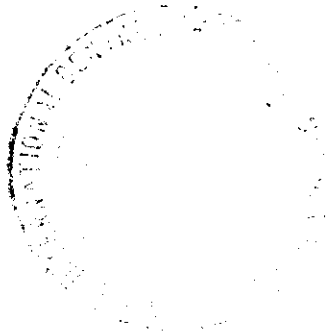


INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONES: 224281/2/3/4/5/6
CABLE: CENTRATOM - TELEX 460392-1

SMR/90 - 36

COLLEGE ON MICROPROCESSORS:

TECHNOLOGY AND APPLICATIONS IN PHYSICS

7 September - 2 October 1981

METHODOLOGY OF SYSTEM DESIGN

C. VERKERK

DD Division
CERN
1211 Geneva 23
Switzerland

These are preliminary lecture notes, intended only for distribution to participants. Missing or extra copies are available from Room 230.

Methodology of System Design

van Dam's Lectures:

- Top-Down Design
- Stepwise Refinement
- Structured Programming
- Modularity

applied to Software Development.

In this lecture we will consider the design of a System, consisting of hardware and software.

A closer look at the hardware design process, some remarks on hardware/software trade-offs and the integration of hardware and software.

Hardware and Software are often developed independently of each other.

This happens when you buy, for instance,

Computer
+ Camac Interface
+ Camac Crate + Controller
+ Camac Modules

But even then, the design of the total system must include hardware and software considerations, before you decide what to buy.

This is even more true for all projects involving microprocessors.

In a microprocessor project the skills needed are in the areas of:

- Applications and System Design
- Hardware (Electronics) Design
- Software Design and Programming

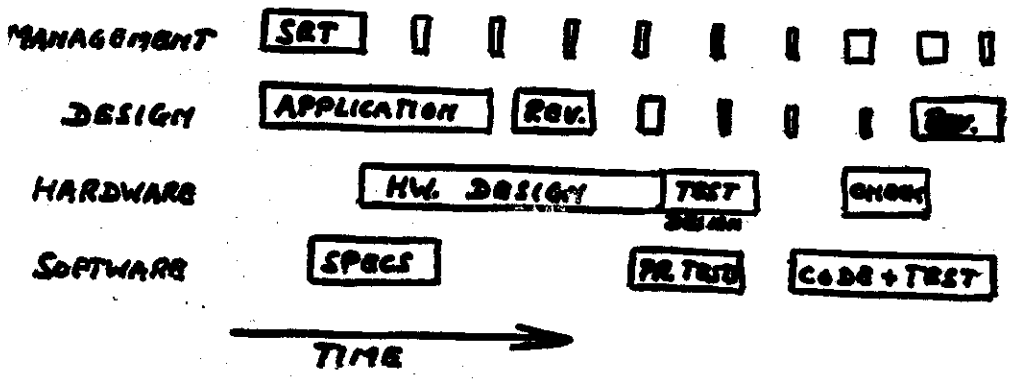
(- Project Management)

Interdisciplinary Knowledge is what you need

You can often tell from a design which discipline dominated.

- Electronics Engineers tend
  - either to make complicated external logic to do things easily done by software
  - or they overestimate the software capabilities and oversimplify the interfaces
- Software people generally get a good estimate of storage requirements, but get the I/O requirements and timing wrong.

IF A SINGLE PERSON WHO POSSESSES ALL THE NECESSARY SKILLS IS NOT AVAILABLE, YOU MAY NEED FOUR PEOPLE, BUT ONLY PART-TIME (AT LEAST TWO).



HARDWARE DESIGN TRADITIONALLY IS DIFFERENT FROM SOFTWARE DESIGN.

- ENGINEERS HAVE ALWAYS BEEN CONSCIOUS OF THE CONSEQUENCES OF A DESIGN ERROR.
- SOFTWARE SEEMS TO BE EASILY MODIFIABLE.

WHEN IT TURNED OUT THAT THE LATTER POINT WAS NOT TRUE FOR LARGE SOFTWARE SYSTEMS, SOFTWARE ENGINEERING WAS BORN.

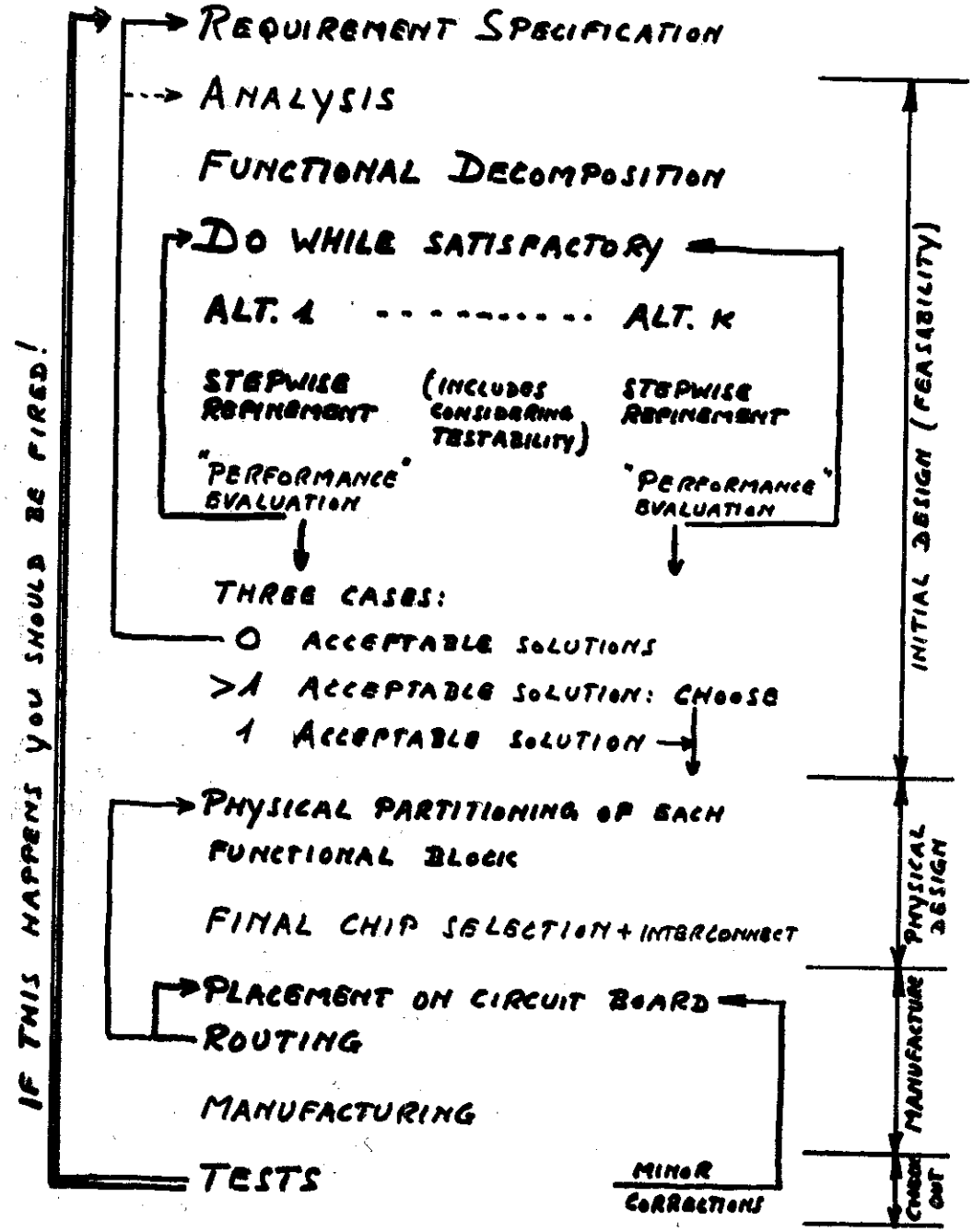
SOFTWARE DEVELOPMENT USED TO BE A BAG OF TRICKS.

SIMILARLY: IN HARDWARE DESIGN

DON'T FALL IN LOVE WITH THE LATEST CHIP ANNOUNCED!

CONSTANCY IS A SIGN OF TRUE LOVE.

THE HARDWARE DESIGN PROCESS.



Requirement Specification:

Write it down!

- Functions performed
- Speed
- User interface
- Interface to equipment
- Data Rates
- Weight, Size, Cost, Power
- Expansion Possibilities
- Testability
- Reliability, Maintainability.
- Modularity
- Standards to adhere to.

This is a very important step in the design! These specs should be as detailed, precise and realistic as you can make them.

The initial design stages of the software are not essentially different from the hardware case:

Requirement
Analysis
Functional Decomposition
Stepwise Refinement
Performance evaluation

The implementation phase is different, of course.

Integrated Hardware-Software Design.

Ideally, design of software should go hand in hand with initial hardware design.

Don't wait for completion of the hardware to start on the software development.

Software design should therefore include design of tests which can be done without the final hardware.

Close collaboration is required during the initial design stage.

Don't underestimate the difficulties of this collaboration.

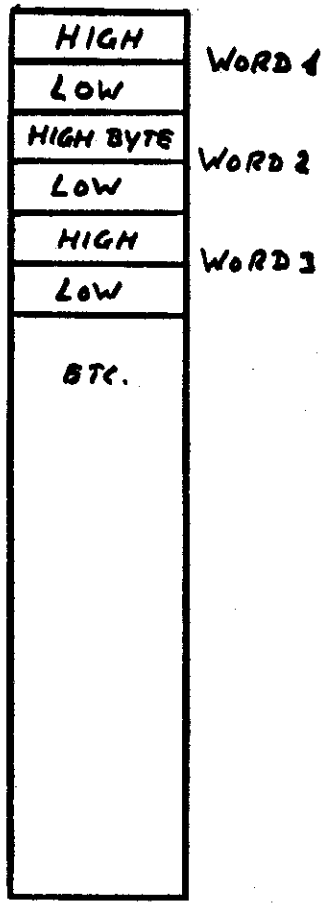
Hardware and Software People can misunderstand each other in numerous ways!

Some sort of management is therefore useful, even for small projects.
How formal this should be is a matter of style and habit.

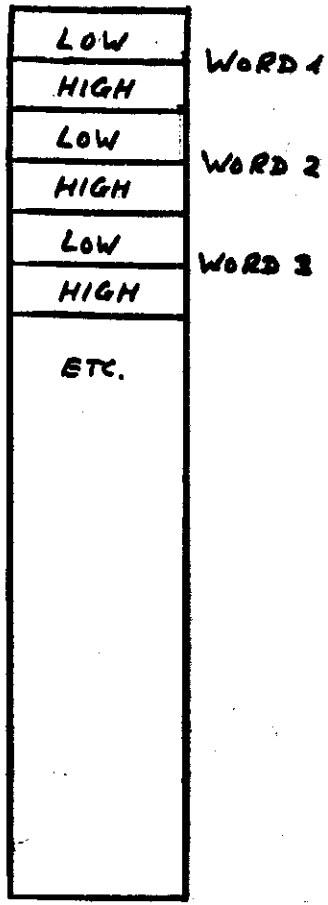
The misunderstandings nearly always concern the order in which events (should) happen:
Bit or byte order, setting, resetting of status

During Physical design stage there is less need for coordination.

A TRIVIAL EXAMPLE:



WHAT THE PROGRAM EXPECTED



WHAT THE HARDWARE DID

CORRECTION:

IN SOFTWARE:

INSTEAD OF
```
INC X
INC X
```
DO
```
INC X
INC X
INC X
DEC X
```

IN HARDWARE: MOVE WIRES AROUND

HARDWARE - SOFTWARE TRADE-OFFS.

THE CRITERIA USED IN INDUSTRY FOR LARGE QUANTITY PRODUCTS ARE DIFFERENT FROM OUR CRITERIA.

THESE INDUSTRY CRITERIA ARE:

- SOFTWARE DEVELOPMENT COST IS PAID ONCE
- AN EXTRA CHIP IS PAID MANY TIMES
- HARDWARE COST SHOULD NOT BE ALLOWED TO MAKE A QUANTUM JUMP
- THE SYSTEM IS TOO FAST ANYWAY, SO SOFTWARE IS ALLOWED TO SLOW IT DOWN.

AN EXAMPLE IS THE MEK6802D5 KIT.

LOOK AT THE PROGRAMMING EXAMPLES IN THE FIRST PAGES AND THE USE OF

PUT ROUTINE AND

MNPTR, A MYSTERIOUS POINTER.

PUT WILL TAKE CONTROL OF THE MACHINE AND "CALL" AT REGULAR INTERVALS THE USER PROGRAM USING MNPTR.

WHY WAS IT DONE THIS WAY?

TO SAVE 6 x 7 LATCHES FOR THE LED DISPLAY

USER CONVENIENCE AND NEATNESS OF DESIGN WERE OVERRULED BY COST CONSIDERATIONS.

ANOTHER EXAMPLE:
ANALOG TO DIGITAL CONVERSION.

ONE OF NEXT WEEK'S EXERCISES:
MAKE AN A TO D CONVERSION, USING A DIGITAL TO ANALOG CONVERTER AND SOFTWARE.

GENERAL PRINCIPLE OF ANALOG TO DIGITAL CONVERSION:

DO UNTILL $V_{IN} = V_{DAC}$

TAKE A DIGITAL NUMBER
PRESENT IT TO DAC
COMPARE OUTPUT VOLTAGE OF DAC (V_{DAC})
WITH VOLTAGE TO BE CONVERTED (V_{IN})
END

THE TRICK IS NOT TO TRY RANDOM NUMBERS, BUT TO FOLLOW A LOGICAL SEQUENCE.

THIS CAN BE DONE BY SOFTWARE, FOR EDUCATIONAL PURPOSES, BUT NORMALLY YOU BUY AN ADC THAT DOES IT BY HARDWARE.

- HARDWARE IS HARD TO MODIFY, BUT FAST
  HARDWARE IS ALSO LONG TO DEVELOP AND BUILD.
- SOFTWARE IS ADAPTABLE*), BUT SLOW

*) NOT TRUE FOR MANY INDUSTRIAL PRODUCTS USING ROMS.

IN OUR ENVIRONMENT HARDWARE VS SOFTWARE TRADE-OFFS MUST GENERALLY BE CONSIDERED IN TWO DISTINCT AREAS:

- THE PROCESSOR ITSELF: DO WE NEED ARITHMETIC? HOW POWERFUL?
  IS INTEGER MULTIPLY ENOUGH OR DO WE NEED FLOATING POINT?
  DO WE NEED SIN, EXP ETC.?
  YOU SHOULD INVESTIGATE HOW MUCH PROCESSOR TIME IS AVAILABLE AND HOW MUCH IS NEEDED.
  ALSO MEMORY REQUIREMENTS FOR SOFTWARE
  IN MOST APPLICATIONS YOU HAVE EXTERNAL CONSTRAINTS ON PROCESSING TIME:
  EVENT RATES, SAMPLING RATES,
  TIME CONSTANTS OF CONTROLLED EQUIPMENT.

- THE INTERFACE TO THE EXPERIMENT:

DO WE NEED

BUFFERING? WHAT SIZE BUFFER?

DMA OR PROGRAMMED TRANSFERS?

PRE-PROCESSING? POST-PROCESSING?

FILTERING TO ELIMINATE NON-VALID DATA?

LOCAL INTELLIGENCE?

HERE YOU SHOULD INVESTIGATE WHAT THE BURDEN ON THE PROCESSOR WILL BE IF YOU DO NOT ADD HARDWARE. WEIGH IT AGAINST THE COMPLEXITY OF THE HARDWARE REQUIRED AND THE DIFFICULTY TO BUILD IT.

DATA RATES, CHARACTERISTICS AND NATURE OF THE DATA, NATURE OF THE PROCESSING WHICH MUST OR MAY BE DONE WILL GUIDE YOU.

ANYTHING THAT CAN BE DONE IN HARDWARE CAN <u>IN PRINCIPLE</u> BE DONE IN SOFTWARE <u>AND VICE-VERSA</u>.

DO (FOR, WHILE, UNTIL)

PROCESS

END

CAN BE IMPLEMENTED IN HARDWARE IF PROCESS IS SUFFICIENTLY SIMPLE (e.g. ADD TO ACCUMULATED SUM). THE SPEED MAY THEN INCREASE FACTOR 100.

```
IF .....
  THEN
     PROCESS 1
  ELSE IF .....
     THEN
        PROCESS 2
     ELSE IF .....
        THEN
           PROCESS 3
        ELSE IF ...
              ETC.
```

MAY BE EXTREMELY SIMPLE, OR EXTREMELY COMPLICATED IN HARDWARE, DEPENDING ON THE IF CONDITIONS AND THE PROCESSES.

SIMPLE: IF BIT 1 SET READ A
IF BIT 2 SET READ B
ETC.

MANY EXAMPLES FROM HIGH ENERGY PHYSICS IN CERN 81-07, CERN 74- , CERN 78- , CERN 81-03

CONCLUSION:

SYSTEMATIC APPROACH TO DESIGN OF HARDWARE-SOFTWARE SYSTEMS IS IMPORTANT TO AVOID ERRORS WHICH MAY BE DIFFICULT TO CORRECT AND TO ENSURE THAT THE SYSTEM WILL HAVE THE SPECIFIED PERFORMANCE.
SOME OF THE INDUSTRY CRITERIA ARE NOT TRANSPORTABLE TO A RESEARCH ENVIRONMENT.