads to the less time and memory consuming computations than the joint consideration.

The notions and theorems presented in this report have straightforward practical interpretations and they constitute a theoretical basis for the algorithms and programs, that can be used for computing the different sorts of decompositions for sequential machines.

REFERENCES

[1] M.A. Arbib: Theories of abstract automata, Englewood Cliffs, N.J.: Prentice-Hall, 1969.

[2] G. Cioffi, E. Constantini, S. de Julio: A new approach to the decomposition of sequential systems, Digital Processes, vol.3, p. 35-48, 1977.

[3] G. Cioffi, S. de Julio, M. Lucertini: Optimal decomposition of sequential machines via integer nonlinear programming: A computational algorithm, Digital Processes, vol.5, p. 27-41, 1979.

[4] A.D. Friedman, P.R. Menon: Theory and design of switching circuits, Woodland Hills, Cal.: Computer Science Press, 1975.

[5] A. Ginzburg: Algebraic theory of automata, N.Y.: Academic Press, 1968.

[6] J. Hartmanis: On the state assignment problem for sequential machines I, IRE Trans. Electron. Comput., vol.EC-10, p. 157-165, 1961.

[7] J. Hartmanis, R.E. Stearns: On the state assignment problem for sequential machines II, IRE Trans. Electron. Comput., vol.EC-10, p. 593-603, 1961.

[8] J. Hartmanis: Loop-free structure of sequential machines, Inf. & Control, vol.5, p. 25-43, 1962.

[9] J. Hartmanis: Further results on the structure of sequential machines, J. Assoc. Comput. Mach., vol.10, p. 78-88, 1963.

[10] J. Hartmanis, R.E. Stearns: Some danger in state reduction of sequential machines, Inf. & Control, vol.5, p. 252-260, 1962.

[11] J. Hartmanis, R.E. Stearns: Pair algabra and its application to automata theory, Inf. & Control, vol.7, p. 485-507, 1964.

[12] J. Hartmanis, R.E. Stearns: Algebraic structure theory of sequential machines, Englewood Cliffs, N.J.: Prentice-Hall, 1966.

[13] W.M.L. Holcombe: Algebraic Automata Theory, Cambridge University Press, 1982. (Cambridge studies in advanced mathematics, vol.1).

[14] Y. Hou: Trinity algebra and full-decompositions of sequential machines, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1986.

[15] Y. Hou: Trinity algebra and its application to machine decompositions, Information Processing Letters, vol.26, p. 127-134, 1987.

[16] L. Jóźwiak: The full-decomposition of sequential machines with the state and output behaviour realization, Eindhoven University of Technology Research Reports, Eindhoven University of Technology, The Netherlands, January 1988. EUT Report 88-E-188

[17] Yu.V. Pottosin, E.A. Shestakov: Approximate algorithms for parallel decomposition of automata, Autom.Contr. & Comput.Sci.,vol.15, No 2, p. 24-31, 1981. (Translation of: Avtom. & Vytchisl.Techn.).

[18] Yu.V. Pottosin, E.A. Shestakov: Decomposition of an automaton into a two-component network with constraints on internal connections, Autom.Contr. & Comput.Sci., vol.16, No 6, p. 24-31, 1982.

[19] Yu.V. Pottosin: Decompositional method for coding the states of a parallel automaton, Autom. Contr. & Comput. Sci., vol.21, p. 78-84, 1987.

[20] M. Yoeli: The cascade decomposition of sequential machines, IRE Trans. Electron. Comput., vol.EC-10, p. 587-592, 1961.

[21] M. Yoeli: Cascade-parallel decompositions of sequential machines, IEEE Trans. Electron. Comput., vol.EC-12, p. 322-324, 1963.

Eindhoven University of Technology Research Reports
Faculty of Electrical Engineering

ISSN 0167-9708
Coden: TEUEDE

(205) Butterweck, H.J. and J.H.F. Ritzerfeld, M.J. Werter
FINITE WORDLENGTH EFFECTS IN DIGITAL FILTERS: A review.
EUT Report 88-E-205. 1988. ISBN 90-6144-205-2

(206) Bollen, M.H.J. and G.A.P. Jacobs
EXTENSIVE TESTING OF AN ALGORITHM FOR TRAVELLING-WAVE-BASED DIRECTIONAL DETECTION AND PHASE-SELECTION BY USING TWONFIL AND EMTP.
EUT Report 88-E-206. 1988. ISBN 90-6144-206-0

(207) Schuurman, W. and M.P.H. Weenink
STABILITY OF A TAYLOR-RELAXED CYLINDRICAL PLASMA SEPARATED FROM THE WALL BY A VACUUM LAYER.
EUT Report 88-E-207. 1988. ISBN 90-6144-207-9

(208) Lucassen, F.H.R. and H.H. van de Ven
A NOTATION CONVENTION IN RIGID ROBOT MODELLING.
EUT Report 88-E-208. 1988. ISBN 90-6144-208-7

(209) Jóźwiak, L.
MINIMAL REALIZATION OF SEQUENTIAL MACHINES: The method of maximal adjacencies.
EUT Report 88-E-209. 1988. ISBN 90-6144-209-5

(210) Lucassen, F.H.R. and H.H. van de Ven
OPTIMAL BODY FIXED COORDINATE SYSTEMS IN NEWTON/EULER MODELLING.
EUT Report 88-E-210. 1988. ISBN 90-6144-210-9

(211) Boom, A.J.J. van den
H_∞-CONTROL: An exploratory study.
EUT Report 88-E-211. 1988. ISBN 90-6144-211-7

(212) Zhu Yu-Cai
ON THE ROBUST STABILITY OF MIMO LINEAR FEEDBACK SYSTEMS.
EUT Report 88-E-212. 1988. ISBN 90-6144-212-5

(213) Zhu Yu-Cai, M.H. Driessen, A.A.H. Damen and P. Eykhoff
A NEW SCHEME FOR IDENTIFICATION AND CONTROL.
EUT Report 88-E-213. 1988. ISBN 90-6144-213-3

(214) Bollen, M.H.J. and G.A.P. Jacobs
IMPLEMENTATION OF AN ALGORITHM FOR TRAVELLING-WAVE-BASED DIRECTIONAL DETECTION.
EUT Report 89-E-214. 1989. ISBN 90-6144-214-1

(215) Hoeijmakers, M.J. en J.M. Vleeshouwers
EEN MODEL VAN DE SYNCHRONE MACHINE MET GELIJKRICHTER, GESCHIKT VOOR REGELDOELEINDEN.
EUT Report 89-E-215. 1989. ISBN 90-6144-215-X

(216) Pineda de Gyvez, J.
LASER: A LAyout Sensitivity ExploreR. Report and user's manual.
EUT Report 89-E-216. 1989. ISBN 90-6144-216-8

(217) Duarte, J.L.
MINAS: An algorithm for systematic state assignment of sequential machines - computational aspects and results.
EUT Report 89-E-217. 1989. ISBN 90-6144-217-6

(218) Kamp, M.M.J.L. van de
SOFTWARE SET-UP FOR DATA PROCESSING OF DEPOLARIZATION DUE TO RAIN AND ICE CRYSTALS IN THE OLYMPUS PROJECT.
EUT Report 89-E-218. 1989. ISBN 90-6144-218-4

(219) Koster, G.J.P. and L. Stok
FROM NETWORK TO ARTWORK: Automatic schematic diagram generation.
EUT Report 89-E-219. 1989. ISBN 90-6144-219-2

(220) Willems, F.M.J.
CONVERSES FOR WRITE-UNIDIRECTIONAL MEMORIES.
EUT Report 89-E-220. 1989. ISBN 90-6144-220-6

(221) Kalasek, V.K.I. and W.M.C. van den Heuvel
L-SWITCH: A PC-program for computing transient voltages and currents during switching off three-phase inductances.
EUT Report 89-E-221. 1989. ISBN 90-6144-221-4