Percy Ludgate's Analytical Machine

Brian Coghlan

Abstract This document attempts to deduce additional detail about Percy Ludgate's Analytical Machine by contextual analysis of both his 1909 paper and recently discovered articles. The aim is to assemble a body of known new design facts. Thus far, this has been found possible in relation to his logarithmic indexes, data format, Index and Mill, carry propagation, timing, division algorithm, storage, instruction set and preemption. The approach has proven to be surprisingly fruitful, and has allowed some progress to be made towards a 3D-printed re-imagining of the machine. **This paper is a work in progress.**

***Index Terms*— Percy Ludgate, Analytical Machine, Irish Logarithms, Multiply-Accumulate (MAC), Division by Convergent Series**

I. INTRODUCTION

Historically, computing is generally viewed as having four fore-fathers: Charles Babbage (1843: the concept an analytical engine [4]), George Boole (1854: Boolean logic [24]), Alan Turing (1936: theory of computing [25]), Claude Shannon (1937: digital logic [26] and 1948: information theory [27]). Boole, Turing & Shannon impact on all aspects of modern society (principally because computing now does), Babbage does not. Nonetheless Babbage is very notable for introducing the concept of the first analytical engine in history. His importance is historical. He also raised the idea of thinking machines, a controversial subject then, like AI is today.

Babbage's "Analytical Engine" [53] was a very novel concept, with processing done in a Mill based on addition, where the Mill and Storage were made of clockwork cogs and gears (Liebniz c.1671 [13]). Programming and input/output were done via punched cards (Jacquard c.1804 [14]). Ada Lovelace published Babbage's design in 1843 [4]. Only a small portion of it was ever built, but Babbage left very extensive 2-d drawings (for 3-d graphic art representations, see [12]), and these are now being put into modern engineering software [15][18], so perhaps more of it will yet be built.

Percy Edwin Ludgate (1883-1922) is notable as the second person to publish a design for his "Analytical Machine" [1][2][16], the first after Lovelace published Babbage's design. Ludgate's machine included a Mill, but also an "Index" to do multiplication. The combination did multiply-accumulation (MAC), which is now important in signal processing [19], e.g. in radar and astronomy, and in deep AI [20]. The multiply embodied a discrete table, as did a few other calculating machines at this time, such as the Millionaire [21]. But it was based on a novel concept now called "Irish Logarithms", and worked analogously to a slide rule (Oughtred c.1622 [23]). The accumulation worked like Napier's "bones" (Napier c.1614 [22]), otherwise called "rods", once widely used to facilitate the accumulation of partial product units and tens digits inscribed on their upper surface.

These were the only two mechanical designs well before the electronic computer era. Subsequently, c.1914 electromechanical designs began, and c.1937 electronic designs began [11]. A third mechanical design arose at the dawn of the electronic computer era: in 1936 Zuse's "V1", later renamed "Z1". Historically there are four types of Analytical Machines: Babbage (mechanical, novel, 1843), Ludgate (mechanical, very different, 1909), Torres y Quevedo and his successors (electromechanical, 1914 onwards [7]), then modern computers and their billions of successors (fully electronic, 1949 onwards). Analytical machines now pervade, running big data, big science, health, business, government, and now domestic appliances, mobile phones, and even toys. Ludgate is notable for the second published analytical machine design in history, and Irish Logarithmic indexes, and the first multiply-accumulator (MAC) in a computer, and the first division by convergent series in a computer, and very novel concepts for storage and programming (e.g. two-operand instructions, pre-emption). Like Babbage, his importance is historical.

Ludgate and Babbage appear to have had no influence on modern computers, since as far as is known there is no record in the literature of modern electronic computers that specifically states inheritance from either of their machines. Randell's 1971 paper [6] was followed by just one attempt, in 1973, by Riches, at a conjectural machine design based on Ludgate's design [42]. Copies of most of the related literature are held by The John Gabriel Byrne Computer Science Collection [8] (and its host department presents an annual Ludgate Prize [9]). This and [10] outline the initial technical results of an investigation by the Collection (and [17] outlines its initial historical results). The extremely few other technically-related efforts are referred to further below.

Dr.Brian Coghlan is with the School of Computer Science and Statistics, Trinity College Dublin, The University of Dublin, Ireland (e-mail: coghlan@cs.tcd.ie).

II. Aspects of Ludgate's Machine

II.1. Features

Only a few features are described in Ludgate's 1909 paper. The rest must be deduced by contextual analysis of the paper, largely a process of logical inference or elimination of false propositions, and mechanical or mathematical speculations. They are summarised in Table 1, but explored further below.

	Base operation is **multiply-accumulate (MAC)** not addition
	Multiply is done with **Irish Logarithms** by an **INDEX**
	Long **multiply starts at left digit** of multiplier
¥	Numbers must be **fixed-point**
	Multiply-accumulate partial-products are added **units first, then tens** by a **MILL**
¥	Timing implies **pipelining** tens carry adds
¥	Instruction set: **ADD, SUBTRACT, MULTIPLY, DIVIDE, LOG, STORE, CONDITIONAL BRANCH**
	Two-operand addressing for load
¥	**Two-operand addressing for store**
	Fast for 1909: ADD / SUB 3 sec, MUL 10 sec, DIV 90 sec, LOG 120 sec
¥	Storage of 192 variables implies **(64 inner + 128 outer) shuttles**, equispaced
¥	Hence storage size implies **binary storage addressing**
	Numbers stored via **rod for sign & every digit** protruding 1–10 units
	Data input / output via **punched number-paper** (or upper keyboard)
	Program input / output via **punched formula-paper** (or lower keyboard), one instruction per row
¥	**Manual preemption**
¥	Measurements in **units of an eighth of an inch**
¥	Main shaft gives **'cycle time' of a third of a second**
	Small size: estimated by Ludgate as 0.5m H x 0.7m L x 0.6m W

Table 1 *Features of Percy Ludgate's analytical engine, with inferences denoted by ¥*

II.2. Unknowns

Almost everything about its construction is unknown, for a small sample of these unknowns, see Table 2. Many such issues are explored further below.

How **shuttles were selected** in storage cylinders	Any **internal dimensions**
How a **shuttle was moved**	Any **internal timing**
How the INDEX **mechanism** worked	Almost everything about **program control**
How the MILL **mechanism** worked	Almost everything about **input & output**

Table 2 *Sample of known unknowns of Percy Ludgate's analytical engine*

II.3. Reducing the Unknowns: New Discoveries

Clearly the discovery of drawings by Ludgate would reduce the unknowns, so this has been an ultimate target of the research, but any discovery of new information about Ludgate's design would help. And in fact, just before Christmas 2019, Ralf Beulow of Heinz Nixdorf MuseumsForum with Eric Hutton [28] discovered a pair of articles about Ludgate's machine in the 1909 issues of the popular science magazine "The English Mechanic and World of Science" [29]. One was a very brief summary [30] (see Appendix II) of Ludgate's 1909 paper, with no new information.

The other article [31] (see Appendix III) did indeed promise new information, if validated, and included the first known diagram of Ludgate's machine! This article was attributed to "Engineering". A search by Jade Ward of University of Leeds Library discovered the article was in fact an abbreviated extract of an article [32] published a month earlier, i.e. just over three months after Ludgate's original Royal Dublin Society paper, in "Engineering" (a London-based monthly magazine founded in 1865). This article, which from here onwards will be referred to as "Engineering [32]", is reproduced in full as Appendix I.

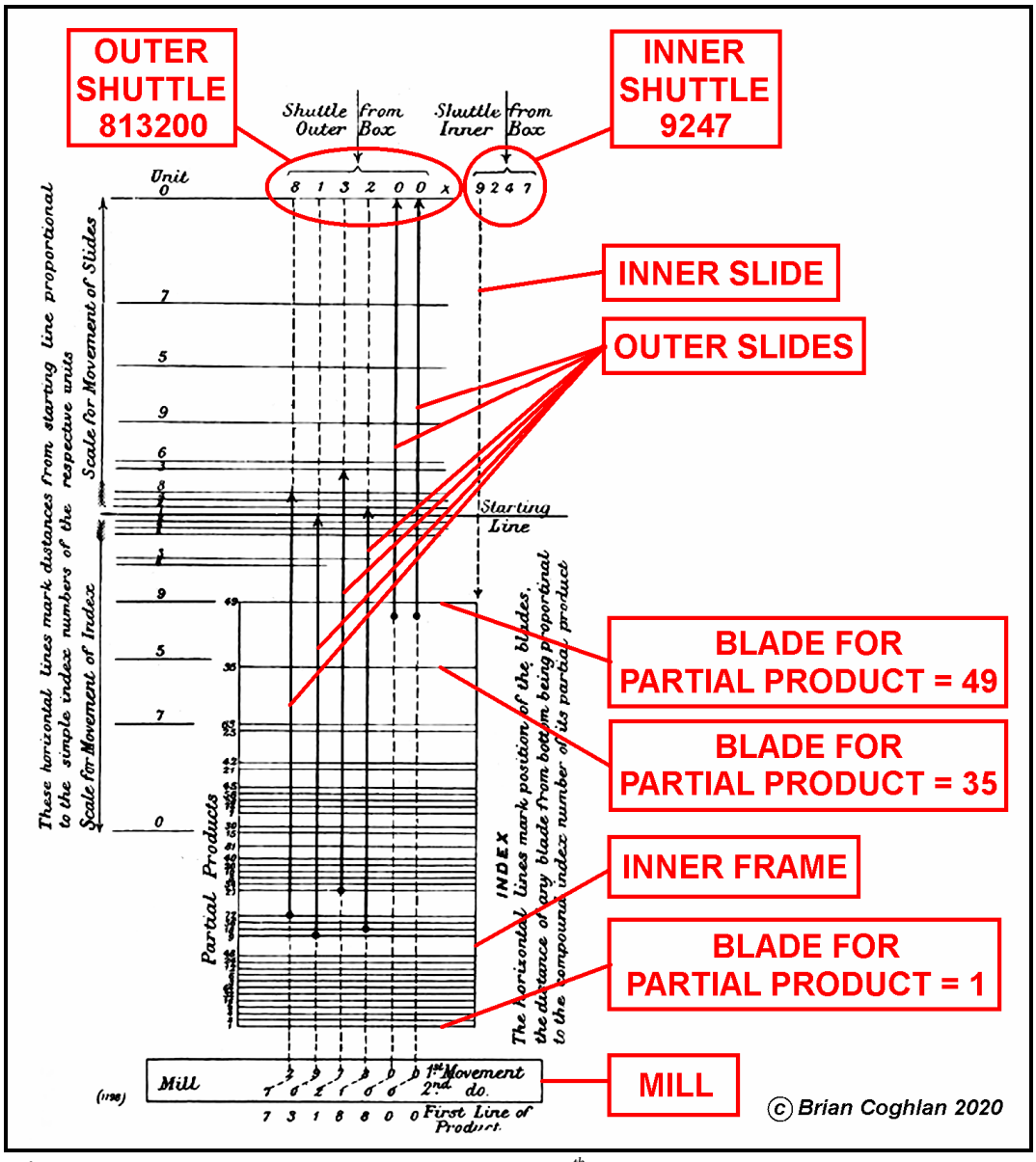


Figure 1 *Annotated version of the diagram from Engineering, 20th Aug. 1909, Pages 256-257 [32]*
See Appendix I for the original full-size image
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Figure 1 shows an annotated version of the diagram from Engineering [32] (an identical copy of which appeared in "The English Mechanic and the World of Science" [31]), which presumably was provided, and perhaps actually drawn, by Ludgate. It is not yet known whether its text is handwritten or typeset. The *'1198'* near the bottom left of the diagram could be the publisher's notation, or could be a datestamp ('11-Sep-1908' encoded as *'1198'*), or could be Ludgate's sheet number (large but perhaps not impossible after years of redesigning). It would also be useful to have the text professionally analysed against excerpts from Percy Ludgate's 1917 Will and Testament [33].

The text of the article of Engineering [32] at first appeared to us to contradict Ludgate's 1909 paper, but as analysis has progressed it has begun to seem more likely that the text was provided by Ludgate too (the article includes a detailed explanation of the diagram). Ludgate and the writer of the article are quite precise in what they say. In fact treating Ludgate's paper and the article as equally valid has uncovered ways the design may have been that would have been very hard to arrive at with only Ludgate's 1909 paper. A surprising amount of understanding has emerged, construction detail has been uncovered, some useful dimensional, geometric and timing inequalities established, and an effort has been made to codify these new facts. All this is described further below. Work has begun on both simulating the machine and "re-imagining" it with modern engineering software.

III. Essential Principles of Ludgate's Machine

III.1.Ludgate's multiply-accumulate operation

The multiplication in Ludgate's multiply-accumulate (MAC) operation is done in reverse to its traditional order. This is best illustrated graphically. For comparison, the traditional long multiplication ordering, starting with the multiplier least significant digit, is first illustrated in Figure 2.

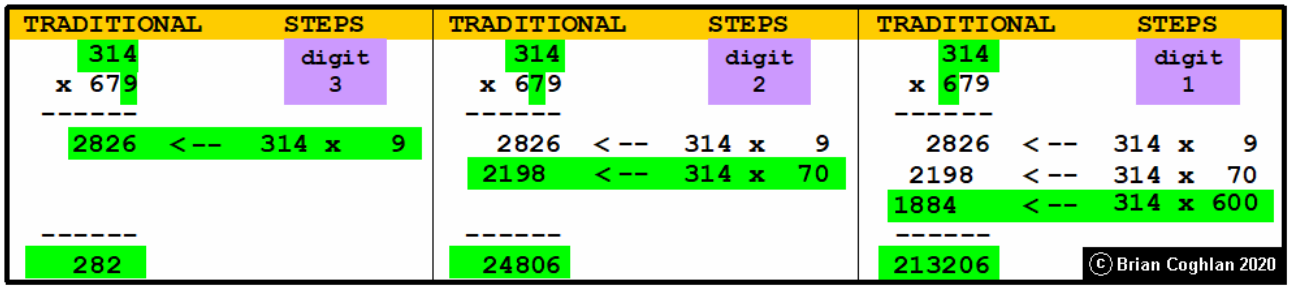


Figure 2 *Traditional long multiplication: multiply by LS digit of multiplier to form LS partial product, then multiply by next left digit of multiplier, and etc. For each step accumulate partial result to yield a final result (where LS and MS denote least and most significant)*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Ludgate's multiply-accumulate is processed jointly in his Index and Mill. The Index multiplies based on logarithms, while the Mill adds via clockwork cogs and gears (Ludgate refers to these as "wheels"). The multiply is done iteratively per digit from the most significant (MS) digit of the multiplier towards the least significant (LS) digit, i.e. in the reverse order to that in Figure 2. Figure 3 shows the first iteration (for the MS digit). For each digit of the multiplier, the units of the partial product are generated and added to the accumulator. Then the tens of the partial product are generated and added to the accumulator.

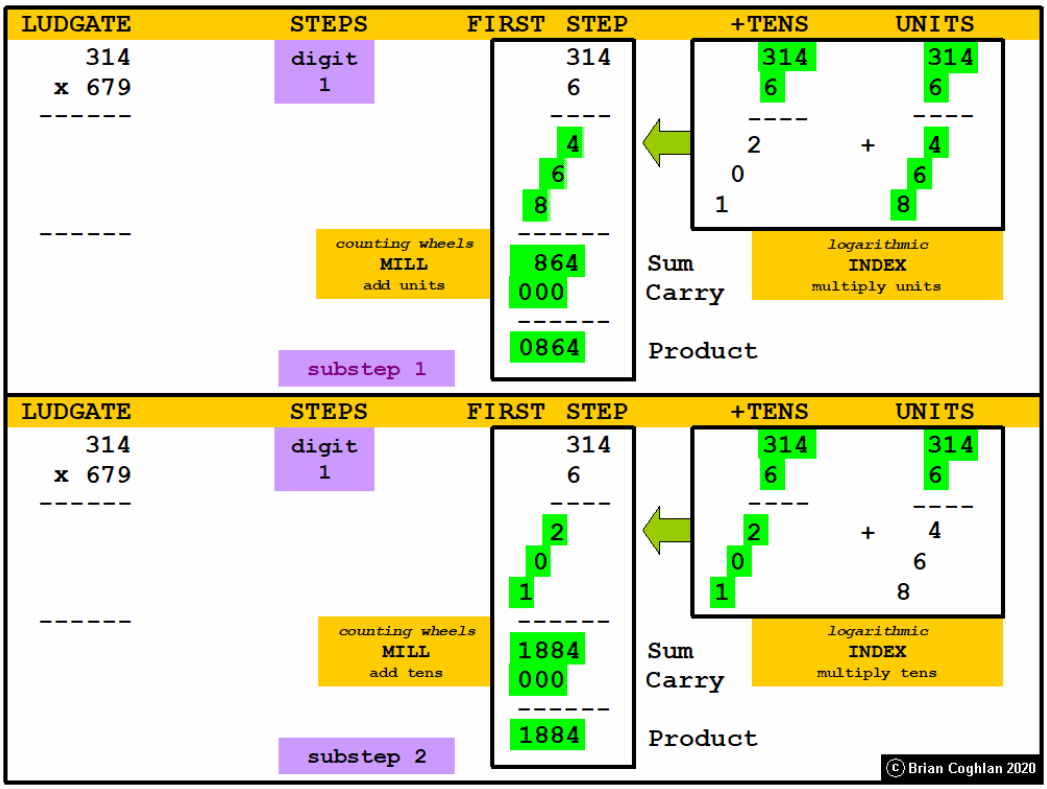


Figure 3 *Ludgate's long multiplication steps for the multiplier MS digit. First multiply-accumulate units, then tens*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Ludgate's 1909 paper stated "A carriage near the index now moves one step to effect multiplication by 10", and "After this the index makes a rapid reciprocating movement". This implies an awkward step left for tens then two steps right for the units of the next multiplier digit. This appears inexplicable, as it seems to be easier to add tens first then units for a smooth traverse from left to right, however it may have been an unknown optimization.

He also stated “I have devised a method in which the carrying is practically in complete mechanical independence of the adding process, so that the two movements proceed simultaneously”. Indeed his stated operation timings imply that carries are added in parallel with subsequent mechanical movements until a final visible addition of carries after the last of the multiplication steps. The implications are considered later; for now, let us ignore these details.

Multiplication for the remaining digits is illustrated in Figure 4.

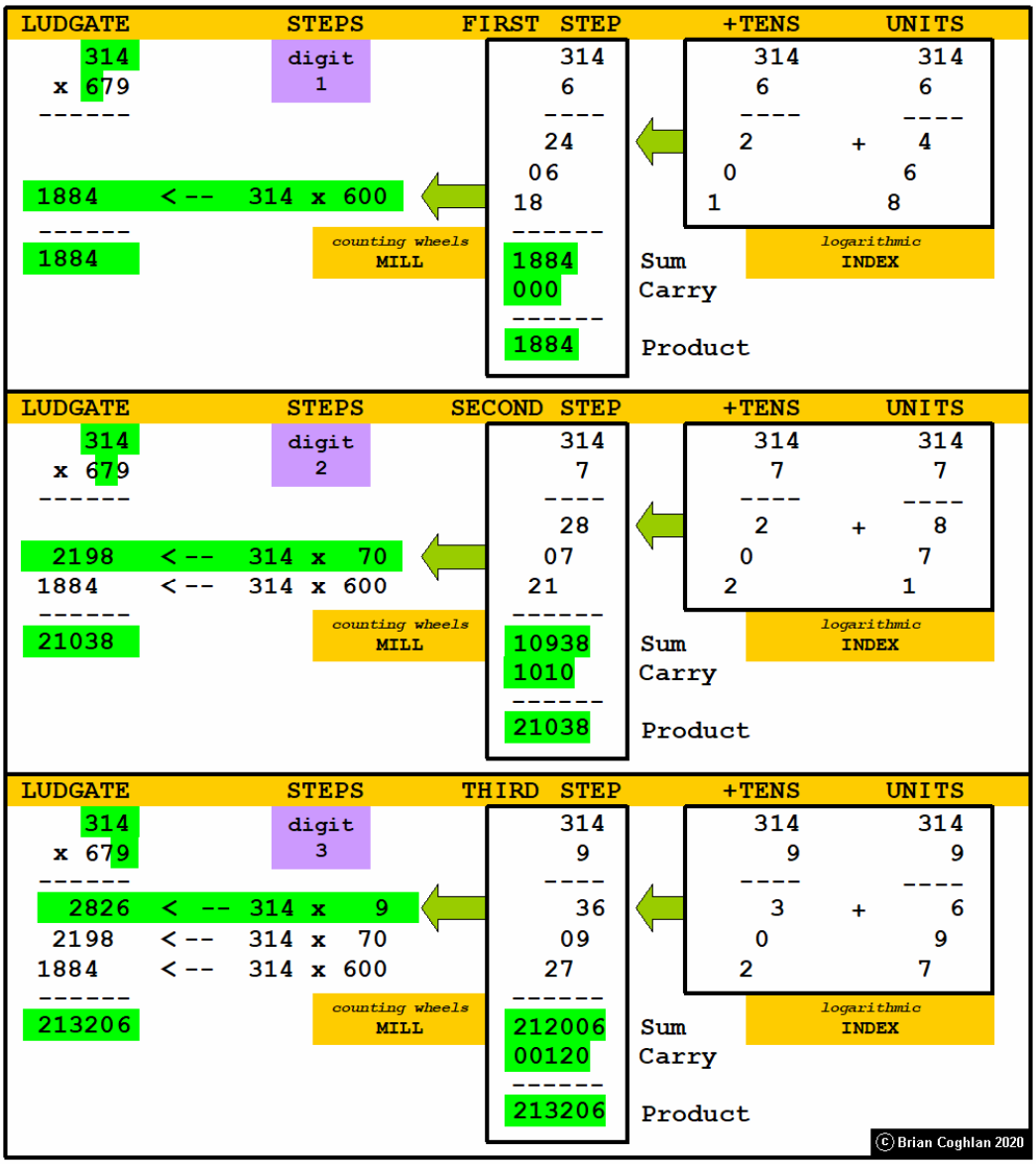


Figure 4 *A possible order in which Ludgate’s machine would have calculated 314×678 with one add and carries per stage. The right hand side shows how the partial products are split into tens and units*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Figure 5 shows a more succinct illustration of another example, but excluding any treatment of Ludgate’s carry propagation, which is explored much further below. This example shows the calculation in Ludgate 1909, illustrating again the novel iteration order and how the partial products are split into units and tens, where the units of the partial product are generated by the Index and added to the accumulator by the Mill, then the tens of the partial product are generated and added to the accumulator.

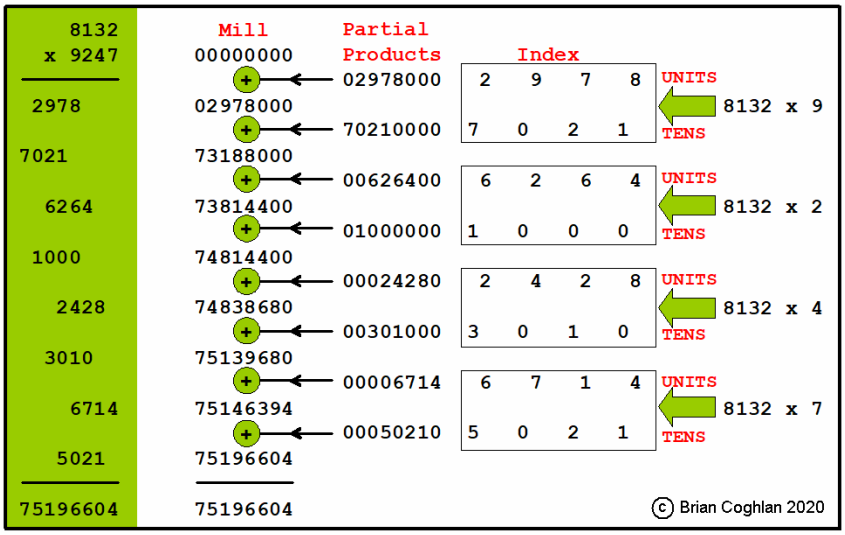


Figure 5 *An example of Ludgate's variant of long multiplication, calculating 8132 x 9247*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

III.2.Ludgate's logarithmic indexes

Ludgate invented his own logarithmic indexes for multiplication, an entirely new result in 1909. As indicated above, these are not ordinary logarithms, but integer values that obey the logarithmic law. For two operands Z_J and Z_K, Ludgate's index numbers ensure $Z_Y = Z_{J*K} = Z_J + Z_K$. For example, indexes $Z_3 = 7$ and $Z_5 = 23$, therefore $Z_{15} = Z_{3*5} = Z_3 + Z_5 = 30$. As can be seen in Table 3, simple indexes form a non-monotonic function of the decimal operands, but a monotonic function of the ordinals (because ordinals represent the rank order of simple indexes). Therefore ordinals may be used as a proxy for the decimal operands in circumstances where monotonicity is desirable, and Ludgate employed this.

Decimal operand	Simple index	Ordinal number
0	50	9
1	0	0
2	1	1
3	7	4
4	2	2
5	23	7
6	8	5
7	33	8
8	3	3
9	14	6

Partial product	Compound index	Partial product	Compound index	Partial product	Compound index
1	0	15	30	36	16
2	1	16	4	40	26
3	7	18	15	42	41
4	2	20	25	45	37
5	23	21	40	48	11
6	8	24	10	49	66
7	33	25	46	54	22
8	3	27	21	56	36
9	14	28	35	63	47
10	24	30	31	64	6
12	9	32	5	72	17
14	34	35	56	81	28

Table 3 *Ludgate's simple and compound logarithmic indexes (reproduced from Ludgate's 1909 paper, Table 1 and 2)*

C.V.Boys said: "Ludgate … uses for each of the prime numbers below ten in a logarithmic system with a different incommensurable base, which as a fact never appears" [2], i.e. each $Z_X = \log_{N_X}(X)$ has a different (invisible) base N_X.

Logarithmic indexes were not new in 1909, as Jacobi/Zech indexes (which can be derived from number theory) were already in use in astronomy, but Ludgate's particular indexes were new. Ludgate said: "The index numbers (which I believe to be the smallest whole numbers that will give the required results)". The largest Z_{J*K} is $Z_{7*7} = 66$. By way of comparison:

(1) Jacobi/Zech indexes (1846/1849 [35]). For example $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 18$, $Z_5 = 44$, $Z_7 = 7$.
An alternative is $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 8$, $Z_5 = 44$, $Z_7 = 27$. In both cases the largest Z_{J*K} is $Z_{5*5} = 88$.
(2) Remak indexes (Von K. Hoecken 1913 [36]), $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 13$, $Z_5 = 21$, $Z_7 = 30$, where the largest Z_{J*K} is $Z_{7*7} = 60$.
(3) Korn indexes (Von K. Hoecken 1913 [36]), $Z_1 = 0$, $Z_2 = 8$, $Z_3 = 13$, $Z_5 = 1$, $Z_7 = 30$, where the largest Z_{J*K} is $Z_{7*7} = 60$.

Ludgate's focus on small numbers makes it possible he knew of Jacobi/Zech indexes. Remak and Korn indexes were reported after Ludgate's 1909 paper was published. Small numbers are an important metric used to minimize the length of any mechanism that is used in the machine design to convert to or from the logarithmic indexes. McQuillan [37][38] has shown that better indexes can readily be computed using modern machinery.

Lemma 1: Ludgate proposed logarithmic indexes with the smallest Z_{J*K} then known.

As happens surprisingly often with novel inventions, in the same year as Ludgate's 1909 paper, a slide rule with Jacobi indexes was designed by Prof.Schumacher of Germany [39] and later manufactured as the Faber Model 366 [40]. For an example of how logarithmic indexes are used in calculations, see Andries de Man's educational emulator [41] for Ludgate's indexes.

III.3. Algorithm to derive Irish Logarithms

Only Boys [2], Riches [42] and de Man [34] offer any guidance as to how Ludgate derived some or all of his indexes. He almost certainly did not derive his indexes from number theory, but developed a systematic "method", such as that in Figure 6. The method of Figure 6 can be expressed as a simple algorithm, for example in Python as in Figure 7. Executing this algorithm yields identical indexes to those given in Tables 1, 2 and 3 of Ludgate's 1909 paper, see Figure 8.

Lemma 2: Ludgate's proposed logarithmic indexes are amenable to construction by a systematic method.
Lemma 3: Ludgate's proposed logarithmic indexes are amenable to algorithmic construction.

Irish Logarithms might exist for the range up to any *N*, not just for 0-9 as in Ludgate's paper. Certainly they have been shown to exist for 0-99 by extending the rather inefficient algorithm of Figure 7 with a basic Sieve of Eratosthenes to generate the seed prime numbers. Beyond that a more efficient algorithm would be necessary, or ideally a number-theoretic proof.

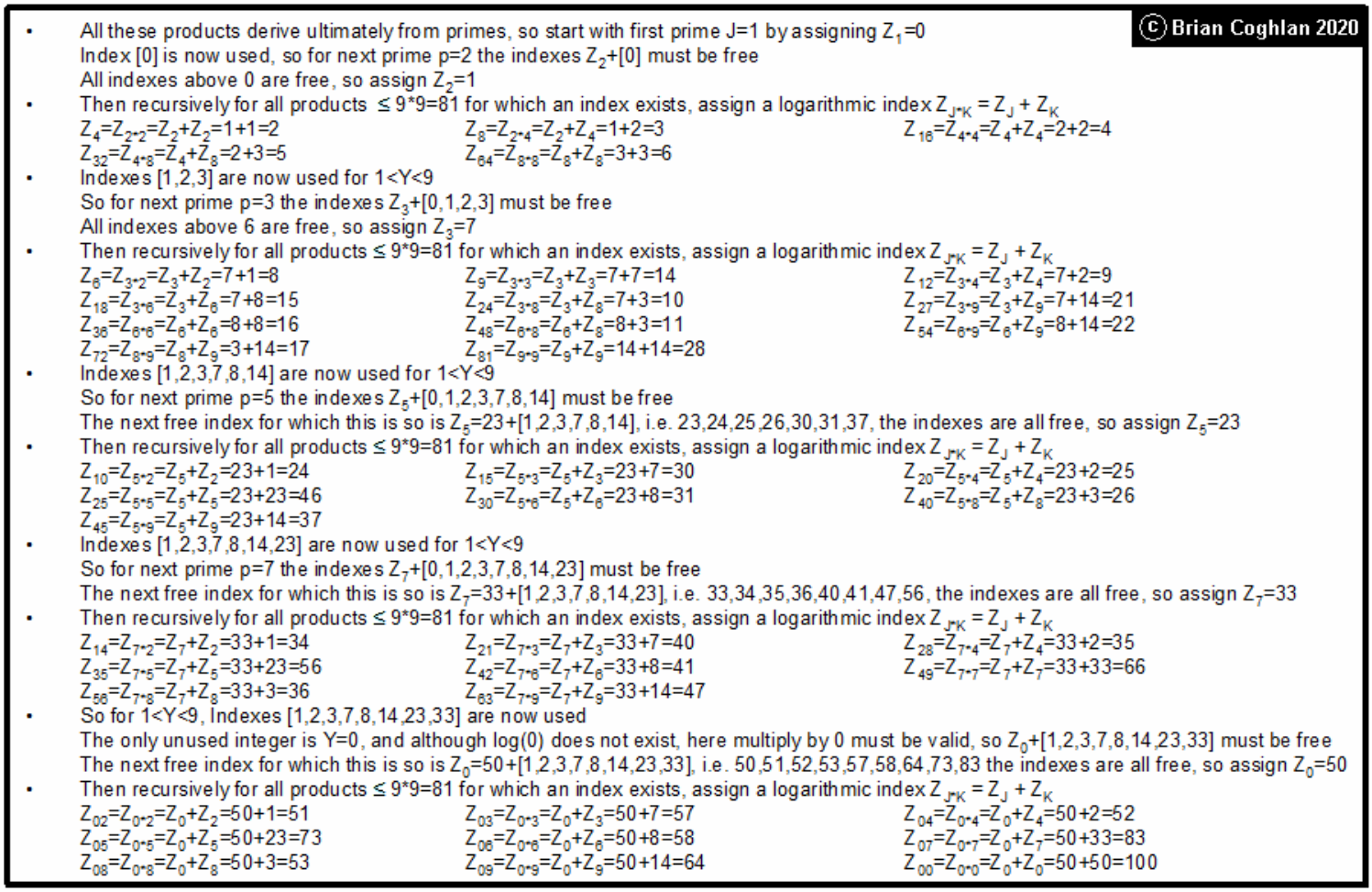


Figure 6 *Systematic method to derive Ludgate's index numbers*
For all products Y = (1<J<9) * (1<K<9) assign logarithmic index numbers $Z_Y = Z_{J*K} = Z_J + Z_K$
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

```
#!/usr/bin/env python                                   © Brian Coghlan 2020

import sys

# initialise variables
Z=[-1]*200   # table of complex indexes
PP=[-1]*200  # table of partial products
i = 0;
for p in (1,2,3,5,7,0):
    if Z[p]==-1:  # prime not indexed yet
        free=False
        while free==False and i<=100:
            free=True
            for j in (1,2,3,4,5,6,7,8,9):
                if free==True:
                    if Z[j]<>-1:               # for existing indexes
                        for k in range (1,100):
                            if Z[k]==(i+Z[j]): # check if complex index exists
                                free=False
                                i=i+1
        if free==True: # OK, found a desired free set of indexes
            Z[p]=i     # create new simple index
            PP[i]=p    # create new partial product
            i=i+1
            for j in (1,2,3,4,5,6,7,8,9,0):
                if Z[j]<>-1:  # multiplicand simple index exists
                    for k in (1,2,3,4,5,6,7,8,9,0):
                        if Z[k]<>-1:  # multiplier simple index exists
                            if PP[Z[j]+Z[k]]==-1:    # product not indexed yet
                                if Z[j*k]==-1:
                                    Z[j*k]=Z[j]+Z[k] # create new complex index
                                PP[Z[j]+Z[k]]=j*k    # create new partial product
```

Figure 7 *Python algorithm to derive Ludgate's index numbers*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

```
final Ludgate Simple Index for each Unit (Table 1):
  0: 50    1:  0    2:  1    3:  7    4:  2    5: 23    6:  8    7: 33    8:  3    9: 14

final Ludgate Complex Index for each Partial Product (Table 2):
  1:  0    2:  1    3:  7    4:  2    5: 23    6:  8    7: 33    8:  3    9: 14   10: 24   12:  9   14: 34
 15: 30   16:  4   18: 15   20: 25   21: 40   24: 10   25: 46   27: 21   28: 35   30: 31   32:  5   35: 56
 36: 16   40: 26   42: 41   45: 37   48: 11   49: 66   54: 22   56: 36   63: 47   64:  6   72: 17   81: 28

final Ludgate Partial Product for each Complex Index (Table 3):
  0:  1    1:  2    2:  4    3:  8    4: 16    5: 32    6: 64    7:  3    8:  6    9: 12
 10: 24   11: 48   12:       13:       14:  9   15: 18   16: 36   17: 72   18:       19:
 20:       21: 27   22: 54   23:  5   24: 10   25: 20   26: 40   27:       28: 81   29:
 30: 15   31: 30   32:       33:  7   34: 14   35: 28   36: 56   37: 45   38:       39:
 40: 21   41: 42   42:       43:       44:       45:       46: 25   47: 63   48:       49:
 50:  0   51:  0   52:  0   53:  0   54:       55:       56: 35   57:  0   58:  0   59:
 60:       61:       62:       63:       64:  0   65:       66: 49   67:       68:       69:
 70:       71:       72:       73:  0   74:       75:       76:       77:       78:       79:
 80:       81:       82:       83:  0   84:       85:       86:       87:       88:       89:
 90:       91:       92:       93:       94:       95:       96:       97:       98:       99:
100:  0                                                                  © Brian Coghlan 2020
```

Figure 8 *Results from running the algorithm of Figure 7*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

III.4.Principles of operation of Ludgate's logarithmic slides

Ludgate employed logarithmic index "slides" to implement his Irish Logarithms. The slides had a logarithmic profile representing a simple index on one axis *X* for each ordinal on another normal axis *Y*. Figure 9 shows a slide with annotations for the decimal, ordinal and index numbers (based on [38]). It also shows a storage shuttle with a type rod representing the decimal value of '8' (represented by the ordinal value '3'). The principle of operation is that the slide moves right to mate with the shuttle and rod. When it does so, it will have been displaced by a simple index of 3 units. By contrast for a shuttle with a type rod representing the decimal value of '0' (represented by the ordinal value '9'), the slide will be displaced by a simple index of 50

units. It is thought quite likely that ordinals were used in the storage shuttles as a proxy for the decimal operands in order that the series of slide profile changes in *Y* would be monotonic, as this would facilitate the progressive movement of the slides along the *X* axis for all possible decimal values.

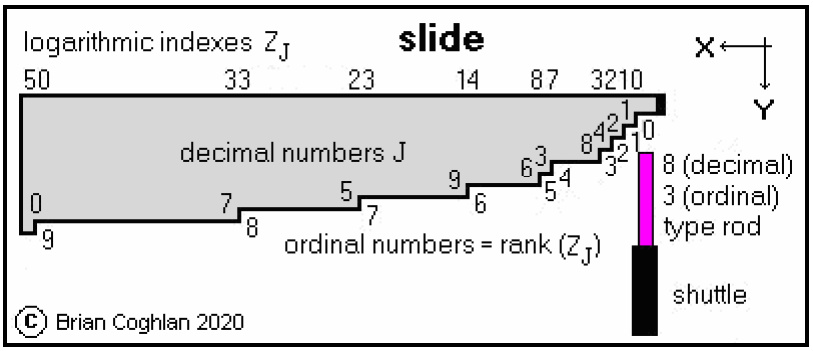


Figure 9 *Ludgate's logarithmic slide*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Note that the slide and shuttle Y-axis of Figure 9 will actually be edge-on, facing directly away from the viewer (reducing their visibility to lines as in Figure 1), but can be illustrated as shown in Figure 10 to aid understanding.

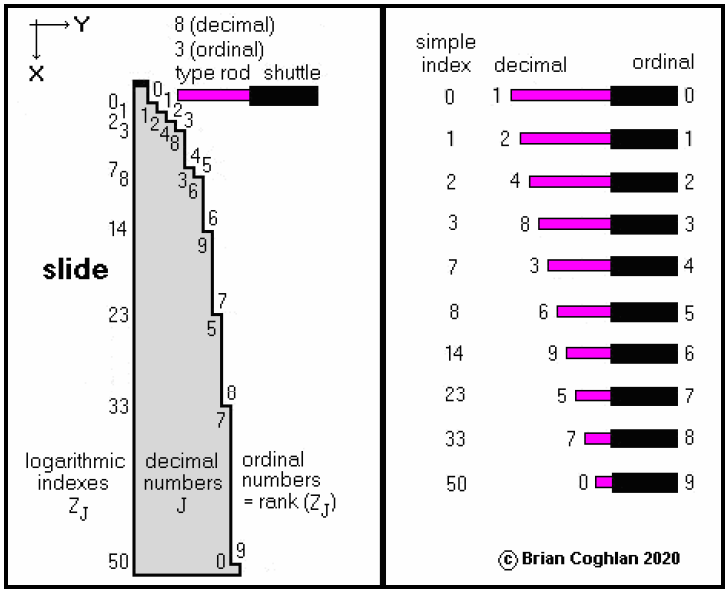


Figure 10 *Re-oriented (left) Ludgate's logarithmic slide, (right) shuttle rod extensions for simple indexes, decimal and ordinal values*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

The upper part of Figure 1 and Figure 12 further below show a 6-digit outer shuttle and a 4-digit inner shuttle, representing a machine with a 6-digit outer operand (the multiplicand 813200) and a 4-digit inner operand (the multiplier 9247) as per Engineering [32]. These correspond to the variables to be operated upon, which are from the outer and inner storage cylinders, respectively. Figure 1 and Figure 12 also show six outer slides and one inner slide. The two types of slide are likely to have had the same logarithmic profile, as is shown. Each of the outer slides was to mate with a corresponding digit of the outer shuttle, and were slightly longer (length = maximum compound index) with a pointer attached to one end. The inner slide (length = maximum simple index) was to mate iteratively with one digit of the inner shuttle at a time, starting with the most significant digit.

The shuttles were in line as shown, and the outer and inner slides faced each other on a common 'starting line' aligned with their simple index '0'. To multiply, the slides were moved towards their respective shuttles (i.e. the slides moved towards each other). The fact that the outer and inner slides move towards each other (rather than apart) is a surprise. Figure 11 and Figure 12 show, for example, the leftmost slide when moved to mate with the outer shuttle left type rod representing decimal value '8', and therefore displaced by simple index $Z_8 = 3$. Figure 11 and Figure 12 also show the inner slide when moved to mate with the inner shuttle type rod representing decimal value '9', and therefore displaced by simple index $Z_9 = 14$. Like a slide rule, the relative displacement of the two slides is then the compound index $Z_{72} = Z_{8*9} = Z_8 + Z_9 = 3 + 14 = 17$, representing the decimal partial product 8*9 = 72.

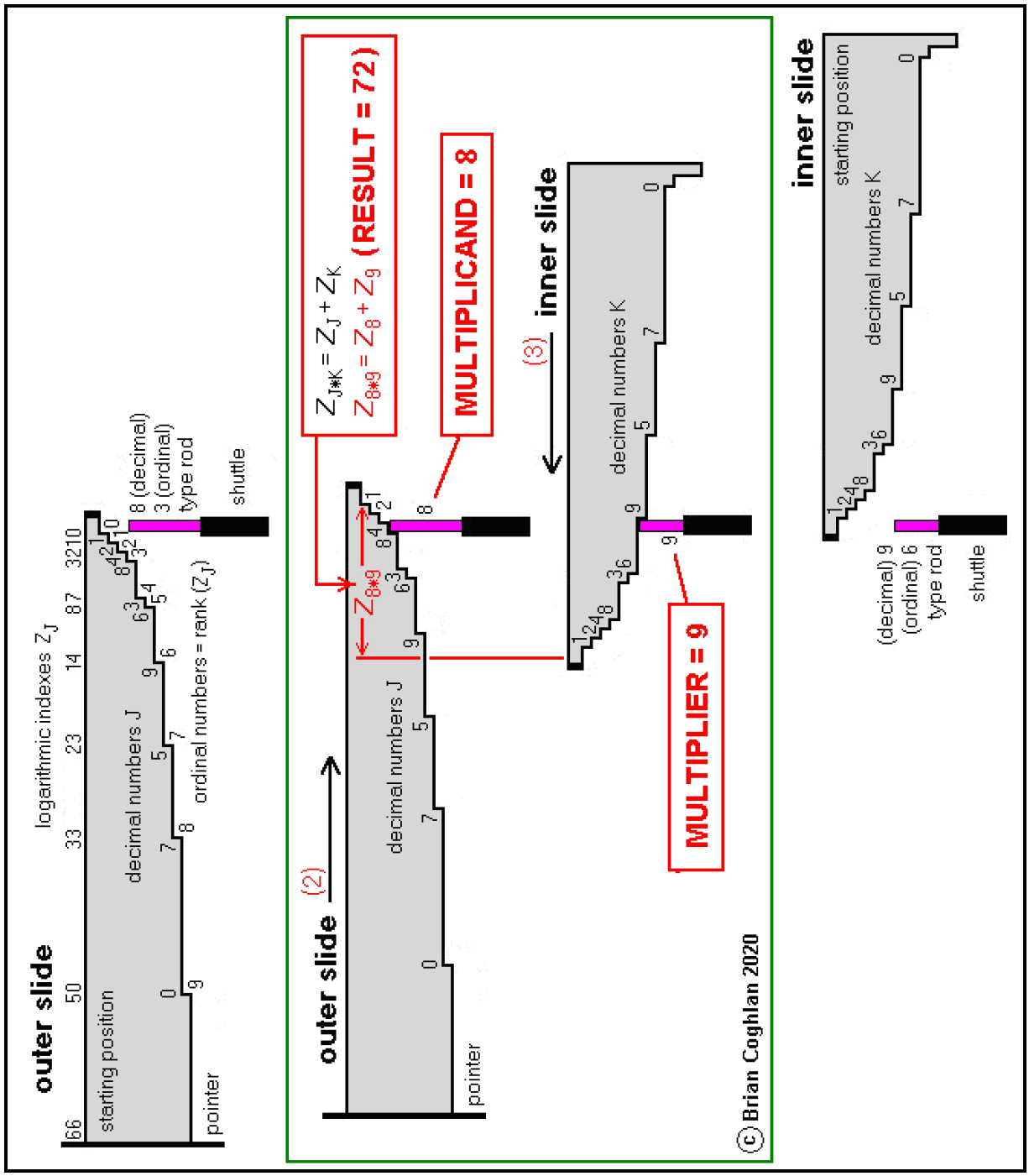


Figure 11 *Multiplication using Ludgate's logarithmic slides:*
Left: outer slide at its starting position,
Right: inner slide at its starting position,
*Middle: movement of slides for multiplication of 8*9 = 72*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

IV. Evidence-based Deduction of Hidden Aspects of Ludgate's Machine

As indicated above, only a few features of his Analytical Machine are described in Ludgate's 1909 paper [1] (***hereafter called Ludgate 1909***), almost everything about its construction is unknown, thus any more must be deduced by contextual analysis of the paper, largely a process of logical inference or elimination of false propositions, and mechanical or mathematical speculations. This document explores that approach, aided by the newly-discovered information of Engineering [32], a copy of which is reproduced as Appendix I.

Objective 1: Contextual analysis of the combined texts plus diagram remains questionable, but this document will attempt to show its potential to deduce lemmas and thereby assemble a body of known new design facts.

The following contextual analysis repeatedly quotes statements from Ludgate 1909 and/or Engineering [32] as evidence to corroborate deductions about the workings of Ludgate's Analytical Machine. This makes the text dense and somewhat turgid, but precise. Where necessary, some speculation will be utilized, but clearly marked as such, in order to explain difficult points.

IV.1. Ludgate's Index mechanism

As mentioned above, Figure 12 reproduces the example multiplication of the outer operand (multiplicand) by the most significant digit of the inner operand (multiplier) of Engineering [32] and also of Figure 1, showing the partial products for all six outer slides. Ludgate's 1909 paper includes an almost identical example. In succeeding iterations the inner slide would mate one multiplier digit at a time with the less significant multiplier digits, while the Mill accumulated the succeeding partial products to produce the result of Ludgate's variant of long multiplication.

Ludgate's index was more than just the slides and shuttles, and in fact was a quite complex multiplicity of moving parts. Much of this related to converting the relative displacement Z_{J*K} to discrete increments or decrements of a result held on the set of counter wheels that Ludgate called the Mill. The two slide types existed within different physical structures. There were twenty of the outer slides (with the pointer attached at one end) in an outer frame but free to move within it (so for illustrative purposes that frame can be treated as invisible). In contrast there was just one inner slide, firmly attached to an inner frame containing multiple blades. There was one blade per compound index, i.e. one per uniquely valid partial product, as per Engineering [32], see Figure 1 or Figure 12. The way in which the blades are composed and utilized is another surprise, but is a very clever arrangement.

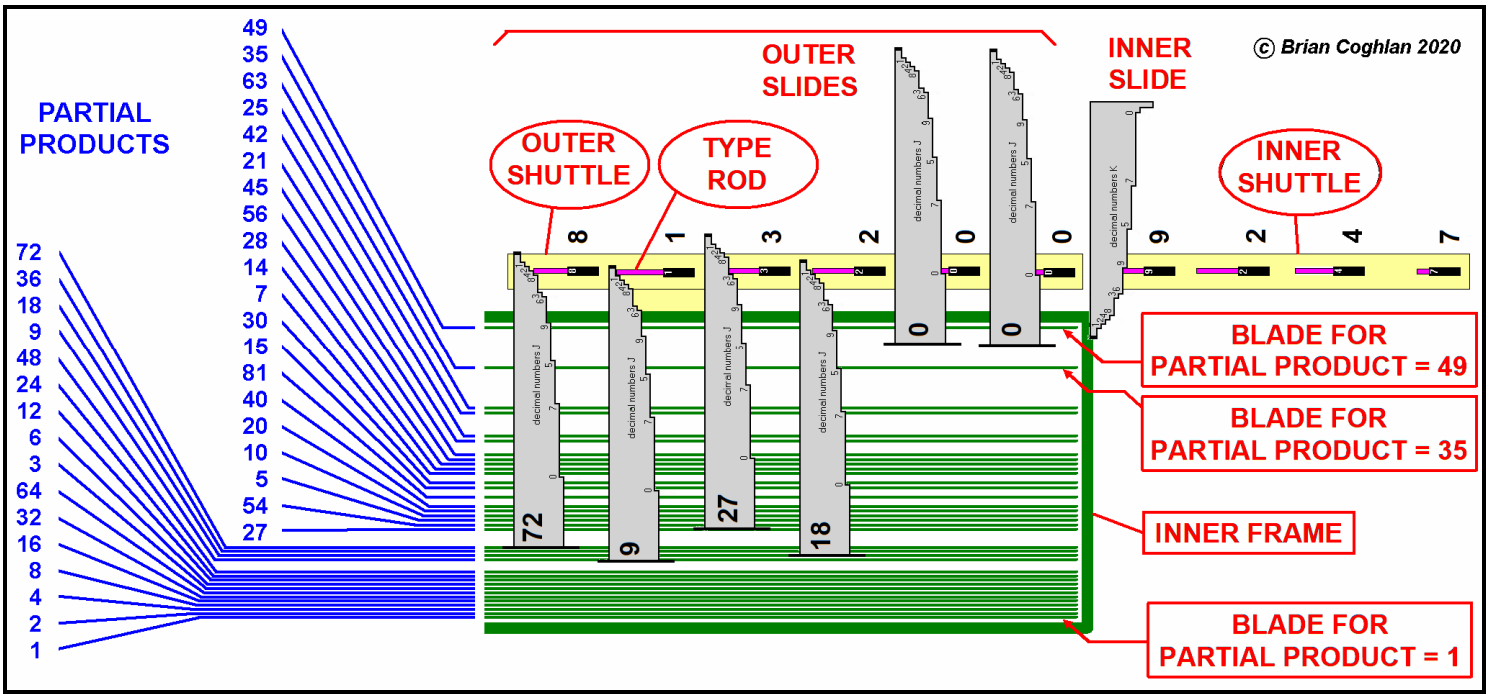


Figure 12 *Ludgate's outer and inner logarithmic index slides, and inner frame with blades for compound indexes, representing the machine with a 6-digit outer operand and a 4-digit inner operand as per Engineering [32]*
Note that the slides and shuttles will actually face directly away from the viewer, but are shown as is to aid understanding
Images reproduced courtesy The John Gabriel Byrne Computer Science Collection

It is clear from Ludgate 1909 that the 20-digit multiply-accumulate procedure was as follows (Figure 13 is cut-down example):

(1) The outer (multiplicand ***b***) and inner (multiplier ***a***) operand shuttles are moved to the starting line "near the Index", with the inner slide aligned with the most-significant rod of the inner shuttle (the left rod).
(2) The set of 20 outer slides move in one direction to mate with the shuttle holding the value of the outer operand. The slides convert the digits to "simple index numbers", essentially the logarithmic indexes of the digits of that operand.
(3) Then the single inner slide is moved in the opposite direction to mate with the rod of the shuttle that it is aligned with; the rod holds the value of that digit of the inner operand. The inner slide converts that digit to a simple index number, i.e. the logarithmic index of the value of that digit of that operand.
(4) "as the index is attached to the last-mentioned slide, and partakes of its motion, the relative displacement of the index and each of the [outer operand] slides … [and] pointers attached to the [outer operand] slides, which normally point to zero on the index, will now point respectively …" to the compound index number, essentially the sum of the simple index numbers, representing multiplication of the operands.
(5) Next all the compound index numbers are mapped by "movable blades" to numerical units and tens of the partial products and "conveyed by the pointer to" the Mill and accumulated in the Mill.
(6) Then the Index and its attached inner slide performs a "rapid reciprocating action" to align with the rod of the inner shuttle that holds the value of the next less significant digit of the inner operand.
(7) Operations (3-6) are repeated for each of the rods of the inner shuttle "until the whole product of ***ab*** is found".
(8) Then "The shuttles are afterwards replaced in the shuttle boxes".

From which may be deduced:

Lemma 4: there is no optimization of shuttle movements, e.g. no early return of the outer shuttle to storage.

Engineering [32] first describes the movement of the outer slides, consistent with step (2) above, but then states "All the slides aforementioned are mounted in a frame, and … this frame is moved up over another frame divided with another set of index numbers" (**hereinafter called outer and inner frames respectively**). From (3) above it is clear that the inner frame and its slide moves as a whole in the opposite direction to the outer slides.

Lemma 5: the outer slides are mounted on an outer frame, and free to move lengthwise along that frame.
Lemma 6: a single inner slide is mounted to an inner frame, and the whole moves in the opposite direction to the outer slides.

The repetitions of steps of operations (3-6) for each of the rods of the inner operand digits until the whole product is found can be observed in the cut-down 6-digit by 4-digit example in Figure 13. Comparison of the quadrants of Figure 13 very clearly shows how the outer slides do not move as the iterations of operations (3-6) above proceed. Only the inner slide and its frame move together in two dimensions (vertically up to retract, then horizontally right to the next digit, then vertically down to hit that digit's type rod) as they perform each "rapid reciprocating action" of operation (6) above to move digit-by-digit from the most to least significant digit of the inner operand.

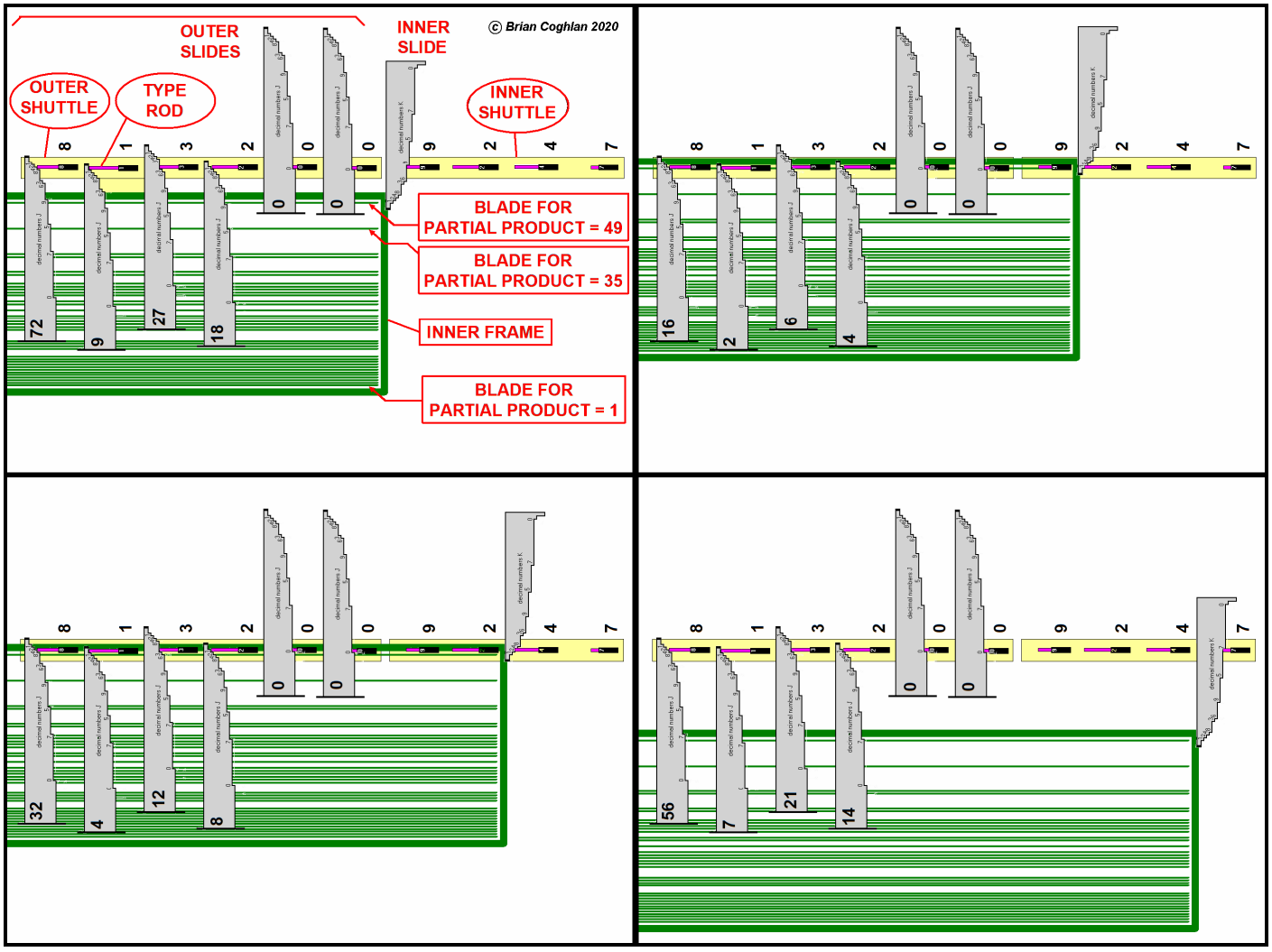


Figure 13 *Ludgate's procedure for multiplication of two operands,*
representing a machine with a 6-digit outer operand and a 4-digit inner operand as per Engineering [32]
Top Left: iteration 1 steps 3-5, multiplicand '813200' multiplied by multiplier digit '9'
Top Right: iteration 2 steps 3-5, multiplicand '813200' multiplied by multiplier digit '2'
Bottom Left: iteration 3 steps 3-5, multiplicand '813200' multiplied by multiplier digit '4'
Bottom Right: iteration 4 steps 3-5, multiplicand '813200' multiplied by multiplier digit '7'
Iteration step 6 occurs between each quadrant of the figure to move the Index and its attached inner slide to the next multiplier digit
Note that the slides and shuttles will actually face directly away from the viewer, but are shown as is to aid understanding
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Figure 14 shows an early 3D rendition of the possible re-imagined Ludgate races, shuttles and Index performing the first iteration of the same multiplication as show in Figure 13, i.e. positioned as shown in Figure 12 and in the top-left quadrant of Figure 13. Figure 15 shows a top view (as per Figure 1) of this 3D design performing the iterations over the four least-significant multiplier digits '9247'. Again a principal characteristic is that the inner frame and slide move together in two dimensions with each iteration to reflect the value of the multiplier digits, while the 20 outer slides remain static.

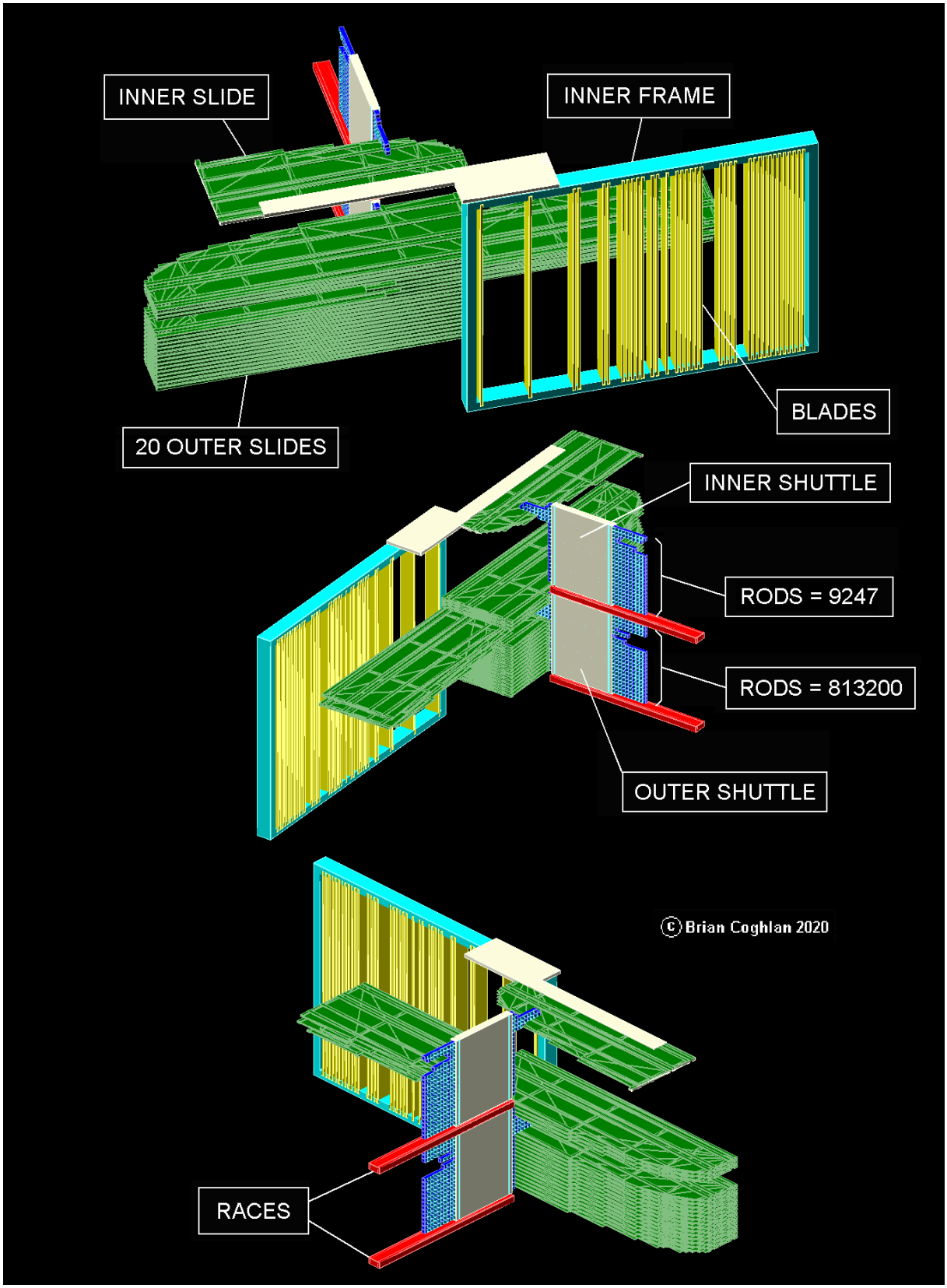


Figure 14 *Early 3D rendition of Ludgate's races, shuttles and Index for multiplication of two 20-digit operands, performing the first iteration of the same multiplication as shown in Figure 13, with multiplicand '813200' multiplied by multiplier '9247' as per Engineering [32] (the sign rod is at the top of each shuttle, next to the least-significant digit's type rod). From top to bottom: front, left-rear and right-rear three-quarter views Image reproduced courtesy The John Gabriel Byrne Computer Science Collection*

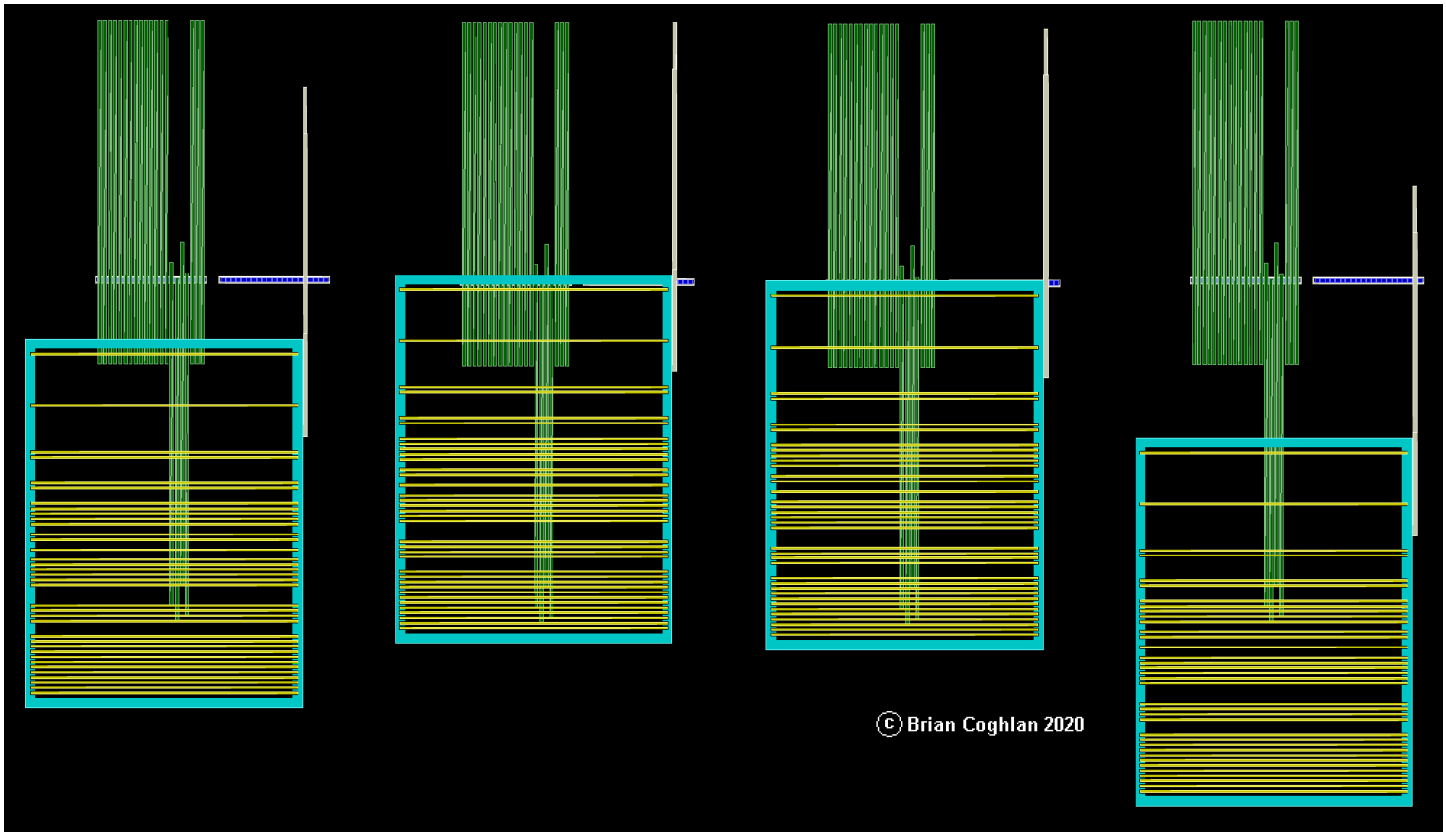


Figure 15 *Early 3D rendition of Ludgate's procedure for multiplication of two 20-digit operands, but with multiplicand '813200' multiplied by multiplier '9247' as per Engineering [32] From left to right: top view of iterations 1 to 4 of steps 3-5 as per Figure 13 Images reproduced courtesy The John Gabriel Byrne Computer Science Collection*

Practical design involves issues that are not relevant to operating principles. For example, Figure 16 shows an early 3D rendition of a re-imagining of a slide. It is very thin to limit weight, but has strengthening struts, and has extra height for slots on each side along which it may slide on bars that fit between each slide. The bars are intended to be part of a frame (which is not shown). Each slide is located by two bars on each side (four bars in all), to guarantee physical alignment,

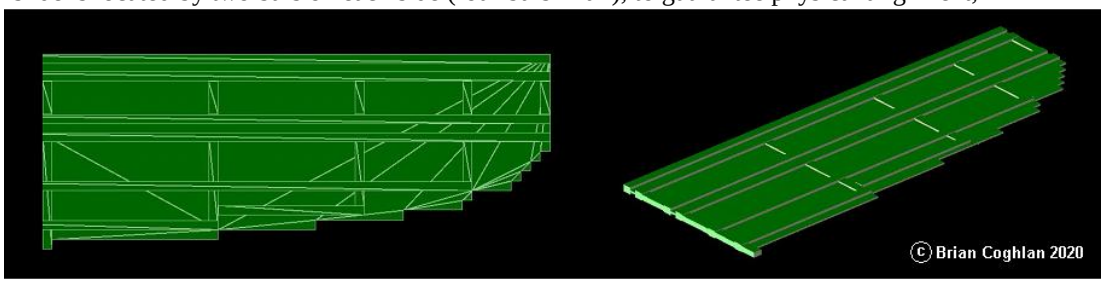


Figure 16 *Early 3D rendition of a re-imagining of Ludgate's logarithmic slide Image reproduced courtesy The John Gabriel Byrne Computer Science Collection*

Further explanation of more surprising aspects of the Index is greatly helped by the text of Engineering [32] and its diagram, shown in Figure 1 (and in Figure 12 and Figure 13). An important clue is disclosed in Engineering [32], which states "A slide to denote decimal 8 is set at ⅜" above the zero or starting line" (i.e. decimal 8 is represented by its proxy ordinal 3). In this and other statements, Engineering [32] clearly demonstrates a fundamental unit of the machine size is ⅛" (not unexpected for Imperial measurements), that the logarithmic index scale is in units of ⅛" and extends over the length of the largest simple index (50 units, i.e. 6¼"), and that the total allowed movement of an outer slide is 50 units, similarly for the inner slide. When both are at maximum relative displacement, the space occupied will be 100 units, but beyond that the inner frame will extend to the length of the largest compound index ($Z_{7*7} = Z_7 + Z_7 = 33 + 33 = 66$) units, i.e. 8¼", so the maximum space occupied is 166 units, i.e. 20¾". Ludgate 1909 states the machine is "26" long, 24" broad, and 20" high", so the available internal space is just adequate.

Engineering [32] clearly implies the slides embody simple indexes versus ordinals, not decimals, see Table 4.

Decimal operand	Ordinal number	Ordinal profile stop	Simple index	Slide profile stop
0	9	0.000”	50	6.250”
7	8	0.125”	33	4.125”
5	7	0.250”	23	2.875”
9	6	0.375”	14	1.750”
6	5	0.500”	8	1.000”
3	4	0.625”	7	0.875”
8	3	0.750”	3	0.375”
4	2	0.875”	2	0.250”
2	1	1.000”	1	0.125”
1	0	1.125”	0	0.000”

Table 4 *Ludgate’s logarithmic slide scales*

Some important spatial relationships on the diagram are not immediately obvious. Mechanically, Figure 1 shows the outer slides (with pointers) are the same length as the inner frame, 66 units, i.e. 16 units beyond the largest simple index. This is because the pointers and blades are used to deduce the partial product. Each blade represents a partial product. The crucial point is that the slides and frame all align along a common starting line, so when idle all the slide tips and the top blade (the partial product line marked “49”, where $Z_{7*7} = Z_7 + Z_7 = 33 + 33 = 66$) all lie on the starting line, and the pointers lie above the bottom blade (the line marked “1”, where $Z_{1*1} = Z_1 + Z_1 = 0 + 0 = 0$). When an outer slide moves up, its pointer moves up over the blades. When the slide mates with the shuttle, the pointer will lie over a blade that indicates the relevant partial product. The “pointers” are not mentioned in the text or diagram of Engineering [32], but Ludgate 1909 states that "pointers attached to the four slides … now point respectively to the 17th ... divisions of the index", i.e. to horizontal lines marked “72 …” that have the round dots at the bottom end of the slides on Figure 1. This confirms that the pointers are at the wide end of the slides, where they lie above $Z_{J*K} = Z_J + Z_K$.

Lemma 7: a fundamental unit of the machine size is **⅛”**.
Lemma 8: the logarithmic index scales are in units of **⅛”**.

Ludgate 1909 states that "pointers attached to the four slides, which normally point to zero in the index, will now point respectively to the 17th, 14th, 21st and 15th divisions of the index … corresponding to the partial products 72, 9, 27 and 18" (as in Fig.14 and the top left of Fig.15), and the partial products "are conveyed by the pointers to the mill". So the output is from the wide end of each slide, i.e. from different positions relative to the starting line. There would be great advantage from fixing the position of the Index outputs and Mill so that transfer of value from one to another was convenient, but this is a false proposition. Ludgate does not imply Index outputs are placed in a fixed or convenient place (e.g. the starting line).

Lemma 9: Index outputs are not placed in a fixed or convenient place relative to the Mill.

How do the pointers know the partial product values? On Figure 1, the pointers lie above blades on the inner “frame” attached to the inner “slide”, and both diagram and accompanying text of Engineering [32] show there are wide blades at partial product positions, not for 'zero' products, and not everywhere. The blades actually must cover (and be slightly wider than) the span across both sets of shuttle rods. Ludgate 1909 states the blades are movable, and move for a “duration” representing the partial product unit or tens value, "the duration of which displacements are recorded in units measured by the driving shaft on a train of wheels called the mill", so a blade moving “duration” *N* leads to a pointer moving, which leads to the Mill incrementing by *N*. The diagram implies (but does not mandate) the pointers lie above the blades, in which case the pointers would be moved or hit by the blade from beneath.

Since pointers are in different positions, several different blade and pointer combinations move simultaneously from different points in space. Figure 1, Figure 12, Figure 13 and Appendices I and III all show just six outer slides with pointers, but in Ludgate’s complete machine the Mill has to register the motion of 20 pointers across the horizontal (*O* for ordinal) axis that are not aligned on the vertical (*I* for index) axis, and where blade movement can only be towards the viewer on the (*P* for product) axis, and where blades must move different “durations” for units and tens of partial products. And the combination of positions and movements must be repeated, but with different values for each digit of the inner operand (multiplier). To do this, Ludgate employed coordinate transformation, where a value along one axis is mapped to another value on an orthogonal axis.

Ludgate reduced the multi-dimensional mapping problem by iterating over one inner-shuttle digit number ***j*** at a time, and mapping units first then tens. With reference to Figure 1 for each ***j*** his approach is equivalent to multiport 2-d mapping combined with 2-d coincident addressing in *O* and *I* axes, with the data (partial product units or tens) in the 3rd dimension *P*. This is very clearly shown when describing the Index as an algorithmic state machine, see Algorithm 1 in pseudo-OCCAM, where SEQ and PAR identify sequential and parallel actions, and ':=' and “=” identify synchronous (“event-triggered” or “clocked”) and asynchronous behaviours.

```
PAR i=1 to 20                                  # do simultaneously for all outer operand digits
   Outer_S.index(i) := map_1(i)                   # S.indexes(i) on I-axis ← outer_operand_digit(i) on O-axis
SEQ j=1 to 20                                  # iterate over inner operand digits
   Inner_S.index(j) := map_1(j)                   # S.indexes(j) on I-axis ← inner_operand_digit (j) on O-axis
   PAR i=1 to 20                               # do simultaneously for all outer operand simple indexes
      SEQ                                         # but for each outer operand S.index do the following sequentially
         C.index(i) =  Inner_S.index(j) + Outer_S.index(i)   # C.index(i) on I-axis = Σ(S.indexes) on I-axis
         p.product_digit(i) := map_2(C.index(i, units))      # p.product_digit(i) on P-axis ← C.index(i, units) on I-axis
         p.product_digit(i+1) := map_2(C.index(i, tens))     # p.product_digit(i+1) on P-axis ← C.index(i, tens) on I-axis
```

Algorithm 1 *Pseudo-OCCAM description of Ludgate's Index mappings*
"S.index" refers to simple index and "C.index" refers to compound index

Ludgate 1909 stated a "carriage" moves to select units or tens. Perhaps this "carriage" can jog blades sideways to select units or tens in the form of a stepped blade profile repeating per multiplicand digit. Perhaps slides could "register" partial product units on a "units pointer" on its one side, and then the "carriage" jog the slides sideways to do likewise for tens on its other side, Perhaps there was a separate blade for units and another for tens, although Ludgate 1909 implies this is not so, and the diagram of Engineering [32] only shows one blade per partial product. Just how the "carriage" was utilised remains an open question.

Regarding the blades themselves, some related if disparate facts can be deduced from Ludgate 1909:

(1) "system of slides called the index", i.e. "index" is the whole system of slides, but confusingly in other instances the Index clearly includes the inner frame and blades too, and in other instances seems to be just the inner frame and blades.
(2) "[the index] may be compared to a slide-rule on which the usual markings are replaced by moveable blades", i.e. the blades determine the partial products.
(3) "blades" move for a "duration" representing a partial product digit. This blade motion can only be along the *P*-axis to/from the viewer of the diagram of Engineering [32].
(4) "index is arranged so as to give several readings simultaneously". It can only do that if it is a planar structure as in the diagram of Engineering [32].
(5) "The numerical values of the readings are indicated by periodic displacements of the blades mentioned", possibly "periodic" increments of ⅛".
(6) "In the index the partial products are expressed mechanically by movable blades placed at the intervals shown in column 2 of the third table". Therefore on the inner frame there are moveable blades protruding along the *P*-axis to/from the viewer, one blade for each of 35 non-zero partial products.
(7) "Now the duration of the first movement of any blade is as the unit figure of the partial product which it represents". This strongly indicates that there is only one blade per partial product (not one for units and another for tens), and it first moves along the *P*-axis for units, and later after carriage motion it performs the same function for tens. Ludgate 1909 states "most of the movements … are derived from set of cams placed on a common shaft parallel to the driving shaft". These displacements could be easily implemented by underlying steps or cams, e.g. a double-lobe cam per blade could create the units then tens movements.
(8) "which movements are conveyed by the pointers to the mill". As it is clear the pointers are attached to the slides and move with the slides, this statement is mysterious.

This conveyance from pointer to Mill in (8) above remains the most mysterious aspect of the Index and Mill. Where is the Mill, does it move, and how are Index output values conveyed to it? Engineering [32] says that the partial products are "registered" before "finally being added in the Mill". Hence "pointers" must "convey" or "register" the "duration" or the existence of the "duration", which Ludgate 1909 says must be "recorded in units measured by the driving shaft on a train of wheels called the mill". And yet when blades move, the pointers (which are in different positions) can only move along the *P*-axis to/from the viewer of the diagram of Engineering [32], not towards the Mill. This poses two classes of proposition.

The first "direct" class assumes "duration" means distance, which might suggest each pointer has separate mechanisms that directly act to "register" the partial product units and tens before conveying them, or alternatively that perhaps the "carriage" can directly shift some "conveyance" mechanism, or that the "conveyance" employs mechanisms to directly convey partial products. The blades could be driven by double-lobe cams from below, where units and tens halves have fixed arc-angle raised segments with a variable height proportional to the partial-product digit magnitude, equivalent to the "duration" (like car engine valve cams but with two opposing lobes), see the left side of Figure 17.

The second "indirect" class assumes that "duration" means time, with indirect action, so in the time a blade moves *N* increments, a driving shaft also indirectly acts to "register" the duration *N*, and when the blade hits the pointer this action is forcibly enabled or disabled. This would nicely decouple the Mill from the Index. In this case, the blades could be driven by double-lobe cams from below, where units and tens halves have fixed height raised segments over a variable arc-angle

proportional to the partial-product digit magnitude, equivalent to the “duration”, see the right side of Figure 17.

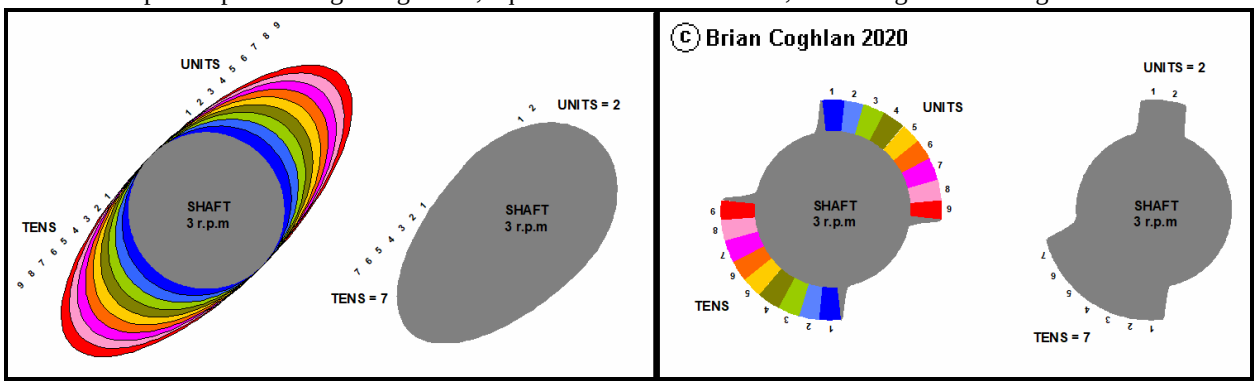


Figure 17 *Left: Double-lobe cam with variable height raised segments (proportional to partial-product digit magnitude) over a fixed arc-angle*
Right: Double-lobe cam with fixed height raised segments over a variable arc-angle (proportional to partial-product digit magnitude)
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Either class of propositions would be consistent with both Ludgate 1909 and Engineering [32], but in the absence of further information it has not yet been possible to differentiate further.

Figure 18 shows early 3D renditions of each type of cam (i.e. for each class of propositions) for all the valid partial products. However the blade movements will be preceded by other mechanical movements (e.g. move frame, release slides), and followed by further mechanical movements (e.g. retract slides, move frame to next multiplier digit), so the blade movements will only occupy a portion of the camshaft revolution. Hence in reality the cam lobes will only occupy a portion of the cam revolution, and the cams will need to have a much larger diameter. Other cams on the same shaft, or on a synchronously related shaft, may control the other mechanical movements.

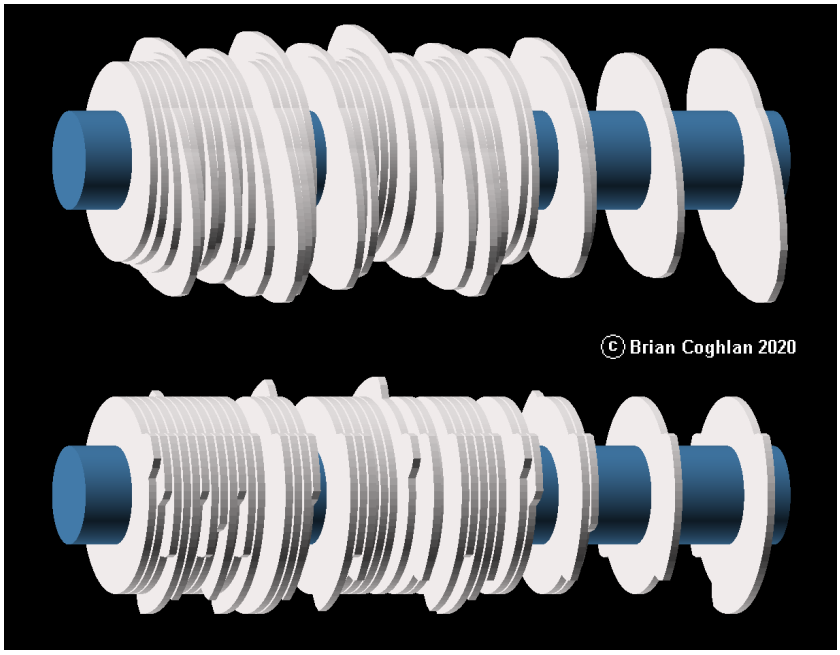


Figure 18 *Top: Double-lobe camshaft with variable height raised segments (proportional to partial-product digit magnitudes) over a fixed arc-angle*
Bottom: Double-lobe camshaft with fixed height raised segments over a variable arc-angle (proportional to partial-product digit magnitudes)
There is one cam per composite index, with the partial product units lobes at the front and tens lobes at the rear
Composite index 0 (partial product 1) is at left, and composite index 66 (partial product 49) is at right
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

In the absence of detail about the mysterious “conveyance”, it is useful to outline three very different speculative but workable solutions that adhere very closely to Ludgate 1909 and Engineering [32]. These cannot be the only possible solutions, but the mechanical details of any other must also adhere.

The first solution assumes "duration" is time, all blades have the same height, blades can independently slide upwards towards pointers, "blade movement" is a fixed distance of variable duration, "register" means to register “enable accumulation”, "carriage near Index" is a bar between Index and Mill, and "move from units to tens" means to move the bar from units to tens digits. The blades are moved up by double-lobe cams with fixed height variable arc-angle raised segments, where the angle is proportional to the partial-product digit magnitude (see Figure 17 right). The “carriage” bar moves between units and tens digits in lockstep with the cams, i.e. the bar jogs, not the Index or Mill. When raised by its blade, a pointer activates its mysterious “conveyance”, a

flexible cable, to stimulate an "enable" lever below the bar to engage escapements or gearing synchronized to the main motor shaft to increment/decrement either its units or tens Mill digit wheel as per the bar position.

An early 3D rendition of this possible “re-imagined” Index is shown in Figure 19, with the Mill assumed to be fixed in position beyond the movable decimal-point carriage bar (at the end of the flexible cables), see further below.

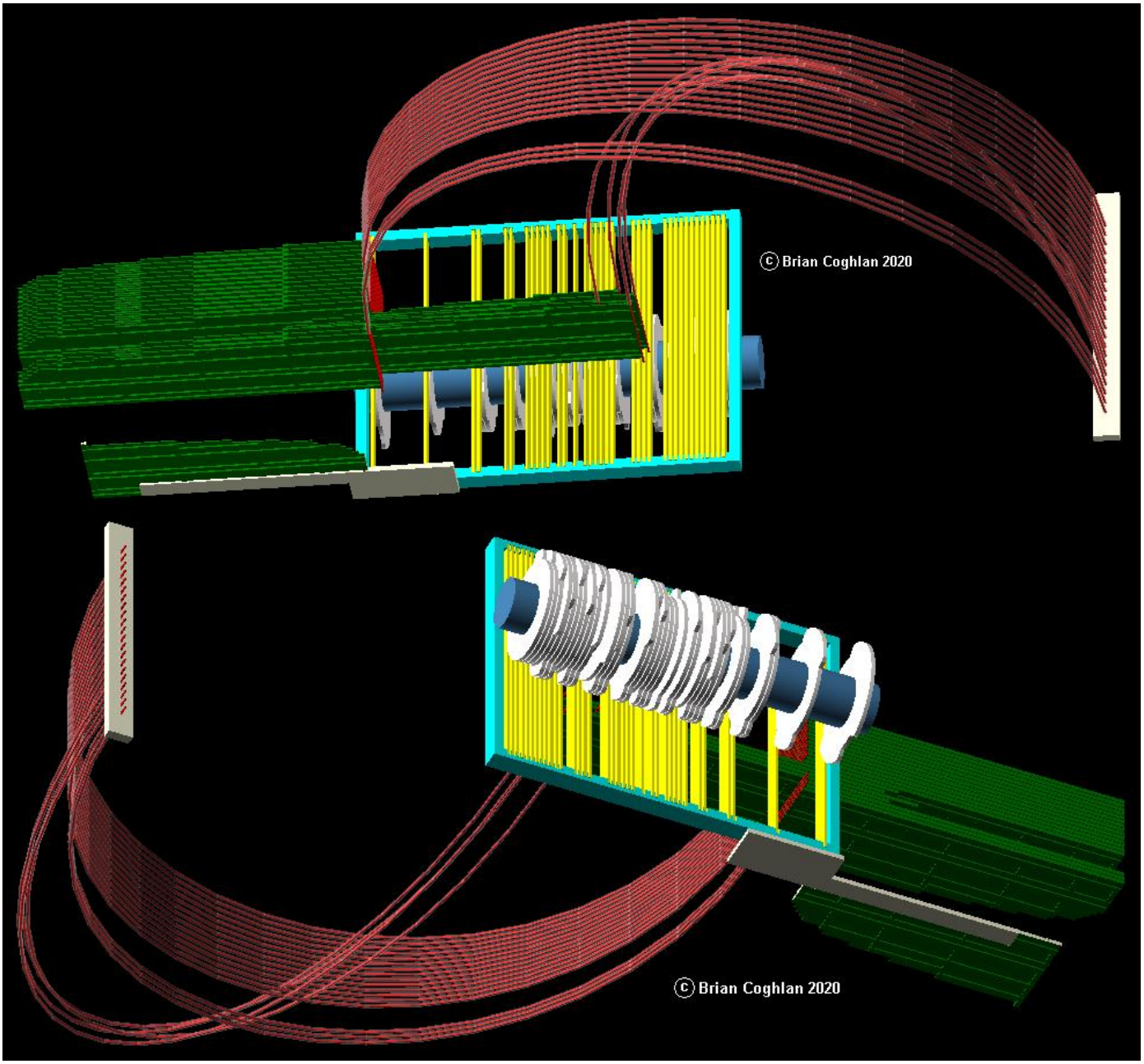


Figure 19 *Early 3D renditions of a possible re-imagined Ludgate Index with flexible cables terminating in decimal-point carriage bar*
Images reproduced courtesy The John Gabriel Byrne Computer Science Collection

As long as the blade is raised, the digit will increment/decrement. “Enable levers” below the decimal-point carriage bar are pushed by the flexible cables to engage a mechanism in one way for increment, and another way with reverse rotation for decrement (as per Ludgate 1909). Figure 20 shows such a reversible escapement mechanism, although reversible gearing is equally applicable. It appears that most mechanical counters use escapement for their first (least significant) stage to guarantee discrete incrementing. In the case of Ludgate’s Mill, each wheel is in effect its own first stage, so an escapement may be the most appropriate mechanism.

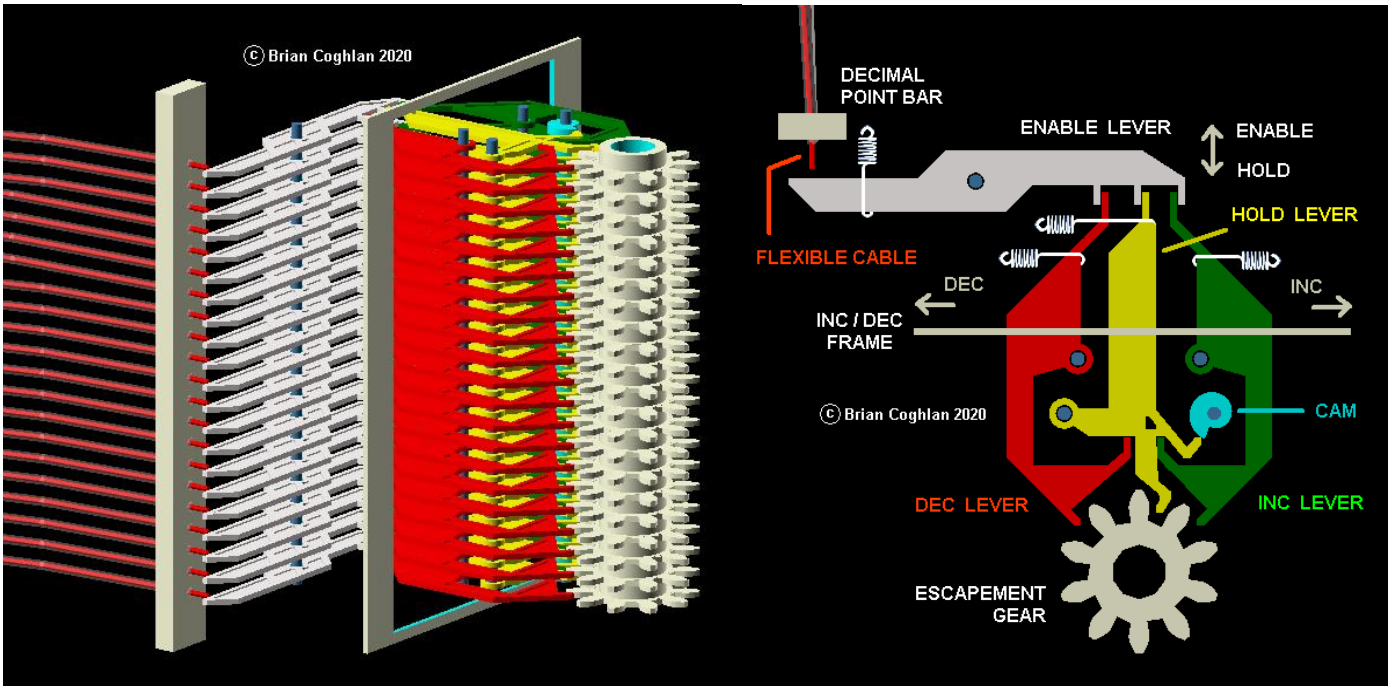


Figure 20 *Speculative "conveyance mechanism using flexible cables to "enable" Mill wheel increment/decrement escapement*
TO BE DONE: *The top layer of escapement levers (included as a placeholder) should be a sign management mechanism*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

The mechanism shown in Figure 20 is very speculative. An enable lever keeps increment, decrement and hold levers in the "hold" state until the flexible cable pushes the enable lever into the "enable" state. A snail cam continuously rotates in synchronism with the blade cams to trigger increment or decrement by releasing the hold lever, which itself allows either an increment or decrement lever to strike an escapement gear. Increment or decrement is selected by moving an "inc/dec frame" sideways such that it keeps either the increment or decrement lever in its "hold" state. The enable lever and inc/dec frame are only moved when the cam is in the "hold" region. The increment, decrement and hold levers are spring-loaded as required for escapement. In contrast the enable lever needs only light spring-loading to maintain the hold state. Thus the flexible cables only require light activation force (an important requirement). This escapement mechanism is replicated for each of the 20 Mill digits as shown. A twenty-first escapement gear is included for overflow detection.

This design is only illustrative, as the cam profile, gear lever involution angles, increment and decrement facets, and their relative positioning, have not been formally designed, and the design does not include a carry mechanism nor a sign management mechanism. Nonetheless, eight common problems are overcome. Firstly, if a slide is at index 50 (representing zero) then there is no blade to raise its pointer, so no increment/decrement will occur. Secondly, any partial product that terminates on any part of the decimal-point carriage bar that is beyond the Mill will not be added. Thirdly, any Mill digit that is beyond the decimal-point carriage bar will be unaffected. Fourthly, the motive power for the blades and pointers comes from the blade cams, and that for the Mill comes from the motor, neither comes from the flexible cable "conveyance" (which is driven by the pointers). Fifthly, the flexible cable "conveyance" has a very tolerant binary action: either "enable" or "disable". Sixthly the Mill design is not constrained. Seventhly, the flexible cables allow the Mill to be wider than the Index, which allows a more complex Mill wheel design and also decouples the Mill design from the Index design. Finally, if the flexible cable cores are made of spring steel then the cables provide the motive power for moving the slides once they are released.

The second solution assumes "duration" is length, all blades have the same height, blades can independently slide upwards towards pointers, "blade movement" is a variable distance of fixed duration, "register" means to register "data", "carriage near Index" is a bar between Index and Mill, and "move from units to tens" means to move the bar from units to tens digits. The blades are moved up by double-lobe cams with fixed arc-angle raised segments with a variable height, where the height is proportional to the partial-product digit magnitude (see Figure 17 left). The "carriage" bar moves between units and tens digits in lockstep with the cams, i.e. the bar jogs, not the Index or Mill. When raised by its blade, a pointer activates its mysterious "conveyance", a tooth to jam a ratcheted rack that is driven by gearing from the main motor shaft to increment or decrement either the units or tens Mill digit wheels, see Figure 21.

TO BE DONE

Figure 21 *Speculative "conveyance mechanism using a tooth to jam a ratcheted rack to halt Mill wheel increment/decrement*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

Until the blade raises the pointer high enough to jam the rack, the digit will increment/decrement. The rack engages gearing for increment, and other gearing with reverse rotation for decrement (as per Ludgate 1909). Similar common problems are overcome by this approach, e.g. if a slide is at index 50 (representing zero) then there is no blade to raise its pointer (so no increment/decrement will occur), nor is the Mill design constrained.

The third but very different solution assumes "duration" is length, all blades are fixed in the inner frame, the whole inner frame with inner slide and blades moves upwards towards pointers, "blade movement" is a variable distance of fixed duration, "register" means to register “data”, "carriage near Index" is what positions the Index relative to the Mill, and "move from units to tens" means to jog the whole inner frame with inner slide and blades sideways from units to tens digits. The blades have a stepped profile representing partial product units and tens in a repeating pattern across the blades, i.e. the units and tens steps each have their own variable height proportional to the partial-product digit magnitude, equivalent to the “duration”. The whole frame is moved up by a double-lobe cam with two identical fixed arc-angle raised segments of height proportional to the maximum partial-product digit magnitude (see Figure 17 left). The “carriage” jogs the inner frame between units and tens digits in lockstep with the cam, i.e. the Index jogs relative to the Mill. When raised by its blade, a pointer activates its mysterious “conveyance” (e.g. either of the conveyances described above) to increment/decrement the Mill. One issue is that the jogging and upward movements of the inner frame decouple the inner slide from the inner shuttle, so the frame will need to be locked in its position on the *I*-axis while the movements take place. An interesting variant would employ separate units and tens pointers that directly ‘register’ the blade displacements on ratcheted racks along the pointers (with blade profiles ensuring only one pointer moves at a time), so that when the twenty outer sliders are returned to their starting position they physically ‘convey’ the pointers to engage with the Mill, where the displacements are reversed, thereby incrementing/decrementing the Mill – all this would be challenging on a ⅛” pitch, but might resolve some mysteries regarding the Mill (see Section IV.3) and carry propagation (see Section IV.6).

All three are entirely speculative solutions to an absence of any relevant detail in Ludgate 1909 or Engineering [32].

IV.2. Ludgate’s special mechanism for ordinals

Ludgate 1909 says: "ordinals are not mathematically important, but refer to a special mechanism which cannot be described in this paper“ (why? patent application?). Ordinals simply rank the logarithmic indexes in increasing order, but the statement implies they are used in some way. Ludgate gives no indication that the shuttles hold ordinals rather than ordinary integers, but Riches (University College Swansea) [42] assumes the shuttles hold ordinals, whilst McQuillan [38] both assumes this and discusses why. Ludgate does make clear that the Mill wheels hold decimal values, so there are two obvious possibilities:

(1) Shuttles hold decimal numerical values, so write-back would be unmapped from Mill to shuttles for Storage, and the special mechanism would convert the shuttle numerical values to ordinals for input to the Index. This would need two converters, one each for outer and inner shuttle, but would allow a fanout from the shuttles to a wider Mill.

(2) Shuttles hold ordinals, so write-back would be mapped via the special mechanism to convert Mill numerical values to ordinals for Storage. This would need just one converter at the output of the Mill.

Simplicity would favour option (2), i.e. shuttles hold ordinals.

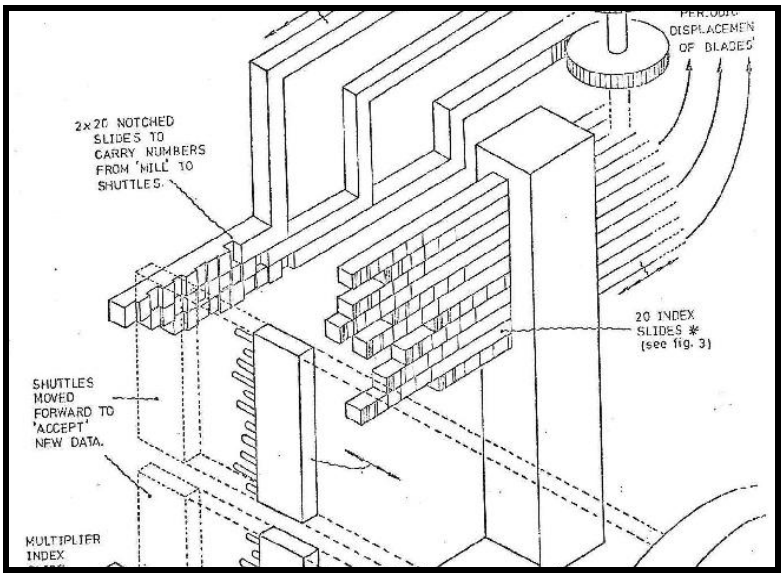
How then could the special mechanism convert from the numerical values to ordinals? Figure 4 on page 76 of Riches report [42], an excerpt of which is shown in Figure 22, is instructive. It proposed that to store a result from the Mill, a shuttle (presumably with all rods extended) should be pushed to mate with notched sliders displaced according to the Mill wheel rotations (perhaps via racks). Shuttle rods would then be backed into the shuttle according to notch depths, so that Mill wheel rotations would map to notched slider displacements, which would then map to notch depths proportional to result values, where the latter mapping could be to ordinals. In this case some further mechanism would be needed to allow Ludgate’s Divide and Log cylinders to store ordinal results directly into shuttles, or perhaps to engage with the notched sliders as needed to achieve this.

This is a viable solution, but could easily be extended to fully integrate the Divide, Log, and any other conceivable cylinders. Ludgate 1909 states a cylinder value “comes opposite to a set of rods. These rods then transfer that number to the proper shuttle” – these “rods” could be Riches’ notched sliders. This could allow integration: each Mill wheel could drive other cylinders, e.g. an extra cylinder[1] with values representing the Mill contents, alongside the Divide and Log seeding cylinders, all sharing the “special mechanism” between the Mill and Storage (see Section IV.3 for how this might resolve other mysteries). As per Ludgate 1909, the 20-digit Divide and Log cylinders would be addressed with the three most-significant Mill digits (see Section IV.8) – the example extra cylinder would need to comprise either 20 cylinders, each independently addressed by its Mill digit, or perhaps 7 cylinders of 0-999, each addressed by a different subset (triple) of the twenty Mill digits (the latter solution would ensure all cylinders had identical diameters and mechanisms, although 0-99 of the Divide and Log cylinders would be unused). All cylinders could output results to a shuttle via the “special mechanism”, which would realize mapping to ordinals. Alternatively that mapping could be done by the cylinders, with hole depths proportional to ordinals and the notched sliders realizing identity mapping. In either case Figure 22 shows the resulting path via which such cylinders would in effect write ordinals to shuttles.

[1] In previous versions of this document this was called an “Ordinal” cylinder, a potential misnomer.

Therefore let us propose that the formats of the numbers in the shuttles, Index and Mill are ordinals, simple/compound indexes, and ordinary integer numerical values respectively, and that all cylinders engage the special mechanism to convert numerical values to ordinals. The propositions need to be carefully scrutinized, as clearly Ludgate could have intended other arrangements.

Proposition 1: Let us propose numbers in Storage are ordinals.
Proposition 2: Let us propose Ludgate's "special mechanism" is implemented for any cylinder to convert from decimals to ordinals.
Proposition 3: Similarly, let us propose Ludgate's Divide, Log and any other cylinders are addressed in decimal.



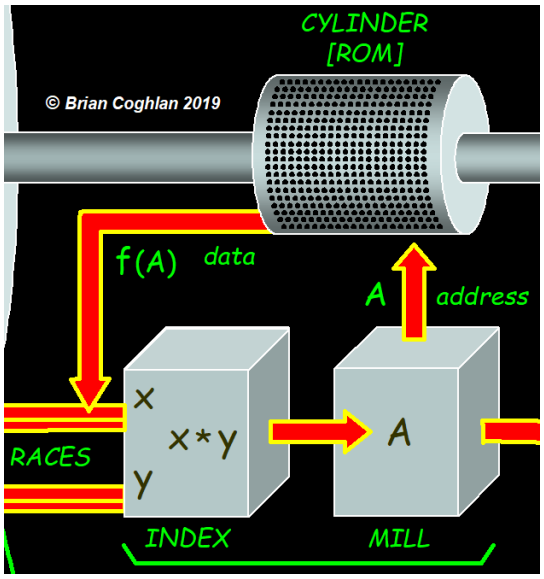


Figure 22 *Left: Figure 4, page 76, of Riches report, showing shuttle rods pushed to mate with notched sliders*
Image reproduced courtesy Professor John Tucker, University College Swansea, UK
Right: Conjectural diagram of path for cylinders to write ordinals to shuttles
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

IV.3. Ludgate's Mill

More complex Mills could not fit within ⅛", but might within ¼" or ⅜". This highlights that it might be convenient to base the Mill on a wider pitch than ⅛" (especially to propagate carries, address Divide, Log and any other conceivable cylinders, and support any related actions).

The mystery of conveyance from pointers also impacts the Mill. Ludgate 1909 gives little detail:

(1) "a carriage near the index now moves one step to effect multiplication by 10". He doesn't say the "mill carriage", so it seems the Mill does not move, instead a separate carriage moves.

(2) "the duration of [blade] displacements are recorded in units measured by the driving shaft on a train of wheels called the mill". As discussed before, this might suggest the Mill is incremented by the driving shaft, not by the pointers, i.e. "duration" is time.

But Engineering [32] includes new and puzzling details of how the Mill is affected:

(3) For the 1st digit "9" of the multiplier "All these partial products are registered in the mill below, as indicated", and "In a final operation these partial products are added together as indicated, giving 7,318,800". Does this "final operation" imply racks or wheels that "register" partial products, and that the "final operation" adds these to further "partial-sum wheels"? Or does it just mean "finally the result is 7,318,800"?

(4) For the 2nd digit "2" of the multiplier "These partial products will then appear on the mill and be added together, giving the result of the multiplication of 813,200 by 2" (not 7,318,800 + 813,200*2). Does the absence of accumulation mean a set of "partial-sum" wheels for each multiplier digit?

(5) Finally "The process is repeated for the remaining figures of the multiplier, and the whole added together so as to give the product of 813,200 x 9247". Does this mean yet another "final-product" set of Mill wheels? Or again does it just mean "finally the result is 813,200 x 9247"?

These sentences appear to be in contradiction with Ludgate 1909, which seems to imply that the partial products are accumulated on just one set of Mill wheels. What does "register" mean? Mechanically, it may mean to physically "convey" the partial products to the Mill, with jogging right then left to add units then tens, perhaps via gears on either side of Mill wheels, or via "open" or "spur" differentials [43]. Alternatively, "register" may reflect a different use of language, e.g. to register an "enable" by activating a mechanism to allow accumulation by the Mill as in Figure 20 or Figure 21.

As a further alternative, “register” could mean transfer to one set of Mill wheels, but accumulation on a second “final-product” set of Mill wheels as per (3) and (5) above. Ludgate’s use of cylinders as lookup tables would allow them to double-act as this second stage of the Mill, rotating to accumulate in decimal (eliminating “addressing” of cylinders), and resolving contradictions. It would be physically efficient if cylinders were grouped as 7 cylinders of 0-999, each accumulating for a different triple of the twenty Mill digits, consistent with suggestions in Section IV.2 above; this would provide a ⅜” rather than ⅛” pitch for the mechanism, and reduce carry mechanisms by a factor of three. The penalty would be increased accumulation time (counting 0-999 rather than 0-9), which would contradict Ludgate’s timing claims, so mystery remains.

In addition, the term “register” may be related to handling decimal-points. If numbers have a decimal-point, then each multiply result decimal-point will depend on its operand decimal-points, and may differ from the accumulated Mill decimal-point. So the result may need to be aligned to the Mill decimal-point before accumulation.

IV.4. Ludgate’s data format

Ludgate’s numbers used a sign-magnitude format. Figure 1 shows an ‘***x***’ beyond the least-significant end of the digits of the outer frame. This can only be the sign, and therefore the sign is stored beyond the least-significant rather than the most-significant digit. Given multiplication begins at the MS digit, this may be mechanically significant.

Lemma 10: sign is stored beyond the least-significant digit.

Were numbers left or right justified? For the case where S.index(8132)=3071 and S.index(0)=50, Ludgate 1909 stated that “the first four slides will therefore move 3, 0, 7, and 1 units respectively, the remainder of the slides indicating zero by moving 50 units." Does “first four” imply numbers are right-justified? The answer is no, since if the “remainder” was all leading zeroes, why say “first four”, as a fifth would not exist.

Ludgate 1909 also stated “Another slide moves … to the simple index number of the first digit of the multiplier”. This (and more) indicates long multiplies started with the multiplier left digit. But does that imply numbers are left-justified? The answer in this case is maybe. But there is a problem with left-justification, as then the decimal point position must be known, otherwise “1” could mean any of 10^0 to 10^{20}.

There are only three remaining possibilities: (a) numbers are floating-point, (b) numbers are fixed-point, and (c) numbers are left-justified. Which of these did Ludgate employ? A floating-point format is likely to have been far too ambitious for the time (although Torres y Quevedo published a paper design in 1911 [7]). However, Ludgate said: "the position of the decimal point in a product is determined by a special mechanism which is independent of both mill and index." This suggests any implied decimal point position may be capable of being varied, and that it was an issue for Ludgate. But he also said “rods for every figure of the Variable, and one to indicate the sign”, i.e. only digits and sign were stored in a shuttle. No rod was used for decimal point. Without rods for decimal point, floating-point or left-justified numbers cannot be used. Hence numbers must be fixed-point, with an implied decimal-point placement.

Lemma 11: numbers must be fixed-point

This suggests the decimal point in operands and results would be known to the programmer. The diagram of Engineering [32] shows a multiplicand of "813200", without indicating an implied decimal-point. It could be after digit 6, or as an arbitrary example, say after digit 18:

+813200.00000000000000 or +000000000000813200.00

In the limit, allowing an implied decimal-point provides an exponent range as follows:

$\pm 0.00000000000000000001 = \pm 1.0 \times 10^{-19}$ to $\pm 10000000000000000000 = \pm 1.0 \times 10^{+19}$

Potentially the decimal-point could be controlled or managed from the upper keyboard (like the IBM 360/44 floating-point precision could be varied with a control-panel selector [44]), but Ludgate 1909 stated the number-paper and upper keyboard are synonymous, so this implies any control was via an instruction that could also be read from the number paper.

IV.5. Ludgate’s decimal-point alignment

Wherever the decimal-point is, since divide and log operations rely on multiply, add and subtract operations, and the latter two preserve any uniformly implied decimal-point, then multiply may be the only operation for which the decimal-point must be aligned. In most accumulator architectures this is done via shift instructions, but this appears to be excluded by Ludgate’s statement that decimal-point position is "determined independently of these [divide and log] formulae", i.e. those formulae (which use multiply intensively) neither know nor care about the decimal-point.

He also states “the position of the decimal-point in a product is determined by a special mechanism which is independent of both mill and index”. This suggests a separate mechanism that can be set to determine the decimal-point for subsequent operations, and that its impact is on the product, not the Mill. The special mechanism would need to be set statically or dynamically as required by the programmer. It would then align subsequent products with the accumulated result.

By way of example, for multiplying 6-digit numbers with the decimal-point in the middle:

005.210 * 002.710 = 000014.119100

This can be represented as integer multiplication followed by scaling (shifting right by 3 digits):

005210 x 002710 = 000014119100 / 1000000 = 000014.119100 ⟶ 014.119

Within the special mechanism, scaling is most easily done by shifting. There are two reasonable options:

a) Pre-shift with a 20-digit product or wider beyond the least-significant digit for more accuracy: repeated shifting during multiply is required, but overflow detection is easy.
b) Post shift with a 40-digit product: overflow detection is harder, but shifting can be a subsequent action.

Any alignment operation is likely to have been designed to allow left or right shifts by 1<N<20 digits.

Lemma 12: the "special mechanism" is likely to support an alignment operation.

A 40-digit product as per (b) raises other issues, e.g. rounding (but Ludgate 1909 does not mention it), or the fact that Ludgate only allowed 20 time-units for the final ripple carry (but it is feasible to do only 20-time-units of the ripple carry).

The early 3D rendition of a “re-imagined” Index shown in Figure 19 includes a bar as the special mechanism. The bar acts on an "enable" lever below it to engage a gear (per digit) from the main motor shaft, so as long as a blade/pointer is raised, its digit(s) will be enabled to increment/decrement. To change the decimal-point position, here the slides, blades, pointers and Mill do not change position, only the bar shifts left/right, but since the principle is relative alignment between bar and Mill, the bar might equally well be fixed and the Mill shifted. An instruction shifts the bar versus Mill so that only the digits that should for that decimal-point setting are affected. This mechanism supports non-accumulation of zeroes, and also "unaffected digits", where depending on the alignment, some Mill digits will be beyond the bar, and so the enable lever for those "unaffected" digits will not move. This workable solution adheres very closely to Ludgate 1909. Again this cannot be the only possible solution, but any others must also adhere.

Alignment might also explain Ludgate’s choice to multiply in reverse to the traditional order (with its awkward step left then two steps right mentioned in Section III.1), as it may have been an optimization to allow early termination of multiplication once the partial products became less significant than the Mill’s least-significant digit (but his fixed timing for multiply implies this was not done), and/or to allow skipping zeroes more quickly, and/or to facilitate carry propagation.

IV.6. Ludgate’s carry propagation

Ludgate said he had a new way to do carrying which decoupled adding and carrying, and he stated that the “carrying is practically in complete mechanical independence of the adding process, so that the two movements proceed simultaneously”, that “the sum of *m* numbers of *n* figures would take (9*m* + *n*) units of time” (“*n*” represents a final ripple carry at the end of a sequence of *m* additions), and for multiply, the *n* = 20 digit multiply-accumulate involving *m* = 40 additions would take ((9 x 40) + 20) = 380 time-units. If carrying is serialized, multiply would take 420 time-units, therefore his timings suggest synchronous pipelining of ripple carries, overlapped with mechanical movements, 30 years before the simple use of pipelining in Zuse’s Z1 and Z3 computers and 60 years before its use in supercomputers [45].

Ludgate’s multiplier inner slide and frame proceeds from MS to LS multiplier digit. For any multiplier digit *N*, after the partial products units register/add there is mechanical movement of a “carriage near the Index" in order to switch to tens. If units carry was synchronously latched, then its propagation could be overlapped with this movement. After the partial products tens registers/adds, the movements back to the units digits occurs, and if their tens carry were synchronously latched then their propagation could be likewise overlapped. In each case (sum + carry) is always less than 19 so double-carries (i.e. >19) are avoided, in contrast to if units and tens carries were propagated together. The “carriage” therefore gives each digit two opportunities to propagate a carry, allowing up to two synchronous carries overlapped ("hidden") behind carriage movements, and optionally another (highly unlikely) one behind the inner slide and frame movement to the next LS digit.

Proposition 4: synchronous ripple carry propagation is overlapped with mechanical carriage movements, effectively taking zero time-units.

The corollary is that the time allowed for these carriage movements must be long enough for the carry to increment the wheel.

Proposition 5: time allowed for the mechanical carriage movement is one time-unit.

A counter-argument is that if the two movements do actually proceed simultaneously, then Ludgate’s solution was different, for example employing a differential in the Mill or decimal-point “special mechanism” as mentioned above, or even a Mill with a differential and second set of wheels.

IV.7. Ludgate's Mill microarchitecture

The data format. alignment, and carry propagation all impact on the Mill mechanism. Its microarchitecture is very clearly shown when describing a single digit of the Mill as an algorithmic state machine, for example as per the speculative pseudo-OCCAM Algorithm 2 below, which assumes Proposition 4 and Proposition 5 directly above are true, and handles both positive and negative numbers, but ignores decimal-point alignment. Table 5 shows its expected behaviour. This demonstrates how within each multiplier iteration, each digit can allow two separate synchronous carries overlapped ("hidden") behind the separate units and tens carriage movements, hence ensuring there is never a carry of two, as outlined in the previous section.

Mechanically, "Wheel" can be each "figure-wheel" (digit wheel) of Ludgate's Mill, where "Wheel = [9 | 0]" signifies the typical lug between "9" and "0" on the wheel, "CYout" is a typical mechanical latch, and ":=" the starting or active transition of Ludgate's "time-unit being the period required to move the figure-wheel through ⅒ revolution". For negative numbers CYin and CYout represent borrow input and output.

```
SEQ  i = 1 to n                                          # n is the number of digits (20)
   ################### UNITS #################
   PAR
      moveTo ( Units );                                  # mechanical "carriage" movement to units
      IF ( CYin = 1 )                                    # overlap carry/borrow propagation if necessary
         IF ( POSITIVE )  PAR
            IF ( Wheel = 9 ) CYout := 1 ELSE CYout := 0; # latch carry output as needed
            Wheel := Mod10 ( Wheel + 1 ) ;               # hidden pipelined ripple carry propagation
         ELSE  PAR
            IF ( Wheel = 0 ) CYout := 1 ELSE CYout := 0; # latch borrow output as needed
            Wheel := Mod10 ( Wheel - 1 );                # hidden pipelined ripple borrow propagation
   SEQ p = 1 to partialProduct ( Units )               # iteratively add partial product units (inc/dec wheel)
      IF (POSITIVE )  PAR
         IF ( Wheel = 9 ) CYout := 1;                  # latch carry output as needed
         Wheel := Mod10 ( Wheel + 1 );                 # add partial product units
      ELSE  PAR
         IF ( Wheel = 0 ) CYout := 1;                  # latch borrow output as needed
         Wheel := Mod10 ( Wheel - 1 );                 # subtract partial product units
   ################### TENS #################
   PAR
      moveTo ( Tens );                                   # mechanical "carriage" movement to tens
      IF ( CYin = 1 )                                    # overlap carry/borrow propagation if necessary
         IF ( POSITIVE )  PAR
            IF ( Wheel = 9 ) CYout := 1 ELSE CYout := 0; # latch carry output as needed
            Wheel := Mod10 ( Wheel + 1 );                # hidden pipelined ripple carry propagation
         ELSE  PAR
            IF ( Wheel = 0 ) CYout := 1 ELSE CYout := 0; # latch borrow output as needed
            Wheel := Mod10 ( Wheel - 1 );                # hidden pipelined ripple borrow propagation
   SEQ p = 1 to partialProduct ( Tens )                # iteratively add partial product tens (inc/dec wheel)
      IF (POSITIVE )  PAR
         IF ( Wheel = 9 ) CYout := 1;                  # latch carry output as needed
         Wheel := Mod10 ( Wheel + 1 );                 # add partial product tens
      ELSE  PAR
         IF ( Wheel = 0 ) CYout := 1;                  # latch borrow output as needed
         Wheel := Mod10 ( Wheel - 1 );                 # subtract partial product tens
################### FINAL RIPPLE CARRY #################
SEQ  k = 1 to n                                          # n is the number of digits (20)
   IF ( CYin = 1 )
      IF ( POSITIVE )  PAR
         IF ( Wheel = 9 ) CYout := 1 ELSE CYout := 0;    # latch carry output as needed
         Wheel := Mod10 ( Wheel + 1 );                   # final pipelined ripple carry propagation
      ELSE  PAR
         IF ( Wheel = 0 ) CYout := 1 ELSE CYout := 0;    # latch borrow output as needed
         Wheel := Mod10 ( Wheel - 1 );                   # final pipelined ripple borrow propagation
```

Algorithm 2 *Example speculative pseudo-OCCAM description of a single digit of Ludgate's Mill*

Example	Before mechanical carriage movement			After mechanical carriage movement			After iterative addition	
	CYin	Wheel	p.Product	Wheel	p.Product	CYout	CYout	Wheel
1	0	4	4	4	4	0	0	8
2	1	4	4	5	4	0	0	9
3	0	4	5	4	5	0	0	9
4	1	4	5	5	5	0	1	0
5	0	9	0	9	0	0	0	9
6	1	9	0	0	0	1	1	0
7	0	9	9	9	9	0	1	8
8	1	9	9	0	9	1	1	9

Table 5 *Example expected behaviour of a single digit of Ludgate's Mill as described by speculative Algorithm 2*

IV.8. Ludgate's division and logarithm algorithms

The way in which Ludgate performed division was very novel for 1909. Seeded with an approximate value A of the reciprocal of the divisor q, the algorithm successively converged to the correct answer, using just multiply/accumulate. Using the Binomial Theorem he expanded division to:

$$\frac{p}{q} = \frac{A\,p}{A\,q} = \frac{A\,p}{1+x} = A\,p\,(1+x)^{-1} = A\,p\,(1 - x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + x^8 - x^9 + x^{10}) + etc \quad \ldots\ldots\ldots [A]$$

$$= A\,p\,(1-x)(1+x^2)(1+x^4)(1+x^8) + etc \quad \ldots\ldots\ldots [B]$$

The seed A is a 20-digit value taken from a lookup table that contains all the reciprocal values $A = \frac{1}{f}$, where ($100 < f < 999$) is equal to $q[1..3]$, i.e. digits 1-3 of q. As a result A will be slightly larger than q, and therefore $A\,q = 1 + x$, where x is a small fractional value. This relation can be reversed to yield $x = A\,q - 1$ in order to evaluate the series above. Ludgate 1909 states that series A up to x^{10} and series B up to x^8 converge to the value of $(1 + x)^{-1}$ correct to at least twenty figures.

The statement that "the quantity A must be the reciprocal of one of the numbers 100 to 999", i.e. f = 100-999, is a clue. For fixed-point it would be expected that f = 000-999, whereas 100-999 clearly indicates the divisor was left-justified, not fixed-point, having excluded divide-by-zero beforehand.

Ludgate 1909 states "the 900 values of A are stored on a cylinder—the individual figures being indicated by holes from one to nine units deep on its periphery", addressed by $q[1..3]$. The row of holes for A "comes opposite to a set of rods. These rods then transfer that number to the proper shuttle, whence it becomes an ordinary Variable". This means the store instruction either must have a "source operand" that specifies that its data comes from the Divide, Log, or any other conceivable cylinder, or there must be an instruction variant for each source. See Section IV.11 below for a discussion about this.

Ludgate 1909 then states A is "used in accordance with the formula" where a "dividing cylinder, on which this formula is represented in the proper notation of perforation" within the machine's sequencer contains the algorithm to evaluate the series. This means A must be stored in a shuttle reserved for use by the division formula, and that A must either be preserved across or precluded from preemption. It also means the division formula uses the standard instruction perforations, so it is a built-in subroutine like DEC AlphaPALcode [46] rather than microcode (which would use a lower-level of micro-instructions).

According to Boys 1909 the machine first calculates the value of the series, then multiplies the result by $A\,p$. Including the lookup of A, the division might be performed as per Algorithm 3, with steps (6-8) calculated at a lower level as per Algorithm 4.

(1) Load q into the Index or Mill
(2) IF (q = 0) THEN error
(3) Left-justify q
(4) Extract the 3-digit value ($100 < f < 999$) = $q[1..3]$
(5) Transfer the 20-digit value A = lookup($\frac{1}{f}$) from the DIV cylinder to storage
(6) Calculate $x = (A\,q - 1)$ and its powers
(7) Calculate the series $S = (1 - x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + x^8 - x^9 + x^{10})$ ..... [C]
or possibly the series $S = (1-x)(1+x^2)(1+x^4)(1+x^8)(1+x^{16})$ ..... [D]
(8) Calculate the final result $\frac{p}{q} = A\,p\,S$

Algorithm 3 *Ludgate's division algorithm*

10	MUL (*A, q*)	# calculate & store $A\ q$	54	MUL (*-x, +1*)	# load $-x$
11	STO (*Aq, -*)		57	MAC (*x2, +1*)	# calculate $-x + x^2$
14	MUL (*+1, +1*)	# calculate & store $-x$	60	MAC (*-x3, +1*)	# calculate $-x + x^2 - x^3$
17	MAC (*Aq, -1*)		70	MAC (*-x3, -x*)	# calculate $-x + x^2 - x^3 + x^4$
18	STO (*-x, -*)		73	MAC (*-x5, +1*)	# calculate $-x + x^2 - x^3 + x^4 - x^5$ & store
28	MUL (*-x, -x*)	# calculate & store x^2	74	STO (*tmp, -*)	
29	STO (*x2, -*)		84	MAC (*tmp, -x5*)	# calculate $-x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + etc$
39	MUL (*x2, -x*)	# calculate & store $-x^3$	87	MAC (*+1, +1*)	# calculate $S = 1 - x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + etc$
40	STO (*-x3, -*)		88	STO (*tmp, -*)	
50	MUL (*-x3, x2*)	# calculate & store $-x^5$	98	MUL (*p, tmp*)	# calculate $p\ S$
51	STO (*-x5, -*)		99	STO (*tmp, -*)	
			109	MUL (*A, tmp*)	# calculate final result $A\ p\ S$

Algorithm 4 *Possible low-level in-line implementation of steps (6-8) of division Algorithm 3*
The numbers before the instructions show the estimated timeline in seconds
Instruction types as speculated in Section IV.11

This approach is common in modern computing, although the algorithms are different. Ludgate's seeds were stored on a special cylinder addressed by a subset of the contents of the Index or Mill, essentially a read-only memory used as a lookup table. He designed for a seed lookup table with 900 entries, enough to possibly require a cylinder of the same size as the storage cylinders (although a spiral mechanism might alleviate this). McQuillan [38] discusses the very significant difficulties of Ludgate's approach in some detail. A simpler prototype design could use 90 (or even 9) seeds and still work as he proposed, but converge more slowly. Ludgate 1909 states that division takes 90 seconds, whereas if addition is assumed to take 3 seconds, multiplication takes 10 seconds, and store takes 1 second, then the un-optimized Algorithm 4 takes 109 seconds.

Why did Ludgate not use logarithms for division? The reason is his logarithmic multiplication was mechanically unidirectional. It was done multiplier digit by digit, to yield a 2-digit partial result, before it moved on to the next digits. To do division another mechanism would be needed to take 2 digits of the dividend and 1 digit of the divisor, to yield a 1-digit partial result. While it could be done, it was mechanically much more efficient to have only one mechanism. Ludgate's divide by convergent series still had much better performance than division by iterative subtraction.

Ludgate also "extended this system to the logarithmic series … a logarithmic cylinder which has the power of working out the logarithmic formula, just as the dividing cylinder directs the dividing process. This system of auxiliary cylinders and tables for special formulae may be indefinitely extended." He does not specify the calculation of logarithms, but for example, the natural logarithm might have been calculated using the MacLaurin expansion:

$$ln\ (1+\mathbf{x}) = \mathbf{x} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} + etc$$

Ludgate 1909 states the logarithm calculation takes 120 seconds.

IV.9. Ludgate's storage

Ludgate's arrangement for storage was novel and a major advance on anything before. Numbers were represented as a rod for sign and for each of 20 digits, with every digit protruding 1-10 units. For example, these could be square rods with notches, with some retention. These 21 rods were stored in shuttles, see Figure 23. Two concentric cylinders (one inner and one outer) held the shuttles, i.e. the inner cylinder was inside the outer cylinder. There were a total of 192 shuttles. Essentially he proposed either two linearly addressed memory partitions, or in a single linearly addressed memory partition.

How were the shuttles arranged? Let us assume the shuttles are all the same size, and occupy the same slice width W at their cylinder inner-radius. If inner and outer cylinder inside-radii are A and B respectively, cylinders hold $N = \frac{2\pi A}{W}$. and $M = \frac{2\pi B}{W}$. shuttles each, and $N + M = 192$, then $N = \frac{192\ A}{A + B}$. Each shuttle holds 21 rods, and if each is the fundamental unit of the machine size of $\frac{1}{8}$", then the minimum height of shuttle is $(B - A) = 2\frac{5}{8}$", realistically 3", i.e. the outside-diameter of the outer cylinder is then $(2A + 12)$". Ludgate 1909 states the machine is machine is 26" long, 24" broad, and 20" high. Therefore $(2A + 12) < 20$, yielding a maximum $A = 4$" and $B = 7$", which sadly leads to an irrational $N = \frac{192\ A}{A + B} = 69.8181\ldots$ But if $A = 3$" and $B = 6$", then $N = 64$, $M = 128$, $M = 2N$, $B = 2A$, and the outside-diameter of the outer cylinder is 18". This simple and very convenient geometric result very strongly suggests Ludgate designed for 64 inner and 128 outer shuttles, most easily addressable in binary.

Lemma 13: storage is arranged as 64 inner and 128 outer shuttles.

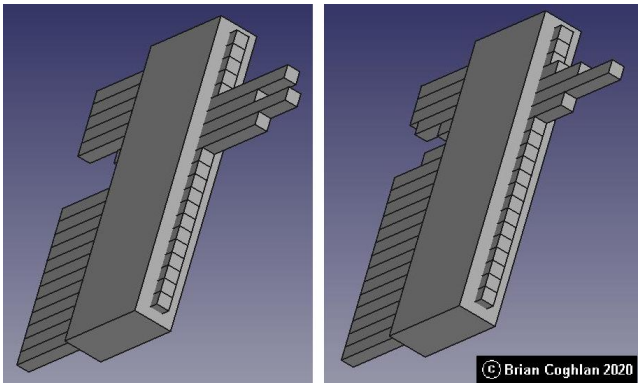


Figure 23 *Speculative schematic of shuttles storing +813200.00 as ordinal +304199.99 (left) and +9167.00 as ordinal +6058.99 (right). The sign is the rod at the top, the next rod is the LS digit, and the bottom rod is the MS digit. The stored profiles are in reverse ranking of ordinals, see Table 4. The shuttle housing and rods are to correct scale but not representative in any other way.. Images reproduced courtesy The John Gabriel Byrne Computer Science Collection*

Ludgate 1909 stated that to access shuttle *S*, the appropriate cylinder must be rotated to line up *S* with the Index input race. He also explained that if an inner shuttle held the value *X* and an outer shuttle held the value *Y*, then to multiply *X* * *Y* the inner cylinder must be rotated to shuttle *X* and the outer cylinder rotated to shuttle *Y*, then the Index instructed to multiply *X* * *Y*, then the Mill instructed to accumulate the result, and finally restore the shuttles to their cylinders. The result could then be stored back simultaneously to up to two shuttles. Ludgate did not state where these two shuttles were allowed to be, only that "they do not belong to the same shuttle-box", where there are "two coaxial cylindrical shuttle boxes". Clearly one must be an inner and one an outer shuttle. Partitioning into 64 inner and 128 outer shuttles implies binary addressing via either 6-bit inner and 7-bit outer, or two generic 8-bit binary address fields (the exclusion of the "same shuttle-box" might imply generic addressing).

Ludgate 1909 states "it is important to remember that it is a function of the formula-paper to select the shuttles to receive the Variables, as well as the shuttles to be operated on, so that (except under certain special circumstances, which arise only in more complicated formulae) any given formula-paper always selects the same shuttles in the same sequence and manner, whatever be the values of the Variables". The text in parentheses implies Ludgate allowed selection of a shuttle other than by a literal operand in the formula paper. For example, indirect addressing could be done quite simply by converting the Mill contents to a shuttle selection with another wheel ("Select" wheel) via the Divide and Log 3-digit addressing mechanism; this would be an important aid to execution of complex algorithms. It must be stressed that this is a possible but not the only interpretation.

Ludgate 1909 further states that the shuttles are removable. Maximum flexibility would imply individual shuttles were removable, or whole storage cylinders were removable. This then implies front-loading, and equally implies the races, Index, Mill and Divide, Log, and any other conceivable cylinders were at the rear of the machine.

In order to ensure reasonably fast rotation to select shuttles, the storage cylinder and shuttle inertias would need to be minimised. Riches proposed rectangular rods which slotted together but their weight would give rise to too much inertia. Instead, the cylinders would need to be skeletal (just slotted cylinders), and the shuttles made of a light and strong material, e.g. spring-steel. Ludgate does not say whether the rods are round or square, which will impinge on how well the rods retain their position.

A possible arrangement might have square rods with retention slots on a regular pitch, representing 0-9. If the shuttle sides were made of spring steel and had internal bumps at the same pitch then they would retain the rods in position as well as allow movement of the rods to change value. Lengthwise wobble of the rods could be avoided by ensuring the rod occupied the full width of the shuttle at all times (which would mean the rods would be almost twice as long as the shuttle), perhaps enhanced by bowing the shuttle sides inwards to apply light pressure on the rods. Side bars or shields extending as far as the maximum rod extents in the shuttle would make it easier to register the shuttles at the destination of each movement and would protect the contents while in transit.

Ludgate 1909 states that shuttles are transferred to the Index via races. Provided shuttles, races and Index are oriented in line, either push/pull or hook/move of shuttles seems feasible.

Selecting a shuttle quickly seems problematic. Storage addresses are binary, so successive approximation might be used, perhaps with differentials to successively phase gear ratios, or epicyclic gears [47] to successively approximate addresses. An obvious option, Gray code, was used in Baudot codes c.1878 [48], but not officially invented until 1947 [49]. Alternative simpler approaches using rebound springs and dampers might prove more convincing, as may non-binary selection mechanisms that respond to a binary address.

Figure 24 and Figure 25 show an early 3D rendition of a possible re-imagined Ludgate Analytical Machine's storage and Index, correctly dimensioned. From this it can be seen that the cylinders are quite skeletal, hence probably quite light, which will help reduce rotation and positioning inertias, and thereby reduce the necessary control forces.

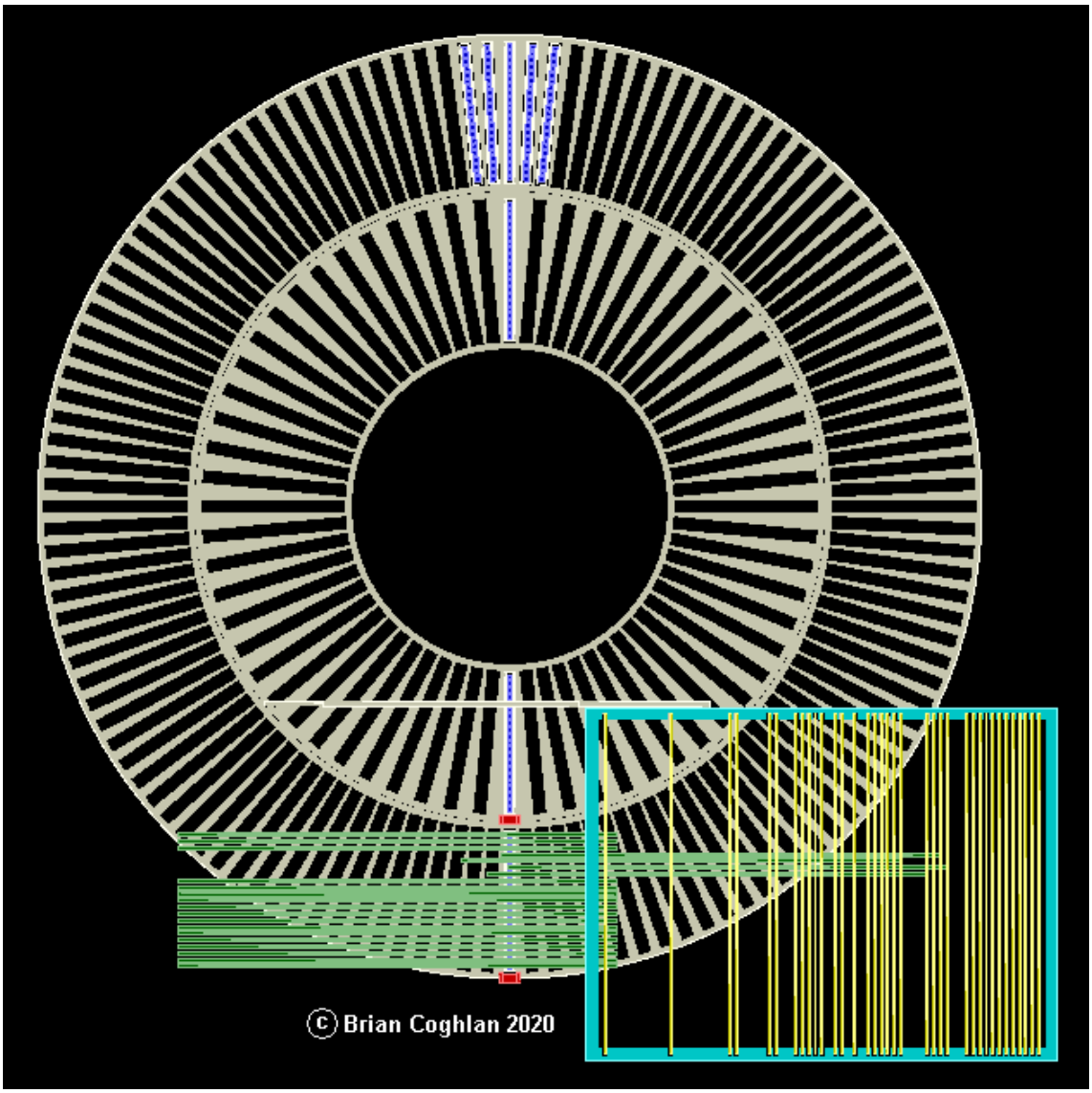


Figure 24 *Early 3D rendition of a possible re-imagined Ludgate's storage and Index*
Top view showing alignment of storage cylinders and shuttles with Index with slides
Showing '813200' multiplied by '9247' as per Engineering [32]
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

In the design shown in Figure 24, the storage address origin (address '0') is at the top centre, with clockwise increasing addresses. Outer shuttles are shown at outer storage addresses 0, 1, 2, 126, 127, while an inner shuttle is shown at inner storage address 0. In addition, Figure 25 shows that an inner shuttle (holding multiplier '9247') has been withdrawn from inner address 32 along the inner race, and an outer shuttle (holding multiplicand '813200') has been withdrawn from outer address 64 along the outer race. The Index is shown having iterated to the digit '9' of the multiplier.

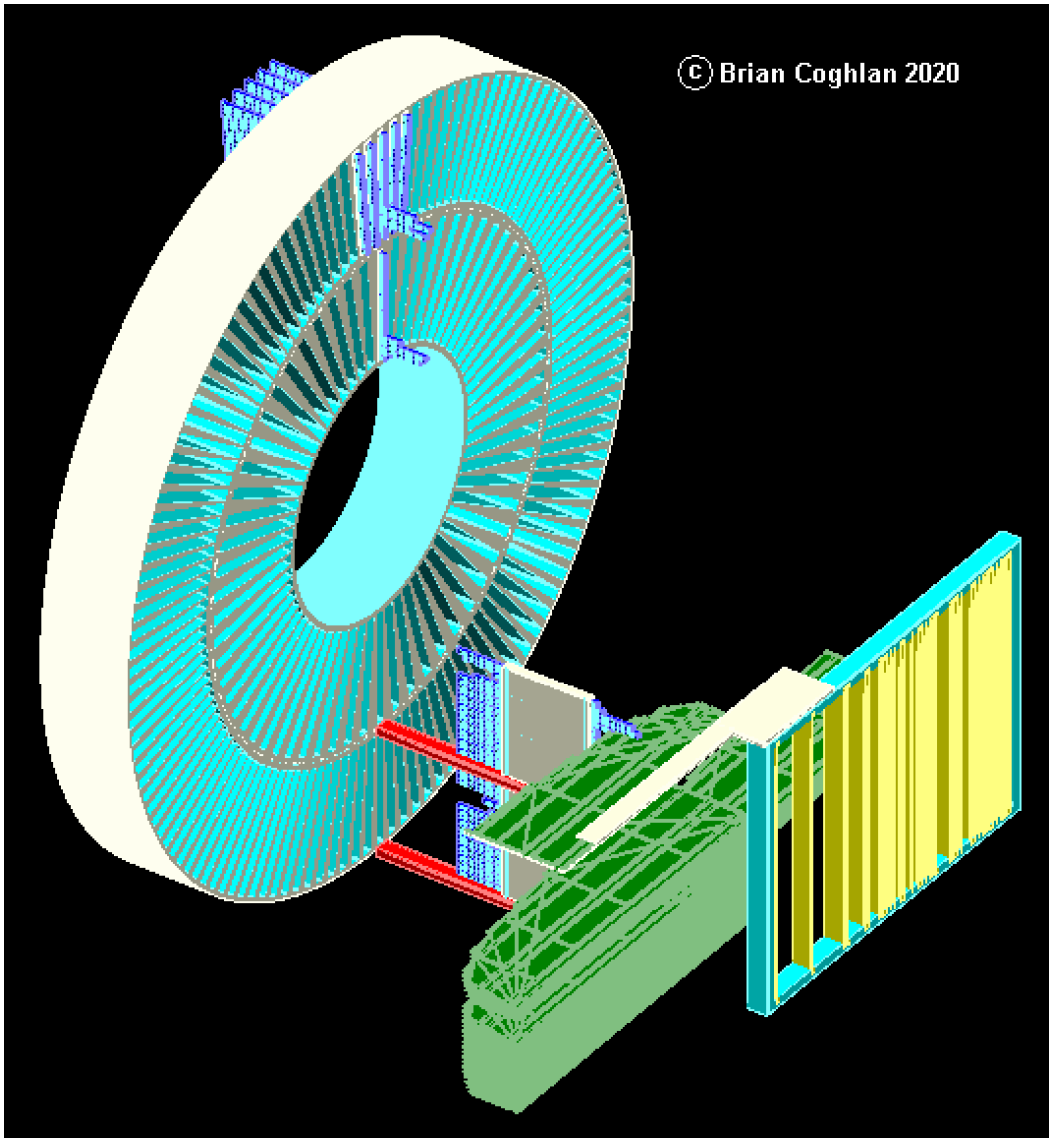


Figure 25 *Early 3D rendition of a possible re-imagined Ludgate's storage and Index*
Three-quarter view showing inner and outer shuttles withdrawn along races
Showing '813200' multiplied by '9247' as per Engineering [32]
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection

IV.10. Ludgate's controlpath

Ludgate 1909 and Engineering [32] are very reticent about the controlpath. Firstly, the former does state "most of the movements … are derived from a set of cams placed on a common shaft parallel to the driving shaft", but says very little else about the mechanism. Secondly and thirdly, and very fortunately, inferences can be made about the core instruction set and preemption.

IV.11. Ludgate's instruction set

What was the format of the basic arithmetic instructions? Ludgate 1909 stated "Each row of perforations across the formula-paper directs the machine in some definite step in the process of calculation—such as, for instance, a complete multiplication, including the selection of the numbers to be multiplied together." Hence each row of the formula tape specifies one instruction, and the arithmetic instruction format must contain an opcode, either a 6-bit inner and 7-bit outer shuttle source operand address, or two 8-bit generic source operand addresses. It is not clear how many arithmetic instructions there were, except for multiply-accumulate (MAC) and in all likelihood multiply (MUL) without accumulation (i.e. pre-clearing Mill). These should be adequate as Ludgate 1909 states that addition is multiply by unity, e.g. MUL(x, 1) and subtraction is, e.g. MUL(x, -1), so the programmer can specify the second operand x as needed. Shifting left could be done with MUL(x, 10^N) and shifting right with

MUL(x, 10^{-N}). This makes sense as mechanically only one arithmetic operation (multiply-accumulate) is implemented, and pre-clearing the accumulator would be very simple. However, it may be that Ludgate defined ADD and SUB variants of MUL and MAC that forced just one inner slide iteration rather than a full multiply, with optimizations to skip zeroes.

Lemma 14: the basic arithmetic instruction format contains an opcode.
Lemma 15: the basic arithmetic instruction format contains at minimum a 6-bit inner shuttle source operand address.
Lemma 16: the basic arithmetic instruction format contains at minimum a 7-bit outer shuttle source operand address.

What was the format of the extended arithmetic instructions like division, logarithm, a^n, etc? While the logarithm instruction could act on Mill contents, that could not be "indefinitely extended" to a^n and other such functions. Furthermore, Ludgate 1909 states "When the arrangement of perforations on the formula-paper indicates that division is to be performed, and the Variables which are to constitute divisor and dividend", so the division instruction must have two operands. In other cases the instruction must have at minimum an opcode, but perhaps nothing more. At minimum there would be DIV and LOG opcodes. Ludgate 1909 goes on: "the formula-paper then allows the dividing cylinder to usurp its functions until that cylinder has caused the machine to complete the division", and states the dividing cylinder has perforations like the formula-paper, i.e. the Dividing cylinder is not the same as the Divider cylinder that has holes of depth proportional to reciprocal values. Ludgate 1909 further says "This system of auxiliary cylinders and tables for special formulae may be indefinitely extended."

Lemma 17: the extended arithmetic instruction format contains at minimum an opcode and up to two source operand addresses.

Do the arithmetic instructions include a result address? Ludgate 1909 stated "Each row of perforations … directs the machine in some definite step … for instance, a complete multiplication, including the selection of the numbers to be multiplied together." There is no mention of result addresses, multiply only includes source operands, not destination operands. Ludgate 1909 also stated: "until the whole product ab is found. The shuttles are afterwards replaced in the shuttle-boxes, the latter being then rotated until the second shuttles of both boxes are opposite to the shuttle-race. These shuttles are brought to the index, as in the former case, and the product of their Variables (21893 x 823) is obtained, which, being added to the previous product (that product having been purposely retained in the mill)". Hence multiply results are retained in the Mill. This is consistent with all typical accumulator architectures. Hence arithmetic instructions do not include a result destination address.

Lemma 18: the arithmetic instruction format does not contain a result destination address.

Section IV.5 explains the necessity for decimal-point alignment, most easily solved via a separate alignment instruction.

Lemma 19: there must be a separate alignment instruction.

Was there a separate store instruction? In an accumulator architecture there is little to be gained by integrating result addresses into arithmetic instructions, and neither Ludgate 1909 nor Engineering [32] mention this, hence there must be a separate instruction (STO) to store the Mill contents.

Lemma 20: there must be a separate store instruction.

How many result operands did the store instruction have? Ludgate 1909 stated "or to two shuttles simultaneously, provided that they do not belong to the same shuttle-box" indicating there must be two result operand addresses. That they must not be at the same address might indicate that that a generic 8-bit address is used for both operands.

Lemma 21: the store instruction format must contain two result operand addresses.

What was the format of the store instructions? From Lemma 20 and Lemma 21, the store instruction format must contain an opcode, and inner and outer shuttle result destination addresses consistent with those for arithmetic instructions. It is possible this should allow separate variants to select Divide, Log, and any other conceivable cylinder outputs, for example for a generic 8-bit address A, when A[7..6]=11 then individual bits of A[5..0] might select outputs of other variants; alternatively extra opcodes could select those.

Lemma 22: the store instruction format contains an opcode, possibly allowing variants.
Lemma 23: the store instruction format contains at minimum a 6-bit inner shuttle destination address.
Lemma 24: the store instruction format contains at minimum a 7-bit outer shuttle destination address.

What was the format of the conditional instructions? Ludgate 1909 stated "It can also "feel" for particular events in the progress of its work—such, for instance, as a change of sign in the value of a function, or its approach to zero or infinity; and it can make any pre-arranged change in its procedure, when any such event occurs." Hence a conditional instruction format must contain an opcode (e.g. BN | BZ | BV) and a target that associates to a program location (but Ludgate did not discuss location, nor form of addressing). McQuillan [38] makes the interesting suggestion of skip markers (as in old printers) for targets.

Lemma 25: the conditional instruction format contains an opcode (e.g. BN | BZ | BV).
Lemma 26: the conditional instruction format contains a target.
Lemma 27: the conditional instruction target associates to a program location.

Very speculatively, the instruction set format could be like that in Table 6 (assuming generic 8-bit shuttle addresses). McQuillan [38] and Riches [42] offer equally speculative but quite different instruction set formats, which highlights the complete absence of any guidance from Ludgate in this regard.

26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC											Shuttle 1 (multiplicand)								Shuttle 2 (multiplier)							
MUL											Shuttle 1 (multiplicand)								Shuttle 2 (multiplier)							
DIV											Shuttle 1 (dividend)								Shuttle 2 (divisor)							
LOG											-								-							
ALN											-								decimal-point position							
STO											Shuttle 1 (destination 1)								Shuttle 2 (destination 2)							
BN											-								target							
BV											-								target							
BZ											-								target							
READ											Shuttle 1 (destination 1)								-							
WRITE											Shuttle 1 (source 1)								-							
STO_DP											-								-							
STO_A											-								-							
HALT											-								-							

Table 6 *Speculative instruction set for Ludgate's analytical machine,*
where for addition (ADD) and subtraction (SUB) the programmer must use MAC or MUL with the multiplier = +1 and -1 respectively
READ, WRITE, STO_DP, STO_A are I/O operations and to support preemption

IV.12. Ludgate's preemption

Does Ludgate's machine support preemption? Ludgate 1909 stated: "Among other powers with which the machine is endowed is that of changing from one formula to another as desired … work of the tabulation can be suspended at any time to allow of the determination by it of one or more results of greater importance or urgency." This is preemption.

Lemma 28: the machine supports preemption.

Has his machine the capabilities required for preemption? If there is only one active set of context objects, then some or all must be physically saved and later restored. Ludgate 1909 stated: "has also the power of recording its results by a system of perforations on a sheet of paper, so that when such a number-paper (as it may be called) is replaced in the machine, the latter can 'read' the numbers indicated thereon, and inscribe them in the shuttles". Hence the machine has the capabilities for preemption.

Lemma 29: the machine has the capabilities required for preemption.

Is this manual or automatic preemption? In its most primitive form, the user must halt execution and run a SUSPEND program that records all the shuttle and Mill contents to a backup number-paper, then the user must mark the formula tape position P with pencil or sticker. To resume, the user must run a RESUME program that reads all the shuttle and Mill contents from the backup number-paper, then the user must place the formula tape at position P and resume execution. Hence at a minimum this is manual preemption just like for early electronic computers.

Lemma 30: at minimum, preemption is manual just like for early electronic computers.

IV.13. Ludgate's input / output

Ludgate 1909 makes it clear his input included two keyboards, and either two perforated paper readers or a single shared reader. For output there are again two perforated paper punches or a single shared punch, and perhaps a printer. With two exceptions, in the absence of new information it does not appear possible to infer any further I/O details, not even whether data I/O interfaced to the datapath and instruction I/O to the controlpath as might be expected. One exception is that Ludgate 1909 stated "Each row of perforations across the formula-paper directs the machine in some definite step in the process of calculation", so each row of the formula tape specifies one instruction. The other exception is that Ludgate 1909 states "The machine prints all results, and, if required, the data, and any noteworthy values which may transpire during the calculation", and then "It also has the power of recording its results by a system of perforations on a sheet of paper"; the fact he discusses one after the other does imply a separate printer of results legible by humans.

IV.14. Ludgate's machine performance

For this initial performance analysis, let us speculatively assume Proposition 4 and Proposition 5 are true, i.e. synchronous ripple carry propagation is overlapped with movements of the "carriage near the index", and is essentially invisible. Ludgate 1909 does not mention the time taken by other mechanical movements, so one can also assume those are excluded from his operational timings. Given these assumptions, the Index and Mill mechanisms would effectively conform to Ludgate's statement that the "carrying is practically in complete mechanical independence of the adding process, so that the two movements proceed simultaneously", and would adhere to his statement that "the sum of *m* numbers of *n* figures would take (9*m* + *n*) units of time", where "*n*" represents a final ripple carry at the end of a sequence of *m* additions. Similarly for multiply, where he states the 20-digit multiply-accumulate would involve *m* = 40 additions and would take ((9 x 40) + 20) = 380 time-units.

Let us now consider the timing when mechanical movements are not ignored.

Ludgate states the multiply instruction takes 10 seconds. Firstly there would be access to shuttles from storage and movement along races. Then there would be outer slide movements with movement of the outer frame up and over the inner frame, and 20 inner frame movements. For each inner frame movement there would also be an inner slide movement, a carriage movement to units (1 time-unit), a units add (9 time-units), a carriage movement to tens (1 time-unit), a tens add (9 time-units), and an inner slide retraction. Finally there would be ripple carry propagation of 20 time-units, and decimal-point alignment (although the latter might be statically set). Restoration of shuttles to storage, and retraction of the outer frame and its slides may all be overlapped with the final carry propagation, and so are ignored below. Therefore the time taken would be:

$T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} + 20*2*T_{inner\_slide} + 20*2*T_{add} + 20*2*T_{carriage} + T_{ripple\_carry} + T_{align}$
$= T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} + 20*2*T_{inner\_slide} + 20*2*9*T_{time\text{-}unit} + 20*2*T_{time\text{-}unit} + 20*T_{time\_unit} + T_{align}$
$= T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} + 40*T_{inner\_slide} + 420*T_{time\_unit} + T_{align}$

Ludgate does not state the duration of a time-unit, only that it is "the period required to move the figure-wheel through $\frac{1}{10}$ revolution". He suggests a "small motor" rotating at 3 revolutions per second, probably an electric (not steam or petrol) motor. Singer produced their first sewing machine with an attached electric motor in 1889. In 1908 when his paper was submitted, Dublin already had 50Hz AC mains electricity (first installed in 1892, substantially upgraded in 1903) [50]. So Ludgate's motor would be 50Hz, synchronous or asynchronous, but likely synchronous as it must rotate at an "approximately uniform rate of rotation". Some sensible possibilities are:

(1) A time-unit is $\frac{1}{10}$ of a motor revolution, i.e. $\frac{1}{30}$ seconds, so then the core arithmetic actions take:

$$(40*T_{inner\_slide} + 420*T_{time\text{-}unit}) > 14 \text{ sec}$$

which is longer than the stated 10 seconds taken by multiply. Therefore a time-unit must be much shorter than $\frac{1}{30}$ seconds.

(2) A time-unit is one cycle of 50Hz, i.e. $\frac{1}{50}$ seconds, so then the core arithmetic actions take:

$$(40*T_{inner\_slide} + 420*T_{time\text{-}unit}) > 8.4 \text{ sec}$$

But 50Hz does not evenly divide to 3 revs/sec. It is approx 50Hz divided by 17 (a prime number, which might have appealed to Ludgate) yielding 2.94 revs/sec, or alternatively divided by 16 yielding 3.125 revs/sec. In addition one iteration of the core arithmetic actions (2 adds + 2 carriage movements) takes:

$$(2*T_{inner\_slide} + 20*T_{time\text{-}unit}) > 0.4 \text{ sec}$$

i.e. iteration proceeds at < 2.5Hz

which does not match well with a 3 revs/sec motor.

(3) One iteration of the core arithmetic actions takes one revolution of the motor at 3 revs/sec:

$$(2*T_{inner\_slide} + 20*T_{time\text{-}unit}) = \tfrac{1}{3} \text{ sec}$$

$$\text{i.e. } 20*T_{time\text{-}unit} = \tfrac{1}{3} - 2*T_{inner\_slide} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{[E]}$$

$$\text{i.e. } T_{time\text{-}unit} < \tfrac{1}{60} \text{ sec} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{[F]}$$

and therefore the core arithmetic actions (20 iterations + ripple carry) take:

$$(20*\tfrac{1}{3} + 20*T_{time\_unit}) = (6.67 + 20*T_{time\_unit}) \text{ sec} < 7 \text{ sec}$$

and hence a useful deduction, that we can use further below, is:

$$T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} < 3 \text{ sec} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{[G]}$$

Option (3) seems the most likely in the absence of other information.

Let us assume option (3) above for an analysis of the addition or subtraction operations, which Ludgate 1909 states take 3 seconds. Ludgate's addition is multiply by +1, and subtraction is multiply by -1 via reversed wheel direction. In both cases partial product Tens will be zero, so no Tens iterations will be done.

When mechanical movements are ignored, "*m*" 1-digit adds will take (9 + *n*) ticks as Ludgate states, so addition or subtraction will take (9 + 20) = 29 time-units.

When mechanical movements are not ignored, firstly there would be access to shuttles from storage and movement along races. Then there would be outer slide and frame movements, but no inner frame iterations. There would only be one inner slide movement, a carriage movement to units (1 time-unit), a units add (9 time-units), a carriage movement to tens (1 time-unit), no tens add, and an outer slide retraction. Finally there would be ripple carry propagation of 20 time-units, and decimal-point alignment (again, the latter might be statically set). Restoration of shuttles to storage, and retraction of the outer frame and its slides may all be overlapped with the final carry propagation, and so are ignored below. Therefore the time taken would be:

$$T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} + 2*T_{inner\_slide} + T_{add} + 2*T_{carriage} + T_{ripple\_carry} + T_{align}$$
$$= T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} + 2*T_{inner\_slide} + 31*T_{time\_unit} + T_{align}$$

Applying the inequaities [F] and [G] above, the time taken for addition would be:

$$< 3 + 31*{}^{1}\!/_{60} \text{ sec, i.e. } < 3.5 \text{ sec}$$

This is a reasonable indication that a time-unit must be $^{1}\!/_{60}$ seconds or less.

Lemma 31: a time-unit must be approximately $^{1}\!/_{60}$ seconds or less.
Lemma 32: $T_{storage\_access} + T_{race\_motion} + T_{outer\_slide} + T_{up\_and\_over} < 3$ sec

But in fact the inner slide is firmly mounted to the lower frame, with far more inertia than outer slides, and hence the time taken to move the inner slide cannot be ignored. From equation [E] above:

$$T_{time\text{-}unit} = {}^{1}\!/_{60} - {}^{1}\!/_{10}*T_{inner\_slide}$$

So the time-unit must be much less than $^{1}\!/_{60}$ seconds. By way of example, for a time-unit of $^{1}\!/_{100}$ seconds, the addition time is:

$$< 3 + 31*{}^{1}\!/_{100} \text{ sec, i.e. } < 3.3 \text{ sec}$$

All the above underrates the time taken shifting the inner frame, exacerbated by its reciprocating nature, and also speculatively assumes Proposition 4 and Proposition 5 are true. Nonetheless, the claims made would indicate the machine was fast for 1909. Ludgate stated that addition and subtraction would each take 3 seconds, multiplication 10 seconds, division 90 seconds, logarithms 120 seconds, and a^n would take 210 seconds..

To compare with a basic (un-optimized) addition-based machine let us ignore mechanical movements and assume it handles the same 20 digits as Ludgate's machine, and adopt his definition of a time-unit as the time to rotate a Mill wheel by a $^{1}\!/_{10}$ turn. For Ludgate's machine to sum m numbers of $n = 20$ figures would take $(9m + n) = (9m + 20)$ time-units, while for a basic addition-based machine it would take $m(9 + n) = 29m$ time-units, i.e. Ludgate's machine would be faster for $m > 1$. For Ludgate's machine to multiply two numbers of 20 figures would take 380 time-units, while a basic addition-based machine would take $N(9 + n) = 29N$ time-units, best when N is the smallest of the two numbers, i.e. Ludgate's machine would be faster for $N > 14$. Since Ludgate's machine multiplies in 10 seconds but divides in 90 seconds, i.e. division is 9 times slower, then by extrapolation division would take approximately 3420 time-units, while for a basic addition-based machine it would take $29N$ time-units, best when N is the largest of the two numbers, i.e. Ludgate's machine would be faster for $N > 118$.

Of course, an addition-based machine could be optimized, for example, Babbage's later analytic engine designs could accumulate a 40-digit number in 3 seconds and multiply 20 digits by 40 digits in 120 seconds [51].

IV.15. Was Ludgate's machine Turing-complete?

An 'analytical machine' is equivalent to a general-purpose computer, and (ignoring physical limits) in theory can be programmed to solve any solvable problem. It is "Turing-complete", or more specifically it is a Universal Turing Machine, i.e. one that can emulate any other Turing Machine, a term invented to reflect Alan Turing's contribution to the theory of computing. The term 'analytical machine' should not be confused with the names of the first two of the only three mechanical designs before the electronic computer era: in 1843 Charles Babbage's "Analytical Engine", then in 1909 Percy Ludgate's very different "Analytical Machine" (the third was in 1936 Zuse's "V1", later renamed "Z1"). From c.1914 electromechanical designs began, and c.1937 electronic designs began [11].

None of these three mechanical computers were "natively" Turing-complete, instead they could be programmed to solve a (substantial) subset of all solvable problems, i.e. they were multi-purpose rather than general-purpose machines. For a discussion of whether and how Ludgate's machine could be rendered Turing-complete, see Appendix IV.

IV.16. Re-imagining Ludgate's machine

Drawing substantially on previous efforts by Riches [42], and McQuillan [37][38], and on the above, it may be possible to at least re-imagine parts of Ludgate's machine.

Objective 2: re-imagine parts of Ludgate's machine.

The aim might be to enable this in several phases:

(1) Datapath Simulation: this is work in progress using Python. The intention is to explore mechanical options, with unit tests for correctness, see Figure 26.

(2) Hybrid CAD Simulation: a 3-D CAD model of a mechanical datapath plus algorithmic controlpath. The intention could be that this should yield online simulations that would create downloadable video files of execution of web user's programs.

(3) Hybrid Engine: a 3D-printed mechanical datapath, with an electronic controlpath processor like a Beaglebone, Arduino or Raspberry Pi, with the controlpath, program and input/output web-enabled and/or on an Android or IOS App.

(4) Full CAD Simulation: a 3-D CAD model of a fully mechanical datapath plus controlpath, taking the results of soft testing controlpath proposals on hybrid implementations. Again the intention could be to yield online simulations and executions.

(5) Pure Engine: fully mechanical engine, manufactured from the 3-D design files either by 3-D printing or using numerically-controlled machine tools. Small programs (e.g. Bernoulli) could be on "real" formula tapes for demonstrations.

The premise might be that mechanical construction of the datapath is easiest, followed by storage, then the controlpath (sequencer), and finally input/output. It is expected it would take several years to fruition, and be very speculative.

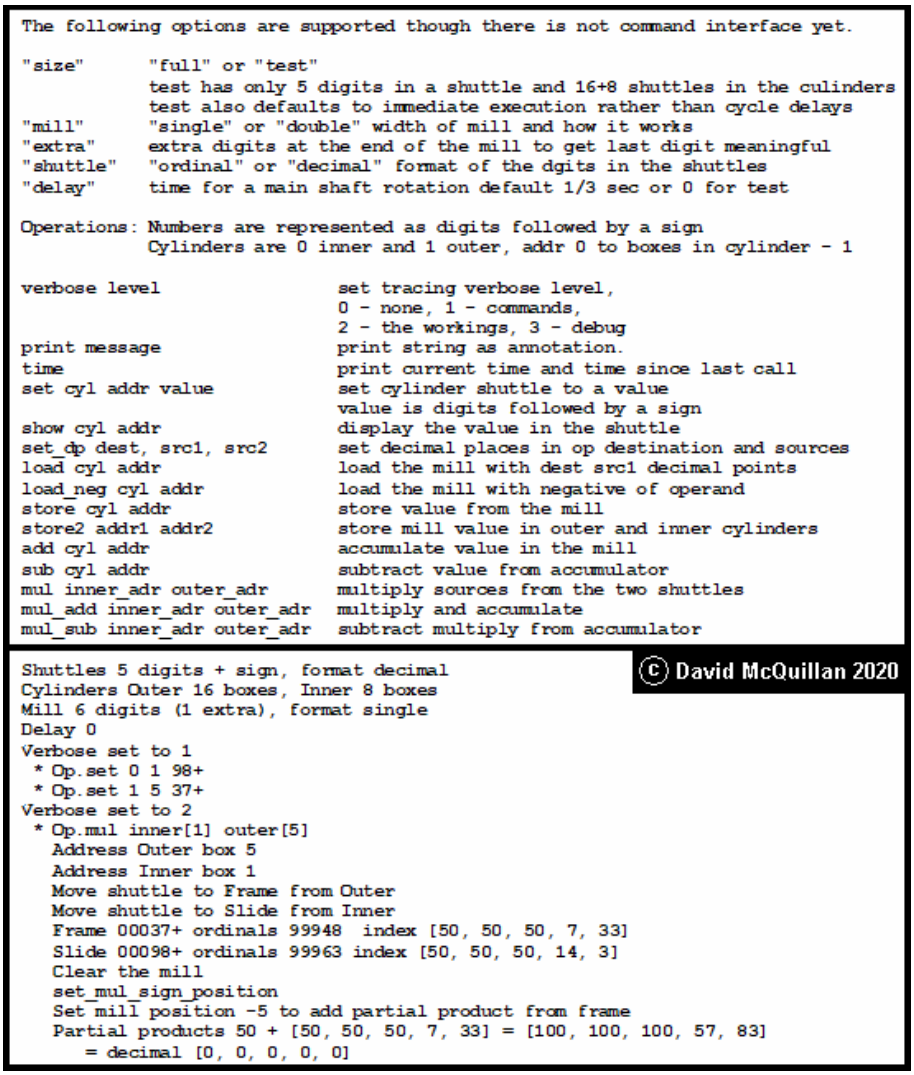


Figure 26 *Help text from an early version of a Python simulator of the datapath of Ludgate's Analytical Machine*
Image reproduced courtesy David McQuillan

V. Concluding Remarks

The Christmas 2019 the discoveries of the articles and especially the diagram in “The English Mechanic”, then the discovery of Engineering [32], have led to intensive efforts to deduce hidden details of Ludgate’s analytical machine. From the latter efforts, thus far a surprising amount of understanding has emerged, construction detail has been uncovered, some useful dimensional, geometric and timing inequalities established, and an effort has been made to codify these new facts. Work has begun on both simulating the machine and “re-imagining” it with modern engineering software.

Acknowledgments

The author would like to thank Professor Brian Randell and David McQuillan (for their engagement, support, and robust debate about the many speculative issues that arise in relation to Ludgate’s machine), Professor John Tucker, University College Swansea, UK (for access to the Riches report on Ludgate), Royal Dublin Society Library (for access to records), Ralf Buelow of Heinz Nixdorf MuseumsForum and Eric Hutton (for discovery of the articles in “The English Mechanic” and permission to publish the images), and Jade Ward of the University of Leeds Library (for discovery of the article of Engineering [32] and permission to publish the images). Finally my thanks for the support of the School of Computer Science and Statistics, Trinity College Dublin, for this work and for The John Gabriel Byrne Computer Science Collection [8].

APPENDIX I

FROM: ENGINEERING, AUG. 20, 1909, PAGES 256-257 [32]

OPTICAL CHARACTER RECOGNITION COURTESY DAVID MCQUILLAN

A PROPOSED ANALYTICAL MACHINE

By name, at any rate, Babbage's famous analytical engine is known to all. It was intended to be a machine for the arithmetical solution of all problems in mathematical physics. Such solutions are generally, perhaps always, feasible, but in most cases when the computations have to be effected by direct human agency, they are so extremely tedious as to be practically, if not theoretically, impossible. Every operation in arithmetic can be reduced to addition, subtraction, multiplication, and division, and, indeed, the two latter operations can be regarded as mere extensions of the two former. The analytical engine was a machine by which these four operations could be performed in any desired sequence ; moreover, a number of partial operations could be combined, and the final results automatically tabulated for any required values of the variable. As is well known, though many years' labour was spent on the machine, it was never even partially completed, Mr. Babbage's scheme being far too ambitious for a first effort. He wished, indeed, to tabulate values to 50 significant figures, thus enormously complicating the mechanism and augmenting the cost of the experiment. In a paper read not long ago before the Royal Dublin Society, Mr. Percy E. Ludgate has revived again the idea of constructing such a machine: As proposed by him, the machine differs from that of Babbage in some fundamental details, though, as in its predecessor, Jacquard cards will be used to control the sequence of operations. Thus if, for instance, a number of values of the series

$$y = x - \frac{x^2}{2^2} + \frac{x^3}{2^2.3^2} - \frac{x^4}{2^2.3^2.4^2} + \&c.$$

were required, the appropriate card would be placed in the machine, which would then, for different values of *x*, calculate each term of the series, add all the positive terms together, subtract from this sum all the negative, and print the result. For a different series a different card would be used.

In Babbage's engine it was proposed to effect multiplication by successive additions, and divisions by successive subtractions, just as is now done in the case of the ordinary arithmometer. Mr. Ludgate, in his engine, proposes to effect these operations on entirely different principles. Multiplication is effected by a series of index numbers analogous to logarithms;

The arrangement is shown diagrammatically in Fig. 1 *[see Figure 29]*. Here the number 813,200 is to be multiplied by 9247. The arrow under 8 represents a slide, which to denote 8 is set at $\frac{3}{8}$ in. above the zero or starting line. The slide representing 1 lies on the starting line, whilst that representing 3 stands $\frac{7}{8}$ in. above this line, and that corresponding to the number two $\frac{1}{8}$ in. above. On the other hand, the slides representing zero are set 50 eighths above the starting line. The number of units above the starting line corresponding to each digit of the multiplicand are known as index numbers, and a complete table of these has been drawn up by Mr. Ludgate. All the slides aforementioned are mounted in a frame, and to multiply by 9, this frame is moved up over another frame divided with another series of index numbers. Thus, as shown, the distance between the lower frame and the starting line is such that the top of this lower frame lies on the index number corresponding to 9; that is, 14 eighths below the starting line. The lower end of the No. 8 slide, represented by the black circle, rests then, it will be seen, on a line marked "72," which is the product of 8 and 9. The digits 7 and 2 appear accordingly on the register below. Similarly, the tail of the No. 1 slide rests on the No. 9 line, that of the No. 3 slide on the No. 27 line, and that of the No. 2 slide on 18, corresponding to the partial products 9 × 1; 9 × 3 ; and 9 × 2. The tails of the zero slides rest on no line in the lower frame, and hence zero is registered for these. All these partial products are registered in the mill below, as indicated. In a final operation these partial products are added together as indicated, giving 7,318,800. If now the frame is moved to the index number below the starting line marked "2," it will be found, on trial with a piece of tracing paper, that the tail of the No. 8 slide rests now on the line marked "16" — i.e., 8 × 2. That of the No. 1 slide on the index-line marked "2," that of slide 3 on the line marked "15," and that of the No. 2 slide on the line marked "10." These partial products will then appear on the mill and be added together, giving the result of the multiplication of 813,200 by 2. The process is repeated for the remaining figures of the multiplier, and the whole added together so as to give the product of 813,200 × 9247. Mr. Ludgate proposes to give such products to twenty significant figures, the time required being, he states, about 10 seconds.

To divide one number by another he proceeds in a different fashion. He notes that the expression $\frac{p}{q}$, where p and q are any two numbers, can always be expressed in the form—

$$\frac{p}{q} = \frac{\mathrm{A}\, p}{1+x},$$

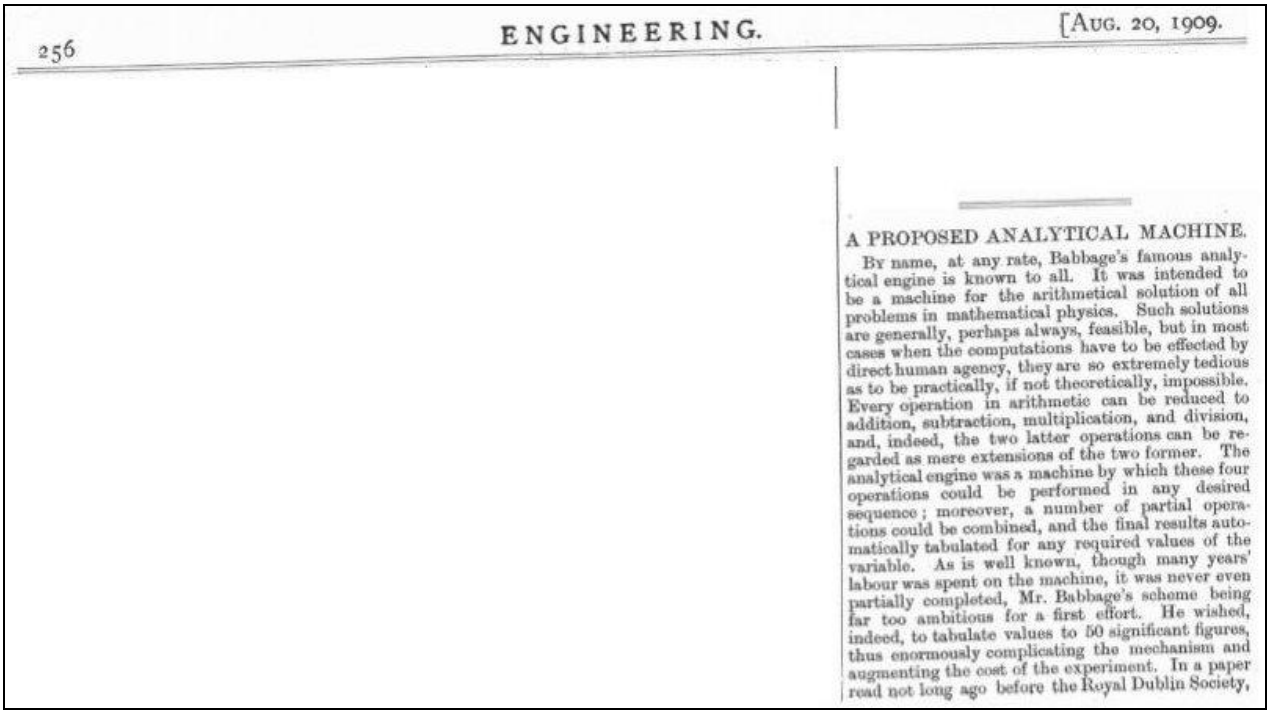
where x is a small quantity, and A is the reciprocal of some number between 100 and 999.

The above expression can also obviously be written

$$\frac{p}{q} = \mathrm{A}\, p\,(1 - x + x^2 - x^3 + x^4 - x^5, + \&c.),$$

the series being very rapidly convergent, the first eleven terms give the value of $\frac{1}{1+x}$ correct to at least twenty figures.

He proposes to perform division, therefore, by making the machine first calculate the value of this series, after which it will multiply $A\ p$ by the value thus found. As a maximum, he considers that this operation giving the result correct to twenty figures might require $1\frac{1}{2}$ minutes.



256 ENGINEERING. [AUG. 20, 1909.

A PROPOSED ANALYTICAL MACHINE.

By name, at any rate, Babbage's famous analytical engine is known to all. It was intended to be a machine for the arithmetical solution of all problems in mathematical physics. Such solutions are generally, perhaps always, feasible, but in most cases when the computations have to be effected by direct human agency, they are so extremely tedious as to be practically, if not theoretically, impossible. Every operation in arithmetic can be reduced to addition, subtraction, multiplication, and division, and, indeed, the two latter operations can be regarded as mere extensions of the two former. The analytical engine was a machine by which these four operations could be performed in any desired sequence; moreover, a number of partial operations could be combined, and the final results automatically tabulated for any required values of the variable. As is well known, though many years' labour was spent on the machine, it was never even partially completed, Mr. Babbage's scheme being far too ambitious for a first effort. He wished, indeed, to tabulate values to 50 significant figures, thus enormously complicating the mechanism and augmenting the cost of the experiment. In a paper read not long ago before the Royal Dublin Society,

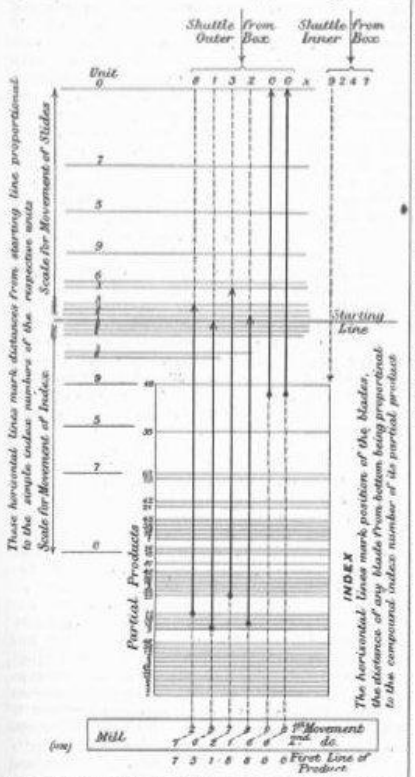
Figure 27 *ENGINEERING, AUG. 20, 1909, PAGE 256*
Image reproduced courtesy Jade Ward, University of Leeds Library

Mr. Percy E. Ludgate has revived again the idea of constructing such a machine. As proposed by him, the machine differs from that of Babbage in some fundamental details, though, as in its predecessor, Jacquard cards will be used to control the sequence of operations. Thus if, for instance, a number of values of the series

$$y = x - \frac{x^2}{2^2} + \frac{x^3}{2^2.3^2} - \frac{x^4}{2^2.3^2.4^2} + \&c.$$

were required, the appropriate card would be placed in the machine, which would then, for different values of x, calculate each term of the series, add all the positive terms together, subtract from this sum all the negative, and print the result. For a different series a different card would be used.

In Babbage's engine it was proposed to effect multiplication by successive additions, and divisions by successive subtractions, just as is now done in the case of the ordinary arithmometer. Mr. Ludgate, in his engine, proposes to effect these operations on entirely different principles. Multiplication is effected by a series of index numbers analogous to logarithms.



The arrangement is shown diagrammatically in Fig. 1. Here the number 813,200 is to be multiplied by 9247. The arrow under 8 represents a slide, which to denote 8 is set at $\frac{8}{8}$ in. above the zero or starting line. The slide representing 1 lies on the starting line, whilst that representing 3 stands $\frac{3}{8}$ in. above this line, and that corresponding to the number two $\frac{2}{8}$ in. above. On the other hand, the slides representing zero are set 50 eighths above the starting line. The number of units above the starting line corresponding to each digit of the multiplicand are known as index numbers, and a complete table of these has been drawn up by Mr. Ludgate. All the slides aforementioned are mounted in a frame, and to multiply by 9, this frame is moved up over another frame divided with another series of index numbers. Thus, as shown, the distance between the lower frame and the starting line is such that the top of this lower frame lies on the index number corresponding to 9; that is, 14 eighths below the starting line. The lower end of the No. 8 slide, represented by the black circle, rests then, it will be seen, on a line marked "72," which is the product of 8 and 9. The digits 7 and 2 appear accordingly on the register below. Similarly, the tail of the No. 1 slide rests on the No. 9 line, that of the No. 3 slide on the No. 27 line, and that of the No. 2 slide on 18, corresponding to the partial products 9×1; 9×3; and 9×2. The tails of the zero slides rest on no line in the lower frame, and hence zero is registered for these. All these partial products are registered in the mill below, as indicated. In a final operation these partial products are added together as indicated, giving 7,318,800. If now the frame is moved to the index number below the starting line marked "2," it will be found, on trial with a piece of tracing paper, that the tail of the No. 8 slide rests now on the line marked "16"—*i.e.*, 8×2. That of the No. 1 slide on the index-line marked "2," that of slide 3 on the line marked "15," and that of the No. 2 slide on the line marked "10." These partial products will then appear on the mill and be added together, giving the result of the multiplication of 813,200 by 2. The process is repeated for the remaining figures of the multiplier, and the whole added together so as to give the product of $813,200 \times 9247$. Mr. Ludgate proposes to give such products to twenty significant figures, the time required being, he states, about 10 seconds.

To divide one number by another he proceeds in a different fashion. He notes that the expression $\frac{p}{q}$, where p and q are any two numbers, can always be expressed in the form—

$$\frac{p}{q} = \frac{A\,p}{1+x},$$

where x is a small quantity, and A is the reciprocal of some number between 100 and 999.

The above expression can also obviously be written

$$\frac{p}{q} = A\,p\,(1 - x + x^2 - x^3 + x^4 - x^5, + \&c.),$$

the series being very rapidly convergent, the first eleven terms give the value of $\frac{1}{1+x}$ correct to at least twenty figures.

He proposes to perform division, therefore, by making the machine first calculate the value of this series, after which it will multiply A p by the value thus found. As a maximum, he considers that this operation giving the result correct to twenty figures might require $1\frac{1}{2}$ minutes.

Figure 28 *ENGINEERING, AUG. 20, 1909, PAGE 257*
Image reproduced courtesy Jade Ward, University of Leeds Library

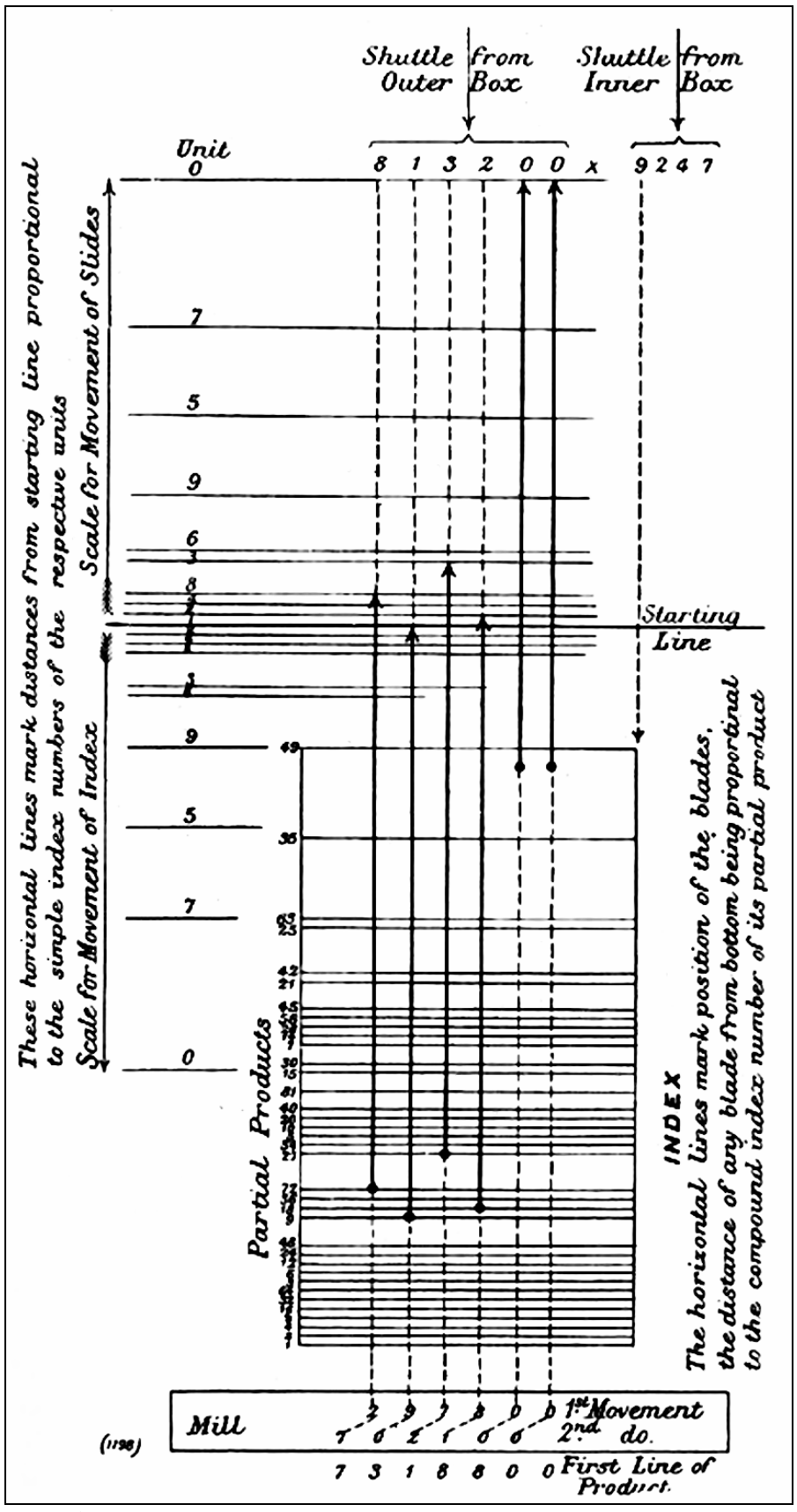


Figure 29 *Diagram from "Engineering", Aug. 20, 1909, Pages 256-257 [32]*
Image reproduced courtesy Jade Ward of University of Leeds Library

APPENDIX II

FROM: ENGLISH MECHANIC AND WORLD OF SCIENCE, NO. 2302, MAY 7, 1909, PAGE 322 [30]

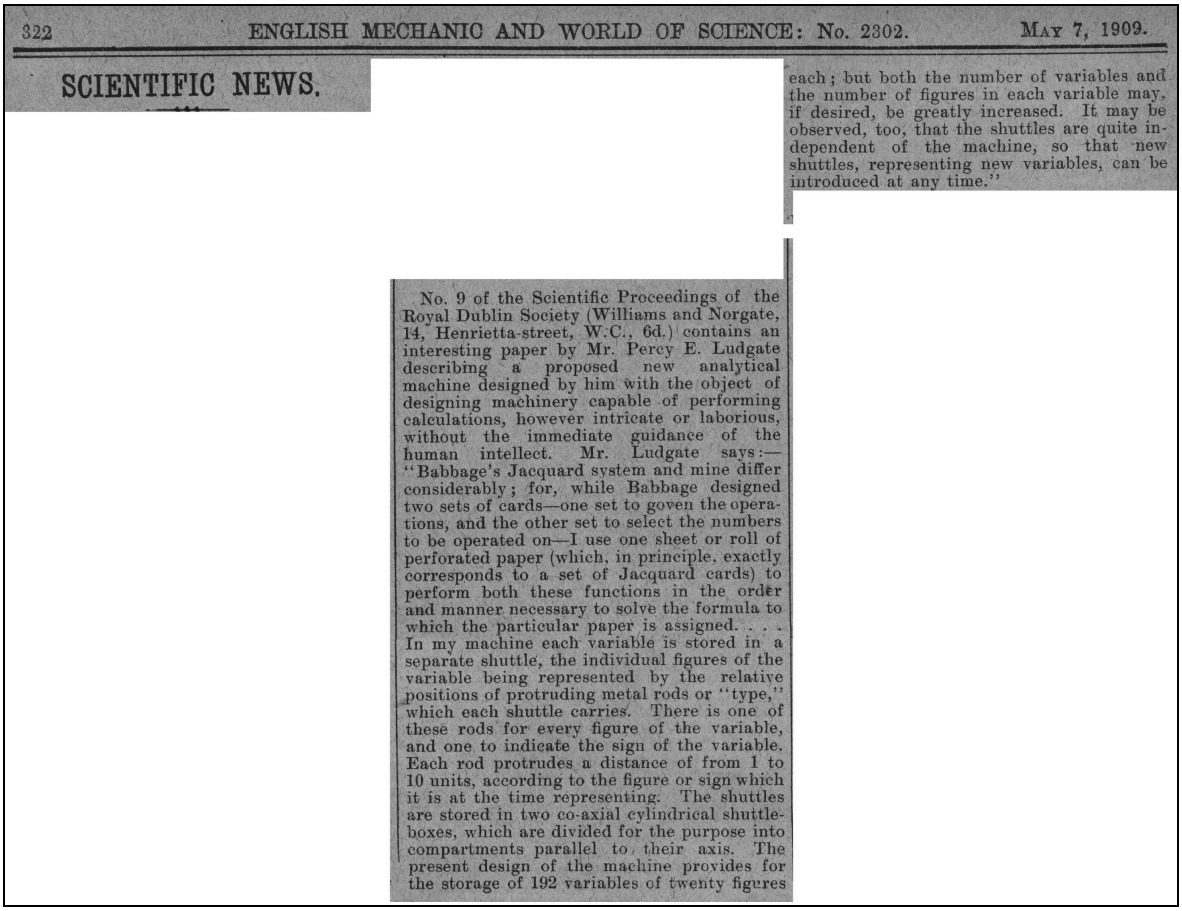
OPTICAL CHARACTER RECOGNITION COURTESY DAVID MCQUILLAN

SCIENTIFIC NEWS

No. 9 of the Scientific Proceedings of the Royal Dublin Society (Williams and Negate, 14, Henrietta-street, WIC., 6d.). contains an interesting paper by Mr. Percy E. Ludgate describing a proposed new analytical machine designed by him with the object of designing machinery capable of performing calculations, however intricate or laborious, without the immediate guidance of the human intellect. Mr. Ludgate says:—

"Babbage's Jacquard system and mine differ considerably ; for, while Babbage designed two sets of cards—one set to govern the operations, and the other set to select the numbers to be operated on—I use one sheet or roll of perforated paper (which, in principle, exactly corresponds to a set of Jacquard cards) to perform both these functions in the order and manner necessary to solve the formula to which the particular paper is assigned. . . .

In my machine each variable is stored in a separate shuttle, the individual figures of the variable being represented by the relative positions of protruding metal rods or "type," which each shuttle carries. There is one of these rods for every figure of the variable, and one to indicate the sign of the variable. Each rod protrudes a distance of from 1 to 10 units, according to the figure or sign which it is at the time representing. The shuttles are stored in two co-axial cylindrical shuttle-boxes, which are divided for the purpose into compartments parallel to their axis. The present design of the machine provides for the storage of 192 variables of twenty figures each ; but both the number of variables and the number of figures in each variable may, if desired, be greatly increased. It may be observed, too, that the shuttles are quite independent of the machine, so that new shuttles, representing new variables, can be introduced at any time."



322 ENGLISH MECHANIC AND WORLD OF SCIENCE: No. 2302. MAY 7, 1909.

SCIENTIFIC NEWS.

each; but both the number of variables and the number of figures in each variable may, if desired, be greatly increased. It may be observed, too, that the shuttles are quite independent of the machine, so that new shuttles, representing new variables, can be introduced at any time."

No. 9 of the Scientific Proceedings of the Royal Dublin Society (Williams and Norgate, 14, Henrietta-street, W.C., 6d.) contains an interesting paper by Mr. Percy E. Ludgate describing a proposed new analytical machine designed by him with the object of designing machinery capable of performing calculations, however intricate or laborious, without the immediate guidance of the human intellect. Mr. Ludgate says:—"Babbage's Jacquard system and mine differ considerably; for, while Babbage designed two sets of cards—one set to goven the operations, and the other set to select the numbers to be operated on—I use one sheet or roll of perforated paper (which, in principle, exactly corresponds to a set of Jacquard cards) to perform both these functions in the order and manner necessary to solve the formula to which the particular paper is assigned. . . . In my machine each variable is stored in a separate shuttle, the individual figures of the variable being represented by the relative positions of protruding metal rods or "type," which each shuttle carries. There is one of these rods for every figure of the variable, and one to indicate the sign of the variable. Each rod protrudes a distance of from 1 to 10 units, according to the figure or sign which it is at the time representing. The shuttles are stored in two co-axial cylindrical shuttle-boxes, which are divided for the purpose into compartments parallel to their axis. The present design of the machine provides for the storage of 192 variables of twenty figures

Figure 30 *ENGLISH MECHANIC AND WORLD OF SCIENCE, NO. 2302, MAY 7, 1909, PAGE 322*
Image reproduced courtesy Ralf Buelow of Heinz Nixdorf MuseumsForum, and Eric Hutton

APPENDIX III

FROM: ENGLISH MECHANIC AND WORLD OF SCIENCE, NO. 2319, SEPT 3, 1909, PAGE 111 [31]

OPTICAL CHARACTER RECOGNITION COURTESY DAVID MCQUILLAN

A PROPOSED ANALYTICAL MACHINE

Mr. Percy E. Ludgate has revived again the idea of constructing such a machine. As proposed by him, the machine differs from that of Babbage in some fundamental details, though, as in its predecessor, Jacquard cards will be used to control the sequence of operations. Thus if, for instance, a number of values of the series

$$y = x - \frac{x^2}{2^2} + \frac{x^3}{2^2.3^2} - \frac{x^4}{2^2.3^2.4^2} + \&c.$$

were required, the appropriate card would be placed in the machine, which would then, for different values of x, calculate each term of the series, add all the positive terms together, subtract from this sum all the negative, and print the result. For a different series a different card would be used.

In Babbage's engine it was proposed to effect multiplication by successive additions, and divisions by successive subtractions, just as is now done in the case of the ordinary arithmometer. Mr. Ludgate, in his engine, proposes to effect these operations on entirely different principles. Multiplication is affected by a series of index numbers analogous to logarithms. Say the number 813,200 is to be multiplied by 9,247. A slide to denote 8 is set at $\frac{3}{8}$ in. above the zero- or starting-line. The slide representing 1 lies on the starting-line, whilst that representing 3 stands $\frac{7}{8}$ in. above this line, and that corresponding to the number two $\frac{1}{8}$ in. above. On the other hand, the slides representing zero are set 50 eighths above the starting-line. The number of units above the starting-line corresponding to each digit of the multiplicand are known as index numbers, and a complete table of these has been drawn up by Mr. Ludgate. All the slides aforementioned are mounted in a frame, and to multiply by 9, this frame is moved up over another frame divided with another series of index numbers. Thus, as shown, the distance between the lower frame and the starting-line is such that the top of this lower frame lies on the index number corresponding to 9; that is, 14 eighths below the starting-line. The lower end of the No. 8 slide, represented by the black circle rests, then, it will be seen, on a line marked "72," which is the product of 8 and 9. The digits 7 and 2 appear accordingly on the register below. Similarly, the tail of the No. 1 slide rests on the No. 9 line, that of the No. 3 slide on the No. 27 line, and that of the No. 2 slide on 18, corresponding to the partial products 9 × 1; 9 × 3; and 9 × 2. The tails of the zero slides rest on no line in the lower frame, and hence zero is registered for these. All these partial products are registered in the mill below, as indicated. In a final operation, these partial products are added together as indicated, giving 7,318,800. If now the frame is moved to the index number below the starting-line marked "2," it will be found, on trial with a piece of tracing-paper, that the tail of the No. 8 slide rests now on the line marked "16 "—i.e., 8 × 2. That of the No. 1 slide on the index-line marked "2," that of slide 3 on the line marked "15," and that of the No. 2 slide on the line marked "10." These partial products will then appear on the mill, and be added together, giving the result of the multiplication of 813,200 by 2. The process is repeated for the remaining figures of the multiplier, and the whole added together so as to give the product of 813,200 × 9,247. Mr. Ludgate proposes to give such products to twenty significant figures, the time required being, he states, about 10 seconds.

To divide one number by another, he proceeds in a different fashion. He notes that the expression $\frac{p}{q}$, where p and q are any two numbers, can always be expressed in the form—

$$\frac{p}{q} = \frac{\mathrm{A}\, p}{1+x},$$

where x is a small quantity, and A is the reciprocal of some number between 100 and 999.

The above expression can also obviously be written

$$\frac{p}{q} = \mathrm{A}\, p\,(1 - x + x^2 - x^3 + x^4 - x^5, + \&c.),$$

the series being very rapidly convergent, the first eleven terms give the value of $\frac{1}{1+x}$ correct to at least twenty figures.

He proposes to perform division, therefore, by making the machine first calculate the value of this series, after which it will multiply $A\ p$ by the value thus found. As a maximum, he considers that this operation, giving the result correct to twenty figures, might require $1\frac{1}{2}$ minutes.—"Engineering."

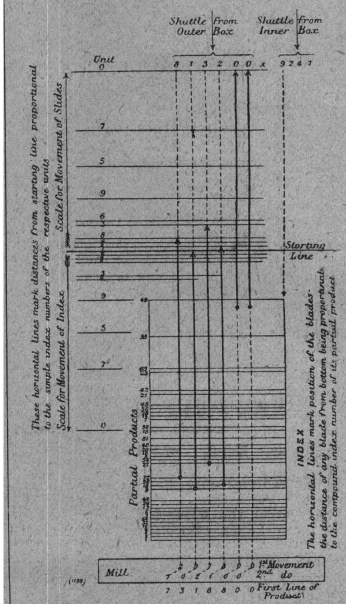
A PROPOSED ANALYTICAL MACHINE.

Mr. Percy E. Ludgate has revived again the idea of constructing such a machine. As proposed by him, the machine differs from that of Babbage in some fundamental details, though, as in its predecessor, Jacquard cards will be used to control the sequence of operations. Thus if, for instance, a number of values of the series

$$y = x - \frac{x^2}{2^2} + \frac{x^3}{2^2 \cdot 3^2} - \frac{x^4}{2 \cdot 3^2 \cdot 4^2} + \&c.$$

were required, the appropriate card would be placed in the machine, which would then, for different values of x, calculate each term of the series, add all the positive terms together, subtract from this sum all the negative, and print the result. For a different series a different card would be used.

In Babbage's engine it was proposed to effect multiplication by successive additions, and divisions by successive subtractions, just as is now done in the case of the ordinary arithmometer. Mr. Ludgate, in his engine, proposes to effect these operations on entirely different principles. Multiplication is effected by a series of index numbers analogous to logarithms. Say the number 813,200 is to be multiplied by 9,247. A slide to denote 8 is set at $\frac{8}{10}$in. above the zero- or starting-line. The slide representing 1 lies on the starting-line, whilst that representing 3 stands $\frac{3}{10}$in. above this line, and that corresponding to the number two $\frac{2}{10}$in. above. On the other hand, the slides representing zero are set 50 eighths above the starting-line. The number of units above the starting-line corresponding to each digit of the multiplicand are known as index numbers, and a complete table of these has been drawn up by Mr. Ludgate. All the slides aforementioned are mounted in a frame, and to multiply by 9, this frame is moved up over another frame divided with another series of index numbers. Thus, as shown, the distance between the lower frame and the starting-line is such that the top of this lower frame lies on the index number corresponding to 9; that is, 14 eighths below the starting-line. The lower end of the No. 8 slide, represented by the black circle rests, then, it will be seen, *on a line* marked "72," which is the product of 8 and 9. The digits 7 and 2 appear accordingly on the register below. Similarly, the tail of the No. 1 slide rests on the No. 9 line, that of the No. 3 slide on the No. 27 line, and that of the No. 2 slide on 18, corresponding to the partial products 9×1; 9×3; and 9×2. The tails of the zero slides rest on no line in the lower frame, and hence zero is registered for these. All these partial products are registered in the mill below, as indicated. In a final operation, these



partial products are added together as indicated, giving 7,318,800. If now the frame is moved to the index number below the starting-line marked "2," it will be found, on trial with a piece of tracing-paper, that the tail of the No. 8 slide rests now on the line marked "16"—i.e., 8×2. That of the No. 1 slide on the index-line marked "2," that of slide 3 on the line marked "15," and that of the No. 2 slide on the line marked "10." These partial products will then appear on the mill, and be added together, giving the result of the multiplication of 813,200 by 2. The process is repeated for the remaining figures of the multiplier, and the whole added together so as to give the product of $813,200 \times 9,247$. Mr. Ludgate proposes to give such products to twenty significant figures, the time required being, he states, about 10 seconds.

To divide one number by another, he proceeds in a different fashion. He notes that the expression $\frac{p}{q}$, where p and q are any two numbers, can always be expressed in the form—

$$\frac{p}{q} = \frac{A\,p}{1+x}$$

where x is a small quantity, and A is the reciprocal of some number between 100 and 999.

The above expression can also obviously be written

$$\frac{p}{q} = A\,p\,(1 - x + x^2 - x^3 + x^4 - x^5, + \&c.),$$

the series being very rapidly convergent, the first eleven terms give the value of $\frac{1}{1+x}$ correct to at least twenty figures.

He proposes to perform division, therefore, by making the machine first calculate the value of this series, after which it will multiply A p by the value thus found. As a maximum, he considers that this operation, giving the result correct to twenty figures, might require 1½ minutes.—"Engineering."

Figure 31 *ENGLISH MECHANIC AND WORLD OF SCIENCE, NO. 2319, SEPT 3, 1909, PAGE 111*
Image reproduced courtesy Ralf Buelow of Heinz Nixdorf MuseumsForum, and Eric Hutton

APPENDIX IV
WAS LUDGATE'S MACHINE A GENERAL-PURPOSE COMPUTER?

The term *general-purpose computer* is somewhat controversial. It has been equated to the von Neumann architecture, but this is questionable, although the converse is more assured, that machines using the von Neumann architecture are general-purpose computers. One could expect that a general-purpose computer could execute any algorithm. But an algorithm is a special-purpose Turing Machine, therefore one might conjecture that a general-purpose computer (itself a Turing Machine) should if truly 'general-purpose' be able to execute any other Turing Machine. A Universal Turing Machine can emulate any other Turing Machine, therefore a reasonable pragmatic definition might be that a general-purpose computer is a physical realization of a Universal Turing Machine.

But a Universal Turing Machine stores both code and data on its (infinite) tape, i.e. code and data share the same address space and access mechanisms. This facilitates self-modifying code, and more importantly access to code or data as a result of prior calculation, i.e. indirect access to code or data, which is crucial for many algorithms. Therefore it is furthermore reasonable to conclude that a general-purpose computer must at minimum facilitate indirect access to code or data.

Raul Rojas of the Freie Universitat Berlin, Germany, has explored the programming architecture of Babbage's engine, and states that Babbage's engine did not have the ability to indirectly access memory [55]. But Rojas also speculates how a sequence of numbers could have been added using an addition loop, e.g. indirectly with one single operation card and combinatorial cards.

Similarly from Ludgate 1909 it can be stated that his Analytical Machine did not have the explicit ability to indirectly access shuttles, since there is no explicit discussion of any support for selection of a shuttle based on the result of a prior calculation. But Ludgate does hint (see Section IV.9 on Ludgate's Storage above) that for complex formulae it may have been possible to select a shuttle other than by a literal operand, i.e. indirectly.

Nevertheless neither of Babbage's or Ludgate's machines explicitly facilitate indirect access to code or data, and hence could not be said to natively be general-purpose machines. In this Appendix we explore to what extent Ludgate's Analytical Machine could have emulated a general-purpose computer, and represent that with some speculative "Ludgate assembly language" code.

Raul Rojas also showed how Zuse's c.1941 "Z3" could be rendered Turing-complete [56][57][58][59]. The Z3 is not considered to be natively Turing-complete because it didn't have conditional branching (it was designed to solve aircraft flow equations). However, Rojas shows the Z3 could be made Turing-complete by forming a loop of the film used in the Z3 to hold programs, plus a lot of mental gymnastics. Perhaps the Z1 could also be made Turing-complete in the same way (the two designs have the same origins), but this is not yet known.

For Ludgate's machine a loop could also be made, in this case of the formula paper used in his machine to hold programs. But instead of mental gymnastics, the loop could contain an emulator program that continuously cycled until halted. For simplicity the emulator could simply emulate the Ludgate machine itself, but with two enhancements: (a) to execute from programs in shuttles, and (b) to execute one extra instruction to do dynamic shuttle selection (indirect addressing). That would allow straightforward emulation of a von Neumann machine.

Rojas' paper [58] also states that recursion was implemented in the Z3 by unrolling code to refer to absolute addresses. This is useful in that it shows a precedent, even if Ludgate was unaware of it. Here we suggest emulating indirect access to Ludgate's 192 storage shuttles by in effect unrolling code that would loop until it found the instruction to access the specific desired shuttle. Ludgate's formulae paper is unlimited ('infinite' a la Turing?) so the loop unrolling for 192 shuttles (about 1.5k rows of instructions on the formulae paper) is of minimal consequence (perhaps successive approximation could be used to reduce the time taken to get to the desired instruction).

```
#
# Stored-program emulator for Ludgate's Analytical Machine.
# IN DEVELOPMENT
#
```

APPENDIX V

OCCAM CODE FOR LUDGATE'S INDEX AND MILL

The behaviour of Ludgate's Index and Mill is relatively concisely represented above as an algorithmic state machine (ASM) using a pseudo-OCCAM. This Appendix expands on how these mechanisms might actually have supported that by transforming the pseudo-OCCAM into a very much more lengthy form of real OCCAM code.

```
#
# IN DEVELOPMENT
#
```

REFERENCES

[1] Percy E.Ludgate, On a Proposed Analytical Machine, Scientific Proceedings of the Royal Dublin Society, Vol.12, No.9, pp.77–91, 28-Apr-1909.
[2] C.V.Boys, A new analytical engine, Nature, Vol.81, pp.14-15, Jul-1909, see elsewhere in the Literature category of this catalog.
[3] Percy E.Ludgate, Automatic Calculating Machines, In "Handbook of the Napier Tercentenary Celebration or modern instruments and methods of calculation", Ed: E.M.Horsburgh, 1914
[4] Lovelace, A., Sketch of the Analytical Engine invented by Charles Babbage, Esq. By L.F. Menabrea, of Turin, Officer of the Military Engineers, with notes by the Translator, Scientific Memoirs, Ed.R.Taylor, Vol.3, pp.666–731, 1843.
[5] Baxandall, D., Calculating machines and instruments: Catalogue of the Collection in the Science Museum, Science Museum, London, 1926.
[6] Brian Randell, "Ludgate's analytical machine of 1909", The Computer Journal, Vol.14, No.3, pp.317-326, 1971.
[7] Brian Randell, "From analytical engine to electronic digital computer: The contributions of Ludgate, Torres, and Bush", Annals of the History of Computing, Vol.4, No.4, IEEE, Oct. 1982.
[8] Trinity College Dublin, The John Gabriel Byrne Computer Science Collection. Available: https://scss.tcd.ie/SCSSTreasuresCatalog/
[9] Trinity College Dublin, Percy E. Ludgate Prize in Computer Science. Available: https://scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/TCD-SCSS-X.20121208.002.pdf
[10] Brian Coghlan, Brian Randell, Paul Hockie, Trish Gonzalez, David McQuillan, Reddy O'Regan, 'Investigating the work and life of Percy Ludgate', *IEEE Annals of the History of Computing*, Vol.43, No.1 pp.19-37, 2021.. Available in file: "IEEE-Annals-FINAL-asPublished-Ludgate-20210307-0849" at https:/scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[11] Brian Randell (Ed.), The Origins of Digital Computers: Selected Papers (3rd ed.), pp.580, Springer-Verlag, Heidelberg, 1982.
[12] Sydney Padua, The Thrilling Adventures of Lovelace and Babbage, pp.320, Penguin, 2015.
[13] Gottfried Wilhelm Liebniz, Stepped Reckoner, c.1671, Available: https://en.wikipedia.org/wiki/Stepped_reckoner
[14] Joseph Marie Jacquard, Jacquard Loom, c.1804. Available at: https://en.wikipedia.org/wiki/Joseph_Marie_Jacquard
[15] Plan28, Building Charles Babbage's Analytical Engine. Available at: http://www.plan28.org/
[16] Royal Dublin Society, Minutes of the RDS Scientific Committee for 1908-1909, RDS Library, Dublin.
[17] B. Coghlan, B. Randell, P. Hockie, T. Gonzalez, D. McQuillan, R. O'Regan, "Percy Ludgate (1883-1922), Ireland's First Computer Designer ", *Proceedings of the Royal Irish Academy: Archaeology, Culture, History, Literature*, Volume 121C, 2021.
[18] Adrian Johnstone, Notions and notation: Babbage's language of thought, Royal Holloway. Available at: https://babbage.csle.cs.rhul.ac.uk/. Also at: https://pure.royalholloway.ac.uk/portal/en/projects/notions-and-notation-babbages-language-of-thought(349c2bbb-9db0-4184-bbce-9b489935520c).html
[19] Wikipedia, Multiply–accumulate operation. Available at: https://en.wikipedia.org/wiki/Multiply-accumulate_operation
[20] Fujitsu, FUJITSU Supercomputer PRIMEHPC FX1000: An HPC System Opening Up an AI and Exascale Era, Available at: https://www.fujitsu.com/downloads/SUPER/primehpc-fx1000-hard-en.pdf
[21] Wikipedia, The Millionaire Calculator. Available at: https://en.wikipedia.org/wiki/The_Millionaire_Calculator
[22] William Rae MacDonald, John Napier, Dictionary of National Biography, Volume 40, 1885-1900.
[23] Peter M.Hopp, Slide Rules: Their History, Models and Makers, ISBN 1879335867, Astragal Press, 1999.
[24] George Boole, The Laws of Thought, 1854. Available at: https://en.wikipedia.org/wiki/George_Boole
[25] Alan Turing, On Computable Numbers, with an Application to the Entscheidungsproblem, 1936. Available at: https://en.wikipedia.org/wiki/Alan_Turing
[26] Claude Shannon, A Symbolic Analysis of Relay and Switching Circuits, 1937, 1938. Available at: https://en.wikipedia.org/wiki/Claude_Shannon
[27] Claude Shannon, A Mathematical Theory of Communication, 1948. Available at: https://en.wikipedia.org/wiki/Claude_Shannon
[28] Eric Hutton, Available: http://www.englishmechanic.com
[29] Ralf Buelow, Percy Ludgate, der unbekannte Computerpionier [Percy Ludgate, the Unknown Computer Pioneer], Heinz Nixdorf MuseumsForum, 17-Jan-2020. Available in German at: https://blog.hnf.de/percy-ludgate-der-unbekannte-computerpionier/ also and including Google translation to English, courtesy Brian Randell at: https:/scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[30] Scientific News: No. 9 of the Scientific Proceedings of the Royal Dublin Society, English Mechanic and World of Science, No. 2302, p. 322, 7-May-1909, discovered by Ralf Buelow, image provided by Eric Hutton, 19-Dec-2019.
[31] Engineering (attribution): A Proposed Analytical Machine, English Mechanic and World of Science, No. 2319, p. 111, Vol. 90 (1909/10), 3-Sep-1909, discovered by Ralf Buelow, image provided by Eric Hutton, 19-Dec-2019.
[32] Engineering: A Proposed Analytical Machine, pp.256-257, 20-Aug-1909, discovered by Jade Ward, Univ.Leeds Library, 14-Jan-2020, OCR by David McQuillan, 3-Feb-2020.
[33] Percy Ludgate, NAI/CS/PO/TR Will of Percy Edwin Ludgate, 26-Jun-1917, National Archives of Ireland.
[34] Andries de Man, Irish Logarithms, Available: https://sites.google.com/site/calculatinghistory/home/irish-logarithms-1 and https://sites.google.com/site/calculatinghistory/home/irish-logarithms-1/irish-logarithms-part-2-1
[35] Wikipedia, Zech's Logarithm, Available: https://en.wikipedia.org/wiki/Zech%27s_logarithm
[36] K. Hoecken, "Die Rechenmaschinen von Pascal bis zur Gegenwart, unter besonderer Berücksichtigung der Multiplikationsmechanismen", Sitzungsberichte Berliner Math. Gesellsch.13, pp.8–29, February 1913. Also available in file: "Hoecken-MultiplierMechanisms-german-1913.pdf" at https:/scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[37] David McQuillan, Percy Ludgate's Analytical Machine, Available: http://fano.co.uk/ludgate/
[38] David McQuillan, The Feasibility of Ludgate's Analytical Machine, Available: http://fano.co.uk/ludgate/Ludgate.html
[39] Dr. Joh. Schumacher, Ein Rechenschieber mit Teilung in gleiche Intervalle auf der Grundlage der zahlentheoretischen Indizes. Für den Unterricht konstruiert, München, 1909.
[40] Dieter von Jezierski, Detlef Zerfowski, Paul Weinmann: A.W. Faber Model 366 - System Schumacher. A Very Unusual Slide Rule, Journal of the Oughtred Society, pp.10-17, Vol.13, No.2, 2004. Available: http://osgalleries.org/journal/displayarticle.cgi?match=13.2/V13.2P10.pdf also at: https:/scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[41] Andries de Man, Irish Logarithms Animation. Available: http://ajmdeman.awardspace.info/t/irishlog.html
[42] D.Riches, An Analysis of Ludgate's Machine Leading to the Design of a Digital Logarithmic Multiplier, Dept.Electrical and Electronic Engineering, University College, Swansea, June 1973.
[43] Wikipedia, Differential (mechanical device), Available: https://en.wikipedia.org/wiki/Differential_(mechanical_device)
[44] Trinity College Dublin, IBM 360/44 console and subsystems, Available: https://www.scss.tcd.ie/SCSSTreasuresCatalog/hardware/TCD-SCSS-T.20121208.018/TCD-SCSS-T.20121208.018.pdf

[45] Wikipedia, Instruction pipelining, Available: https://en.wikipedia.org/wiki/Instruction_pipelining
[46] Wikipedia, PALcode, Available: https://en.wikipedia.org/wiki/PALcode
[47] Wikipedia, Epicyclic gearing, Available: https://en.wikipedia.org/wiki/Epicyclic_gearing
[48] Wikipedia, Baudot code, Available: https://en.wikipedia.org/wiki/Baudot_code
[49] Wikipedia, Gray code, Avaialble: https://en.wikipedia.org/wiki/Gray_code
[50] Maurice Manning and Moore McDowell, Electricity Supply in Ireland, The History of the ESB, Gill and MacMillan, Goldenbridge, Dublin 8, 1984.
[51] Bruce Collier and James MacLachlan, Charles Babbage: And the Engines of Perfection, pp.128, Oxford University Press, January 1999.
[52] Noel Baker, Potential €37m impact of digital life on Skibbereen, Irish Examiner, p.7, 5-Nov-2015.
[53] Bruce Collier, Little Engines That Could've: The Calculating Machines of Charles Babbage, Ph.D. Thesis, Harvard University, 1970. Available at: http://robroy.dyndns.info/collier/
[54] Brian Coghlan, An exploration of the life of Percy Ludgate, West Cork History Festival, Skibbereen, 10-Aug-2019. Available in file: "BrianCoghlan-An-exploration-of-the-life-of-Percy-Ludgate-wAnim.pdf" at https:/scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[55] Raul Rojas, The Computer Programs of Charles Babbage, Annals of the History of Computing, IEEE, March 2021.
[56] Raul Rojas, How to Make Zuse's Z3 a Universal Computer, 14-Jan-1998.
[57] Raul Rojas, Conditional Branching is not Necessary for Universal Computation in von Newmann Computers, Journal of Universal Computer Science, pp.756-768, Vol.2, No.11, 28-Nov-1996.
[58] Oscar Ibarra, Shlomo Moran, Louis Rosier, On the Control Power of Integer Division, Journal of Theoretical Computer Science, pp.35-52, Vol.24, 1983.
[59] David Harel, On Folk Theorems, Communications of the ACM, Vol.23, No.7, July, 1980.