# A CONCEPT FOR THE EDITING OF HARDWARE RESULTING IN AN AUTOMATIC HARDWARE-EDITOR

Franz J. Rammig
University of Dortmund
Federal Republic of Germany

## Abstract

It is the aim of this paper to investigate a system, which, with no manual or mechanical interference, permits the building, changing, processing and destruction of real (not simulated !) digital hardware.
It can be seen, that by the above nothing else than a hardware-editor is outlined. Therefore first a "mini-model" of a usual text-editor is given within this paper. Then this model is transferred to problems of the editing of hardware.
The main problem in particular, i.e. the problem of keeping the timing under control, will be discussed on the basis of an exact algebraic model of the time-behaviour of digital circuits.
The result of these discussions will be the concept of an automatic hardware-editor.
Finally a realization of this concept built by the author within the University of Dortmund, W. Germany, will be discussed.

## A "Mini-Model" of a usual text-editor

A text-editor may be described by a triple (A,C,F) where:
A is an alphabet (e.g. the set of representable ASCII-characters) and
C is a finite, non-empty set of commands.
The semantics of a command is given by a function associated with each command:

$$\bigwedge_{c\varepsilon C} f_c : A^* \rightarrow A^*$$

which defines how a string $s\varepsilon A^*$ is manipulated by the command $c\varepsilon C$.
Let $\underline{F} := \{f_c | c\varepsilon C\}$ be the set of all those semantic defining functions.

Given a string $s\varepsilon A^*$ and a sequence of commands

$$\tilde{c} := c_n \ldots c_o \;,\; \bigwedge_{o\leq j\leq n} c_j \varepsilon C$$

$f_{\tilde{c}}(s) := f_{c_n}(\ldots(f_{c_o}(s))\ldots)$ is called an editing of a string s.

Due to the fact that you have only a fixed finite memory available, strings may be represented as elements of $A^n$ istead of $A^*$ where n is the number of memory-cells available.
This may be done by introducing a special symbol "end of text" (EOT). Now, e.g., the string abcd $\varepsilon A^*$ may be represented by any element of the set
$\{abcdEOTx' | x' \varepsilon A^{n-4}\}$.

## A "Mini-Model" of a Hardware-Editor

Given an ordered set of modules $M := \{m_1, m_2, \ldots, m_k\}$, each with an ordered set of inputs and outputs $IO_{m_i} := \{i_{m_i,1}, \ldots, i_{m_i,r}, o_{m_i,1}, \ldots, o_{m_i,s}\}$, an ordered set of pins
$P := \{p_1, \ldots, p_n\}$ and a one to one mapping
$b: \{io_{m_i,j} | io_{m_i,j} \varepsilon IO_{m_i} \wedge m_i \varepsilon M\} \rightarrow P$,
a hardware-editor may be modelled in a similar manner:
Let R be the maximal nonreflective antisymetric relation on P, i.e.

$R := \{(p_i,p_j) \varepsilon PxP | i>j\}$ and $k := (n^2-n)/2 = (R)$.

Let HA := {w,u} (w stands for "wired", u for "unwired") be the alphabet,
HC the finite non-empty set of commands, each with the semantic defining function

$f_{hc} : HA^k \rightarrow HA^k$ and $\underline{HF} := \{f_{hc} | hc\varepsilon HC\}$.

Then (HA,HC,HF) defines a hardware-editor.
Relative to the set of modules M and the mapping b any circuit is uniquely defined by a "string" $c\varepsilon HA^k$.

As module-outputs may not be connected together this may further be simplified:
Let (I,O) a partition on P, I being the ordered set of module-inputs, $I := \{i_1, \ldots, i_k\}$, O the ordered set of module-outputs, $O := \{o_1, \ldots, o_e\}$ then HA can be replaced by $HA' := \{1,2,\ldots,e\}$, HC by a set of commands HC' with a set of semantic describing functions HF' with:

$$\bigwedge_{hf'\varepsilon HF'} hf' : HA'^{k'} \rightarrow HA'^{k'}.$$

Again, relative to a given set of modules and a mapping b, any circuit c is uniquely defined by a "string" $c'\varepsilon HA'^{k'}$.

## From description to construction

From the above discussion one could imagine that the problem of building an automatic hardware-editor is a triviale one; the problem should be solved by providing k' selectors "1 out of e", connecting the outputs of the modules with the inputs of all selectors and the outputs of the selectors with the inputs of the modules. All editing would be done by manipulating the data on the address-lines of the selectors.
In fact, this will be the heart of our hardware-editor.
But unfortunately some nontrivial problems arise:

a) There is a severe distortion of the time-behaviour of the circuit because of the time-behaviour of the selectors and interfacing-logic. Thus the behaviour of a circuit built in this manner will be rather different from the behaviour of a hard-wired one.
b) Even this irrelevant behaviour of the circuit cannot be readily observed because the timing is controlled by the circuit instead of being dictated by an observation mechanism.

Therefore in addition to an interconnection mechanism a time-controlling mechanism must also be provided.

To do this, one must have available an exact model of the time-behaviour of digital circuits.

## Simplified Quasi Real Boolean Functions

Besides more global models [1,2,3,4] such a model is given by the so called "Quasi Real Boolean Functions"[5]. Those functions allow a very precise description of the logic and the time-behaviour of digital circuits. In this paper a slightly modified approach will be presented. The modification has been done to allow an easier implementation of a hardware-editing-system based upon this model. On the other hand the modelling power is decreased very little by the modification.

As we want to describe the time-behaviour as well as the logic behaviour we must treat the set of functions

$\Theta := f:(\sigma_T)^n \to (\sigma_T)^m \quad n,m \in \mathbb{N}$

mapping an n-tupel of signal-sets $\sigma_T$ into an m-tupel of these signal-sets, instead of treating Boolean Functions $b : \{0,1\}^n \to \{0,1\}^m$.

By a signal-set we mean the set $[0,1]^T$, where T is the set $\mathbb{R}$ of real numbers or a denumberable set with the ordering and metric adapted from $\mathbb{R}$.

In the following we restrict ourselves on signal-sets $_{T'}$, with

a) T' denumberable

b) $|t_{i+1}-t_i| = |t_i-t_{i-1}|$ for all $i\in\mathbb{Z}$ (equidistancy)

c) $a_{t_i} \in\{0,1\}$ for all $i\in\mathbb{Z}$

and an functions mapping those sets.

Def 1 A function $h : (\sigma_{T'})^n \to (\sigma_{T'})^m$ is called Ideal Boolean Function iff

$$\bigvee_{h':\{0,1\}^n\to\{0,1\}^m} \ \bigwedge_{x\in(\sigma_{T'})^n} \ \bigwedge_{t_o\in T'} pr_{t_o}(h(x))=h'(pr_{t_o}(x))$$

where $pr_{t_o}$ is the projection at the time $t_o$.

With $h'_j$ we will denote the j-th image-component.

By an inertial-delay-specification we mean a triple of finite integers, ID := {up,dn,in}, "up" specifying the delay of an "up"-transition, "dn" of a "down"-transition and "in" specifying the minimum number of consecutive points of time for which a signal must have an identical constant value to be accepted (inertia). In addition "in" must be less than the minimum of "up" and "dn".

Def 2 Let ID := {up,dn,in} be an inertial-delay-specification, md := max{up,dn}. With respect to ID we call $idf : \sigma_{T'} \to \sigma_{T'}$ inertial-delay-function iff

$$\bigvee_{idf':\{0,1\}^{md+in}\to\{0,1\}} \ \bigwedge_{x\in\sigma_{T'}} \ \bigwedge_{t_o\in T'} pr_{t_o}(idf(x)) = idf'(pr_{t_o}(x),\dots,pr_{t_{o-(md+in)}}(x))$$

where

$$idf'(pr_{t_o}(x),\dots,pr_{t_{o-(md+in)}}(x)) = \begin{cases} pr'_{t_{o-up}}(x) & \text{if } pr_{t_o}(x) = 1 \\ pr'_{t_{o-dn}}(x) & \text{if } pr_{t_o}(x) = 0 \end{cases}$$

and $pr'_{t_{o-\{{up \atop dn}\}}}(x) \neq pr_{t_{o-\{{up \atop dn}\}}} \iff$

$$\bigvee_{t_o>t_k>t_{o-\{{up \atop dn}\}}} \ \bigwedge \ \bigvee_{t_{o-\{{up \atop dn}\}}>t_j>t_{o-(\{{up \atop dn}\}+in)}} \quad \text{with}$$

$$pr_{t_k}(x)\neq pr_{t_{o-\{{up \atop dn}\}}} \wedge pr_{t_o}(x)\neq pr_{t_{o-\{{up \atop dn}\}}}$$

An inertial-delay-specification is called pure-delay-specification if "in" has the value 0. An inertial-delay-function defined with respect to a pure-delay-specification is called pure-delay-function.

Def 3 Let $ID_i = \{iup,idn,in\}_i =: \{iup_i,idn_i,in_i\}$, $i=1,\dots,n$, be inertial-delay-specifications and $PD_j = \{oup,odn,0\}_j = \{oup_j,odn_j,0\}$, $j=1,\dots,m$, be pure-delay-specifications.

Let be $udi := \max\limits_{1\le i\le n} \{iup_i,idn_i\}$,

$udo := \max\limits_{1\le j\le m} \{oup_j,odn_j\}$,

$uin := \max\limits_{1\le i\le n} \{in_i\}$,

$his := udi + udo + uin$.

A function $q : (\sigma_{T'})^n \to (\sigma_{T'})^m$ is called a Simplified Quasi Real Boolean Function (SQRBF) iff

$$\bigvee_{q':\{\{0,1\}^n\}^{his}\to\{0,1\}^m} \ \bigwedge_{x\in(\sigma_{T'})^n} \ \bigwedge_{t_o\in T'} pr_{t_o}(q(x)) = q'(pr_{t_o}(x),\dots,pr_{t_{o-his}}(x)).$$

q' is defined by $(q'_1,q'_2,\dots,q'_m)$ where

$q'_j(pr_{t_o}(x),\dots,pr_{t_{o-his}}(x)) =$

$odf_j(h'_j(idf_1(pr_{t_o}(x),\dots,pr_{t_{o-(ud_i+in_i)}}(x)),\dots,idf_n(pr_{t_o}(x),$
$\dots,pr_{t_{o-(ud_i+in_i)}}(x))),\dots,h'_j(idf_1(pr_{t_{o-udo}}(x),\dots,pr_{t_{o-his}}(x)),\dots,$
$idf_n(pr_{t_{o-udo}}(x),\dots,pr_{t_{o-his}}(x))))$,

h' with the image-components $h'_1,\dots,h'_m$ being the function used above to define Ideal Boolean Functions, $idf_i$ being the inertial-delay-function defined with respect to $ID_i$ and $odf_j$ being the pure-delay-function defined with respect to $PD_j$.

If we want to describe the behaviour of a module relative only to the input-values at a single point of time we have to replace the function by an automaton. It can easily be seen that the following automaton describes the behaviour of a SQRBF:

$A := (X,Y,S,\delta,\lambda)$ with

$X := \{0,1\}^n$

$Y := \{0,1\}^m$

$S := si_{1,0},si_{1,1},\dots,si_{1,udi},si_{2,0},\dots,si_{n,udi},\dots,$
$so_{1,0},\dots,so_{1,udo},\dots,so_{m,udo}$

$\delta$ is defined by :

1) $\bigwedge\limits_{1\le j\le m} \ \bigwedge\limits_{0\le k\le udo} \delta(so_{j,k}) = so_{j,k-1}$

2) $\bigwedge\limits_{1\le i\le n} \ \bigwedge\limits_{0\le k\le in_i} \delta(si_{i,k}) = si_{i,k-1}$

3) $\bigwedge\limits_{1\le i\le n} \delta(si_{in_i+1}) = \begin{cases} \overline{si_{i,in_i}} & \text{if } \bigvee\limits_{0\le k\le in_i} si_{i,k}\neq si_{i,k-1} \wedge si_{i,in_i+1}\neq si_{i,in_i} \\ si_{i,in_i} & \text{otherwise} \end{cases}$

4) $\bigwedge\limits_{1\le i\le n} \ \bigwedge\limits_{in_i\le k\le udi} \delta(si_{i,k})=si_{i,k-1}$

5) $\bigwedge\limits_{1\le i\le n} \delta(si_{i,0}) = x_i$

6) $\bigwedge\limits_{1\le j\le m} \delta(so_{j,1})=f'_j(a_1,\dots,a_n)$ with $\bigwedge\limits_{1\le i\le n} a_i=\begin{cases} si_{i,upi_i} & \text{if } x_i=1 \\ si_{i,dni_i} & \text{if } x_i=0 \end{cases}$

$\lambda$ is defined by

$$\bigwedge_{1\le j\le m} \lambda_{|Y_j} = \begin{cases} so_{j,upo_j} & \text{if } so_{j,1} = 1 \\ so_{j,dno_j} & \text{if } so_{j,1} = 0 \end{cases}$$

Note that the behaviour of this automaton is undefined with respect to word-prefixes of a certain length, as we allow each state to be an initial one. One sees immediately that this automaton has an extremly simple structure which can be realized with a set of shift-registers and very simple additional logic.
Now this simple automaton allows us to get the time under control.
The basic idea is to represent only the logic behaviour of the modules by the modules themselves. The time-behaviour is represented by providing a circuit realizing a pure-delay-function for every module-output (="crossbar"-input) and an inertial-delay-function for every module-input (="crossbar"-output).
In this way time is under full control of the system, allowing us to stretch, distort and even to stop and restart time without any falsification of the modelling result.
Our system therefore is not only a hardware-editor but also includes an "All-channel State Analyser" without any time-limit.

## The META-46 "GOLDLAC", a realization of the above concept

At the University of Dortmund, we decided to build a system realizing the above concepts in standard TTL-technology. We have restricted ourselves to a system allowing us to construct automatically a digital circuit from a given but changeable set of modules where the sum of module-outputs must not be greater than 128 and the sum of the module-inputs not greater than 192.
From the above discussion it follows that for the basic system we need
- a "crossbar" : "128 x 192" (simplified)
- 128 times a realization of a universal pure-delay-function
- 192 times a realization of a universal inertial-delay-function.

### The "crossbar"

From the above discussion it follows, that the "crossbar" consists simply of 192 selectors "1 out of 128". Such a selector can be built very easily from standard TTL-chips using a two-level realization.
In addition, a 7-bit register is needed to store the address of the actual selected line. The actual data within these 192 7-bit registers specifies, relative to the set of modules plugged in at that moment, the circuit being built by the system.
From this it follows that, based on a given set of modules, up to $2^{1344}$ different circuits may be built without any manual interference.
Our "crossbar" consists of 24 PC-boards, each containing 8 selectors "1 out of 128" with their registers and some bus-interfacing logic.

### The realization of a universal pure delay function (see fig. 1)

We wanted to be able to assign a delay to every module-output with the aid of the system. The system should allow us to have different delays for up- and down-transitions.
We have restricted ourselves to a possible range of delay between 1 and 16 time-units. A time-unit may stand for one nano-second (for example) and is represented by one clock-cycle within the system.
Thus, a universal pure-delay-function can be built very easily out of a parallel readable 16-bit shift-register, a selector "1 out of 16", two 4-bit registers to store the actual "up"-and "down"-delay and some additional circuitry. (this circuitry consists only of 4 EXOR-gates and 4 AND-gates, as we store the bit-wise boolean difference between the "up"-delay and the "down"-delay in the register for the "up"-delay).

In our system the set of 128 pure-delay-functions is distributed on 8 PC-boards, each containing 16 pure-delay-functions.

### The realization of a universal inertial-delay-function (see fig. 2)

As we wanted to be able to assign an inertial-delay to each module-input, we had to construct a realization of a universal inertial-delay-function. This has been done by adding some circuitry for a universal inertia representation to the realization of a pure-delay-function. As this complicates the circuit, we restricted ourselves to a range for delay and inertia between 1 and 8 time-units.
The inertia is implemented by adding to a pure-delay-function the ability to store into the n-th cell of the shift-register not only the contents of the n-1-th cell but also the complemented value of this cell.
The latter is done iff
1) the n-th output of an inertia specifying demultiplexor carries a "high",
2) the n-th and the n-1-th cell carry different values,
3) any two other adjacent cells i,i-1 with i<n-1 carry different values.

In our system the set of 192 universal inertial-delay-functions is distributed on 24 PC-boards, each containing 8 such functions.

### Additional components of the META-46 "GOLDLAC"

Since circuits of interest are generally not autonomous, the inputs into a circuit under test must be representable.
We do this with the aid of an additional 128-bit register. Each of the lines going into the "crossbar" may be connected with the corresponding cell of this register instead of being connected with a module-output. This is controlled by selectors "1 out of 2" which in turn are controlled by another 128-bit register.
Secondly, observation of the automatically built circuit must be possible, of course. For this purpose, an arbitrary subset of the 128 lines going into the "crossbar" can be selected as "signalset of interest". Any transition on one of these selected lines causes an interrupt signal that stops the machine, preserving its actual state. This state may then be read by a controlling mini-computer, which is also used to fill all the registers of the system.
For our system we use a PDP-11 as "mother-computer", coupled with the META-46 "GOLDLAC" by a parallel interface.
On this PDP-11 a small software-package that allows an interactive processing of the system is in operation.
Relative to a given set of modules the user defines a circuit by specifying the connections and assigning delays and inertias to lines. If he supplies his own set of modules, he first must describe it.
In addition he defines some circuit-inputs and a set of lines of interest. He may also change all his specifications interactively as in usual text-editors.

Having specified his circuit, he loads and starts the system by a simple command. At any time when one of his "signals of interest" changes its value, the values of all "signals of interest" are outputted on the uer's terminal together with the system-time (simulated time). The user may now simply register the new values or react by changing the input-values or the circuit itself.
The complete dialoque is recorded on a log-file.

Future software-packages will immediately process circuit-descriptions in a Computer Hardware Description Language (DIGITEST [6]) and will allow single user time-sharing for the simulation of larger circuits as well as multi user time-sharing for a more economic use of the system.

## References

/1/ Beister, J.
A Unified approach to combinational hazards
IEEE ToC C-23 (1974), pp 566-575

/2/ Brzozowski, J. A. & Yoely, M.
Digital Networks
(Prentice-Hall, 1976)

/3/ Miller, R. E.
Switching Theory, Vol. II
(J. Wiley & Sons, New York, 1965)

/4/ Noguchi, A. et al.
Mathematical Theory of asynchronous circuits I-III
(Waseda University Press, Tokyo 1963-1966)

/5/ Rammig, F. J.
Quasi Reale Boolesche Funktionen: Ein Versuch, zeitliche Effekte physikalisch realisierter Bauteile zu algebraisieren
Digital Processes, 2 (1976) pp. 27-45

/6/ Rammig, F. J.
Der Übersetzer DIGITEST (Version 2.1)
Bericht Nr. 8 der Abteilung Informatik der Universität Dortmund (1975)

Fig. 1   Draft of the realization of a pure delay function

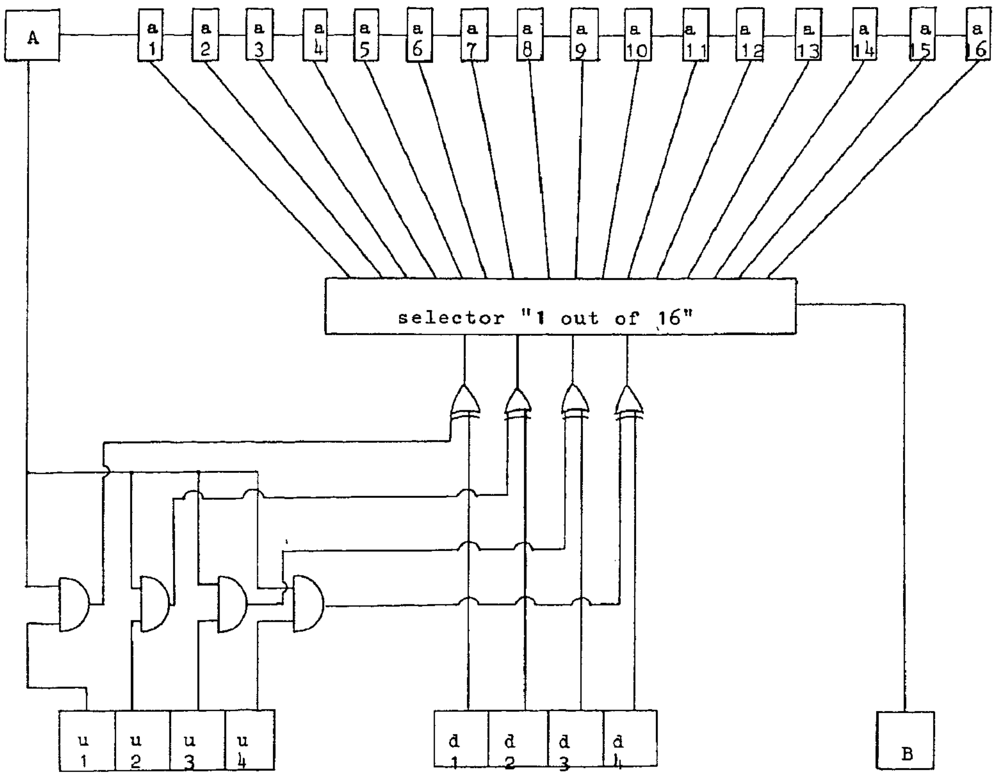


Fig. 2  Draft of the realization of an inertial delay function

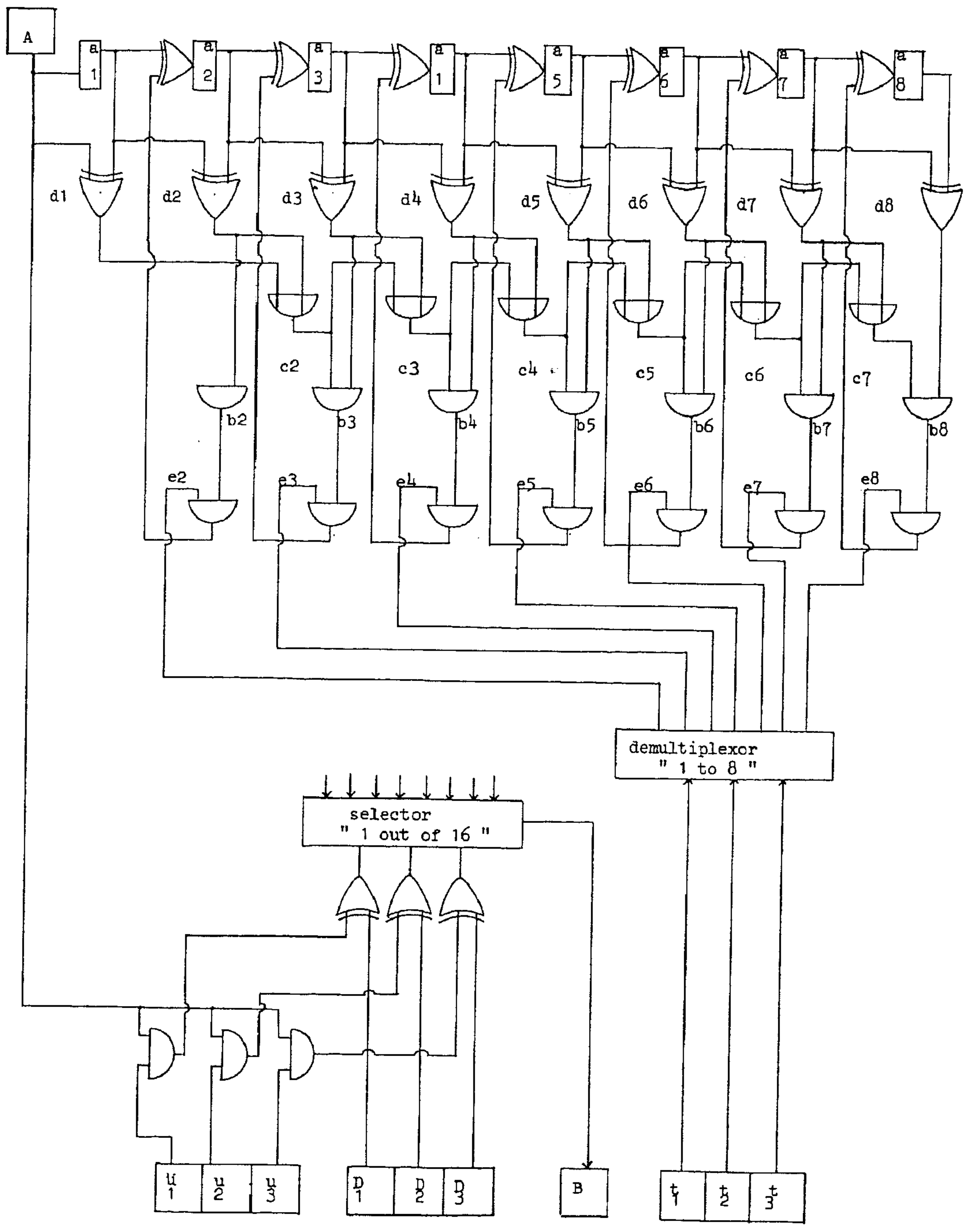
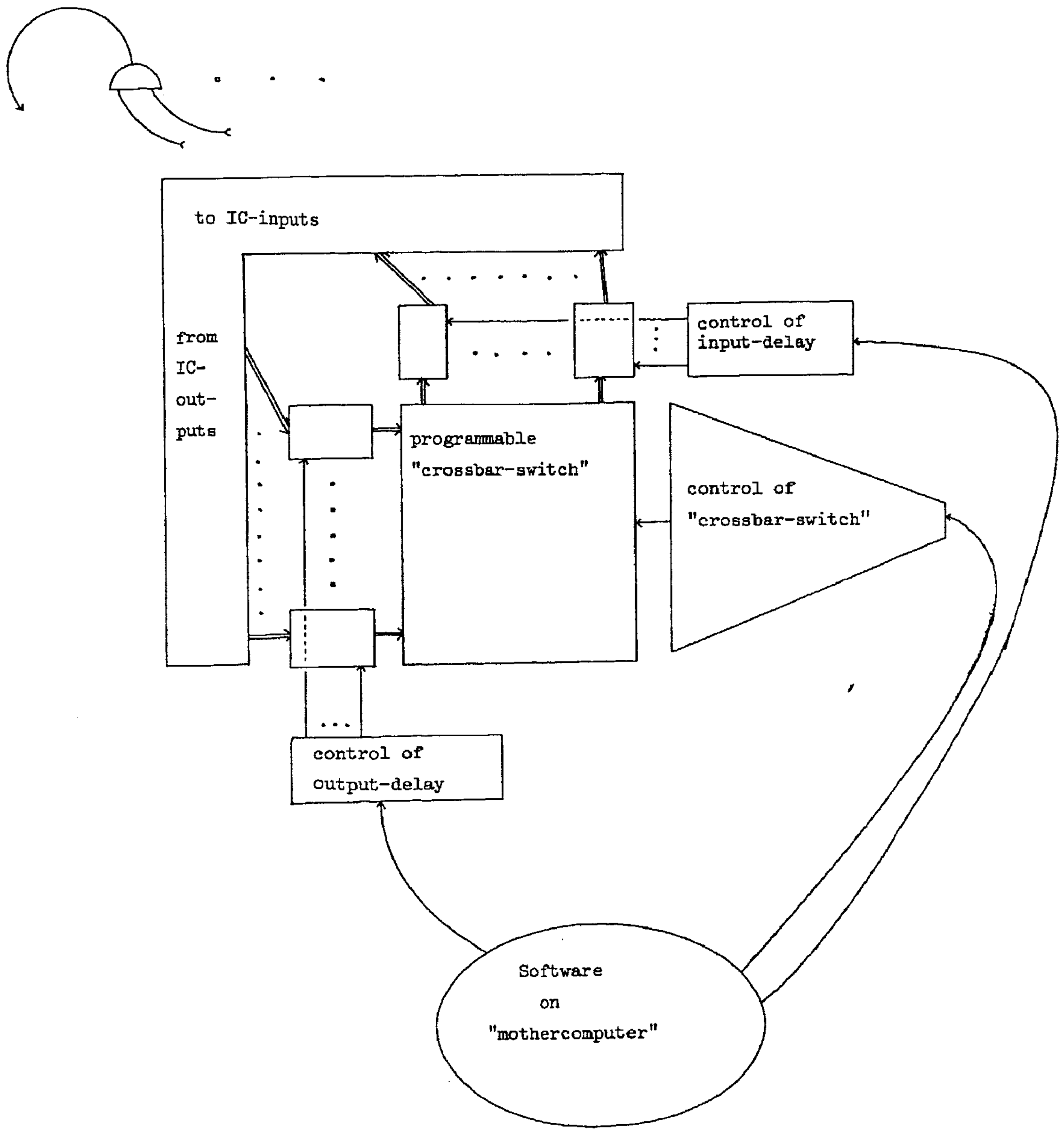


Fig. 3 Draft of the total system (only basic components)


to IC-inputs
from IC-out-puts
control of input-delay
programmable "crossbar-switch"
control of "crossbar-switch"
control of output-delay
Software on "mothercomputer"