Trends in computer technology

E.H. Dooijes
Department of Computer Science, University of Amsterdam, 1994

Introduction

In this article we will discuss some of the technical problems and developments which had a major significance for the evolution of the digital computer, since the ENIAC became operational in January 1946. It goes without saying that the developments in software were at least as important, if only because software defines the link between the computer as an universal machine on one hand, and the particular application on the other hand. Software however will not be the subject of this article.

The ENIAC, the first *electronic* computer (see box 1) has been a turning point in a course of development that started long before 1946.

An illustrious - though never completed - predecessor was Charles Babbage's Analytical Engine (about 1845). Babbage's experiments with computing automata didn't find continuation until ninety years later, when - in the 1930's - electrical technology could provide an alternative for the exorbitantly expensive and difficult to handle mechanical systems Babbage had to rely on.

The first more or less useful results in general purpose electromechanical computing machines were obtained around the beginning of World War II, by Konrad Zuse in Germany and Howard H. Aiken in the U.S.A. The ideas of Charles Babbage were largely forgotten by that time.

In the pre-war decades, and particularly during the war, computing was exclusively associated with number crunching. In fact, the word 'computer' had the meaning of 'a human performing computations', either or not using mechanical equipment. Work in the (then) seemingly unrelated field of mathematical logic, emerging in the early 1930's, turned out to be very consequential for the development of the symbol-manipulating, information processing computer as we know it today. The British logician Alan Turing played a crucial role here. Some historians hold that the modern computer was really born in 1936, when Turing published his paper on what is called today the Universal Turing Machine.

Before the world war, scientific computational work was characterized by being sparing in calculations, but supported by virtually unlimited memory (scrap paper, mathematical tables); with the coming of ENIAC this state of affairs was completely reversed. This gave a dramatic turn to the field of numerical mathematics. This development, and others triggered by the ENIAC would ultimately lead to the 'second industrial revolution' which became reality when the integrated circuit appeared on the stage, making the computer a wide-spread phenomenon in a short time.

What is a computer?

In what follows we will use the word 'computer' to indicate a physical machine: the invariant of the computer *systems* which can be built around the computer. A computer becomes a computer system by adding software and - often dependent on the application - peripheral equipment like printers, multi-media equipment, tape drives, robot arms and the like.

Box 1: The ENIAC (Electronic Numeric Integrator and Calculator)

Work on the ENIAC started 1943 at the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, under the direction of J.P. Eckert en J. Mauchly. The machine was switched on for the first time in January 1946 and remained operational until 1955. There were 18000 electron tubes in the ENIAC. It took 233 m^2 of floor space, its energy consumption was 140 kW, its weight 30 tons.

In the beginning the ENIAC had only 20 memory locations, meant for input and output data and intermediate results, represented by 10-digits decimal numbers. In the original design programming was done using flip switches and patch cables (like in an old-fashioned telephone exchange). Eventually, the stored-program mode of operation, as proposed (but not invented) by Von Neumann was adopted, after the machine's switches and patch cables were put in a fixed setting.

Developed for the computation of ballistics tables during the world war, the ENIAC came too late for that purpose, and was mainly used for another military application: the Manhattan hydrogen-bomb project. For the solution of a certain differential equation, 8 million multiplications and a comparable number of additions were needed. This took 15 system-hours, 20% of the time being used for testruns [1]. Nevertheless computing was done more than 1000 times faster than was possible with any equipment available then.

It is important to realize that there never has been a sharp boundary between hardware and software. Recall the programming-by-wiring of the ENIAC, and modern concepts like firmware, microcoding etc. With neural networks - another kind of computer - the traditional concept of programming isn't even applicable.

From about 1930 to 1970 an important role in industrial, military and scientific applications was played by *analog* computers (or *differential analyzers*). This type of machine will not be discussed in this paper, though it should be mentioned that the ENIAC was designed as a 'digitalized' differential analyzer.

We won't go into details of the operation of a digital computer. In the present context the following fundamental observation suffices: a computer is an organized collection of switches of the 'gate' type. A gate passes or stops electrical current under the control of another electrical current, which is switched in turn by still another gate. How one set of gates affects the state (on or off) of other gates depends both on the architecture of the computer, i.e., the way the gates are interconnected, and on the input and program data fed into the computer. Each possible combination of 'microscopic' gate states defines a unique state of the computer as a whole.

Box 2: Relays, tubes and transistors.

A relay is a switch with one or more pairs of contacts, which can be opened or closed by a built-in electro-magnet. In an electron tube, there is a stream of electrons from an electrically heated electrode (cathode) to a positively charged second electrode (anode). The stream of electrons can be interrupted by connecting a negative voltage to a third gauze-like electrode placed between cathode and anode. A transistor operates on roughly the same principle; however here the electrons are moving through semiconducting material. Most of the time this is silicon, with an accurately controlled number of 'impurity' atoms built into the crystal lattice.

Switching takes time: for a relay at least 1 millisecond, for an electron tube 1 microsecond, for a transistor (imbedded in a modern integrated circuit) 1 nanosecond.

The first prototypes of computers were equipped with electromechanical gates (called 'relays', see box 2). In the ENIAC electron tubes were used, and since 1959 gates are implemented exclusively by transistors.
There is no intrinsic difference between data and programs; this is the essence of the *stored-program* concept. Not only numbers, but texts, abstract concepts and programs as well are represented in a digital computer by strings of binary symbols or *bits*. The pair of values a bit can assume are usually called on/off, true/false or 1/0. Feeding a stream of bits into a computer means that specific gates are set to the open or closed state before the processing starts.

Box 4: How are chips made

Chips are made using a lithographic process. A slice of pure silicon (wafer) is covered by an insulating layer of silicon dioxide by exposing it to oxygen at high temperature. Next this layer is coated by a light-sensitive material. The chip design (in the form of a mask or template) is projected onto this layer, using light of short wavelength (ultraviolet). After being developed, like photographic film, the coating can be removed by a chemical solvent from the illuminated parts of the surface. Here the silicon dioxyde is etched away and the exposed silicon is doped: 'impurity' atoms (arsenic or antimony) are implanted into the crystal lattice thus changing its electrical characteristics. This treatment can be repeated many times with different masks. Connections to the periphery are made using evaporated aluminium.

A single wafer contains dozens of chips which are separated and then tested individually. Usually more than half of the chips are rejected at this stage. The remaining chips are glued to plastic or ceramic carriers and provided with terminal connections. In order to reduce the number of chips thrown away, sometimes redundant subsystems are built on each chip. These spare subsystems are activated (in the factory) when the original subsystems don't pass the testing.

As they have a very regular structure, memory chips usually have a much higher packing density than processor chips.

As was mentioned earlier, the *architecture* of a computer specifies which (groups of) gates can affect each other's states. A very important consideration is here, that it must be perspicuous for the human user how to program the system. This is one of the reasons of the long-time success of the Von Neuman architecture model, which discerns mutually communicating groups of gates each of which has a well-defined function (arithmetic/logic unit, register, central processor, memory etc). The well known fetch-decode-execute cycle is very characteristic for the Von Neumann architecture. An extreme counterpart of the Von Neumann computer is the artificial neural network, whose architecture is inspired by biological neural networks. Instead of being programmed in the traditional sense, an artificial neural network knows what it is expected to do by learning from examples.

Box 3: Numbers of 'switches' in computers

Zuse Z3 (1943): 2000 relays

ENIAC (1945) 18.000 electron tubes

IBM 709 mainframe (1959): last machine using tubes, about 20.000

IBM 7090 mainframe (1959): 20.000 (separate!) transistors

Motorola 68000 processor (1980): 68.000 transistors on a single chip

DEC Alpha processor (1992) 1.7 million transistors on a single chip

To the present day, the developments in computer hardware can roughly be described as a scaling-up of the Von Neumann model; apparently this has been sufficient to make possible the evolution in computer applications we have witnessed during the past decennia. However, this development is now meeting its limits, and we observe a quickly growing interest in alternative architectures. These are often characterized by a 'symbiosis' of - sometimes large - numbers of processing units (ranging from 1-bit processors to complete workstations), operating more or less independenly.

The following assertion, loosely based on the notion of the Universal Turing Machine, could be named the 'First Law of Computer Science': *all computers can do in principle the same things, provided that the number of states (i.e., the amount of memory) and/or the available time is large enough*. Of course, in practice this condition is usually not fulfilled! Also, practical constraints related to the (peripheral) hardware cannot be taken into account: it is certainly bot possible to have an 8" floppy disk read by a modern laptop computer…
This 'law' does tell us, however, that an artificial neural network can be simulated by a classical (Von Neumann) computer, and in fact this is the way artificial neural networks are usually implemented today; the inverse must be possible as well!

Milestones: hardware

1943 relay machines Zuse (Germany), Aiken (USA).
1946 ENIAC operational.
1947 invention of the transistor (Bardeen & Brittain, Bell Labs), announced to the public as useful 'for the development of better hearing aids for the deaf'.
1951 Univac I, the first commercial machine (sold to the US Bureau of Census).
1952 IBM 701, the first commercial machine manufactured in a series (19 machines installed); rent $15000 per month.
1953 invention of core memory.
1956 introduction harddisk (IBM RAMAC).
1958 invention of the planar transistor, suitable for integration (Fairchild Semiconductor).
1959 IBM 7090, the first commercial transistor machine (about 20.000 transistors costing $80 each).
1961 first integrated circuit (4 transistors, Fairchild).
1964 Introduction of IBM 360 series of mainframe computers.
1964 CDC-6600 scientific computer, designed by Seymour Cray.
1965 DEC PDP-8, the first commercially successful minicomputer.
1970 Illiac IV, the first supercomputer (64 processors operating in parallel, 20 MFLOPS, University of Illinois; in use until 1981).
1971 Intel 4004 microprocessor, 60.000 instructions per second, 2300 transistors.
1972 Tektronix introduces the Direct View Storage Tube graphics terminal. By reducing the graphics-console cost from $150 to $20 per hour, this was the starting point for the rapid development of computer graphics.
1976 Cray-1 supercomputer 'the fastest computer in the world': 100 MFLOPS.
1976 Ethernet (Xerox), Token Ring network (IBM).

1977 the first commercial microcomputers Apple II, TRS-80, PET, using 8-bits microprocessors.
1978 16-bit microprocessor (20000 transistors).
1980 the era of VLSI heralded by the textbook *Introduction to VLSI Design* by Mead en Conway [2].
1981 introduction of the IBM PC.
1981 Xerox Star: first commercial application of windows with mouse-based interaction (Graphical User Interface, GUI); lack of commercial success due to high price and large size.
1984 Apple Macintosh, the first small system using GUI.
1990 Connection Machine: massively-parallel machine, up to 64000 'basic' processors.
1993 Alpha, PowerPC, Pentium: 32-bits microprocessors.

Milestones: Programming languages and programming methodology

Complementary to the development of computer hardware is the evolution of programming languages. Without sophisticated programming tools (including operating systems), the range of possible applications would have been very small, and there would have been little reason for the development of computer hardware that we have actually witnessed. It should be noted that through the years a problem has been that the possibilities of the hardware coudn't be fully exploited by the programming tools existing at the same time. This 'software crisis' is topical to this very day, the more so because the Von Neumann architecture model is currently being abandoned in favour of distributed processing, with some (or many) processors operating simultaneously on a single problem.

The exponential technology growth model

Experience of the past 30 years shows that the cost of computer equipment has decreased by about 20-30 % per year. That is a factor 10^5 over 30 years! Cost can be interpreted here here as price-performance ratio, price per bit of memory or the like. In 1964, G.E. Moore [3] predicted that the number of components on a chip would increase each year with a factor between 1.5 and 2. This 'law' seems to be valid up till now. (The area of a chip has remained of the order of 1 cm^2 ; why this is so we will discuss shortly).

From these examples - to which many more can added - it appears that the computer industry is a perfect example of exponential technology development which is also found to some extent in other industries. Exponential technology development can be modelled mathematically as

$$T(t) = T(t_0) * r^{t-t_0} .$$

The technology-measure T in year t is, for instance, the price per bit of memory for computers, or the fuel consumption per watt of effective power for jet engines, or the light output per watt for light bulbs.

Exponential growth is found when the following conditions are fulfilled [4]:

1. the product in question must be fabricated for many years;
2. the manufacturing process is constantly improved by learning effects;
3. the design is constantly improved.

The conditions imply that the product has an short economical life (quick 'turnaround', no saturation of the market), and that growth is not impeded by physical limits (as it is the case with jet engines and automobiles). Besides the 'push' from the semiconductor industry, IBM's strategy to migrate computer products from high-price/small-volume to low-price/high-volume has played an important role in the realization of conditions 2 and 3. Also the interaction of industry with academic and industrial computer users has been important in that respect, for example through the development of programming languages.

An example of exponential growth: memory elements

For memory magnetic cores were used since their invention in 1952. Before that time memory was implemented by mercury delay lines, magnetic drums and 'Williams tubes'. The price per bit of core memory was 20 ¢ (dollar-cent) in 1960 and decreased from there with 19% per year. In 1974 was the 'turnover' to semiconductor (transistor) memory with the advent of the 4 kbit chip; the cost for both techniques was then 1 ¢ per bit. Core memory has been in use until recently for special purposes, because it retains the information when the power is switched off, and it is resistant against radiation.

For semiconductor memory, we have the empirical law

$$\text{price of production per bit in } ¢ = 0.3 * 0.72^{(t - 1974)}$$

which means that the price decreases with 28% each year. Today (1994) one bit costs 1 m¢, at 256 Mbits on a single chip. A consequence of the exponential decrease of memory prices (and of the fact that this was well exploited by the end users) was that the address space of processors had to be increased by one bit (i.e. a factor 2) every 2-3 years. This fact has eventually determined the life span of many families of computers. An illustration: the PDP-11 family of minicomputers (1970-1985) had 18 address bits an hence room for 256 kilobytes of memory; today's PowerPC processor chip has 32 adres bits; this suffices for addressing no less than 4 gigabytes.

What is the use of this computing power?

The world never had any problem with utilizing the equipment offered by the computer industry. Every time the machines got smaller, cheaper and more powerfull, new groups of users presented themselves. (Indeed the 'technology push' has always been more important than the 'market pull' in the computer scene). A recent example is the entrance of the computer into the printing industry.

There are many applications of the computer where performance improvements would be useful even in the distant future.

Difficult problems: simulation, for instance for weather forecasting; visualization and image processing (in medicine, printing industry); interactive systems (virtual reality), robots, automatic control of cars and airplanes. What makes these problems difficult is the very large amount of data to be handled, the requirement that this handling is done in 'real time', and also that is is done without any errors. Clearly these requirements not only concern the computer hardware but the software as well.

Intrinsically difficult problems: these are often found in situations where all possible combinations of things have to be checked, leading to 'combinatorial explosion'. The classical example is the 'traveling salesman' problem: find the shortest tour of n given cities. For n=7 we have 360 different tours; for n=120 there are 10^{197}! Practical examples which are

interesting in the present context are the testing of chips and the geometrical design of chips and chip carriers (printed circuit boards, multi-chip modules).

Why ever smaller?

That computers have become small is clear when we compare the physical characteristics of the ENIAC (box 1) to those of a commonplace PC. Not only is there an immense difference in size, but also in electricity consumption (150 W for a PC). On the other hand, the computing power of the PC is hundreds of times the ENIAC's, while the PC is many, many times more reliable.

What is getting smaller in the first place is the size of the components brought together on a chip. Of course this means that complete systems can be smaller as well; but this is not the main thrust behind this development. Globally it can be said that a centimeter-squared of silicium always costs the same, whatever is put onto it, assuming that this is something justifying large production volumes! Hence the performance/price ratio is proportional to the level of integration. Moreover, computers become faster and more reliable with increasing level of integration; below we will go deeper into these partially counter-intuitive aspects. Finally there is a host of applications for small and/or special-purpose computers, like lap-top computers, computers imbedded in measuring equipment, CD-players, calculators, washing machines, smart cards and so on.

A closer look at chips

The number of components on a chip is growing annually with a factor 1.5 - 2 (Moore's law). On the other hand, the size of a chip has not significantly increased during the years, the chip area having been of order 1 cm^2 all the time. This is because with increasing area, the probability that the silicon contains an undesired impurity or a mechanical flaw grows quickly. Larger chips would therefore cause an unacceptably high rejection rate. Hence the growing of the number of components is accompanied by (or rather is caused by) reduction of the component sizes, and by increasing the packing density.

The latter is desirable for various reasons:

- increased system speed by smaller transistors, shorter on-chip interconnections, and less inter-chip connections;
- improved reliability, by less long and error-prone connections between chips, and because less chips are needed for a given system design;
- less heat generation, as the number of high-power line-driving transistors can be reduced;
- lower system price: the price of a single chip is almost independent of what is on it.

The production of heat on a chip can not be avoided. An important task of the chip package is to allow this heat to be removed, because the temperature of the silicon may not exceed 75-85 C. By air cooling some watts can be removed through the package. A impression of the problem is given by a common 60 W lightbulb, which produces 0.5 W/cm^2 and is much too hot to touch. The Alpha processorchip dissipates 30 W and has only 2.3 cm^2 of area!

The speed of signal transportation over metallic wires is 0.5 - 0.8 times the velocity of light in vacuo (c), that is 1.5 - 2.4 $*$ 10^8 m/s. With a cycling time of about 10 ns, the characteristic length is then L = 1.5 - 2.4 m. The length differences of connections in a computer system must not exceed a small fraction of L in order to guarantee synchronous action of the various components of a computer system; this problem is encountered mainly in the field of

supercomputers. On a chip the situation is different because no metallic interconnections are used. Here the signal propagation speed can be up to a factor 1000 below the speed of light, resulting in a characteristic length of some millimeters. This can cause large problems, for instance in the distribution of the clock signals responsible for synchronous operation of all subsystems on the chip.
The higher the level of integration of a chip, the more contact pins are needed for making connections to the outside world. An empirical relation between the number of logical units on a chip and the required number of pins is given by Rent's rule (box 5). Roughly a system using n chips needs n^2 inter-chip connections. It is difficult to find the proper balance between the level of integration and the number of chips needed, considering that in modern computers 'wires' takes most of the place, form a speed bottleneck and are very expensive.
In early computer systems, wiring was a relatively negligeable factor.

Other aspects

The progress of computer technology has certainly not only been a matter of chip evolution. Other components have similar (often exponential-type) development histories. Important examples are packaging technology, communications (glass fiber), mass storage devices: diskette, tape, hard-disk, CD-ROM.

Box 5: Rent's rule

This rule, first proposed by E.F. Rent (1960) expresses the empirical relationship between the number T of terminal pins needed by a group of N processing elements to communicate with the rest of the system:

$$T = C.\ N^p$$

where C is the avarage number of terminals associated with a single processing element. Consider, for example, Motorola's 68000 microprocessor (1980) with about N = 68000 transistors and T = 64 terminal pins. (We neglect the fact that some of these pins are needed for the 'trivial' task of supplying power).
With C = 3 (the processing elements being simple transistors) we find a Rent exponent p = 0.27. For the PowerPC 620 chip (IBM/Motorola), introduced in 1995, we have $N = 7.10^6$ and T = 482, corresponding to a Rent exponent p = 0.32. For memory chips, which possess a more regular structure than processor chips, the Rent exponent is of order 0.13.
Rent's law appears to apply for each of the hierarchical levels found in a computer: from the chip level, via (sometimes) the multichip module and the printed-circuit board up to the backplane level. It has been noted [5] that in all these levels the wiring structure has comparable properties, implying that the wiring of a computer can be described as a fractal system with dimension D = 3 - 2p, p being the computer's overall Rent exponent. This p shouldn't be much different from the chip-level Rent exponent, if the wiring technology in the higher levels is chosen appropriately.

The life span of a computer

The technical and economical life expectancy of computers are vastly different. Earlier it was remarked that a short economical life is an essential aspect of exponential technology growth. On the other hand, we know that the reliability of a computer should be beyond any doubt. During their economical life computers are supposed to remain intact and not to show any signs of wear. In many commercial and industrial applications, computer down time is very

expensive. Also the cost of maintenance - either preventive or in case of trouble - is very high. Household equipment, on the other hand, is usually being used until the moment that problems are going out of control. Can computers wear out anyway?

Let us first consider the elementary components: the gates (electron tubes, transistors).

The 18000 tubes in the ENIAC had a mean time before failure (MTBF) of about one year. This means that the time between malfunctions was half an hour on the average. Therefore, the ENIAC could be used for serious computations only because every now and then preventive maintainance was done, for detecting tubes about to fail (this was possible with tubes because they usually degenerated gradually). We know that a modern microprocessor with a million transistors can be operational for years without failure. Supposing that the failure-free period is one year on the average, and supposing that failure of a single transistor leads to failure of the chip as a whole, then we find that the MTBF of the individual transistors must be of the order of a million years! (Of course we are dealing with the chips which have survived the initial testing procedure in the factory). This astronomical number is essential for the success of micro-electronics. Semiconductor physics tells us that transistors are really tend to deteriorate by their internal electrical fields and heat generation, though it is difficult to put this into quantitative terms. The aforementioned number is therefore an empirical fact.

Despite the reliability of semiconductor electronics it is sometimes desirable to perform run-time checks on correct performance. For instance, random-access memory is usually equipped with a 'redundancy check' mechanism. One reason is that cosmic radiation can cause so-called soft errors: errors which are not caused or accompanied by physical destruction of the memory cell in question.

As to the remaining parts of a computer: the MTBF of mechanical parts like keyboard and harddisk, as well as the MTBF of relatively vulnerable electronic components (large capacitors, cathode ray tube) are measured in years today.

Hence, the ecomical life of a computer is more determined by the fact that after some time the following problems become manifest: decreasing availability of (supported) software; high cost of maintenance (usually 10 % of the purchasing price of the equipment in question); lack of 'upward' compatibility; excessive power consumption, size or weight ; low speed. The consequence is that computers have to be replaced after 3-5 years of service. It has become apparent that this is causing an appreciable waste problem. With the ever larger number of small an relatively cheap computers in circulation, it is hardly rewarding for computer destruction enterprises to collect selectively the various types of scrap (plastic cases!) and reusable parts after breaking down the equipment. Some manufacturers attempt to enhance the life of computers by applying modular architectures making it possible to extend the system and to upgrade critical components like the central processor. Also in some countries the government requires that computer vendors take back abandoned computers, which of course encourages a destructor-friendly way of construction in the first place.

In this context, we don't have the opportunity to discuss health problems of the computer caused by viruses. Another interesting and important topic that we have to skip is the life expectancy of information carriers (from punched card to CD-ROM).

Future developments

With regard to future developments we have to restrict ourselves to a number of short remarks.

Presuming that the exponential growth law will not be impeded by external factors (like the general state of the world economy), the current chip technology will find its limits around the year 2010 Transistors will have become so small by then, that their mass production will be very difficult and expensive. Moreover, the operation of 'nanotransistors' cannot be described any more in terms of classical electronics, because one will have to deal with quantum phenomena. Thus, further miniaturization will require a complete new technology which largely is a *terra incognita* today.

It should be realized that even 'straightforward' extension of today's technology is a multi-million dollar affair.

In popular magazines it is sometimes stated, based on the thoughtless extrapolation of Moore's Law, that around 2020 a chip will have 100 billion (10^{11}) components, that is the number of neurons in the human brain. Even less plausible is the conclusion that such a chip would exhibit human-like intelligence. For, as we have seen before, there is a vast gap between our capabilities to built complex hardware, and our insight in how to exploit this complexity. The existence of a chronical 'software crisis', and in particular the lack of significant progress in Artificial Intelligence research during the past 25 years should make us very cautious in making predictions of this kind.

New alleys in the field of hardware technology are being explored under the denominators of cryogenic and optical systems (the nanostructures refered to above are very suitable to couple light-based computing systems to electronic computers), and 'quantum computers'. However, it seems that today the main thrust in computer technology development is based on the exploration and exploitation of the capabilities of existing hardware techniques - at the same time making the investments in these techniques productive. Parallel and distributed computing are very important topics of research and some of the results have been commercialized; it should be noted that the development of suitable software is the main bottelneck in these endeavors. New application domains like virtual reality, multimedia technology (games!), medical image processing, intelligent autonomous systems (robots) for industrial use, large scale simulation are eager customers for the developments in these areas.

Also we can expect to have in the foreseeable future facilities for the production of small quantities of customized chips for very special applications, owing to the automatization of design- and manufacturing techniques.

References

[1] J.G. Brainerd, T.K. Sharpless: The ENIAC. Electrical Engineering 67(1948). Reprint in Proc IEEE 72 (1984) pp 1203-1212.

[2] C. Mead, L. Conway: Introduction to VLSI systems. Addison-Wesley 1980.

[3] G.E. Moore: Cramming more components onto integrated circuits. Electronics 38 (1965) no.8

[4] C. Gordon Bell eo.: Computer Engineering: A DEC view of hardware systems design. Digital Equipment Corporation 1978.

[5] P. Christie: A fractal analysis of interconnection complexity. Proc IEEE 81 (1993) pp 1492-1499.

Note added 2016

This article was conceived in 1994 as internal report CS-94-17. The present version has some minor corrections, however no attempt was made to include recent developments.
The substance of the text is believed to be still valid today.