Int. J. Engng Ed. Vol. 11, No. 2, pp. 119–124, 1995
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1995 TEMPUS Publications.

A Course on Electronic Design Within a CIM Environment*

A. H. D. BIDLOT
F. BOERSMA
Department of Electronic Engineering, Hogeschool Eindhoven, PO Box 347, 5600 AH Eindhoven, The Netherlands

This paper presents the course 'Computer-Aided Design of Electronic Systems', which is a module of the Hogeschool Eindhoven B.Eng. course 'Computer Integrated Manufacturing of Electronic Systems'. Students are trained in the design of complex and large electronic systems by using a top-down design methodology. Furthermore they are trained in engineering the systems onto a printed circuit board while coping with design and production constraints set by the CIM environment. Because the development system used in this course is integrated with other CIM subsystems for manufacturing and logistics, students gain experience in using industrial CIM-based development systems for electronic products

INTRODUCTION

TODAY, in many institutes for higher education, students in electronic engineering are trained to design electronic systems using computer-aided design (CAD) tools. At the Hogeschool Eindhoven an extra dimension has been added, namely the integration of all the data necessary to specify, design and manufacture an electronic product. Within the course 'Computer-Aided Design of Electronic Systems' students are trained in using a top-down methodology for the design and engineering of electronic systems.

The course is centered around the following components:

1. The hierarchical design of electronic systems utilizing behavioral descriptions (HDL-models) for recurrent simulation and verification.
2. The engineering of a hierarchically designed electronic system onto a printed circuit board (PCB) while coping with several design rules and production constraints.
3. The integration of this CAD development process with logistics (CAL) and manufacturing (CAM) via a CIM database (see Fig. 1).

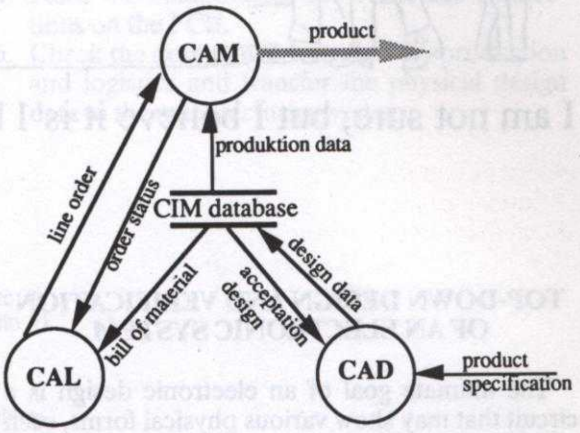

Fig. 1. Integration of CAD with the other CIM subsystems.

OBJECTIVES OF THE COURSE

This integrative course has been developed with the following objectives concerning the training of students in design and engineering:

1. Building up a practical experience in meeting quality requirements during the design and engineering cycle of an electronic product by:
   (a) The integrative and recurrent use of simulation and verification at each step during the refinement of the design. This results in meeting the functional specifications in each phase of the design cycle and leads to a product that is 'designed for test'.
   (b) Coping with logistic and production constraints during the design and engineering. This leads to an electronic product that is 'designed for manufacturing'
2. Acquiring knowledge of, and practical experience in using computer-integated facilities for the design, engineering and manufacturing of an electronic product.

The paper describes the way in which students reach these objectives by applying case studies. As a simple example the design and engineering of a digital 'Kürfurst_clock' is described (see Fig. 2).

* Accepted 1 February 1994

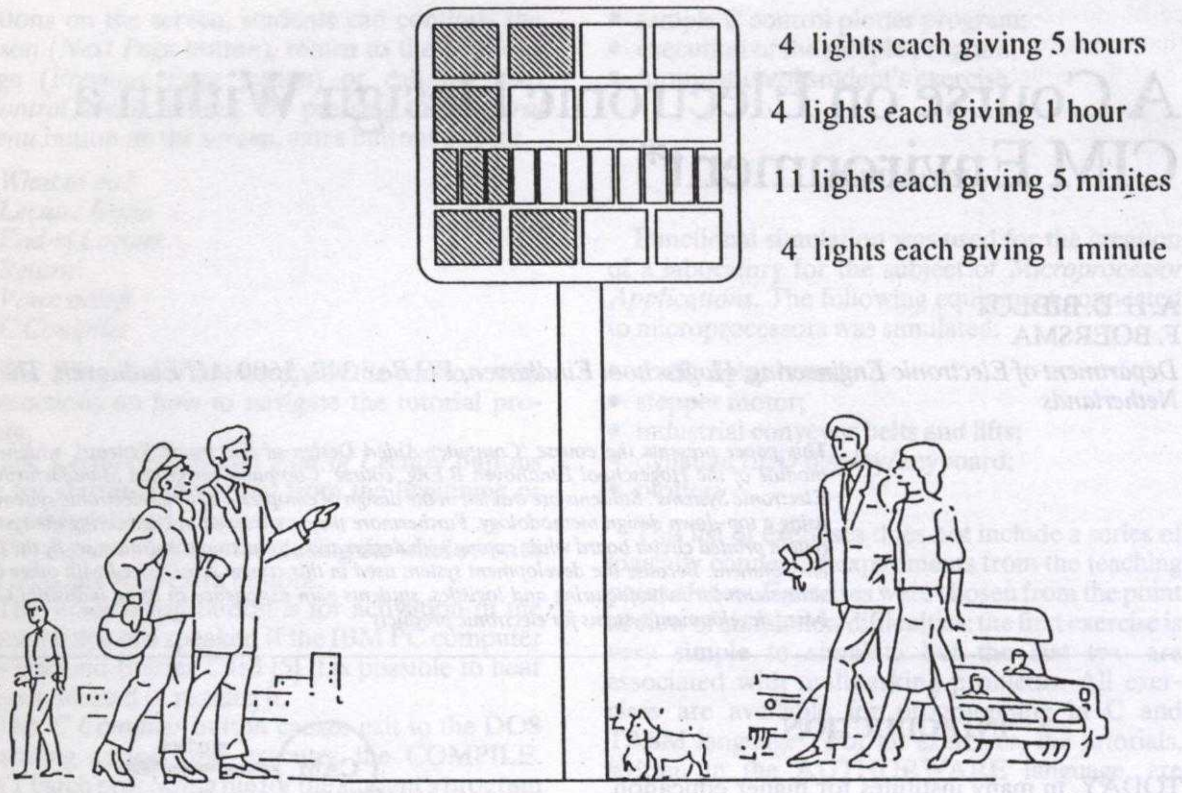
4 lights each giving 5 hours
4 lights each giving 1 hour
11 lights each giving 5 minutes
4 lights each giving 1 minute
I am not sure, but I believe it is 11.17

Fig. 2.

TOP-DOWN DESIGN AND VERIFICATION OF AN ELECTRONIC SYSTEM

The ultimate goal of an electronic design is a circuit that may show various physical forms, such as a PCB, a thin-film circuit, a chip or even combinations of different forms.

A top-down approach results in a hierarchical system design that is very powerful in controlling the complexity of large systems. Hierarchical design also has advantages for rapid parallel design of subsystems with 'clean' interfaces.

To some extent, the physical realization of the system under design is not important for an electronic designer. Especially at the first stage of the design project, he or she will most likely use a model of the system with as few details as possible. For reasons of analysis, simulation and verification of the behavior of the system early in the design cycle, such an abstract model will commonly be made by means of a hardware description language.

A formal description also has advantages with respect to communication about, and documentation of the system. As a consequence of the method of design, the hardware description language that is used has to support hierarchy, e.g. VHDL.

Hierarchical design means that the system is broken down into manageable subsystems, each subsystem only containing information necessary at the specific level of description, and hiding all, often more detailed, information at lower levels. The following levels of abstraction can be distinguished:

1. Software—an abstract behavioral model (system specification) for the entire system is used and simulated as a 'black box'.
2. Architecture—the basic design architecture is simulated at the 'block level'.
3. Logic—the design is broken down to gate-level primitives (which may be gates or functional primitives such as TTL parts). Detailed timing and functionality are verified at this stage.

The software level

At the software level, the model describes the overall behavior of the system under design. In fact the specification of the system is written in a formal language and simulated.

Usually at this level only the worst-case performance of the system is of interest. In fact it is even impossible to predict the exact timing of the system without knowing the lower-level implementation.

In practice, there already often exists some idea of the architecture that will be used. This is especially true for our students when they practice with the design of a relatively small system; in the case study of the 'Kürfurst_ clock', the architecture that will be used later on can already be recognized from the model description (see Fig. 3).

The architecture level

The next step in the design cycle is to determine an architecture for the system (see Fig. 4). For each of the building blocks in the architecture the behavior has to be described (see Fig. 5).

The system is then simulated again, using the same stimuli as during the run for the higher-level specification. If the results of the two simulations are not identical, then an error in the implementation (or in the system specification) has occurred. Since at the higher level the prediction of the timing is difficult or impossible, 'identical' here means that the sequences of the patterns are identical whilst the time intervals between the patterns are irrelevant. At high levels of description of the design even this comparison may be too restrictive.

The logic level

The process described in the previous section is repeated until the system is broken down into logic building blocks; this can be seen as a physical implementation, such as TTL parts or gate-array macros (see Fig. 6).

The final result is the hierarchical design, which is stored in the CIM database.

THE ENGINEERING OF AN ELECTRONIC SYSTEM

During the engineering process the students transform the functional design into a physical design. The latter is now also being stored in the CIM database. After that they convey the final physical design data that are relevant for production and test to the manufacturing system (CAM). Since this process takes place in a CIM environment, the students learn to cope with the design constraints regarding production and logistics, set by the CIM environment. All constraints regarding physical components, PCBs and aperture sets have been stored in the CIM database.

The engineering process can be considered as consisting of five consecutive steps:

1. Remove the hierarchy in the functional design.
2. Allocate the physical components to each of the low-level functional symbols in the functional design.
3. Choose a PCB and a set of constraints regarding the CIM environment.
4. Place the components and route the connections on the PCB.
5. Check the design on constraints for production and logistics and transfer the physical design data to the manufacturing system.

```
entity  kurfurst_clock  is
  port  ( tune                  : in Std_logic ;
          Fhour,Hour,  Min      : inout Std_logic_vector( 4  downto 1);
          Fmin                  : inout Std_logic_vector( 11 downto 1)      ) ;
end  kurfurst_clock;

Architecture  software_level  of  kurfurst_clock  is
constant tclock   : time := 30 sec;    -- The clock period is 1 minute   --
................................       -- Definition of other constants --
begin
  Internal_clock:       -- Generation of an internal clock --
     process(tune,  clk)
     begin  if tune = '0'  then    clk <= not clk after  tclock;
                           else    clk <= not clk after  ttune ;  end if; -- much faster --
     end process;
  Minit_lights:        -- Update the minit_lights on the falling edge of the clock --
     ................................
  Five_minit_lights:  -- Update the five_minit_lights on the falling edge of  min(4) --
     process(min(4))
     begin    if falling_edge(min(4))    then
                 if   fmin(11) = '0'   then fmin <= fmin(10 downto 1)&'1'    after tplh;              -- shift a '1' in --
                                       else fmin <= ( others => '0')         after tphl;  end if;  --  reset  --
           end if;
     end process;
  Hour_lights:          -- Update the hour_lights on the falling edge of  fmin(11)  --
                        -- Reset if 24 hours is reached --
     process(fmin(11))
     begin if  falling_edge(fmin(11))   then
              if hour(4) = '1'    or (fhour(4) = '1' and hour(3) = '1')  then
                   hour  <= "0000"                after  tphl;             -- reset   --
              else  hour   <= hour(3 downto 1)&'1'  after tplh;  end if;    -- shift a '1' in --
           end if;
     end process;
  Five_hour_lights:    -- Update the five_hour_lights on the falling edge of  hour(3) --
     ......................................
end software_level;
```

Fig. 3. A part of the model of the system 'Kurfürst_clock' at the software level.

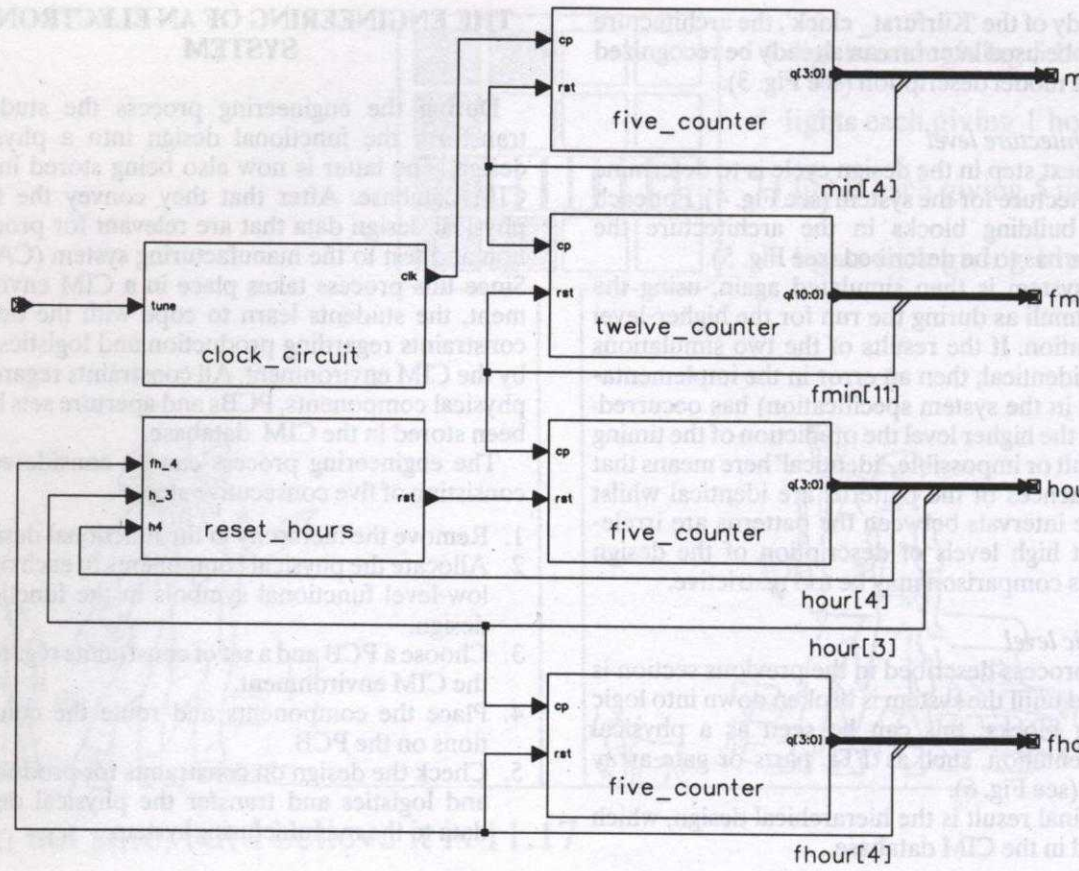

Fig. 4. The first architecture.

```
entity twelve_counter is
   port ( cp, rst              : in Std_logic ;
          q                    : inout Std_logic_vector( 10 downto 0)       ) ;
end twelve_counter;

Architecture behave of twelve_counter is
constant tplh  : time := 10 ns ;  - - The  low_to_high delay is 10 ns --
constant tphl  : time := 15 ns ;  - - The  high_to_low delay is 15 ns --
begin
   process(cp)
   begin  if  falling_edge(cp)  then
               if     rst = '0'  then q <= q(9 downto 0)&'1'  after tplh;    -- shift a '1' in --
               elsif  rst = '1'       q <= ( others => '0')   after tphl;    --  reset
               else                   q <= ( others => 'X');  end if;
          end if;
   end process;
end behave;
```

Fig. 5. The model of the 'twelve_counter' with synchronous reset.

Removing the hierarchy

The necessity for removing the hierarchy of a functional design is due to the fact that, in general, software tools (including ours) for engineering are not able to handle hierarchy in a design.

Removal of the hierarchy is a critical process. Great care must be taken not to modify the functional behavior of the design. Therefore, after having removed the hierarchy, the behavior of the functional design should be verified by simulating it over again.

Possibilities for automatic removal of hierarchy will be implemented in the future.

Allocating the physical components

To each of the functional symbols in the low-level schematic diagrams of the functional design, a physical subsystem (component) has to be assigned that is commercially available. In many cases a choice can be made from different components forming the same function.

The choice of a component depends on many production constraints such as quality requirements, overall dimensions, orientation (axial/radial), and, last but not least, logistic constraints such as cost and supplier reliability.

It is therefore necessary to assign a component to

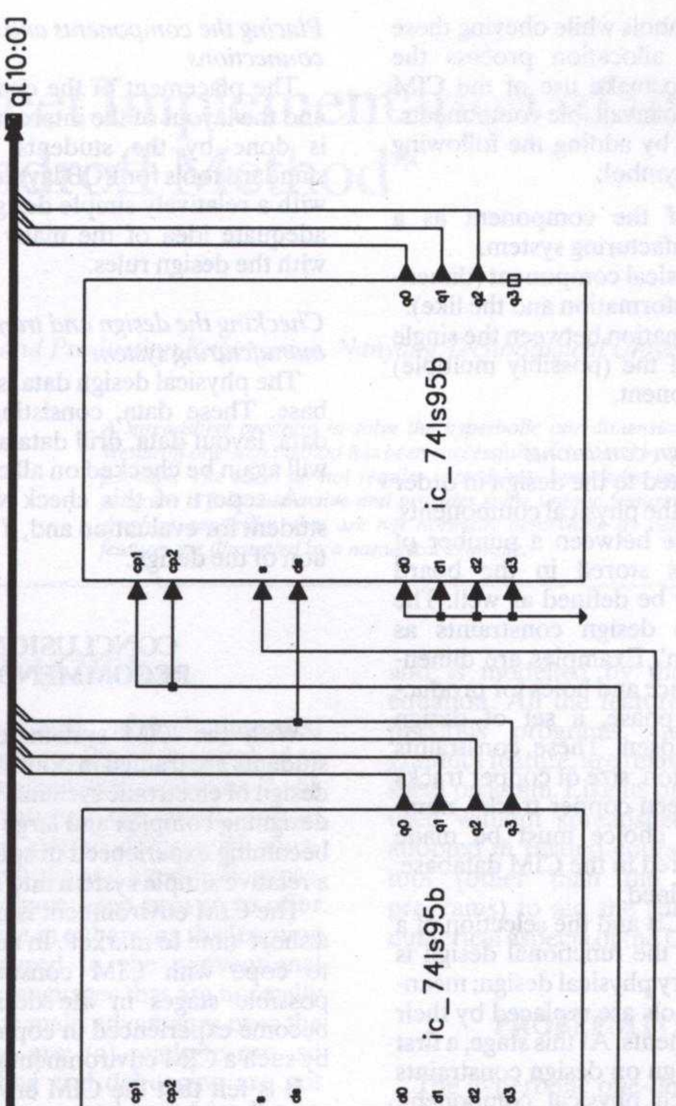

Fig. 6. The logic (implementation) level of the 'twelve_counter'

each of the functional symbols while obeying these constraints. During the allocation process the students are instructed to make use of the CIM database for an overview of available components.

The allocation is done by adding the following labels to each functional symbol:

1. The stock-number of the component as a reference for the manufacturing system.
2. A reference to the physical component (dimensional and swapping information and the like).
3. Cross-reference information between the single symbolic function and the (possibly multiple) functions of the component.

Choosing a PCB and design constraints

A PCB has to be allocated to the design in order to place and interconnect the physical components. A choice has to be made between a number of different existing boards stored in the board library; a new board may be defined as well. The PCB should match the design constraints as prescribed by 'production'. Examples are dimensions of the board, free space and holes for production tools. In the next phase, a set of design constraints has to be chosen. These constraints concern component location, size of copper tracks and pads, distances between copper tracks, number of layers. Again, a choice must be made between different sets stored in the CIM database. Also a new set may be defined.

After the choice of a PCB and the selection of a set of design constraints, the functional design is transferred to a preliminary physical design; meanwhile the functional symbols are replaced by their allocated physical components. At this stage, a first check of the physical design on design constraints is done by checking each physical component, whether it is released for production or not. This check is done by extracting a list of components from the physical design; this list is compared with the components in the CIM database.

Placing the components and routing the connections

The placement of the components on the PCB and the layout of the interconnecting coppertracks is done by the students themselves, applying standard tools for PCB layout. By experiencing this with a relatively simple design, the students get an adequate idea of the many difficulties in coping with the design rules.

Checking the design and transferring it to the manufcturing system

The physical design data is sent to the CIM database. These data, consisting of PCB placement data, layout data, drill data and a 'bill of materials' will again be checked on all constraints.

A report of this check will be mailed to the student for evaluation and, if necessary, for correction of the design.

CONCLUSIONS AND RECOMMENDATIONS

With the CIM system, discussed so far, the students are trained in both efficient and structured design of electronic systems. They are prepared for designing complex and large electronic systems by becoming experienced in splitting up the design of a relative simple system into modular subsystems.

The CIM environment is meant to contribute to a short 'time to market'. In addition, students learn to cope with CIM constraints at the earliest possible stages in the design cycle. They also become experienced in coping with limitations set by such a CIM environment in general.

It is felt that the CIM environment used in the course can be improved still further. The level of integration between the libraries in the CIM database and the consistency between them result in unnecessary and inconvenient constraints. In the near future our R&D effort will focus on a further integration of the CIM subsystems via the CIM database.

Tony Bidlot is a Senior Lecturer at the Hogeschool Eindhoven in The Netherlands. For six years he was involved in the design of integrated circuits and the development of software used by integrated circuit designers at the Eindhoven University. His current interests are integrated circuit design and the integration of designing and manufacturing electronic systems with a PCB realization.

Feike Boersma is a Senior Lecturer at the Hogeschool Eindhoven in The Netherlands. For 15 years he was involved in the design of electronic systems for computer systems at Philips and in computer applications in governmental projects. His current interests include the integration of design and manufacture of electronic systems.