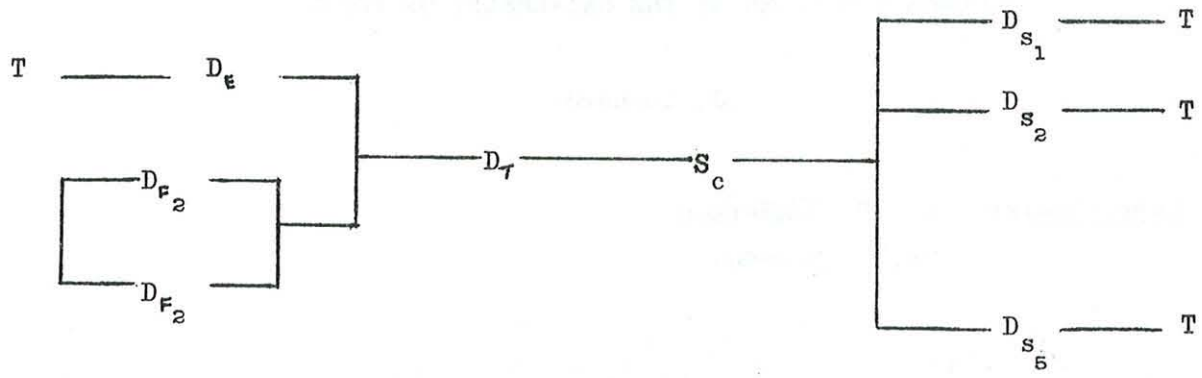
COURSE STRUCTURE AT THE UNIVERSITY OF PARIS

J. Suchard

Rapporteurs: Dr. P. Henderson
Mr. R. Snowdon

I speak from a long, large and wide experience of the integration of hardware and software courses. We began our courses 10 years ago with 30 students and now have more than 3000 students. We now offer courses varying over a wide range from applied algebra to computer technology with essential emphasis on systems engineering and systems design. We do not specifically teach computer design because for us, a system includes hardware and software.

To process so many students efficiently, offering such a wide range of courses, a very powerful processor is needed. We have chosen a pipe-line model which I shall describe using Professor Bell's notation. We have a processor P with two inputs and several outputs. At the next level of detail we have a main trunk D_T and a switch S_C with several specialised branches D_{S_1}, D_{S_2}, ..., D_{S_5} which lead to T modules (T for thesis). At the other end we have two kinds of input, high school issue and students who have followed the first two years of a university course. The latter have a better background, at least in theoretical aspects. Thus we need separate preprocessors for these two types of input. Preprocessing of high school issue is represented by the two successive steps T and D_E with a flow rate of approximately 200 students per year. The remaining students are preprocessed by the modules D_{F_1}, D_{F_2} according to background. D covers theoretical and mathematical aspects of informatics (computing science) while D_{F_2} covers the practical and pragmatic side. The flow rate of this section is 500 - 1000 students per year.



The main trunk D_T consists of a bunch of parallel processes $D_{T_1},\ldots,D_{T_{10}}$ from which the students choose from 2 to 4 to be executed in serial, parallel or a combination of both, according to external duties. An important section of our students are working full or part time in industry. These parallel branches of teaching cover widely differing topics: algebra, formal logic, syntactic analysis, compiling techniques, system engineering, system design, file systems and data base design, computer structure, data transmission, graphics and operations research. Evidently there are distributed controls and tests not represented on our diagrams which maintain the stable functioning of the system, in addition to the lateral outputs for stucents who, for many reasons, find it impossible to proceed.

What is the place of hardware in this system? It is present everywhere except in the mathematical sections, and is closely related to the programming courses. In fact hardware is essentially presented as a particular form of programming, the most basic level of it. We emphasise the fundamental mechanisms which can be implemented by wiring, microprogramming and different levels of programming. Microprogramming plays a very important role here and is used to link hardware and software concepts. It is my contention that microprogramming does not lie fully in the domain of software. There are many aspects of microprogramming ranging from the uncoded microprogramming of the Elliot 4100 series up to the IBM 360/25 which has an almost normal instruction set. The Elliot 4120 for example does not have the essential features of software system (e.g. conditional branches, subroutines etc.), having no micro instructions, simply sets of

control signals. The existence of a complete range of examples between these two extremes suggests that it is irrelevant to try to classify micro programming as either hardware or software. Indeed the distinction between hardware and software is obscure.

To introduce the students to the detailed structure and behaviour of a computer in the D_E and D_{f_2} modules, we effectively make use of an Elliot 4120. The students program on an Elliot 4130 multi programming machine where each user sees exactly a 4120 both in batch and conversational mode. The 4130 itself is too complicated an example. The students analyse the circuits (which are very straightforward), the clock system and the micro programming and so discover the details of execution of instructions in an elementary peripheral exchange.

For practical work we use a system which is in fact intermediate between real (direct) experimentation and simulation, it can be called a hardware simulator. It consists of basic modules which the student can connect together, using pre-wired supports, to demonstrate elementary computer components.

The first basic module has three active elements which are cold cathode trigger tubes, and some passive ones, resistors and capacitors principally. Two of the trigger tubes form a flip flop which can be used as SR or JK, the third is a one-shot; added to that each tube provides the pulse gate function which is the logic part of the module. All input and output is through a 10-way connector which plugs into the supports. The logic levels are ground and 110 volts and the impedance of a module contains a 1 M-ohm resistor so the problem of bad contacts is practically non existent. The problem of noise similarly disappears, the margin exceeds 40 volts, and so does that of fan-out. Thus there is unlimited flexibility in the design.

The second type of module is double decked and has four trigger tubes which form a flip flop with independant double input. This provides either double SR, double JK or SR and JK operation.

The modules make it possible to construct a great variety of models, from a simple 1 bit register to simple computers, using suitable supports.

The modules are used in courses T, D_{F_1} and D_{F_2} of the student preprocessing phase. In course T they are used to illustrate the general structure of computers and the fine details of machine operations, from the simple shift to the floating point. In modules D_{F_1} and D_{F_2} they are used essentially to introduce precise models of registers, links, buses and general operations and to demonstrate the mechanisms for transfer of information and for transfer of information and for control of execution. Finally, the modules are used in the main trunk of the course D_T for a project to design a subsystem, lasting about 6 months.

Although not directly the subject of this seminar, I would like to say a few words about the Mathematical courses in the Computer Science Department. Basically the courses are in Algebra and Algorithmics. In Algebra the students study Graph Theory, Turing Machines, and Normal Languages. They also are taught sufficient Numerical Analysis to enable them to understand and construct algorithms. Simulation techniques were used extensively in the software parts of the course and also logic simulation for hardware. We believe in simulating as exactly as possible. We have also very close connections with industry particularly with regard to design and this has helped us considerably with graph plotters and terminals. Two documents are available which describe these courses.

Finally I should like to put the following question to the seminar for its consideration - where should we place the frontier between hardware and software? I think it should be between component engineers and system engineers, in other words between those who design and build the hardware and those who use it. This is because the design of integrated circuits in 60-70% chemistry are very sophisticated.

Professor Horning I suggest a different place for the frontier between hardware and software. Hardware is what you cannot change and software is what you can change. In some computers the wiring can easily be altered so it can be looked on as software, in others the software system is so massive and difficult to alter it should be considered to be hardware.

Dr. Kinnement I disagree with Professor Suchard that integrated circuits are 60% chemistry, what is important about them is not the chemistry but the process steps through which the circuit goes.

Professor Suchard If that is the case then such a process is not design at all but simply fitting things together according to a predefined pattern.

Professor Michaelson I do not think it matters where you draw the frontier provided that the designer is aware where the interface is in a particular situation. Different people at different times will want to draw the frontier in a different position with regard to the current manufacturing position.