ENGAGEMENT OF INTERACTIVE GRAPHIC TOOLS IN A CAD-SYSTEM FOR DIGITAL UNITS

Dr.E.Hörbst Project Manager
R.Prechtl Design Engineer
B.Will Systemanalyst

Siemens AG, Forschungslabors
München
Germany

This work has been sponsored by the Data Processing Program (Sign DV 2.03) of the Federal Department of Research and Technology of the FRG. The authors alone are responsible for the contents.

Abstract: The paper describes a graphical system which was developed at Siemens and the use of this system for the design automation of digital processing units.

1. Introduction

Program systems have been used for a long time now in an effort to automate the business of designing and developing electrical circuits. However, these program systems have not been able to automate the process entirely. This is expecially the case with very complex circuits and large data volumes, where the great many bottlenecks within the system oblige the development engineer to intervene. This intervention invariably takes the form of a dialogue with the computer. For a long time attempts were made to conduct the dialogue by means of lists and tables, but this "computer-oriented" dialogue approach conflicts with the working habits of the development engineer who draws his circuits. Nowadays, however, cheaper graphics hardware and, more especially, improved software systems make it possible to conduct a "user-oriented" dialogue using graphics. This paper describes a graphic system and its use in a CAD system. Particular attention is devoted to the dialogue.

2. The graphic methods base - GMB

To tackle the various tasks arising within a CAD system, graphical data processing must meet certain requirements. Since the individual cases where graphical data processing is applied clearly cannot be solved by different special systems, a system must be developed which has the capability to handle every application. This consideration produced four important criteria which exercised considerable influence on the development of the graphic system:[1]

- The system must be capable of serving the needs of a large number of different graphics hardware facilities without the interface to the user software having to be changed.
- The system must be capable of running on various computer systems (mini- and maxicomputers) and must be easily portable across different computers.
- The maintenance of all graphic data must be handled by the system and must not impose extra burdens on the programmer who prepares the user software.
- The programming of the graphics and the interactive dialogue must be done in a familiar language and be easy to learn.

2.1. Independence from graphics hardware and computer system

Over the last few years there has been a sharp increase in the amount of graphics hardware available. Special devices have been developed for the most varied applications and these are now on the market in great abundance and the a reasonable price. This hardware is offered by a wide variety of manufacturers and unfortunately there has so far been no standardization of the hardware interface or of the software.[2,6]

The use of different graphics terminal equipment in a large CAD system is both desirable and necessary, but is only possible at great development costs in view of the great differences in hardware and software.

The first requirement of a universally applicable graphic system, therefore, is that it must be totally independent of the graphics devices.

This means that the interface to the user software must possess standard coordinates, standard graphics calls and a device-independent data structure.

These characteristics alone, however, are not in themselves sufficient to free the system from dependence on the hardware. Rather, the system itself must be independent. This means that the separate program modules in the system must be prepared once only and be valid across the whole device range. Only in this way is it possible to ensure the avoidance of excessive redundancy demands on computer storage space. Such a system allows the range of devices in use to be expanded at any time without the system itself needing to be modified.

In such a system, the individual device-specific characteristics are first taken into account at a physical level in the form of code conversion. Very often this adaptation can be made by means of hardware modules (adapters), in which case use is made of microprocessors and microprograms.
Figure 1 shows the schematic layout of the GMB system. As can be seen, the device adaptations can be implemented with hardware. Where this is not the case, a code table must be prepared in Assembler language. All other programs of the system are written in FORTRAN and can therefore be easily implemented of different computer systems.

2.2. Maintenance of the graphic data

The maintenance of the graphic data plays an important role in interactive graphic dialogue programs. This data can be very extensive and linked with physical or mechanical files by means of pointers. Rapid access to the data is an essential requirement for the dialogue. The modifications to the data in consequence of the dialogue must be taken into account instantly. Various models have been devised in recent times for this form of data maintenance [2]. An important factor here is that the maintenance of the data must be dealt with by the system and must not be allowed to make demands on the programmer. The reorganization and overflow of the individual lists and files must be handled by the system without the need for a user program. All the data and pointer lists, both for the master file and for the different image files of the individual devices, are maintained centrally in the GMB. The total device-independence of the system greatly simplifies this central maintenance function.

2.3. Calls of the GMB

The calls of the GMB are formulated in such a way that they can be easily understood and quickly learned by the programmers of the user software. The format of the calls corresponds to FORTRAN conventions. They are subdivided into

- management calls
- graphic calls
- dialogue calls.

Figure 2 shows examples of each type of call.

3. The CAD system REGENT-D

This CAD system enables the development engineer to check digital circuits by simulation of the logic functions, the delay times of signals and the effect of faults before the design is ever committed to hardware.

The development data which results is processed further by the system to produce test and manufacturing data. This, in turn, provides a source from which the documents for production and test departments are derived.

The modular structure of REGENT-D is shown in Figure 3.

REGENT-D's software is subdivided into several system modules which are interlinked via standardized data interfaces. Each system module is assigned unique functions which must be executed in the course of a product run through the system.

An interactive satellite system, on which the GMB is implemented, is used for collection and updating of the data and for the output and analysis of results.

All other activities, namely the logic and delay time simulations, the fault simulation, the design programs for board population and connecting path finding and the generation of all documents, are handled in the batch-processing mode on a lareg-size computer.

The interactive satellite system is hooked up to the large computer via a data communication link.

However, this connection is only ever activated briefly when data is required for batch processing or when the result of a batch run is to be analysed on the screen or output via the plotter.

3.1. The libraries

Three different libraries must be available to ensure a product has a smooth run through the REGENT-D system.
The arrangement of the libraries is shown in Figure 4.

The graphics library is resident on the disk storage of the satellite system. It contains all the information on the contours of the circuit symbols, the relative coordinates of the inputs and outputs and also their assignment to the corresponding symbolic pin designations in the logic library. The latter-mentioned is resident in the data base of the large computer in a pool which contains all the development data. The library contains the description of the logic functions of the modules as well as their dynamic properties.

All mechanical master data is grouped together in the third library. This belongs to the pool of manufacturing data, which is also stored in the data base of the large computer. The data in this library describes, for example, the module dimensions, the layer structure of the multilayer boards, the position of the mounting locations, the designation of the module pins and so forth.

3.2. Data collection

The circuit diagrams contain modules and their logic interconnections. These must be input to the computer for further processing. The system allows for three different input methods for this:

- encoding and input of the circuit diagram on punched cards;
- generating the circuit diagram on a refresh storage tube;
- plotting out the circuit diagram on the digitizer.

The last method mentioned is the one most frequently applied and will now be described more closely.

The circuit diagram drawing is placed on the digitizer and plotted. This task is facilitated by a menu area which can be freely selected by the user. The menu area indicates the available module repertoire and the various dialogue commands such as "DELETE", "POSITION" etc.

Firstly, the modules are input. For this, the appropriate element on the menu area must be determined and the coordinate reference point of the module symbol must be indicated on the circuit diagram. The logic, mechanical and graphic data of the module are ascertained by the computer by means of the link with the libraries. Once all the modules have been input, the logic interconnections are entered by plotting the interconnect lines on the diagram. Finally, the lettering is entered via an alphanumeric keyboard. Troughout the operation the input values are displayed on a screen for checking purposes. Plotting errors can be immediately corrected by way of the menu command "DELETE".

Once the work is completed, the entire logic of the diagram as well as its geometry is contained in storage and is made available for the simulation.

3.3. The simulation

The REGENT-D system operates with two simulators. The first of these permits logic simulation - with correct delay times - of all or any logic networks up to and including the total complex, while the second verifies the results of the first by simulating faults in the individual pc boards making up the total complex.

Each of the simulators is specially suited to its task and so provides optimum results at all times.

3.3.1. Logic simulation. This simulation mode reproduces the behaviour of the digital circuit under conditions of unfaulted operation.[4] The result of this simulation gives the timing patterns of all the signals occurring in the circuit. The development engineer uses the logic and delay simulation for

- checking the digital circuit for formal errors,
- discovering logic and delay time errors,
- checking microprograms for logic errors,
- determining test bit pattern sequences.

3.3.2. Fault simulation. The fault simulation assists the development engineer in determining the behaviour of a pc board circuit in the event of a fault.

To this end, faults are built into the circuit model systematically by the simulator, and their effect on the input and output signals of the pc board is monitored. The input bit patterns and circuit data necessary for the fault simulation are obtained from the preceding logic simulation of the total complex.[7,8]

The fault simulation enables a check to be made as to whether

- all functions were simulated during the preceding logic simulation of the total complex,
- the circuit contains logic redundancies,
- the test bit patterns generated are suitable for detecting all possible hardware faults on the pc board.

In addition, the fault simulation produces diagnostic data which is used to prepare a fault diagnosis catalog to aid in automatic fault tracing on the circuit board tester.

3.4. Results analysis

After termination of the simulation program, the results have to be analysed by the development engineer. Just as the simulation of a circuit is intended to replace the trial model of the hardware, so a substitute must also be found for the familiar oscilloscope when it comes to the analysis of the results. The most suitable replacement for this is a screen. This allows the pulse patterns of the individual signals to be represented and checked in a similar way to the oscilloscope[5]. Forms of representation familiar to the development engineer from the oscilloscope are selected to represent the individual logic states. The various forms of representation are shown in Figure 5. The pulse patterns of the output signals are the first things to be checked on the screen. To represent the signals on the screen, the user must input the following:

- time interval
- time increment
- signal name.

The time interval defines the time within which the signal is observed, e.g. from 0 to 10,000 nanoseconds. The time increment serves to indicate the time steps at which possible pulse transitions are displayed. Assuming, for example, a time increment of 100 nanoseconds is indicated in the case of the above time interval, 100 possible pulse transitions are represented in all on the screen. As when triggering with a delayed time base, a more precise representation can be achieved by choosing a smaller time interval. Once the signal names are input, the signal patterns for each signal are shown. Figure 6 shows the output as displayed on the screen. The individual signal edges can now be checked precisely by means of a time scale which can be moved over the entire display. If an error is discovered at the output signals, it must now be pinpointed. By tracing back from the output signal, all the internal signals involved can be represented on the screen. For this, the respective signal names must be input by the user.

If a module or other circuit configuration is found whose inputs are correct but whose outputs are incorrect, the error has been located and the circuit diagram can be corrected at this point.

There are two advantages in using the screen for the analysis: Firstly, the development engineer has in front of him familiar signal patterns instead of line printer listings. Secondly, the user has the entire data volume at his disposal. In a simple dialogue he can rapidly call up and view any signal he chooses in any time interval. The data volume so far implemented comprises 10,000 signals with a time interval of 0 to 10^6 time steps. To access a signal and build it up on the screen takes less than a second.

3.5. Modification of the data

If the errors contained in the circuit diagram are found following the analysis on the screen, they can be corrected immediately via the screen by displaying the whole circuit diagram or parts of it. Wrong modules or connections can be deleted and input anew. The procedure is similar to that described in 3.2., except that here the dialogue is carried on directly via the screen without the intermediary of the digitizer. Besides the purely graphical alternation of the circuit diagram on the screen, all logic, mechanical and graphic data is corrected accordingly and rewritten to storage in a manner resembling that for the initial input. Following the error correction, the simulation program is rerun. This process is repeated until no more logic errors are found. Only then is the stored data released for further processing.

3.6. The design programs

Once the formal logic development data has been checked for freedom from error with the aid of the simulation, it is processed by a suite of programs into physical manufacturing data.

This is done for all the pc boards making up the total complex and also for its printed or wrapped wiring. The geometric data required for this operation and the technical descriptions of hardware structure are taken from the design data library.

On termination of the programs for board population and connecting path finding, the positions of the modules on the circuit boards, the arrangement of the latter in the device and the routing of all conductors and wires have been established.

3.7. Production of documentation

Several postprocessors generate artwork, assembly diagrams and control media from the manufacturing data for the production stages: etching, marking, drilling and testing of the unpopulated pc boards. ANAS test programs and diagnostic docomentation for automatic fault location on the logic tester are provided for testing and servicing the fully populated pc boards.

4. Concluding remarks

Some of the CAD programs described have been fully-developed and in use for five years. The role of REGENT-D is to integrate existing programs into a system and to close any gaps with new programs. The most significant new developments have been the graphic dialogue and the interfaces to the existing data files. This dialogue greatly increases the effectiveness and user convenience of the system as a whole. Programming work for the system has proved the GMB to be an easily acquired and highly practical programming language.

References:

/ 1 / Hörbst E., Geitz G., Gonauser M.:
'An integrated graphic system which provides the use of various graphic terminals'.
Eurocomp, Interactive Systems, October 1975, pp. 44-56

/ 2 / W.N. Newmann, R.F. Sproull:
'Principles of Interactive Computer Graphics'
McGraw-Hill 1973

/ 3 / J. Encarnacao:
'Computer Graphics'
Oldenbourg Verlag, 1975

/ 4 / Katzmayr, K.:
'BAPSI - ein Programm zur Logiksimulation',
Nachrichtentechnische Fachberichte, Bd. 49, 1974, pp. 78-82.

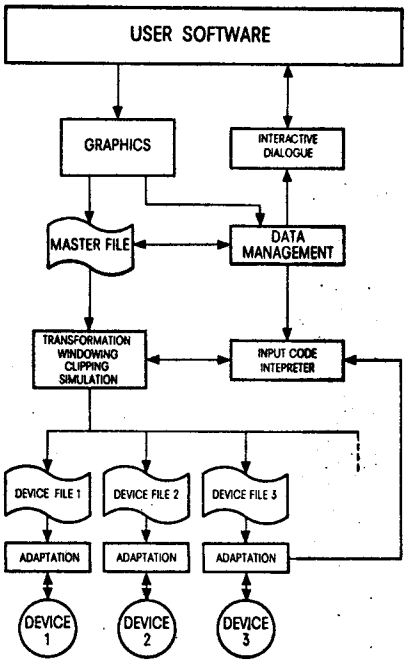
/ 5 / Hörbst E.:
'Computer-aided development of computers by means of display units',
On-line 72, Volume 2, 1972, pp. 799-817

/ 6 / Keydata Corporation, Computer Display Review,
Keydata Corporation, Yearly

/ 7 / Jost W.:
'PROFET' - ein Programmsystem zur Simulation von Hardwarefehlern in Digitalrechnern'.
Informationen Fernsprechvermittlungstechnik 8, 1972, pp. 105-117

/ 8 / Bachinger R., Jost W.:
'Nachbildung logischer Elemente bei der parallelen Simulation von Hardwarefehlern'
Angewandte Informatik, Heft 1, 1975, pp. 33-37.

Fig.1



SCHEMATIC LAYOUT OF THE GMB SYSTEM

13.4.76 Siegl N ZL kl. 663 BL1

Fig.2

MANAGEMENT CALLS

```
      CALL   INIT    (ISEN, ITR)
      CALL   BEPIC   (ISEN, X, Y, ID)
      CALL   EPIC    (ISEN, ID)
```

GRAPHIC CALLS

```
      CALL   LINE    (X, Y, IABS)
      CALL   CIRC    (X, Y, RAD, IABS)
      CALL   TEXT    (X, Y, :n)
n     FORMAT   ('EXAMPLE')
      CALL   LNTYP   (ISEN, LTYP)
```

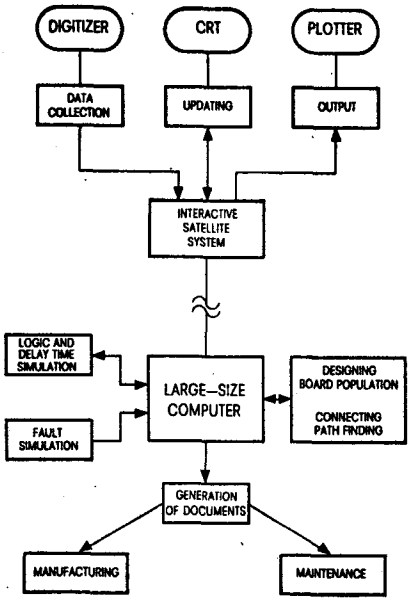
DIALOGUE CALLS

```
      CALL   DELETE  (ISEN, ID, IZU)
      CALL   MOVE    (ISEN, X, Y, IABS, ID, IZU)
      CALL   ENINT   (ISEN, INTM)
      CALL   SPECI   (ISEN, IWAIT, ISPEC, ID, NRGR, IFELD)
      CALL   ALCOP   (ISEN 1, ISEN 2)
```

EXAMPLES OF CALLS OF THE GMB

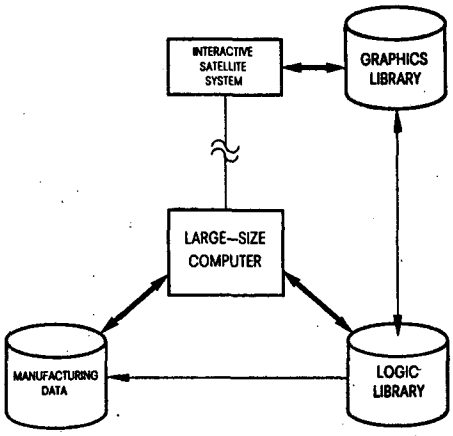
13.4.76 Siegl N ZL kl. 663 BL2

Fig.3



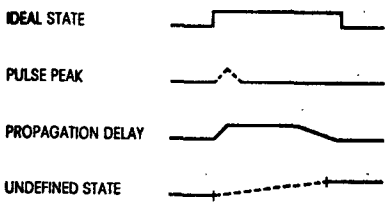
MODULAR STRUCTURE OF REGENT–D

Fig.4



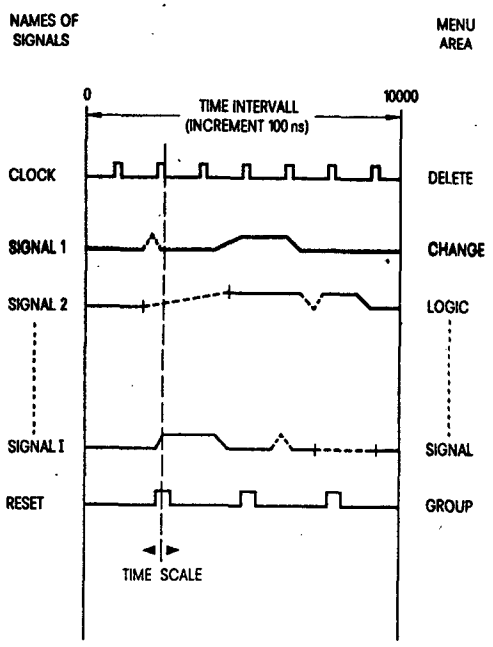
ARRANGEMENT OF THE LIBRARIES

Fig.5



REPRESENTATION OF LOGIC STATES OF THE SIGNALS

Fig.6



EXAMPLE OF SIGNAL PATTERNS ON THE SCREEN