The Belgian Electronic Mathematical Machine (1951-1962): An Account

Pierre-Jacques Courtois

Institute of Information & Communication Technologies,
Electronics & Applied Mathematics (ICTEAM),
Université catholique de Louvain (UCL),
Louvain-la-Neuve, Belgium

Abstract. Dedicated to Brian Randell, Emeritus Professor of Newcastle upon Tyne University, in commemoration of his 75th birthday, this paper evokes the origins and the development of an electronic computer realized in Antwerp (Belgium) in the early 1950's. Our main focus is on some interesting archetypes and advanced aspects of the design, and on how a small group of Belgians engineers developed this early machine.

Keywords: computer history, IRSIA FNRS computer, Belevitch, Belgium.

1 Introduction

This paper is a contribution to a *Festschrift für Brian Randell, Emeritus Professor der Universität Newcastle am Tyne*, to commemorate his 75th birthday. Very early on in his professional life, Brian became interested and deeply involved in the history of computing machines. In 1982, the third edition of his important book appeared [15]: an impressive collection of papers describing, in great detail, the origins and the designs of a wide variety of early machines, developed during a period that started from the Babbage analytical engine in 1834 and ran until May 1949 when the EDSAC computer executed its first program at Cambridge University. The ESDAC was built by a team headed by Maurice Wilkes and is generally accepted as the first practical stored program computer to be completed ([15], page 353).

In his introduction, Brian explains that he did not attempt to cover the growth of electronic computers which followed, in the late 1940's and early 1950's. In that period, several projects started based on the EDVAC, the computer which began operation in 1951 at the US Army's Ballistic Research Laboratory, Aberdeen, Maryland, and to which John Mauchly, J. Presper Eckert and John von Neumann had initially greatly contributed.

Here, we briefly evoke the origins and the development of an electronic machine, also realized in the early fifties, in Antwerp. The machine remained largely unknown outside Belgium, at least until recently [14]. The paper sheds light on some interesting archetypes and clever features of the design. It is also about the pioneering work of a small group of Belgians engineers who developed the hardware and the basic software.

C.B. Jones and J.L. Lloyd (Eds.): Festschrift Randell, LNCS 6875, pp. 225–237, 2011.

2 The Bell Telephone (BTMC), Antwerp

In the middle of the last century, the development of programmable computers in many parts of the world was often initiated under very similar conditions. The UK and Belgium were no exception. During the last world war, on both sides of the channel, some mathematicians and engineers were confronted with giant manual calculations. They had, however, different objectives: breaking military codes in Bletchley Park, and synthesizing complex electrical filters in Antwerp. Although they work independently, their problems and their manual procedures were quite similar. The computing machines that they eventually invented to solve their problems were put into service at times less than five years apart.

In Belgium, the history of computing goes back to the middle of the last world war, at the *Bell Telephone Manufacturing Company* (BTMC), in Antwerp. BTMC, a Belgian company created in 1882, was a major subsidiary of the *International Telephone and Telegraph Company* (ITT) in New York. ITT had another subsidiary in Berlin: *Mix & Genest,* a former German company. At the beginning of the war, ITT changed the status of BTMC from a direct subsidiary of ITT, New York, to a subsidiary of Mix *& Genest* in such a way that at the time of the invasion of Belgium, BTMC was regarded German, not enemy property.

At that time, the calculation of the properties or parameters of an electrical filter was manual, intricate, requiring high numerical accuracy and thus was time-consuming. Alfred Fettweis was then working at BTMC Laboratory of Transmission. He is now Professor Emeritus at the University of Bochum in Germany and an internationally renowned expert in circuit theory. In [8], he talks about long hours of human labour on the task of designing a new type of filter in a higher frequency range. Equipped with a desk electro-mechanical calculator only, the job took him two full months. A young girl was then assigned to help with the calculations. Because she had only a secondary school diploma and not the slightest notion of what the poles of a polynomial equation might be, the computation had to be decomposed into small elementary steps that she could understand. It was an embryonic form of programming.

The situation was not different elsewhere. For instance, in Bletchley Park, England, where Alan Turing and other mathematicians worked during the Second World War on decipherment in the British army General Code and Cipher School, and managed to break the complex German Enigma code, a large number of women were employed from the Women's Royal Naval Service [9]. In those days the word 'computer' had in fact a different meaning from today. It simply meant a person, most often a woman[1], engaged in calculations aided by such tools as an abacus or a desk machine.

3 The Project

In 1951, the Belgian National Fund for Scientific Research (FNRS) and the Institute for the Promotion of Scientific Research in Industry and Agriculture (IRSIA)[2] decided

[1] This might be one of the origins of the use of the pronoun 'she' to refer to a computer (according to [9], page 55).

[2] *Le Fonds National de la Recherche Scientifique* (FNRS) et l'*Institut pour l'Encouragement de la Recherche Scientifique dans l'Industrie et l'Agriculture* (IRSIA).

to subsidize the construction of an "electronic mathematical machine in order to promote the study and construction of similar equipments in Belgium". One should not forget that programmable calculation machines were radically new at the time; as radically different from their precursors as the jet engine, which appeared about at the same time, was from the piston engine.

Charles Manneback (1894-1975) [5] was then professor of mathematical physics and electricity at the engineering school of Louvain university. Before the war Manneback had met some of the most distinguished physicists of this extraordinary period: Werner Heisenberg, Niels Bohr, and Enrico Fermi. He was also a colleague and friend of Georges Lemaitre (1894-1966), the priest who invented the big-bang.

Manneback had been to Harvard in 1946 to become familiar with the early developments of electronic computers and, in 1947, had published with Léon Brillouin a study report called "*Les Machines Mathématiques aux Etats-Unis*" [13]. Quite naturally, he was thereby appointed chairman of a committee responsible for the FNRS-IRSIA project. Two members of this committee again visited the Harvard laboratory of Professor Howard Aiken. In 1943, with funding from IBM and the US Navy, Aiken had built an electromechanical computer: the ASCC later renamed Harvard Mark I. In 1952, at the time of the visit, he was completing the construction of the Mark IV.

Right after this visit, the design and implementation of the Belgian computer was assigned to BTMC. Although the times were similar there would be differences from the UK and US projects mentioned above, because of the nature of the driving applications. The Belgian one required numerical calculations approximating real numbers - today floating point numbers. Turing's application did not require that. Aitken was experienced in, partially funded by, and motivated by calculation of utilities bills. This would lean more towards fixed point calculations.

The computer was to be one of the first electronic stored program computers built in Europe and regarded as a universal calculator [17]. Left unnamed, people referred to it as the FNRS-IRSIA machine. It followed the machine Z3 built in 1941 by Konrad Zuse in Germany, the Small Scale Experimental Machine *("Baby")* in 1948 at the University of Manchester, and the machines EDSAC (Cambridge, 1949), Mark 1 (Manchester, 1949) and ACE (NPL, 1951) in England.

Early in the project, Manneback remembered one of his former students, Vitold Belevitch (1921-1999), a young engineer from Louvain University who had joined BTMC in 1942, immediately after his studies. A bright and precocious student, Belevitch had enrolled at the university at the age of 16. There, starting in 1939, he had followed the teaching of Manneback and presented, under Manneback's direction both his master thesis in electrical and mechanical engineering in 1942, and his doctoral thesis in Applied Sciences in 1945, at the age of 24. He was working at BTMC on the design of equipment for telephony and telegraphy; and had become chief of the Laboratory of Transmission. Belevitch is still well known world-wide for his discoveries and fundamental work in mathematics, circuit theory and electrical filters design [6].

Because Belevitch's technical and mathematical skills were well recognized in Bell Telephone, Manneback called him to participate in the project, as part of the group in charge of designing the computer software. Belevitch became the soul of the project. Later, he did not fail, however, to recognize that Manneback's extensive experience in

numerical analysis had been a precious help. Other key protagonists were M. Linsman and W. Pouliart who directed the project, A. Fisher, P. Dagnelie and Claude Fosséprez in charge of the design and maintenance of the circuits and storage devices, and F. Storrer who worked on the design of the numerical programs.

The preponderant role that Belevitch played in the design and the use of the prototype is shown by several of his publications of the period: *Machines Mathématiques Electroniques,* published in the *Journal des Ecoles Spéciales* in Leuven in April 1952; *Le trafic des nombres et des ordres dans la machine IRSIA-FNRS* published in 1955 in *Nachrichtentechnische Fachberichte*, and what became the "breviary" of the engineers and programmers: *Le calcul numérique des fonctions élémentaires dans la machine mathématique IRSIA-FNRS,* which appeared in 1956 in the *Bulletin de l'Académie Royale Belge des Sciences.*

These publications, and others by his colleagues, demonstrate expertise and knowledge of computational problems and computer architecture which compete on equal terms with the experience then acquired in the United States and Europe.

In the following, on the basis of these publications, we briefly evoke some of the most salient and interesting features of the design.

4 The Machinery

The IRSIA-FNRS computer was made of approximately 1000 triodes, 1200 cold cathode gas tubes, 400 relays, 1000 selenium rectifier diodes and 500 germanium diodes [11][3]. There was no core memory. The main memory was a magnetic drum and the auxiliary memory consisted of tape drives. The drum had two fixed physical partitions: one for the instructions and one for the data, stored separately like those in the machines of the Mark series. To improve performances, the two partitions could be accessed in parallel by the processing unit. Their roles could be exchanged; the content of a register of the processing unit determined which contained what. In this sense the machine had globally the major characteristic of a "von Neumann machine", i.e. the use of one memory for both code and data. Each drum partition had 100 tracks, each divided in 20 sectors of one word each. Thus, each partition contained 2000 words; a word contained either a number of 18 tetrads or two instructions of 9 tetrads. The drum was directly addressable by the processing unit. A word position was defined by an address of four decimal digits: two (00 to 99) for the longitudinal address (the track number) and two (00 to 95) for the angular address (the sector number) with an increment of five so as to jump automatically in sequence to the next track. Four index registers of the central processing unit were available for the calculation of drum address displacements and for the memorization of return addresses.

The drum cylinder, in nickel-plated aluminum, was one meter long with a 50cms diameter, and turned at a speed of approximately 4100 RPM. Each partition had 100 tracks 3mms apart, each of 1440 bits (i.e. of dipoles) 1mm apart that could be read or written in 10μs by a reading and a writing coil respectively, in diametrically opposed positions on the track.

---

[3] For comparison, the total of valves and thyratrons of CPU custom circuits were approximately of 1400 in the Mark I and 2400 in the Mark II computers.

The auxiliary memory consisted of six tape loops, each 100 meters long, and of a capacity of about one Mbits (10 bits per mm). The tapes stored data from input coding devices and for output printing machines. Each tape was wound around a hollow driving barrel which was covered by a cap. The tape movement was controlled by a pneumatic clutch. Air valves controlled the relative air pressures in the barrel and in the cap. A depression in the barrel pressed the tape against the barrel and drove it; on the contrary, an overpressure in the barrel – i.e. a depression in the cap - applied the tape against the cap and stopped it. Coming from a driving speed of about 1m/sec to a brutal halt and vice-versa took no more than about 2,5 msec [10]. Each tape fell and loosely accumulated in a magazine under the barrel.

The final version of the machine was 13 meters long and 2.5 meters high. A prototype was completed in late 1954 and inaugurated in the presence of the Belgian King *Baudouin* in February 1955. The demonstration prepared for the royal visit was the calculation of the square root of 2. An unfortunate crash of the hardware during the demonstration forced resort to a simulacrum of the calculations with the help of a pre-recorded tape containing the result data; an artifice that Belevitch later never failed to recommend to his engineers as a back-up for official demonstrations. The construction of the machine was eventually completed by the end of 1956 [5].

5 The Architecture

All the information, data and addresses, was coded in decimal notation, every decimal digit being a tetrad of a bi-quinary code [3], with one ‘bi’ bit and three ‘quinary’ bits:

	abcd		abcd
(0)	0000	(5)	1000
(1)	0001	(6)	1001
(2)	0010	(7)	1010
(3)	0101	(8)	1101
(4)	0100	(9)	1100

Biquinary codes had been used in other early machines such as the Colossus code breaking computer operational at Bletchey Park during the war, and the 7 bit IBM 650 machine, but with other representations. The particular choice of the Belgian code was the result of a compromise between several constraints: to keep to a minimum the number of symbols; to have 0000 for the digit (0); to minimize the number of 1’s so as to minimize energy consumption in the electronic tubes, and the number of successive 1’s to increase the reliability of magnetic recording materials; and, more importantly, for an easy calculation of the complement to 9. This complement was used to reduce the subtraction operation to the addition, and was obtained by simply replacing 1 by 0 and vice-versa in the positions a and b. The error made between the digits (2) and (7) was easily rectified since these two tetrads can be uniquely identified by the c and d positions only.

Belevitch also observed in [3] that the tetrads excluded by the code remain unused in the machine and suggested that they make possible the transformation of numbers

in instructions and vice-versa, as well as calculations on instruction codes, practices that would be distasteful to software engineers today.

The machine was a single address instruction computer. A word was either a number of 18 tetrads or a pair of single address instructions of 9 tetrads each (5 for the code and 4 for the address). We discuss some of the motivations for this particular arrangement at the end of Section 7.

The decimal tetrad code gave the possibility of both fixed and floating point calculations. A machine word was also a number of 18 tetrads with 15 tetrads for the mantissa, two for the exponent, and one for the signs. The exponent was limited to 49 so as to avoid a carry over in the addition of two exponents; the occurrence of a value above 49 caused an alarm. In a previous proposal, the mantissa had 18 tetrads with two additional tetrads for the exponent and signs. The exponent could vary from – 29 to +29, but a negative exponent was coded as the complement to 100; the exponent thus took values 00 to 29 and 71 to 99. Its first decimal digit could thus take 6 distinct values for which three distinct binary values were sufficient; the fourth symbol of the corresponding tetrad was then reserved for the sign of the number. This clever proposal, however, was not retained. The accuracy increase of the mantissa probably justified neither the reduction in number range nor the additional complexity of the logic for the exponent calculations.

Calculations were made in decimal floating point so as to keep a maximum of significant digits in the mantissa. A result such as $0{,}004\ldots10^{00}$ was realigned by the machine as $0{,}4\ldots10^{-02}$. Belevitch remarks in [3] that the last decimal digit of such re-alignments must be "invented", even if it is a priori decided to assign it with the value 0. In order to avoid the possible accumulation of systematic errors that might compromise the advantage of floating point calculations, he suggested the use of a special tetrad, noted "*u*" (perhaps meant for "unknown"), that would signal the end of the number. The computer would calculate on "*u*" as if it were 0, but would reassign an "*u"* tetrad to the result at a rank which would be function of the operation and of the *u* ranks in the terms. In the calculator, he wrote, "*u*" would somehow be an "isotope" of 0. Whether or not this proposal eventually influenced the final design or the programming of numerical functions remains also an "*u*".

6 The Flow of Numbers and Instructions

The design of the processing units was mainly constrained by two objectives: balancing the great difference in speed between the slow magnetic memories and the electronic tube circuits, and keeping the logic of these circuits simple.

A track of the drum memory contained 20 sectors of one word, thus $20\times18=360$ tetrads, or one tetrad per degree, making time synchronization easier. A tetrad was sequentially read or written in 40 µs and processed in parallel in the logic circuits. The total time for reading/writing a word on a cylinder was thus $40\times18=720$µs, the total time of a drum revolution $20\times720=14{,}4$ ms, so that the average access time to a word, half a revolution, took 7,2 ms. A special track on the drum, continuously read, provided a 10µs time base synchronization and signals of period 720µs indicating the origins of the sectors. Each signal was followed by a code giving the angular address of the corresponding sector. As for the magnetic tapes, the average time to read or write a number was about 7 ms.

Accommodating these slow memory access times with the comparatively high switching rate of the processor electronic tube circuits was not an easy task. It was achieved by means of two input buffer registers O and T_1, and one output buffer T_2, each containing one word. The word with the next two instructions (the French word *ordres* was used) of a program sequence was read into the register O from the instruction drum partition, while T_1 contained the operand (number), read from the number drum partition, for the instruction being currently executed. Similarly the result of an instruction was buffered and written back on the number drum via an output register T_2. The four bits of a tetrad were transferred and processed in parallel. The separation of the drum into two partitions allowed the flows of instructions and of numbers between the drum and the processor to be independent and asynchronous. In addition, the input buffer T_1 was organized as a shift register. The head instruction contained in its first half word, once decoded and the address index calculations done, was replaced by the tail instruction. The latter was decoded and its execution also started as soon as the operand buffer T_1 was freed from the previous operand, its contents having been exploited by the arithmetic unit. An instruction, such as a jump, involving no calculation by the arithmetic unit took at most one cycle of 720μs. As we will see in the next section, for most other instructions, on the average, the execution time was essentially that of the arithmetic operations, and not access times to the drum.

Thus, it was a sort of look-ahead pipeline processor, ahead of its time. The price to pay was the obligation for jump instructions to always address the first of the pair of instructions contained in a word; when necessary, dummy instructions could be inserted in the code to circumvent the constraint [3]. Address calculations used four index registers which could be loaded with the current address and results of the arithmetic unit. They provided the necessary functionality for making jumps with the memorization of the return address, for transforming numbers into instructions and conversely, and for programming floating point calculations. Altogether a bunch of quite complex control circuits.

7 The Arithmetic Unit

In comparison with the control and synchronizing circuits, the design of the arithmetic unit is much simpler, primarily because it was based on a rigorous model.

The unit consisted of arrays of selenium diodes. Two arrays delivered sums of the two decimal digit tetrads on their inputs; one delivered a product; one computed a 9-complement and a last one retained and delayed the carry over of the operations. These arrays were assembled in a single circuit which processed one pair of digits at a time (see Fig.1), starting from the less significant tetrad of a number, and executing the operation:

$$A \times B + C + R \tag{1}$$

where A, B and C are three shift input registers of cold cathode gas tubes, each containing a number. R is an output register retaining the most significant tetrad possibly carried over from the operation on the previous pair of tetrads. Operation (1) reduces to an addition for B = 1, and to a multiplication for C = 0.

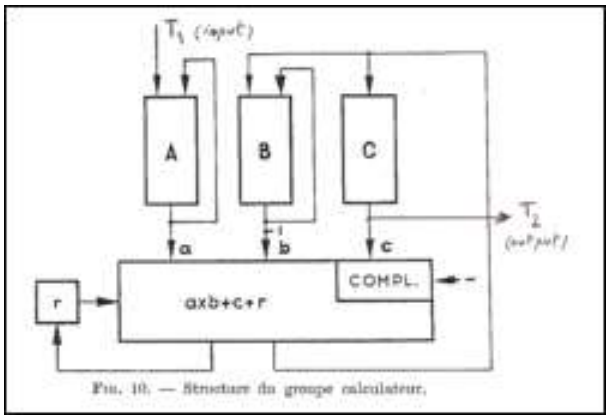

Fig. 1. The structure of the arithmetic unit. *a, b, c, r* are symbols for tetrads.

The new number to be processed is loaded into the register A from the register T_1. C is the accumulator which eventually retains the final result of the last completed arithmetic operation, and is transferred to the register T_2; T_1 and T_2 are the transfer registers mentioned in the previous section. For an addition, the two numbers in A and C are entered, one tetrad at a time being unlatched into the addition circuit, starting with the less significant digits, while 1 is loaded onto B; the results are accumulated in C. For subtraction, done by the method of complements, the number in C is first replaced by its complement to 9. At the end, the contents of R, if equal to 1, are added to C; otherwise the content of C is replaced by its 9's complement. In the first case, the result has the opposite sign of the initial content of C; and the same sign in the second case.

The multiplication was similar to the way we calculate on paper. The multiplicand is loaded into A, the multiplier into B, and C is emptied. Synchronizing pulses first unlatch the least significant tetrad of B and successively all the tetrads from A, each being multiplied by B, the carry over of each two tetrads multiplication being delayed and added to the result of the next one. A is shifted onto itself so as to retain its contents. After a first round of A, the first partial product is obtained and added to the contents of C. The next tetrad of B is then loaded into the multiplier and again multiplied to the contents of A. The new partial product obtained is shifted one tetrad and added with C. In the end, the final result is in C; A and C have made 20 rounds while B has been shifted one tetrad each round.

The time for an elementary operation on two tetrads was about 25 µs. The time to make one tetrad shift in registers was 20 µs. Tetrads were shifted and unlatched from the registers into the table circuits with a period of 40µs. The total execution times of the addition and multiplication of two floating point numbers amount to approximately 3 and 15 milliseconds, respectively [11].

Thus, the execution times for multiplications and for the processing of words of pairs of instructions were approximately in the same range of magnitude as (multiples of) the mean access time to a drum sector (7,2 ms). As a result and thanks to the option of loading the arithmetic unit with two instructions at a time, there was no pressing need for optimizing the arrangement of instructions on the drum to prevent rotating memory latency from slowing the program down. In contrast, this optimization was required on the IBM 650, another decimal computer of

the same period (1953), which also used a rotating direct access memory and had instructions with two addresses; the second address was for the next instruction. The programmer, or an optimizing assembler, had to figure out where the drum would be when the current instruction finishes executing, and store the next instruction at that sector.

8 Elementary Functions

The arithmetic unit could combine numbers by addition or multiplication only, but could process either fixed or floating point numbers, since the exponents, mantissas and signs were stored separately in the shift registers A, B and C. The operations of division, square root, the trigonometric functions and other usual transcendental functions were executed by programmed sequences that Belevitch baptised *fonctions élémentaires*. With Freddy Storrer, a young mathematician from Ghent University, he paid much attention to the selection and numerical implementation of a set of such basic functions, sufficiently universal so that many others could be reduced to the set. In the process, they improved classical numerical procedures and even discovered original results, in particular an improvement of the division process using the Newton-Raphson method [18].

Their objective was to guarantee that all functions are evaluated with a relative accuracy of 10^{-14}, an optimum since the machine was capable of representing results with 15 decimals in floating point notation. Their main preoccupation was to maintain small the cumulative relative error in the calculations by avoiding subtractions of very close numbers. They had mainly recourse to Tchebycheff approximations of Mac Laurin developments around the zeros of the functions. So, among the basic functions selected were sin x, arc tg x, 10^x, and $\log_{10} x$; and for small values of the argument x, the last two functions were obtained by the intermediary calculations of the functions $10^x - 1$ and $\log_{10} (1 + x)$, respectively.

The problem of division was reduced to the subtraction of exponents and the calculation of the inverse of the mantissa $y = 1/x$ in the x-interval [0,1, 1]. The calculation was done by the iteration

$$y_n = y_{n-1} (a_n - x y_{n-1}), \tag{2}$$

where y_0 is a first polynomial approximation, and a_n is the constant 2 in the Newton-Raphson iteration. Storrer found a formula [18] which improves this constant; a_n is derived at each iteration from the relative error made at the previous one. They also observed [2] that, when the degree of y_0 increases, the number of iterations (2) needed to reach an accuracy of 10^{-14} decreases and that, for a certain degree, the number of instructions of the program is minimal. By trial and error, they found that the minimum is obtained for a polynomial expression y_0 of first degree, and that five iterations only were then necessary to guarantee a relative error $e_5 = 3.10^{-16}$.

For the square root calculation, they used a formula of type similar to (2) but found more convenient to adopt $y = 1/\sqrt{x}$ as the elementary function and to obtain $\sqrt{x}$ by xy. A detailed analysis [2] proves a final relative error $e_4 < 1{,}75.10^{-14}$ after only four iterations.

Because of the poor reliability of the electronic circuits, there was already a great awareness of the necessity of controlling the validity of the results. To improve the dependability of the machine, systematic verifications were implemented[4]. For instance, the calculation of the inverse $y = 1/x$ was monitored at run-time by verifying the result with a formula that was derived from the expression of the relative error e_5 mentioned above:

$$|xy - 1| \leq 2.10^{-14} . \quad (3)$$

Similarly, square roots were checked against the tolerance derived from the relative error e_4:

$$|xy^2 - 1| \leq 4.10^{-14} . \quad (4)$$

Other checks were performed by computing functions in two different ways and comparing results; or at two close points and comparing the difference between the two values with that of a known delta of the function. Should any of these tests fail, an alarm was issued or a jump made to an additional iteration or to an alternative computation.

9 The Art of Logic: Programming

This splendid definition of programming is the title of a short paper published by Belevitch in 1958 [4]; a paper which stands the test of time and would still be of interest to those who wish to learn or recall the basics of the design of a processing unit and the nature of the difficulties inherent to its programming[5].

In contrast with the classic search for systems and languages that bring the machine closer to the level of the user, Belevitch takes the opposing view. His premises are the basic arithmetic and elementary logic operations that are needed to reduce irrational or transcendental operations to a succession of finite algorithms. From these necessary operations, he carefully and systematically derives the functional and structural characteristics of an efficient processor and of its interconnections with the memory: accumulator, floating point and index registers, address calculations, etc. On this basis, he describes what is then felt as being the most difficult work for the programmer: a tricky task of organization, independent of any language. A work which is function of the machine structure, of the application problem, which must be reduced into a topological structure of interrelated simple computation sequences, and of the problem of dynamic allocation of memory space to intermediary results. The latter issue is considered similar to the operation research problem of allocating coaches in a railway network.

---

[4] Such verifications should be especially appealing to Brian Randell who devoted quite of his research efforts to software fault-tolerance and dependable computing. These checks would have been effective instruments to activate his “recovery blocks” [16].

[5] An exception is the discussion of certain optimizations, like the possibilities of modifying instructions during execution; today they would be against good practice and no longer justified except perhaps under severe performances constraints.

In accordance with such austere views, the capabilities of the arithmetic unit were restricted to the combination of numbers and of their exponents by addition, subtraction and multiplication. At the level of the machine code, the tools made available for assisting programming were also extremely limited. A few special instructions called "*alterations*" had been selected for their usefulness in the calculations of the elementary functions. *Alterations* could change the sign of an exponent (useful for division) or a mantissa; an exponent could be changed into a mantissa (useful for the logarithm), etc…

Two kinds of jumps were available to transfer control between computation sequences: "*ordres de renvoi*" were unconditional jumps with memorization of the return address; "*bifurcations*" were jumps conditioned on the sign ± of a number. *Bifurcations* were, in particular, useful to program the elementary functions and the verifications mentioned in the previous section.

10 Epilogue

In 1955, the IRSIA and the FNRS set up in Antwerp BTMC premises an Institute, the *Comité d'Etude et d'exploitation des Calculateurs* (CECE), with the mission of maintaining the machine and prospecting the opportunities for Belgium to exploit and develop computers. Belevitch was appointed director. He oriented the activities to research in programming and numerical analysis. Access to the machine was free for national institutions such as the Belgian meteorological and the military geographical institutes.

In the spring of 1958 the calculator was transferred from Antwerp to the *Institut National de Statistiques*, in Brussels, where it remained operational until 1962. Despite the mediocre reliability of the components and problems caused by the heat generated by the electronic tubes, a variety of scientific complex calculations were successfully made [7]. The machine, in spite of its hardware and software complexity, had been realized by a small group of engineers. The logic had been well thought of and was rather simple, but the implementation full of thorny issues. It was yet another example of Maurice V. Wilkes's observation, the architect of the ESDAC, when he compared in one of his last interviews the contributions of mathematicians and engineers in the design of the early computers: "… *Mathematicians weren't particularly qualified… You can't design or build a computer, unless you are an engineer. … There wasn't any use [for Boolean algebra]. Boolean algebra has no time element to it and while it is good for shaking up a bit of complex logic we didn't have complex logic. We all had very simple logic in the early days.* [1]*"*

Meanwhile, taking advantage of its experience, in 1956 BTMC started the development of a second computer, in response to a tender from the New-York *First National City Bank.* This computer had a similar structure but was made of transistors, which eventually turn out to be more reliable than the electronic tubes. Jacques Loeckx was responsible for the development of these transistor circuits [12]. The computer was supposed to control specific electromechanical input-output devices for the automatic handling of magnetised paper banking documents. Unfortunately, these devices never worked satisfactorily and the City Bank eventually abandoned the project. The story goes that, as the computer had already been shipped

to the US, in order to avoid custom taxes on both sides, both parties agreed that the best solution was to dump the machine in the international waters of the Atlantic.

Nevertheless, the pioneering work achieved with the construction of the IRSIA-FNRS machine was the catalyst and the origin of the development of computer science in Belgium in the following years. When the CECE was dissolved at the end of 1962, Belevitch set up and became director of the MBLE[6]/Philips Research Laboratory in Brussels. Rapidly, the laboratory acquired an international reputation for its achievements in filter theory and analysis, applied mathematics, coding theory, logical circuit design, programming languages and operating systems [6,12]. In Belgium as elsewhere, those years were the "Golden Sixties", "*Les Trente Glorieuses*". Another era, another story.

Acknowledgments. Dave Parnas made very useful comments on an earlier draft. Stephane Deconinck, in charge of the UCL Computer Science department library was of great help to find and make available the necessary documentation, and John Lloyd, University of Newcastle upon Tyne, suggested helpful editorial changes.

References

1. Anderson, D.P.: An Interview with Maurice Wilkes. Communications of the ACM 52(9), 39–42 (2009)
2. Belevitch, V., Storrer, F.: Le calcul numérique des fonctions élémentaires dans la machine mathématique IRSIA-FNRS. Bull. Acad. Roy. Belg., Classe des Sc., t. XLII, Bruxelles, pp. 543–578 (1956)
3. Belevitch, V.: Le Trafic des Nombres et des Ordres dans la Machine IRSIA-FNRS. NTF 4 (1956)
4. Belevitch, V.: L'Art de la Logique: La Programmation. Encyclopédie des Sciences Modernes de Genève, Kister, VIII, pp. 89–90 (1958)
5. Biot, M.A., Manneback, C.: Florilège des Sciences en Belgique, Académie Royale de Belgique, Classe des Sciences, pp. 371–377 (1980)
6. Courtois, P.-J.: Notice Biographique de Vitold Belevitch, Nouvelle Biographie Nationale, Tome X, Académie Royale des sciences, des lettres et des beaux-arts de Belgique, Bruxelles, pp. 35–42 (2010)
7. Courtois, P.J.: Vitold Belevitch (1921-1999), Témoignages et Réminisciences. Institutional Repository of the 'Académie Universitaire Louvain' (DIAL), (en préparation)
8. Fettweis, A.: In Memoriam, Vitold Belevitch. IEEE Transactions on Circuits and Systems 47, 613–614 (2000)
9. Leavitt, D.: The Man Who Knew Too much. Alan Turing and the Invention of the Computer, Phoenix (2006)
10. Linsman, M.: Les Machines Mathématiques. Bull. A.E.E.S. Liège, Technique et Humanisme 6, 279–294 (1954-1955)
11. Linsman, M., Pouliart, W.: La Machine mathématique IRSIA – FNRS, *en construction* à la Bell Telephone Mfg C. Industrie 8, 505–509 (1953)

[6] Manufacture Belge de Lampes Electroniques.

12. Loeckx, J.: L'informatique en Belgique dans les années 1950 à 1970. Une rétrospective basée sur des réminiscences personnelles. Presses Universitaires de Namur, Namur (to appear, 2011)
13. Manneback, C., Brillouin L.: Les machines mathématiques aux Etats-Unis. Archives Dockx, Facultés Universitaires Notre-Dame de la Paix, Namur, typed-script report (1947)
14. Mols, S., d'Udekem-Gevers, M.: Disseminating electronics: Bell Telephone and the emergence of electronic computing expertise in post-war Belgium, c1945-c1960. In: Histelcon Conference, Paris (2008)
15. Randell, B. (ed.): The Origins of Digital Computers. Selected Papers, 2nd edn. Springer, Heidelberg (1975)
16. Randell, B., Xu, J.: The Evolution of the Recovery Block Concept. In: Lyu, M. (ed.) Software Fault Tolerance. Trends in Software, pp. 1–22. J. Wiley, Chichester (1994)
17. Rojas, R.: How to make Zuse's Z3 a universal computer. IEEE Annals of the History of Computing 20(3), 51–54 (1998)
18. Storrer, F.: Amélioration du procédé de division utilisant l'itération de Newton-Raphson. Bull.Acad. Roy. Belg., Classe des Sc., 5^{e} série, t. XLII, Bruxelles, pp. 30–33 (1956)