Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

126

Microcomputer System Design

An Advanced Course
Trinity College Dublin, June 1981

Edited by M.J. Flynn, N.R. Harris, and D.P. McCarthy

Springer-Verlag
Berlin Heidelberg New York 1982

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

126

Microcomputer System Design

An Advanced Course
Trinity College Dublin, June 1981

Edited by M.J. Flynn, N.R. Harris, and D.P. McCarthy

Springer-Verlag
Berlin Heidelberg New York 1982

Editors

Michael J. Flynn
Computer Systems Laboratory
Stanford University
Stanford, CA 94305, USA

Neville R. Harris
Department of Computer Science
School of Engineering
Trinity College
University of Dublin
Dublin 2
Ireland

Daniel P. McCarthy
Department of Computer Science
School of Engineering
Trinity College
University of Dublin
Dublin 2
Ireland

AMS Subject Classifications (1979): 68-02, 68A05, 68B20
CR Subject Classifications (1981): 4.1, 4.2, 4.3, 6.1, 6.2, 6.3

ISBN 3-540-11172-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-11172-7 Springer-Verlag New York Heidelberg Berlin

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

PREFACE

The main European tradition is for computer people to be educated in either the software or the hardware side of computing. The Computer Scientist graduates with his knowledge of operating systems, compilers, data structures, etc. and happily enters the field of software and has no contact with computer hardware except as a user. Engineers generally graduate with a knowledge of Electronics and very little knowledge of the software side.

The advent of microcomputers has seen the Computer Scientist trying to increase his hardware knowledge in order to implement systems. It has also seen the Engineer implementing large systems using assembly language or struggling with interfaces to operating systems and high level languages. It is now evident that in order to use microcomputers effectively system designers require a broad knowledge of computer hardware, interfacing, software, and design tools.

Because of this we proposed a microcomputer system design course which would integrate the hardware and software sides of microcomputers and bring together academics/practitioners from both disciplines. In 1979 a course proposal was made to the National Board of Science and Technology (N.B.S.T.) Ireland and to the Informatics Training Group of the EEC Scientific and Technical Research Committee (CREST). The proposal was enthusiastically received and supported.

Lecturers were chosen to cover the main areas of the syllabus and during the last week of September 1980 a week-long meeting took place in County Wicklow for the detailed planning. It was agreed that the syllabus should "span development from silicon technology to software and should bring together current techniques in LSI/VLSI design, computer structures and languages and show their application to, and implication for, microcomputer system designs". A detailed syllabus was prepared resulting in this set of course notes.

The course ran in July 1981 and had approximately 75 attendees from ten countries whose enthusiasm and interest made the program all the more interesting for all. A preliminary version of these notes was published at Trinity College for use in the course. This Springer-Verlag edition includes revisions based on presentations, corrections and some new material.

Acknowledgements:

With great pleasure we take this opportunity to express our gratitude and appreciation to:

- **the EEC for their financial sponsorship and to the Informatics Training Group of the EEC Scientific and Technical Research Committee (CREST) for their support and encouragement.**
- **the National Board of Science and Technology for their financial sponsorship, advice and encouragement. In this regard special thanks are due to Dr. B. O'Shea.**
- **the lecturers for their full co-operation during the preparatory seminar, planning of the course, and submission of course notes and the course itself.**
- **Mrs. Janet Hogan, our Administrative Secretary, for her cheerful organisation of these course notes and her efficient, willing administration of the course.**
- **Professor J. G. Byrne, Head of Department of Computer Science, Mrs. H. Smith and Miss Susie Pakenham-Walsh, and to all the members of our staff who have helped in the organisation of this course.**
- **Prof. F. Sumner of the University of Manchester who provided early council on the course organisation and lecturers. Events conspired to prevent him from a later, more direct involvement with the course.**
- **The course presentation was assisted by the use of several video taped lectures originally presented at Stanford University. We are grateful to the Stanford Computer Systems Laboratory for making these available and to the TCD Communications Centre for use of its video facilities during the course.**

M. J. Flynn
N. R. Harris
D. P. McCarthy

TABLE OF CONTENTS

HIGH LEVEL SEQUENTIAL AND CONCURRENT PROGRAMMING

R. H. Perrott, Department of Computer Science, The Queen's University, Belfast.

MICROCOMPUTER OPERATING SYSTEMS

N. R. Harris, Department of Computer Science, Trinity College, University of Dublin, Dublin 2.

NOTES ON DISTRIBUTED SYSTEMS OF MICROPROCESSORS
G. J. Popek, University of California at Los Angeles

LILITH : A PERSONAL COMPUTER FOR THE SOFTWARE ENGINEER
N. Wirth, Federal Institute of Technology (ETH), Zurich.

PERSPECTIVE ON MICROCOMPUTERS

Michael J. Flynn
Department of Computer Science
Trinity College, Dublin
(on leave - Stanford University)

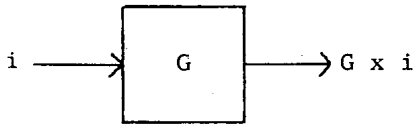
It is the thesis of this introduction, and indeed this Course, that the study of Microcomputers is not simply the study of very small computers. It is that the radical decrease in the cost of technology has significantly altered the designer's and user's view of realizing systems' applications.

From the point of view of the designer the Microcomputer is an integrated computer engineering discipline encompassing and relating such studies as:

- technology
- computer aided design (CAD)
- computer architecture
- operating systems
- language
- environment/function

From the users' point of view the Microcomputer is an amplifying systems component. Unlike simple signal amplifiers, the Microcomputer amplifies behaviour - or "intelligence" (Fig. 1). Given an input, i, the output is any response based on the behaviour of (i), (B(i)); not just a simple gain multiplier as in more familiar amplifiers. These components (designated μBj on Fig. 2 - representing the jth behaviour or programmed response to an input signal) may be connected in arbitary patterns to create complex system interactions. The reader will note the need for a new "circuit theory" - a theory of complex behaviour interaction.

Coupling these views is the realization that the more integration of function (behaviour) that occurs during the design the better the response (i.e. performance) of the system's component. Offsetting this is that the very integration specializes the resultant design, limiting its overall applicability and increasing effective per unit cost - since design costs are fixed. Only by reducing design costs (through CAD) can we hope to realize the potential inherent in the technology.



A familiar amplifier increase signal energy

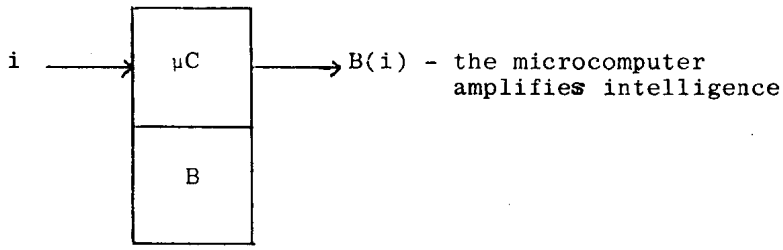


Figure 1 : Microcomputer : An intelligence amplifying component

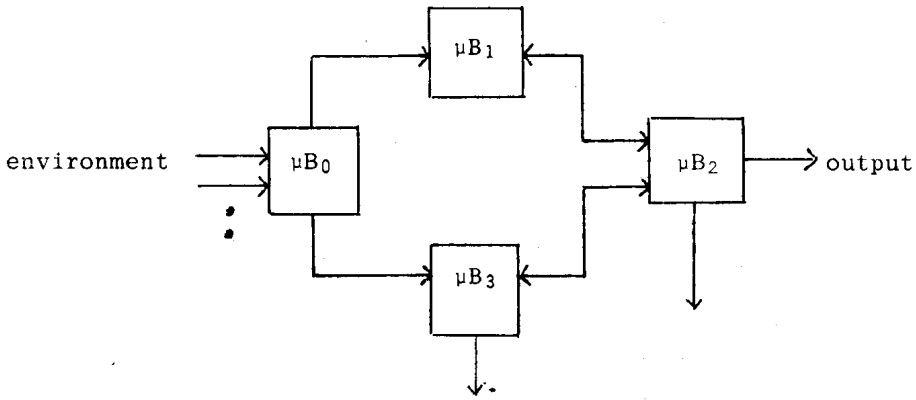


Figure 2 : Interconnected microcomputers

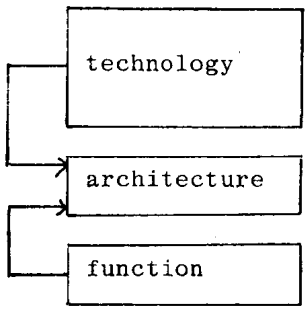


Figure 3 : Early computers

It is useful to review the changing nature of computer design in an historical perspective; to situate the Microcomputer environment.

Early computer development (pre-transistor)

Perhaps one might even call this the "classical period" of computer design because there was a relatively high degree of inter-disciplinary understanding on design (Fig. 3). With cost dominated by technology the architecture closely reflected the particular function within severe technological limitations: the "form closely followed the function". Of course designers considerations were much simplified by modern standards - there were no CAD, language or operating systems considerations. In these early systems the user was required to display considerable familiarity with the technology for efficient functional execution. For example, in many drum based computers proper placement of the next instruction depended on the execution time of the current instruction.

The Middle Ages - Batch Computing (Fig. 4)

As technology evolved and the cost per computation decreased secondary functions were taken over by the computer, viz. high level languages to improve programmer productivity, operating systems to manage system resources. By the early 1960's the "batch processing system" had fully evolved. In this environment the programmer would write a complete program in a suitable higher level language (only two were in really common use: Fortran and Cobol), submit it to the system which would translate all programs written in a particular language at the same time, then one by one load and execute them (batched together), and finally print all the results. The batch system was a further evolution in the recognition of the decreasing cost of the computer vis-a-vis the human being.

A few points about these early systems are worth recalling to mind.

The costs were dominated by the "system"; including processor technology, input-output equipment and the supporting management facilities. In fact, because the (human) management function was so important at this time it was very slow to respond to technology which would decentralize this responsibility. Even today the residue of this centralized "computer centre" management philosophy still persists.

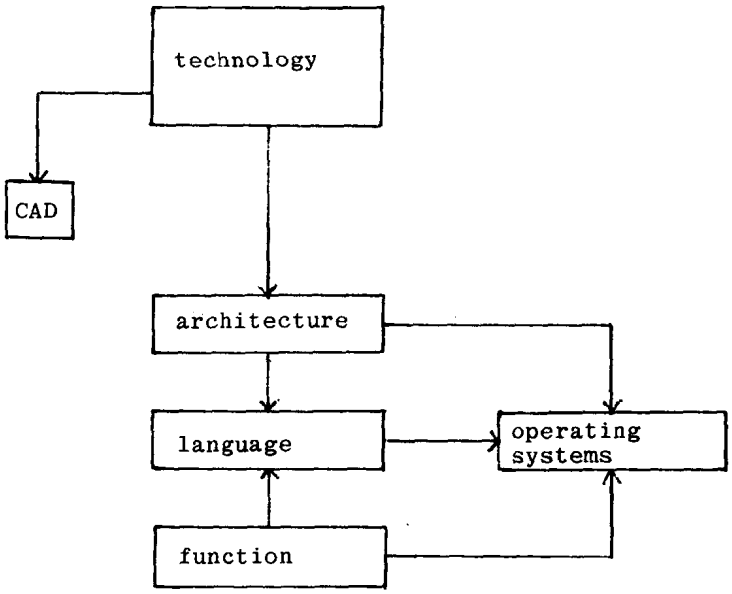


Figure 4 : Middle Ages - Batch Computing

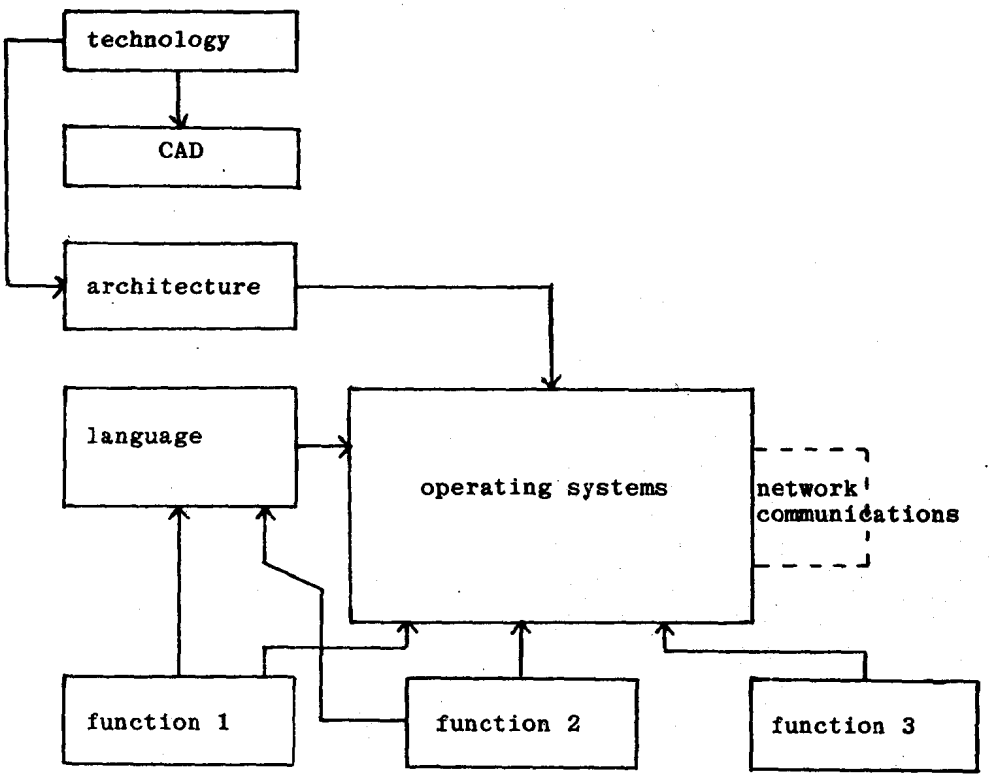


Figure 5 : Later Ages - Time Sharing

•The genesis of CAD was developed at this time - mainly for the rather unglamorous role of production control. "Unglamorous", perhaps, but a number of otherwise successful commercial ventures failed due to the lack of systematic production (quality) control.

•Early higher level languages were "low level" by latter standards - they were sensitive to machine architecture and execution performance.

The latter ages - "time sharing" and beyond (Fig. 5)

Within ten years the batch model had largely been replaced by the time sharing system. As processor technology costs continued to decline, the relative emphasis on direct user costs increased. Thus time sharing was introduced as an aid to programmer productivity accompanied by increasing diversity and sophistication in high level languages. Quick response to multiple users using diverse languages/functional environments required still further sophistication in operating systems. One may truly speak of this age as the "baroque" era of system design. The very complexity of such systems - especially the operating systems - designed to serve a universal user - is its great weakness. Again many commercial ventures failed because of inability to constrain the complexity and universality of the system.

Language design became almost as complex; with some (nameless) efforts proving to be equally unsuited to man or machine ("man and machine independent").

At the same time, CAD tools improved at a modest pace to include many useful hardware design aids - simulation, test, wiring printed circuit layout, etc.

VLSI - The era of distributed intelligence

Question : How does one design a system with a very low cost technology?
Answer : Use as much of it as you can.

The message for today's designers is to distribute function/capability through low cost technology while sharing only expensive resources (e.g. centralized data banks). But, as many have already noticed, neither a small computer or program is "small" when fully situated with interface hardware and software. Clearly, the interface problem limits the applicability of the technology. And design costs limit

the use of the technology in solving the interface problem. The CAD of earlier times is inadequate in coping with the complex requirements of modern technology. Linear improvements in minimizing a feature dimension result in a squared improvement in logic per unit area. Increases in logic gates increase the CAD complexity (for layout, simulation, test, etc.) by a power function.

Some problems (and opportunities) for the VLSI era

The growth in complexity illustrated by CAD (above) continues the higher one goes into the design process - architecture, operating systems, language - but it is basically also an increase in design opportunity, the ability to realize complex design which would have been otherwise infeasible.

We can summarize some of these problems/opportunities as follows:

1. CAD tools to allow ready use (verified and testable) of the technology.
2. Architectural designs which are optimized to the execution of particular language/operating system environments.
3. Operating systems which provide efficient local services as well as reliable communications to central resources.
4. Languages which are matched to function (comprehensible by both man and machine).
5. Data communications networks which are flexible and reliable and allow a variable balance between distributed and central intelligence.

The above is certainly not a comprehensive list - it is a departure point for those participating in the use of these notes and lectures.

The ant

We are frequently concerned with the limits of technology - the ultimate number or speed of logic gates per unit volume or energy. A more interesting question concerns the limits of design - independent of technology: our ability to represent ideas, languages and architectures. If we confront today's technologist with the possibility of a 100,000 gate chip, each gate switching with energy of 1pJ(1×10^{-12}

watt-seconds), he would certainly regard this as highly probable, if not now, then in the not far distant future. Yet the brain of the common ant consists of 100,000 neurons, each switching with energy of about 1pJ. And the architect/designer would not be at all sanguine about duplicating anything like the sophistication or complexity of such a creature. Thus even with the advantage of over a million times in speed (logic gates are much faster and use correspondingly more power than neurons giving the same energy product), we - the users of technology - have a long way to go before approaching a design limit.

Well, perhaps with several hundred million years of engineering effort .....

INTEGRATED CIRCUIT PHYSICS AND TECHNOLOGY

J. F. Gibbons and J. D. Shott
Stanford Electronic Laboratories
Stanford University

Stanford, CA 94305/USA

Our purpose in these lectures is to introduce the physical principles on which integrated circuit behavior depends and to show how semiconductor technology is employed to fabricate integrated circuits that behave according to these principles. Our presentation of this subject will necessarily be highly condensed and oversimplified. However, the analysis to be offered is conceptually correct and in any case is intended to provide only a broad understanding of the role that semiconductor physics and technology play in the solution of electronic systems problems. References are provided for those wishing a deeper appreciation of the topics to be presented.

We will begin with a discussion of the basic principles of MOS integrated circuits, leading to a discussion of both CMOS gate arrays and the "sticks" approach to IC chip design that has been so successfully pioneered by Mead and Conway. We follow this discussion with an outline of the fabrication processes that are used to make integrated circuits. We conclude with a discussion of bipolar digital technology and some remarks that compare this technology with its MOS counterpart.

1. Basic Principles of MOS Devices and Integrated Circuits

1.1 What is a Semiconductor ?

Solid state materials can be divided broadly into three categories: conductors (i.e., metals), semiconductors, and insulators. Integrated circuits contain representatives of each of these types of material. However it is the special properties of crystalline semiconductors that provide the signal amplification and switching characteristics that are necessary for the construction of electronic systems. It is therefore appropriate to begin with a brief description of those properties of crystalline semiconductors that are essential to the operation of integrated circuits.

The Conduction Process. On a microscopic scale, current consists of a flow of charged particles, called carriers. The flow of electrons in a metal in response to an applied electric field is a familiar example. Metals have an abundance of free electrons, that is, electrons which can readily move within the material. Because the number of free electrons is large, metals can conduct currents readily with the application of very small electric fields.