



(51) International Patent Classification:

G06F 9/302 (2006.01) G06F 5/10 (2006.01)
G06F 7/544 (2006.01) G06F 9/38 (2006.01)

(21) International Application Number:

PCT/EP2011/055963

(22) International Filing Date:

14 April 2011 (14.04.2011)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PCT/EP2011/052929

28 February 2011 (28.02.2011)

EP

(71) Applicant (for all designated States except US): ST. JUDE MEDICAL AB [SE/SE]; S-175 84 Järfälla (SE).

(72) Inventor; and

(75) Inventor/Applicant (for US only): TULLBERG, Mattias [SE/SE]; Åkerholmsvägen 1, S-755 98 Uppsala (SE).

(74) Common Representative: ST. JUDE MEDICAL AB; Patent Department, S-175 84 Järfälla (SE).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— of inventorship (Rule 4.17(iv))

Published:

— with international search report (Art. 21(3))

(54) Title: MULTIPLY-AND-ACCUMULATE OPERATION IN AN IMPLANTABLE MICROCONTROLLER

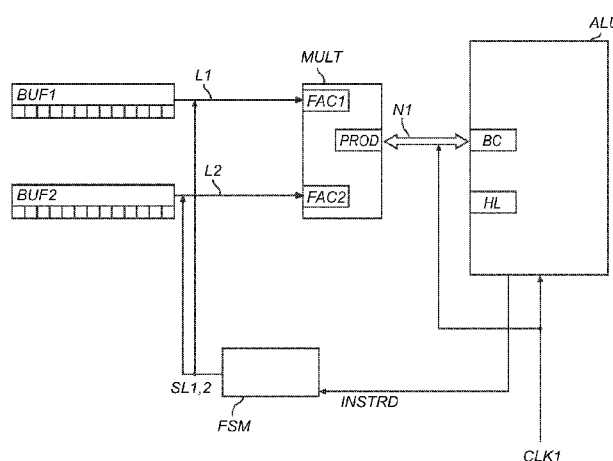


Fig. 8

(57) Abstract: The invention provides microprocessor extensions for cooperating with a sequential arithmetic-logic unit (ALU) to execute a multiply-and-accumulate operation (MAc). The ALU performs a continuous sequence of accumulation instructions synchronously with a clock signal (CLK1). Buffers (BUF1, BUF2) store input data which are fed to a combinatorial multiplier (MULT) by first buses (L1, L2). A second bus (N1) forwards the product to the ALU, where it is accumulated with previous data. Since at least the first buses operate independently of the clock signal, they do not limit the speed of the MAc operation. In particular embodiments, a finite state machine (FSM) controls the buses on the basis of triggers, e.g., signals from the multiplier and/or ALU indicating the completion of their respective instructions. The FSM may be operable in a low-power mode. The invention also relates to methods, computer programs and the use of a sequential ALU for executing MAc operations.

MULTIPLY-AND-ACCUMULATE OPERATION IN AN IMPLANTABLE MICROCONTROLLER

Technical field

The invention disclosed herein generally relates to an implementation of a multiply-and-accumulate (MAc) operation in a microprocessor that is subject to size and/or power constraints. In particular, the invention provides a MAc implementation suitable for an implantable medical device (IMD), such as a microcontroller associated with a pacemaker.

Background

IMDs for diagnostic or therapeutic purposes do not fully benefit from the accumulated advances in signal processing, data analysis and similar arts. This is because IMDs, just like embedded microprocessors in certain other applications, are subject to extreme constraints on size and/or power consumption. These quantities are correlated and cannot be reduced at the same time, considering that a faster circuit, with a higher degree of parallelization, will have a larger circuit footprint and higher power consumption. Thus, realistic IMDs sometimes fail to offer their programmers certain computational tools that would enable them to achieve clinical goals, to add new functionalities or to work around hardware limitations.

One example of such missing computational tools is the MAc operation, which maps given input vectors $f = (f_0, f_1, \dots, f_{n-1})$, $g = (g_0, g_1, \dots, g_{n-1})$ to a scalar output

$$f \cdot g = \sum_{k=0}^{n-1} f_k g_k. \quad (1)$$

This operation, which is also known as product-sum, multiply-add, scalar product, dot product and inner product, occurs in finite-impulse response filters (FIR) and more complex operations such as convolution. More precisely, if the above vectors are regarded as discrete functions $f(k) = f_k$, $g(k) = g_k$, $0 \leq k \leq n-1$, then a (discrete) convolution may be defined as the mapping from these functions to the function

2

$$(f * g)(l) = \sum_{k=0}^{n-1} f(k)g_{\text{per}}(l-k), \quad 0 \leq l \leq n-1, \quad (2)$$

where $g_{\text{per}}(k) = g(k \bmod n)$.

The design of multifunctional components is one way to increase the efficiency of microprocessors. US 2008/0229075 A1 discloses a MAc unit as part of a general-purpose central processing unit (CPU). In the interest of cost reduction, the MAc unit utilizes both dedicated hardware and existing CPU hardware to carry out a MAc operation. In particular, two CPU registers are reused and extended by further registers to accommodate wide operands. The MAc uses a sequential multiplier already provided within the CPU and performs the accumulation instruction by means of a dedicated adder and adder registers. The existing CPU bus has been extended in order to serve also the added, dedicated hardware. This known MAc unit offers the programmer a MAc operation similar to that available in dedicated digital signal processors (DSPs) without access to the full hardware set in such processors. A dedicated DSP will obviously perform better than this MAc unit, e.g., by requiring fewer clock cycles to accomplish one MAc operation. Alternative MAc units, that either perform better or are more economical, would therefore be an interesting prospect.

Summary

It is in view of the above concerns that the present invention has been made. One object of the invention is to provide a MAc unit comprising not only dedicated hardware, as an alternative to the prior art. Another object is to provide microprocessor extensions which form a MAc unit together with hardware already available in a microprocessor, particularly with hardware in a microcontroller in an embedded device, such as an IMD. A further object is to provide a way of operating a microprocessor that lacks a dedicated MAc functionality so that it performs or facilitates a MAc operation at an energy cost acceptable in practical IMD conditions.

Accordingly, the invention provides devices and methods with the features recited in the independent claims. Particular embodiments of the invention are defined by the dependent claims.

Typical IMD designs comprise a large number of interacting components, and both their performance and reliability critically depend on how efficiently

these interactions run. A central component like the microprocessor may have been selected at an early stage, typically several years before the release of the product, to allow sufficient time for successive prototypes to be tested and tuned. The designer may have attempted to select a processor with the smallest possible circuit footprint (size), yet with sufficient computing ability (e.g., its maximum number of instructions per second or number of floating-point operations per second) that the processor utilization (e.g., its duty cycle) would be well below 100 per cent in its expected normal operating regime. Neither was it desirable to include significant excess capacity, nor was it apparent several years in advance where such excess capacity would have been wisely spent among the various functionalities of the microprocessor. As a consequence of this workflow, new data-processing features that go beyond the original plan – or are introduced in subsequent product releases – can be enabled only to the extent available computational resources in the IMD permit.

The inventor has realized that reusing both the multiplier and accumulator functionalities in an existing microprocessor in a straightforward manner, such as by “for” loops in a high-level programming language, would lead to an overly slow implementation, extending the duty cycle and shortening battery life prohibitively already at a modest data sampling frequency. In terms of circuit footprint, it is not satisfactory either to use dedicated components for both the multiplication and accumulation tasks. The invention balances these considerations by providing, in a first aspect, a MAc unit having a multiplier implemented as a dedicated component with combinatorial logic, while utilizing the accumulation instruction in a sequential arithmetic-logic unit (ALU). The ALU is a shared resource in the sense that it is available for other duties when no MAc operation is being carried out. As the independent claims define in greater detail, the MAc unit includes extensions implemented in dedicated hardware for storage, communication and/or control purposes.

This configuration is advantageous since, firstly, accumulation can be efficiently implemented in sequential logic, wherein one memory can fulfil a double purpose, as a combined operand and result register; the ALU stores the result of an accumulation as one of the two terms for the next accumulation. Secondly, multiplication is a binary operation that lends itself well to a hardware implemen-

tation by combinatorial logic circuitry, which makes the product available a very short period after input of the operands and independently of any clock signal. The delay between input and reliable output is sometimes referred to as the FIFO depth (or instruction queue depth) of the combinatorial component. Further, the

5 buffers and/or buses operate independently of the clock signal of the ALU (that is, without a regular or predictable time relationship to this signal, or in a non-synchronized manner, or without time coincidence, or without justification or alignment; alternatively, the buffers and/or buses are not driven by the ALU clock signal; alternatively, the buffers and/or buses initiate their operations at points in

10 time which are variable with respect to the clock signal and which are located arbitrarily with respect to this signal, such as in suitable time intervals of non-zero length). This makes it possible to feed the ALU with new data for the next accumulation cycle in a sequence as soon as it has accomplished the previous one, thereby allowing that the ALU to be run continuously. As already noted, the de-

15 sign of a typical IMD may include a purposeful minimization of the computing ability of the ALU subject to other requirements, and the invention proposes a configuration allowing the limiting resource of the computing system to be run at maximum capacity for the duration of the MAc operation. In contrast, if a clock-synchronous (sequential) bus were used for the data fetching, several clock cy-

20 cles would be occupied during which the ALU would not complete MAc-related tasks. Yet another advantage lies in the fact that simple, low-width communication buses may be used without detriment to the overall performance; indeed, by virtue of the ALU's independence from the buses (and obviously from the non-sequential combinatorial multiplier as well), the buses may complete the data

25 fetch of an operand in two or more batches, which may still be accommodated within an accumulation instruction in the ALU. For these reasons, the solution proposed by the invention is an advantageous alternative to the device described in the prior art cited previously, and so the invention achieves at least one of its objects.

30 The microprocessor extensions alone and the MAc processor that they potentially form with an ALU independently fall within the scope of the invention. The microprocessor extensions comprise at least the buffers, the combinatorial multiplier and the first and second communication buses.

In a second aspect, the invention provides a method, preferably for implementation in a finite state machine (FSM), and more preferably for implementation in a FSM within the processing resources in an IMD, for performing a MAc operation. The method comprises:

- 5 • clearing the combined register in a sequential ALU operating synchronously with a first clock signal;
- transferring input data from buffers into operand registers of a combinatorial multiplier using first communication buses, which preferably are direct;
- transferring intermediate product data from a result register of the multiplier
- 10 into an operand register of the ALU using a second communication bus, which preferably is direct, wherein the ALU comprises a further register, which is a combined operand and result register;
- repeating the second and third steps until all input data have been processed; and
- 15 • allowing the ALU to complete the last accumulation operation and extracting the output of the MAc operation from the combined operand and result register of the ALU.

It is understood that the ALU will be continuously carrying out accumulation instructions for the duration of the MAc operation. By the invention, the first and

20 second communication buses operate independently of the first clock signal.

In a third aspect of the invention, the method may be made available as a computer-program product, or more precisely as computer-readable instructions stored on a data carrier.

In a fourth aspect, the invention relates to use of a sequential ALU in a

25 MAc processor, wherein input data are transferred from buffers to operand registers in a combinatorial multiplier using first communication buses, and intermediate product data from a result register of the multiplier are transferred into the operand register of the ALU using a second communication bus. These communication buses operate independently of the first clock signal. Preferably, the

30 buses are direct, in the sense that they form dedicated transmission lines without passing through a bus controller.

The invention further provides, in a fifth aspect, a MAc unit comprising:

- an ALU operating synchronously with a first clock signal and adapted to perform an accumulation instruction;
 - buffers as described above;
 - a combinatorial multiplier as described above;
- 5 • first communication buses operating independently of the first clock signal and being adapted to transfer input data from buffers to the multiplier; and
- a sequential bus operating synchronously with the first clock signal and being adapted to transfer intermediate product data from the product register of the multiplier and the operand register of the ALU.
- 10 Clearly, this fifth aspect differs from the first aspect in that the task of the asynchronous second communication bus now is fulfilled by a synchronous (sequential) bus, such as a system bus. This structure may be implementable in a broader range of available ALUs, especially if access to the ALU registers is subject to restrictions. In use, the ALU and the synchronous bus perform an uninterrupted sequence of intermediate product data transfers (sequential bus) and accumulation instructions (ALU) cumulating the intermediate product data thus
- 15 transferred. In a context where both the sequential bus and the sequential ALU are controlled by the same program, generally there is no simultaneity available. Instead the ALU and the sequential bus take turns, so that data transfers and accumulations are alternated. Preferably, the alternation takes place in a one-to-one
- 20 fashion, so that clock cycles devoted to the accumulation instruction are immediately followed by one or more clock cycle of data transfer over the sequential bus and vice versa. The program controlling the ALU and sequential bus may be an list of assembler instructions executed by a central processing unit (CPU) that
- 25 includes the ALU.

The microprocessor extensions alone and the MAc processor that they potentially form with an ALU independently fall within the scope of the invention. The microprocessor extensions comprise at least the buffers and the first communication buses.

- 30 Analogously with the fifth aspect, a sixth, seventh and eighth aspect of the invention respectively provide a method, computer-program product and an advantageous use of a sequential ALU with an accumulation faculty, a combinatorial multiplier and a sequential bus connecting these.

The variations and further developments which will be outlined below may be applied to any aspect of the present invention.

The first communication buses may extend from the buffers directly to the operand registers of the combinatorial multiplier. This means that the buses offer
5 a dedicated communication line between the buffers and the multiplier, without passing through a bus controller or similar device. In particular, the bus may be direct in the sense that it is independent from a system bus or equivalent device that serves the ALU or a microprocessor that the ALU is part of. The first communication buses may further have a layout allowing data from two buffers to be fed
10 into the operand registers of the multiplier. Also the second communication bus may run from the result register of the combinatorial multiplier directly to the operand register of the ALU.

The microprocessor extensions, and consequently the MAc system as well, may further comprise a FSM adapted to control the first communication
15 buses (and second communication buses, if such are provided) in such manner that the operand register of the ALU stores fresh intermediate product data at initiation of each accumulation instruction in the sequence. More precisely, if only first communication buses are controlled by the FSM, the FSM may be adapted to provided new input data to the multiplier operand registers a suitable period
20 before the clock cycle in which the sequential bus write data from the result register of the multiplier into the operand register of the ALU. Alternatively, if both first and second communication buses are controlled by the FSM, the latter may apply the relevant data at the inputs of the ALU some period before the next accumulation instruction is to be initiated. The FSM may verify that it provides the intermediate product data at the appropriate instants by fetching or receiving a signal that
25 indicates the completion of an accumulation instruction. For instance, the ALU may store this information in status flag registers, which the FSM can poll.

As another option for ensuring correct timing, the FSM may listen to the first clock signal that controls the ALU. Assuming that the ALU effects a new accumulation instruction at every N^{th} positive or, as the case may be, negative clock
30 signal edge, the FSM may verify that the intermediate product data were indeed provided to the ALU at the correct interval or point in time. The verifying may take place after each accumulation instruction or intermittently at regular or irregular

time intervals. It is noted that the clock signal alone contains less information than the status signal, since the FSM may then need to count all clock cycles continuously to keep track of the beginnings of new accumulation instructions.

Additionally or alternatively, the FSM may retrieve an indication of whether
5 intermediate product data are available in the result register of the combinatorial multiplier. This indication may consist in a change of the value of a status register or an output signal of the multiplier. In the absence of such an indication, the FSM may predict the instant at which the multiplication will have been completed. More
precisely, this instant is likely to occur a period, which corresponds to the FIFO
10 depth of the multiplier, after the latest change to the operands of the multiplier. When the indication has been received, the FSM may activate the second communication bus, so that intermediate product data are transferred to the ALU.

The FSM may be operable in a low-power mode. Hence, in addition to its normal mode, in which it controls the communication buses connecting the multiplier with buffers and/or ALU, it may enter a low-power mode until a subsequent
15 MAc operation is to be executed. Such low-power mode may include suspending the output signals (see below) or logical components of the FSM or in interrupting any polling for data from the multiplier and/or the ALU.

There are several ways in which the FSM may control the communication
20 buses. Advantageously, some or all of the buses are controlled by means of strobes. As used herein, a strobe is a selection signal that is active when data are correct on a bus. The strobe may also indicate the start or end of the data. It may be encoded with a different potential or time-length or may use a dedicated wire in a parallel bus. In general, a strobe is used to synchronize the data in an electric
25 bus when the bus components lack a common clock. In the invention, one strobe may be used to facilitate the FSM's control of a first communication bus or both first communication buses; when the signal is active, the relevant bus(es) establish(es) a link (pass-through) between its endpoints, so that the operand register of the multiplier coincide with the data word at a current memory position in the
30 corresponding buffer. Further, the FSM may use a different strobe to control the second communication bus, which in the active state of the strobe equates the data words at its endpoints.

If a low-width communication bus is employed, the FSM may be adapted to cause the bus to transfer data to an operand register by portions smaller than a number in the input data, that is, in two or more batches for each accumulation instruction.

- 5 The FSM may be a Moore machine (input-independent) or Mealy machine (input-dependent). Preferably, the FSM is a Mealy machine. A Mealy machine is typically capable to generate a pulse in response to a state transition, which may provide for a more efficient implementation.

10 As an alternative, while being independent from the first clock signal, which controls the operations of the ALU, the buses and/or buffers may be controlled by a different, second clock signal. The second clock signal may be generated by the FSM or some other component. There is no obvious advantage in synchronizing the first and second clock signals. Rather, a non-zero phase difference between these signals is advantageous and may allow use of a lower, less energy-
15 consuming bus frequency, as will be explained in more detail below. In typical implementations, the second clock frequency is equal to the first frequency or higher.

 It is to be noted that the FSM does not necessarily control all the communication buses. A possible alternative solution is the following: The FSM controls
20 the first communication buses, whereas the second communication buses are activated by a separate controller adapted to activate the second communication bus in response to a change into an active state of a signal indicating that the multiplier output (intermediate product data) has stabilized and is available.

 The buffers may include buffer logic for facilitating the addressing of the
25 storage locations in the buffers. In each buffer, the buffer logic may include a read pointer register, which stores an effective address to which the next buffer read operation will refer and which will therefore be updated at every buffer read operation. The logic may further include a modifier register, which is responsible for said updates by storing an increment by which the read pointer register is modified
30 between consecutive read operations. Preferably, the increment is signed and may thus express either a forward or backward shift of the pointer in the address space. The logic may as well include a data length register, by which a periodicity may be imposed on the input data. The joint update action defined by a

pointer register PTR, a modifier PMOD and a data length register PLEN, may be expressed as

$$\text{PTR} := (\text{PTR} + \text{PMOD}) \bmod \text{PLEN}. \quad (3)$$

5 A buffer read operation triggering said updating operation may be evidenced by an active period of a strobe signal controlling the first communication bus connecting the buffer to the multiplier. An active bus strobe signal may be easily detected by the buffer logic.

10 In a further development, the buffer logic may include a write pointer register as well and associated modifier and data length registers, analogously to the description already given. As a simple alternative, all buffer write operations may be directed to the same memory location, while shifting the already stored data away from the input location along the buffer.

The ALU may be a general-purpose component. Preferably it offers the set of instructions generally encountered in a Z80 architecture. This not only enables 15 MAC-related operations but also makes the ALU well suited for the computational tasks other than MAC. Hence, an IMD equipped with a Z80-architecture ALU and the extensions proposed by the invention will be able to fulfil the normal computational duties expected from such a device while offering an efficient implementation of the MAC functionality. Neither does this require modifications to existing 20 software modules, nor does it necessitate the provision of further hardware components. As used herein, Z80 architecture refers to the microprocessor Zilog Z80, which was originally conceived and sold by Zilog, Inc., San José CA, United States, and its successors. The term Z80 architecture is also meant to cover devices which lack a relationship to Zilog Z80 or Zilog, Inc. but which are nevertheless 25 suitable for replacing Zilog Z80 by offering similar capabilities, having a similar instruction set, or by being equipped with similar I/O interfaces or internal hardware and software. In particular, the term is meant to cover any microprocessor that offers a superset of the Zilog Z80 instruction set, since this microprocessor will be able to replace Zilog Z80. Alternatively, the ALU may be one of the 30 following architectures: 6502, 6800, 68000, 8051, x86 and RISC.

Also with the aim of providing a general-purpose processing device, there may be a further connection between the ALU and the combinatorial multiplier. The connection is preferably provided in the form of a communication bus operating synchronously with the first clock signal. This communication bus may be (a
5 portion of) a system bus. This allows the ALU to use the multiplier for speedy multiplication of large numbers even outside the context of MAc operations.

Further to fulfil the same aim, the ALU may be adapted to offer a minimal set of instructions. Such minimal set preferably includes the following instructions: and, compare, multiply, or, subtract, xor.

10 It is economical to activate only those components of the processor that are actually required to perform an operation or instruction. To this end, one or more of the buffers, multiplier, communication buses and, if such is provided, the finite state machine may be suppressed unless a MAc operation is being carried out. The suppression may simply consist in interrupting the electric power supply
15 to these devices, possibly with the exception for power sufficient to avoid loss of stored data.

A processor formed by an ALU and the microprocessor extensions in accordance with the above teachings may advantageously form part of an implantable medical device. In particular, the processor thus formed may be used in an
20 implantable pacemaker.

In one embodiment, there are provided microprocessor extensions for performing a MAc operation in cooperation with an ALU, which operates synchronously with a first clock signal, and a combinatorial multiplier. The ALU and the combinatorial multiplier are connected by a communication bus allowing intermediate product data to be transferred from a result register of the multiplier to an
25 operand register of the ALU. The ALU is operable to perform an accumulation operation in respect of the operand register and a combined operand and result register. The extensions include:

- buffers for storing sets of input data on which said MAc operation is to be
30 performed; and
- first communication buses for transferring input data from the buffers into operand registers of the combinatorial multiplier, wherein the first communication buses operate independently of the first clock signal.

The communication bus connecting the ALU and the multiplier either operates synchronously with the first clock signal or operates independently of the first clock signal. In either case, the first communication buses are adapted to provide the input data in such manner that the ALU (and the second bus, if this operates

5 synchronously with the first clock signal) is allowed to operate at maximum speed, so that for the duration of the MAC operation no clock cycle is wasted. A finite-state machine may be responsible for activating the communication buses in appropriate time intervals.

In one embodiment, the invention provides a method for performing a MAC

10 operation by means of a sequential arithmetic-logic unit, which operates synchronously with a first clock signal and is adapted to perform an accumulation instruction in respect of an operand register and a combined operand and result register. The method includes the steps of:

- i) clearing the combined operand and result register;
- 15 ii) transferring input data from buffers into operand registers of a combinatorial multiplier using first communication buses;
- iii) transferring intermediate product data from a result register of the combinatorial multiplier into the operand register of the ALU using a second communication bus;
- 20 iv) repeating steps ii) and iii) until all input data have been processed; and
- v) allowing the arithmetic-logic unit to complete the last accumulation instruction and extracting output data from the combined operand and result register.

In this embodiment, step ii) includes using communication buses operating independently of the first clock signal. Step iii) may be performed by means of a

25 communication bus operating synchronously with the first clock signal or a bus operating independently of this clock signal. Similarly to the previous embodiment, the ALU and possible connected clock-synchronous devices are allowed to operate at full speed for the duration of the MAC operation. A finite-state machine

30 may control the first communication buses, and, possibly, the second communication buses as well.

Brief description of the drawings

These and other aspects of the invention will be apparent from and further elucidated by the following description of particular embodiments. Reference is made to the accompanying drawings, on which:

5 figure 1 is a generalized block diagram of MAc processor which is formed by an ALU and extensions to this, in accordance with a first embodiment of the present invention;

 figure 2 is a detailed view of a data buffer for deployment in the embodiment shown in figure 1 or a similar device;

10 figure 3 is a generalized block diagram of a MAc processor, which may be regarded as a further development of the processor of figure 1;

 figure 4 is a more complete block diagram showing either of the MAc processors in figure 1 or 3, in which there is indicated a data connection between the ALU and the combinatorial multiplier, wherein said connection enables a more
15 versatile use of the processor or may even replace the second communication bus;

 figure 5 is a flowchart of a method for performing a MAc operation, in accordance with an embodiment of the invention;

 figure 6 is time plot of control signals and memory content in a processor
20 similar to the one shown in figure 1;

 figure 7 comprises time plots of control signals in processors wherein the microprocessor operate synchronously with a second clock signal; and

 figure 8 is a generalized block diagram of a MAc processor, in which the multiplier and the ALU are directly connected by a clock-synchronous communication bus.
25

Detailed description of embodiments

Figure 1 shows a microcontroller unit MCU comprising an arithmetic-logic unit ALU and a combinatorial multiplier MULT. The ALU may be one of the architectures mentioned above, in particular the Z80 architecture, and so possesses
30 basic functionalities, including a set of simple arithmetic and I/O instructions. It comprises two multi-bit registers BC, HL for storing the operands in an accumulation instruction, wherein the values of the operand registers BC, HL are added

and the result is written to one of the registers HL, which is, therefore, a combined operand and result register in respect of the accumulation instruction. The reference signs, e.g., "BC", used in the figure are not to be construed as an incentive to use a register carrying this particular label in an actual, physical processor. The register may also be formed of a combination of several sub-registers. The ALU is a sequential component, which is a precondition for the twofold use of the operand and result register HL insofar as the significance of the stored data can be separated in time by belonging to clock cycles. The ALU is supplied with a clock signal CLK1, which may have the general appearance shown in figure 6.

10 The clock signal CLK1 may function as a system clock of the microcontroller unit MCU. For the purposes of this description, it will be assumed that a clock cycle begins at a positive edge of the clock signal unless otherwise stated.

The combinatorial multiplier MULT has two operand inputs FAC1, FAC2 and one result output PROD. Since the multiplier is not clock-operated, the result of the multiplication appears at the output PROD instantly in an idealized model. A real multiplier may however be composed of cascaded logic components operating at finite speed, and so the output signal from the multiplier MULT as a whole may not stabilize until after a finite period of time. The period after which the output is reliable, the FIFO depth of the component, may be predetermined.

20 An indication to a similar effect may be had from a select signal MULT_SEL which changes into a "ready" state when sufficient time has passed from the latest change of the input signals to the multiplier MULT.

The microcontroller unit MCU may comprise several other sections than the multiplier MULT and the ALU, including data lines (not shown) for linking these components to one another, in particular one or more system buses.

25

The microprocessor extensions that the invention proposes additionally include buffers BUF1 and BUF2 for storing the vectors that form the inputs to the MAc operation. As will be described in more detail below, the buffers BUF1, BUF2 comprises a plurality of memory spaces for storing vector entries (components). Hence, preferably, the size (width) of individual memory spaces is compatible with those of the inputs to the multiplier MULT. The buffers BUF1, BUF2 are adapted to provide input data (entries of the input vectors) to the operand inputs FAC1, FAC2 of the multiplier MULT via two first communication buses L1,

30

L2 operating independently of the first clock signal CLK1. A second communication bus M1 is operable to connect the result register PROD of the multiplier to the (pure) operand register BC of the ALU. Its purpose is to forward intermediate product data corresponding to each term of the sum in equation (1) above. The second communication bus M1 forms part of the extensions provided by the invention and not of the microcontroller unit MCU itself; if the ALU and multiplier MULT are already connected through a system bus or the like, the second communication bus M1 is intended to provide a dedicated, direct data connection that is independent of the first clock signal CLK1.

10 The embodiment shown in figure 1 further comprises a finite state machine FSM, which may be a Mealy machine, or preferably a Moore machine, that is adapted to control the communication buses L1, L2, M1. The FSM controls the first communication buses L1, L2 by means of first strobes SL1, SL2, which connect the FSM to each communication bus, and similarly controls the second bus M1 by a second strobe SM1. In embodiments wherein the values of the first strobes SL1, SL2 coincide at all times, or may do so without inconvenience, the two first strobes may be replaced by one single strobe extending from the FSM to both first communication buses L1, L2. Examples of signals occurring at these strobes and generated by the FSM are illustrated in figure 6. In this embodiment, the FSM generates the strobes on the basis of a status signal INSTRD, which may be encoded with a status flag in the ALU that is susceptible of polling, and which indicates that an accumulation instruction has been completed in the previous cycle. As figure 6 illustrates, if the accumulation instruction takes two cycles to complete, the status signal will be low in the first clock cycle and high in the second cycle, thereby indicating that the relevant result register now contains the result of the operation. Using this information, the FSM is adapted to react to a change into the positive state of the status signal INSTRD by activating the first communication buses L1, L2 and then also the second communication bus M1, so that the next portion of intermediate product data is applied to the ALU input before the status signal INSTRD changes into its negative state. The activation of the first communication buses L1, L2 is achieved by setting the first strobes SL1, SL2 to their positive state, which is indicated as step 502. The activation of the second communication bus M1 corresponds to changing the second strobe SM1

to its positive state, in step 503. This way, said intermediate product data are written to the operand register BC of the ALU at the beginning of the next clock cycle, which is indicated in figure 6 by a dashed vertical line. It is emphasized that the respective activations of the communication buses L1, L2, M1 need not happen at a special instant in time, but may vary independently of one another as long as they are contained within a single clock cycle with a positive status signal INSTRD value. In the example shown in figure 6, the second communication bus M1 is activated a short while after the first communication buses L1, L2, which follows by the fact that the positive edges of the first strobes SL1, SL2 are not closer to the positive edges of the second strobe SM1 than a predetermined period denoted Δ , which corresponds to the FIFO depth of the combinatorial multiplier MULT. This way of operating the buses has the advantage that unreliable or obsolete intermediate product data are never made available to the ALU. However, if it can be tolerated that the ALU is fed with such unreliable data in between the instants at which data are actually written to the ALU operand register BC – this may be the case if the ALU is configured to be unaffected by values provided to it between the write instants – then the first and second communication buses may be activated simultaneously or even in the reverse order. In one embodiment, the FSM may be adapted to keep the second communication bus M1 active for the whole duration of a MAC operation, whereby the value of the result register PROD of the multiplier MULT is constantly applied to the input to the operand register BC of the ALU. In these variations, it is suitable to take proper account of the FIFO depth of the combinatorial multiplier MULT, that is, the input values fed to the multiplier are preferably not allowed to change in a final segment of length Δ prior to the instant at which data are written from the multiplier MULT into the operand register BC of the ALU.

In another embodiment, the FSM may, as an alternative or a supplement, use a select signal MULT_SEL from the multiplier MULT in order to determine suitable time intervals in which to activate the communication buses L1, L2, M1. Such a signal MULT_SEL, which is not necessarily available, has been indicated by a dashed connection line in figure 1. If the select signal MULT_SEL changes into its “ready” state after new input data have been supplied to the multiplier

MULT, then the FSM may consider the data supplied at the output of the multiplier MULT as reliable and thus susceptible of being forwarded to the ALU.

With reference now to figure 2, an arrangement for addressing memory spaces 20 within the buffers BUF1, BUF2 will now be described. In a typical use
5 situation, each of the buffers contains an operand (input vector) to the MAC operation, from which vector individual numbers are extracted, multiplied pairwise and added. As suggested by equation (2), this may amount to data being read from memory locations that are successive in one direction in one buffer and are successive in the reverse direction in the other buffer. More generally, the mem-
10 ory locations may be consecutive or separated by predetermined intervals. The buffer memories are preferably memory-mapped so that arbitrary spaces within them can be addressed and accessed. However, in a processor adapted for a given application, it may be expedient to equip the buffers BUF1, BUF2 with logic that is responsible for the addressing. More precisely, each buffer may include a
15 read pointer register 23 storing the address PTR1 of a read pointer 22 determining from where data are extracted, via the internal connection line 21 leading up to the connection point of the communication bus L1, the next time the communication bus L1 is active. The read pointer is updated between consecutive activation periods of the communication bus, and to this end the buffer logic further
20 comprises a modifier register 24 for storing a signed increment PMOD1, such as ± 1 , ± 2 , ± 3 , ..., and a data length register 25 for storing a number PLEN1 which indicates to the logic when one of the ends of the set of currently used memory spaces has been reached, wherein the pointer should initiate a new round starting from the opposite end. In one embodiment, the value PTR1 of the read pointer register 23 is updated by equation (3) above. Equation (3) may be evaluated
25 by an address generator (not shown) in the buffer BUF1. Figure 1 shows an exemplary case, wherein the increment PMOD1 is +4 and the data length is 19. Consequently, the read pointer 22 will jump back to a position at the left end of the memory spaces 20 after every fourth or fifth increment. If for some reason it is
30 desirable to begin the sequence of memory spaces at some offset from the zeroth space, then the skilled person will readily be able to modify this arrangement accordingly.

As already noted, the updating may be triggered by a deactivation of the first communication bus L1 connected to the buffer BUF1, as evidenced by a change between active and inactive values of a corresponding strobe SL1. Alternatively, it may be triggered by the inverse of the second strobe SM1. Figure 6 shows values of the read pointers PTR1, PTR2 of the first and second buffers BUF1, BUF2. It can be seen that the first read pointer PTR1 is increased by one unit when the strobes change into their low value. Meanwhile, the second read pointer PTR2 is decreased by one unit. Other triggers may be conceived and applied within the scope of the present invention. As one example, the FSM may be adapted to utilize a dedicated trigger signal in order to cause the buffer logic to advance to the next stored value in the buffer. The second buffer BUF2 may have a similar structure as that shown in figure 2. Similar logic for addressing the memory during write operations may be included in one or both buffers, especially in a buffer intended for measured data since this will be subject to frequent write operations. The updating of a write pointer register (analogous to the read pointer register) may be triggered by the action of addressing a memory address within the concerned buffer, as evidenced, e.g., by the value of an address bus (not shown) in the microprocessor, to which by the buffer and the measuring means are connected.

The functioning of the extensions described above, as well as their cooperation with an ALU performing a continuous sequence of accumulation instructions, will now be summarized with reference to the flowchart in figure 5. A particular use envisioned for the present invention, which may serve as an example for in this description, is for subjecting measured data points to a finite impulse response (FIR) filter. The data points may have been captured by a transducer within a sensor or measuring device communicatively connected to the processor, possibly via an analog-to-digital converter and other devices that the skilled person will select and deploy without difficulty. Such data points may be stored in one buffer BUF1, which is then updated with new data (e.g., by shifting out the oldest data) in connection with measurements. The filter coefficients may be stored in the other buffer BUF1, the content of which is therefore relatively more constant and has a length corresponding to the tap number of the filter. An evaluation of the currently stored data points with respect to the filter coefficients is

described by equation (1), where vectors f, g correspond to filter coefficients and data points, respectively.

In a first step 501, the combined operand and result register HL in the ALU is cleared of its previous values, e.g., by writing a zero.

5 In a second step 502, the FSM transfers the first pair f_1, g_1 of input data points from the respective buffers BUF1, BUF2 to the inputs of the multiplier MULT. To this end, the FSM may activate the first communication buses L1, L2, as discussed above. After a small or negligible delay corresponding to the FIFO depth of the multiplier, a reliable result of the multiplication is available at the out-
10 put PROD, namely the intermediate product $f_1 g_1$.

In a third step 503 then, the FSM effects the transfer of the intermediate product data to the operand register BC of the ALU via the second communication bus M1. The third step is to be completed a short while prior to the next instant at which the ALU initiates the next accumulation instruction in the sequence,
15 this instant being indicated by a vertical dashed line in figure 6. Because this is the first accumulation instruction after the clearing of the combined register HL, the ALU will then add the intermediate product $f_1 g_1$ to zero and store the result in the combined register HL. It is noted that the second and third steps 502, 503 of *transferring* data to and from the multiplier MULT are necessarily carried out in
20 the order set out here. However, while the relevant communication buses have to be active when the transfer is to take place, it may not always be necessary to deactivate the buses otherwise. It is also possible to activate the communication buses over extended intervals having start points and end points with an order that is different from the order of the transferring steps 502, 503. In particular, the
25 start points and/or end points of the intervals may coincide. As noted above, the second communication bus may even be maintained active throughout the MAC operation if it can be tolerated that incorrect data are fed to the ALU in between the write instants.

In a fourth step 504, it is assessed whether or not all data points have
30 been processed. In terms of equation (2), it is checked whether the summation index has reached its final value $k = n - 1$. If not, the second and third steps 502, 503 are repeated. On the first repetition, the ALU will be presented with the second intermediate product $f_2 g_2$, which will be added to the already stored first in-

intermediate product, from which it will result that the combined register HL will contain $f_1g_1 + f_2g_2$ after completion of the second accumulation instruction.

In a fifth step 505, it will have been established in the last repetition of the fourth step 504 that no more intermediate products need to be fed to the ALU.

- 5 Since the ALU will perform the accumulation instruction in finite time, such as two clock cycles, this time must elapse before the result of the MAc operation is extracted from the combined register HL. This is the endpoint of the method.

Figure 3 shows an ALU and extensions which cooperate with the ALU and represent an alternative embodiment of the present invention. Figure 3 clarifies
10 that it is the second buffer BUF2 that receives external measurement data, while the first buffer BUF1 contains previously stored data, such as filter coefficients. In addition to the circuitry shown in figure 1 (wherein any optional entities in that embodiment remain optional here), this embodiment includes a controller
ON/OFF for activating or deactivating the FSM. The controller may help optimize
15 the use of energy, which is particularly important in battery-powered devices. The controller may be configured to so that it automatically turns the FSM off after completion of a MAc operation. It may also be timer controlled, so that the FSM is turned off after a predetermined period.

Another additional, optional feature shown in figure 3 is a connection DA-
20 TA_REC from the second buffer BUF2 and the FSM. Such connection may be used to inform the FSM that a new data item has been written to the second buffer BUF2. This may facilitate the programming of the FSM in practical application. For instance, in a situation where the device shown in figure 3 is used to process successive data values arriving at time instants separated by more time than the
25 duration of one MAc operation, the second buffer BUF2 may notify the FSM directly via the connection DATA_REC when there are updated data due to be processed.

A particular embodiment includes the connection DATA_REC from the second buffer BUF2 to the FSM, the select signal MULT_SEL from the multiplier
30 MULT to the FSM and the status signal INSTRD from the ALU to the FSM. The devices may then be configured as follows:

- The FSM activates the first communication buses L1, L2 in response to either a notification from the buffer BUF2 of new data or in response to a positive status signal INSTRD from the ALU.
- The FSM activates the second communication bus M1 in response to a change into the “ready” state of the select signal MULT_SEL from the multiplier MULT.
- The ALU carries out only a number PLEN2 (or PLEN1) of accumulation instructions corresponding to the number of stored data values (or filter taps). Hence, when the end the MAC operation has been reached, the FSM stops automatically.

Figure 4 shows a MAC processor, wherein the ALU and multiplier MULT have been drawn together with the sequential system bus N1 of the microcontroller unit MCU. The system bus N1 operates synchronously with the first clock signal CLK1 and is thus synchronized with the ALU as well. This device represents a versatile processor which, in addition to the MAC operations described above, carries out simple standard instructions, such as and, compare, decrement, increment, load, multiply, or, subtract, shift, xor etc. The system bus N1 also enables the processor to multiply wide operands efficiently by using the combinatorial multiplier MULT. With a view to energy efficiency, the processor is preferably adapted to suppress (or power off) the not-needed components when simple standard instructions are carried out. As such, the communication buses L1, L2, M1 and FSM may be suppressed during a wide-operand multiplication. For the simple standard instructions exemplified above, the multiplier MULT may be suppressed as well.

Still with reference to figure 4, a further embodiment of the invention includes use of the system bus N1 for forwarding data from the multiplier MULT to the operand register BC of the ALU. This approach may be advisable in a situation where access to the ALU registers is limited, e.g., due to logical restrictions imposed by the manufacturer. In terms of Z80 pseudo-instructions, this may amount to substituting

```
LD BC,[mul_prod_reg_lo]
```

```
ADD HL,BC
```

for every occurrence of

ADD HL,BC

that would normally be given to a central processing unit controlling the ALU. The above LD instruction loads the content from the PROD register in the combinatorial multiplier MULT via the system bus N1 into the BC register of the ALU. As such, this embodiment provides a MAc processor which, in addition to a sequential ALU that operates synchronously with a first clock signal CLK1, comprises:

- buffers BUF1, BUF2 as described above;
- a combinatorial multiplier MULT as described above;
- first communication buses L1, L2 operating independently of the first clock signal and being adapted to transfer input data from buffers to the multiplier; and
- a system bus N1 operating synchronously with the first clock signal CLK1 and being adapted to transfer intermediate product data from the product register PROD of the multiplier MULT and the operand register BC of the ALU.

It is noted that the second communication bus M1 is never active in this embodiment and may be omitted. Likewise, the FSM need not generate the second strobe SM1 for controlling the second communication bus M1. In comparison with the previously disclosed embodiments, the present one may perform less well under similar conditions considering that the intermediate product data are forwarded by means of the synchronous system bus N1 instead of a clock-independent bus. Clearly, at least one clock cycle between consecutive accumulation instructions is devoted to fetching the intermediate product data. Hence, if each accumulation instruction requires two cycles, the total MAc instruction may take 50 per cent more time to accomplish.

As figure 8 shows, the clock-synchronous communication bus N1 responsible for forwarding intermediate product data need not be a system bus but may also be configured as a dedicated bus extending directly between the multiplier MULT and the ALU. Similarly to the system bus in the preceding embodiment, this dedicated communication bus N1 may be controlled by a central processing unit adapted to control the ALU as well. It is noted that the FSM in the MAc processor of figure 8 only requires the status signal INSTRD as input in order to generate control signals SL1, SL2 intended for the first communication buses L1, L2.

From the value of this signal, the FSM is able to derive a suitable time interval in which to feed new input data to the operand registers FAC1, FAC2 of the multiplier MULT using the first communication buses L1, L2.

Figure 7a is a time plot of five binary control signals CLK1, SL1, SL2 (wherein SL2 = SL1), SM1 as described above and a second clock signal CLK2, which is provided to the microprocessor extensions. The vertical dashed lines indicate instants at which the ALU initiates an accumulation instruction, which is when data provided to the ALU by means of the second communication bus M1 is written into its operand register BC. In this example, the frequency of the second clock signal CLK2 is $3/2$ times the frequency of the first clock signal CLK1, and certain pairs of positive edges coincide in time. In this embodiment, the control signals SL1, SL2, SM1 to the communication buses L1, L2, M1 are strobes in the sense that they activate the concerned communication bus so that data words at its endpoints are equated. The first communication buses L1, L2 are activated in the 2nd cycle, prior to the activation of the second communication bus M1, which happens in the 3rd cycle. Hence, the output (intermediate product data) of the multiplier MULT will be allowed to stabilize during the 2nd cycle before it is forwarded to the ALU, thereby avoiding the risk of incorrect data being introduced into the MAc computation. Without any obvious inconvenience, the second clock signal CLK2 may operate at a higher frequency than $3/2$ of the first clock frequency, as disclosed in figure 7a; the use of such higher frequencies may however increase the energy consumption of the components.

Figure 7b is a time plot of the same control signals as in figure 7a. The two clock signals CLK1, CLK2 have equal frequencies but are separated by a non-zero phase difference. In this embodiment, the microprocessor extensions are implemented in sequential logic, which also falls within the scope of the present invention. References to the strobes SL1, SL2, SM1 are to be interpreted accordingly, in the context of sequential buses. The components of the microprocessor extensions are configured as follows:

- A falling signal (e.g., a status signal INSTRD) triggers, with a one-cycle delay, a falling first strobe signal SM1 to allow ALU prefetch.
- The second communication bus M1 is triggered by a negative edge on the second strobe SM1.

- The first strobe SL1 is the logical inverse of the second strobe SM1.
 - The transfers from the buffers BUF1, BUF2 are triggered by a negative edge on the first strobe SL1. Thus, the first communication buses L1, L2 will write their input data a half second-clock CLK2 cycle after the transfer
- 5 over the second communication bus M1.

Preferably, the unit responsible for generating the second clock signal CLK2 (e.g., the FSM) verifies at appropriate intervals that the phase difference with respect to the first clock signal CLK1 stays within suitable limits.

A use currently envisioned for the processor formed by an ALU and the extensions disclosed herein is in an IMD and particularly a pacemaker device. The data processing may include subjecting cardiac data collected by sensors connected to the IMD to appropriate digital filters, so as to extract data relevant to diagnosis or therapy. The cardiac data may be obtained by sampling physiological electric signals.

10

Further embodiments of the present invention will become apparent to a person skilled in the art after studying the description above. Even though the present description and drawings disclose embodiments and examples, the invention is not restricted to these specific examples. Numerous modifications and variations can be made without departing from the scope of the present invention, which is defined by the accompanying claims. Technical features may be combined to advantage even though they are recited in different claims or in connection with different embodiments.

15

20

The systems and methods disclosed hereinabove may be implemented as software, firmware, hardware or a combination thereof. In a hardware implementation, the division of tasks between functional units referred to in the above description does not necessarily correspond to the division into physical units; to the contrary, one physical component may have multiple functionalities, and one task may be carried out by several physical components in cooperation. Certain components or all components may be implemented as software executed by a digital signal processor or microprocessor, or be implemented as hardware or as an application-specific integrated circuit. Such software may be distributed on computer readable media (or data carriers), which may comprise computer storage media (or non-transitory media) and communication media (or transitory media). As is

25

30

well known to a person skilled in the art, computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage

5 media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer. Further, it is well

10 known to the skilled person that communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media.

CLAIMS

1. Microprocessor extensions for performing a multiply-and-accumulate, MAc, operation, by cooperating with:
 - 5 a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform an accumulation instruction in respect of an operand register (BC) and a combined operand and result register (HL);
 - a combinatorial multiplier (MULT) comprising two operand registers (FAC1,
 - 10 FAC2) and a result register (PROD); and
 - a sequential bus (N1) operating synchronously with the first clock signal (CLK1) and connecting the result register (PROD) of the combinatorial multiplier to the operand register (BC) of the arithmetic-logic unit,
 - wherein the arithmetic-logic unit (ALU) and the sequential bus (N1) are
 - 15 configured to perform a continuous sequence of transfers of intermediate product data from the result register (PROD) of the multiplier to the operand register (BC) of the arithmetic-logic unit, in alternation with accumulation instructions in respect of the operand register (BC) and the combined operand and result register (HL) of the arithmetic-logic unit,
 - 20 the microprocessor extensions comprising:
 - buffers (BUF1, BUF2) for storing sets of input data on which said MAc operation is to be performed; and
 - first communication buses (L1, L2) for transferring input data from buffers into the operand registers (FAC1, FAC2) of the multiplier,
 - 25 wherein the first communication buses (L1, L2) operate independently of the first clock signal (CLK1).
2. A processor for performing a multiply-and-accumulate, MAc, operation, comprising:
 - 30 a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform an accumulation instruction in respect of an operand register (BC) and a combined operand and result register (HL);

a combinatorial multiplier (MULT) comprising two operand registers (FAC1, FAC2) and a result register (PROD);

a sequential bus (N1) operating synchronously with the first clock signal (CLK1) and connecting the result register (PROD) of the combinatorial multiplier to the operand register (BC) of the arithmetic-logic unit; and

the microprocessor extensions of claim 1,

wherein the arithmetic-logic unit (ALU) and the sequential bus (N1) are configured to perform a continuous sequence of transfers of intermediate product data from the result register (PROD) of the multiplier to the operand register (BC) of the arithmetic-logic unit, in alternation with accumulation instructions in respect of the operand register (BC) and the combined operand and result register (HL) of the arithmetic-logic unit.

3. The device of claim 1 or 2, wherein the arithmetic-logic unit (ALU) and the sequential bus (N1) are configured to perform said transfers and accumulation instructions in one-to-one alternation.

4. The device of any one of claim 1 to 3, further comprising:

a finite state machine (FSM) adapted to receive a signal (INSTRD) indicating completion of an accumulation instruction and, based thereon, to control the communication buses (L1, L2) in such manner that the operand register (BC) of the arithmetic-logic unit stores fresh intermediate product data at initiation of each accumulation instruction in the sequence.

5. The device of claim 4, wherein the finite state machine (FSM) is operable in a normal mode and a low-power mode.

6. The device of claim 4 or 5, wherein the finite state machine (FSM) is adapted to control said first communication buses (L1, L2) by means of strobes (SL1, SL2).

7. The device of any one of claims 4 to 6, wherein the finite state machine (FSM) is a Mealy machine.

8. The device of any one of claims 4 to 7, wherein the buffers and communication buses operate synchronously with a second clock signal (CLK2), distinct from the first clock signal (CLK1), wherein the frequency of the second clock signal is greater than or equal to the frequency of the first clock signal.
9. The device of claim 1 or 2, wherein a buffer (BUF1, BUF2) is associated with buffer logic comprising:
- a read pointer register (PTR1, PTR2) for storing an effective address reference to the memory location of the buffer from which data is read;
 - a modifier register (PMOD1, PMOD2) for storing an increment by which the read pointer register is modified between consecutive read operations; and
 - a data length register (PLEN1, PLEN2) for controlling cyclic rotation which said incremental pointer register modifications are subjected to.
10. The device of claim 1 or 2, wherein the arithmetic-logic unit (ALU) is a Z80 architecture.
11. The device of claim 1 or 2, wherein the finite state machine (FSM) is adapted to receive a notification signal (DATA_REC) indicating that input data stored in a buffer (BUF1, BUF2) have changed and, based thereon, to initiate a MAc operation.
12. The processor of claim 2, further adapted to respond to instructions in the group comprising:
- and, compare, decrement, increment, load, multiply, or, subtract, shift, xor by operating the arithmetic-logic unit independently and suppressing said buffers, communication buses and, if any, said finite state machine.
13. An implantable medical device including the processor of any one of claims 2 to 12.

14. A method for performing a multiply-and-accumulate, MAC, operation by means of a sequential arithmetic-logic unit (ALU) and a sequential second communication bus (N1), which operate synchronously with a first clock signal (CLK1),

5 said method comprising the steps of:

i) clearing (501) a combined operand and result register (HL) of the arithmetic-logic unit (ALU);

10 ii) transferring (502) input data from buffers (BUF1, BUF2) into operand registers (FAC1, FAC2) of a combinatorial multiplier (MULT) using first communication buses (L1, L2) operating independently of the first clock signal (CLK1);

iii) transferring (503) intermediate product data from a result register (PROD) of the combinatorial multiplier (MULT) into an operand register (BC) of an arithmetic-logic unit (ALU) using the sequential second communication bus (N1);

15 iv) performing an accumulation instruction in respect of the operand register (BC) and a combined operand and result register (HL) of the arithmetic-logic unit (ALU);

v) repeating steps ii), iii) and iv) until all input data have been processed, wherein a continuous sequence of instances of step iii) in alternation with
20 instances of step iv) is performed.

15. The method of claim 14, wherein a continuous sequence of instances of step iii) in one-to-one alternation with instances of step iv) is performed.

25 16. The method of claim 14 or 15, wherein the second communication bus (N1) when in continuous operation is adapted to initiate a transfer of intermediate product data in response to every Nth edge of the first clock signal (CLK1),

the method comprising completing step ii) prior to an edge at which the second communication bus (N1) initiates a transfer.

30

17. The method of any one of claims 14 to 16, wherein step ii) includes using communication buses (L1, L2) operating synchronously with a second clock signal (CLK2), distinct from the first clock signal (CLK1), wherein the frequency of

the second clock signal is greater than or equal to the frequency of the first clock signal.

18. The method of any one of claims 14 to 17, wherein the arithmetic-logic unit
5 (ALU) is a Z80 architecture.

19. A data carrier storing computer-readable instructions for performing the method of any one of claims 14 to 18.

10 20. Use of circuitry in a processor for performing a multiply-and-accumulate, MAc, operation, said circuitry comprising:

a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform an accumulation instruction in respect of an operand register (BC) and a combined operand and result register (HL);
15

a combinatorial multiplier (MULT) comprising two operand registers (FAC1, FAC2) and a result register (PROD); and

a sequential bus (N1) operating synchronously with the first clock signal (CLK1) and connecting the result register (PROD) of the combinatorial multiplier to the operand register (BC) of the arithmetic-logic unit,
20

wherein the arithmetic-logic unit (ALU) and the sequential bus (N1) are configured to perform a continuous sequence of transfers of intermediate product data from the result register (PROD) of the multiplier to the operand register (BC) of the arithmetic-logic unit, in alternation with accumulation instructions in respect of the operand register (BC) and the combined operand and result register (HL) of the arithmetic-logic unit,
25

wherein:

input data from buffers (BUF1, BUF2) are transferred into operand registers (FAC1, FAC2) of a combinatorial multiplier (MULT) using first communication buses (L1, L2); and
30

said first communication buses (L1, L2) operate independently of the first clock signal (CLK1).

21. Microprocessor extensions for cooperating with a sequential arithmetic-logic unit (ALU) to perform a multiply-and-accumulate, MAc, operation,
wherein the arithmetic-logic unit (ALU) operates synchronously with a first clock signal (CLK1) and is adapted to perform a continuous sequence of accumulation instructions in respect of an operand register (BC) and a combined operand and result register (HL),
the microprocessor extensions comprising:
buffers (BUF1, BUF2) for storing sets of input data on which said MAc operation is to be performed;
a combinatorial multiplier (MULT) comprising two operand registers (FAC1, FAC2) and a result register (PROD);
first communication buses (L1, L2) for transferring input data from buffers into the operand registers (FAC1, FAC2) of the multiplier; and
a second communication bus (M1) for transferring intermediate product data from the result register (PROD) of the multiplier into the operand register (BC) of the arithmetic-logic unit,
wherein the buses operate independently of the first clock signal (CLK1).
22. A processor for performing a multiply-and-accumulate, MAc, operation, comprising:
a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform a continuous sequence of accumulation instructions in respect of an operand register (BC) and a combined operand and result register (HL), and
the microprocessor extensions of claim 21.
23. The device of claim 21 or 22, further comprising:
a finite state machine (FSM) adapted to receive a signal (INSTRD) indicating completion of an accumulation instruction and, based thereon, to control the communication buses (L1, L2, M1) in such manner that the operand register (BC) of the arithmetic-logic unit stores fresh intermediate product data at initiation of each accumulation instruction in the sequence.

24. The device of claim 23, wherein the finite state machine (FSM) is further adapted to receive a signal (MULT_SEL) from the multiplier indicating that intermediate product data are available in the result register (PROD) and, based thereon, to control the second communication bus (M1).

5

25. The device of claim 23 or 24, wherein the finite state machine (FSM) is operable in a normal mode and a low-power mode.

26. The device of any one of claims 23 to 25, wherein the finite state machine (FSM) is adapted to control said first and second communication buses (L1, L2, M1) by means of strobes (SL1, SL2, SM1).

10

27. The device of any one of claims 23 to 26, wherein the finite state machine (FSM) is a Mealy machine.

15

28. The device of any one of claims 23 to 27, wherein the buffers and communication buses operate synchronously with a second clock signal (CLK2), distinct from the first clock signal (CLK1), wherein the frequency of the second clock signal is greater than or equal to the frequency of the first clock signal.

20

29. The device of claim 21 or 22, wherein a buffer (BUF1, BUF2) is associated with buffer logic comprising:

a read pointer register (PTR1, PTR2) for storing an effective address reference to the memory location of the buffer from which data is read;

25

a modifier register (PMOD1, PMOD2) for storing an increment by which the read pointer register is modified between consecutive read operations; and

a data length register (PLEN1, PLEN2) for controlling cyclic rotation which said incremental pointer register modifications are subjected to.

30

30. The device of claim 21 or 22, wherein the arithmetic-logic unit (ALU) is a Z80 architecture.

31. The device of claim 21 or 22, wherein the arithmetic-logic unit (ALU) is further connected to the multiplier (MULT) via an internal bus (N1) operating synchronously with the first clock signal (CLK1).
- 5 32. The device of claim 21 or 22, wherein the finite state machine (FSM) is adapted to receive a notification signal (DATA_REC) indicating that input data stored in a buffer (BUF1, BUF2) have changed and, based thereon, to initiate a MAc operation.
- 10 33. The processor of claim 22, further adapted to respond to instructions in the group comprising:
and, compare, decrement, increment, load, multiply, or, subtract, shift, xor
by operating the arithmetic-logic unit independently and suppressing said buffers, communication buses and, if any, said finite state machine.
- 15 34. An implantable medical device including the processor of any one of claims 22 to 33.
- 20 35. A method for performing a multiply-and-accumulate, MAc, operation by means of a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform a continuous sequence of accumulation instructions in respect of an operand register (BC) and a combined operand and result register (HL),
said method comprising the steps of:
- 25 i) clearing (501) the combined operand and result register (HL);
ii) transferring (502) input data from buffers (BUF1, BUF2) into operand registers (FAC1, FAC2) of a combinatorial multiplier (MULT) using first communication buses (L1, L2);
iii) transferring (503) intermediate product data from a result register
30 (PROD) of the combinatorial multiplier (MULT) into the operand register (BC) of the arithmetic-logic unit (ALU) using a second communication bus (M1);
iv) repeating steps ii) and iii) until all input data have been processed; and

v) allowing (505) the arithmetic-logic unit (ALU) to complete the last accumulation instruction and extracting output data from the combined operand and result register (HL),

5 wherein steps ii) and iii) include using communication buses operating independently of the first clock signal (CLK1).

36. The method of claim 35, wherein the arithmetic-logic unit (ALU) when in continuous operation is adapted to initiate an accumulation instruction in response to every Nth edge of the first clock signal (CLK1),

10 the method comprising completing step iii) prior to an edge at which the arithmetic-logic unit (ALU) initiates an accumulation instruction.

37. The method of claim 35 or 36, wherein step iv) comprises a first sub-step of

15 iv-1) polling the arithmetic-logic unit (ALU) for a signal (INSTRD) indicating completion of an accumulation instruction,
and a subsequent, second sub-step of
iv-2) repeating steps ii) and iii).

20 38. The method of any one of claims 35 to 37, wherein steps ii) and iii) include using communication buses operating synchronously with a second clock signal (CLK2), distinct from the first clock signal (CLK1), wherein the frequency of the second clock signal is greater than or equal to the frequency of the first clock signal.

25

39. The method of any one of claims 35 to 38, wherein the arithmetic-logic unit (ALU) is a Z80 architecture.

40. A data carrier storing computer-readable instructions for performing the
30 method of any one of claims 35 to 39.

41. Use of a sequential arithmetic-logic unit (ALU), which operates synchronously with a first clock signal (CLK1) and is adapted to perform a continuous

sequence of accumulation instructions in respect of an operand register (BC) and a combined operand and result register (HL), in a processor for performing a multiply-and-accumulate, MAc, operation, wherein:

- input data from buffers (BUF1, BUF2) are transferred into operand registers (FAC1, FAC2) of a combinatorial multiplier (MULT) using first communication buses (L1, L2); and
- intermediate product data from a result register (PROD) of the combinatorial multiplier (MULT) are transferred into the operand register (BC) of the arithmetic-logic unit (ALU) using a second communication bus (M1); and
- said communication buses operate independently of the first clock signal (CLK1).

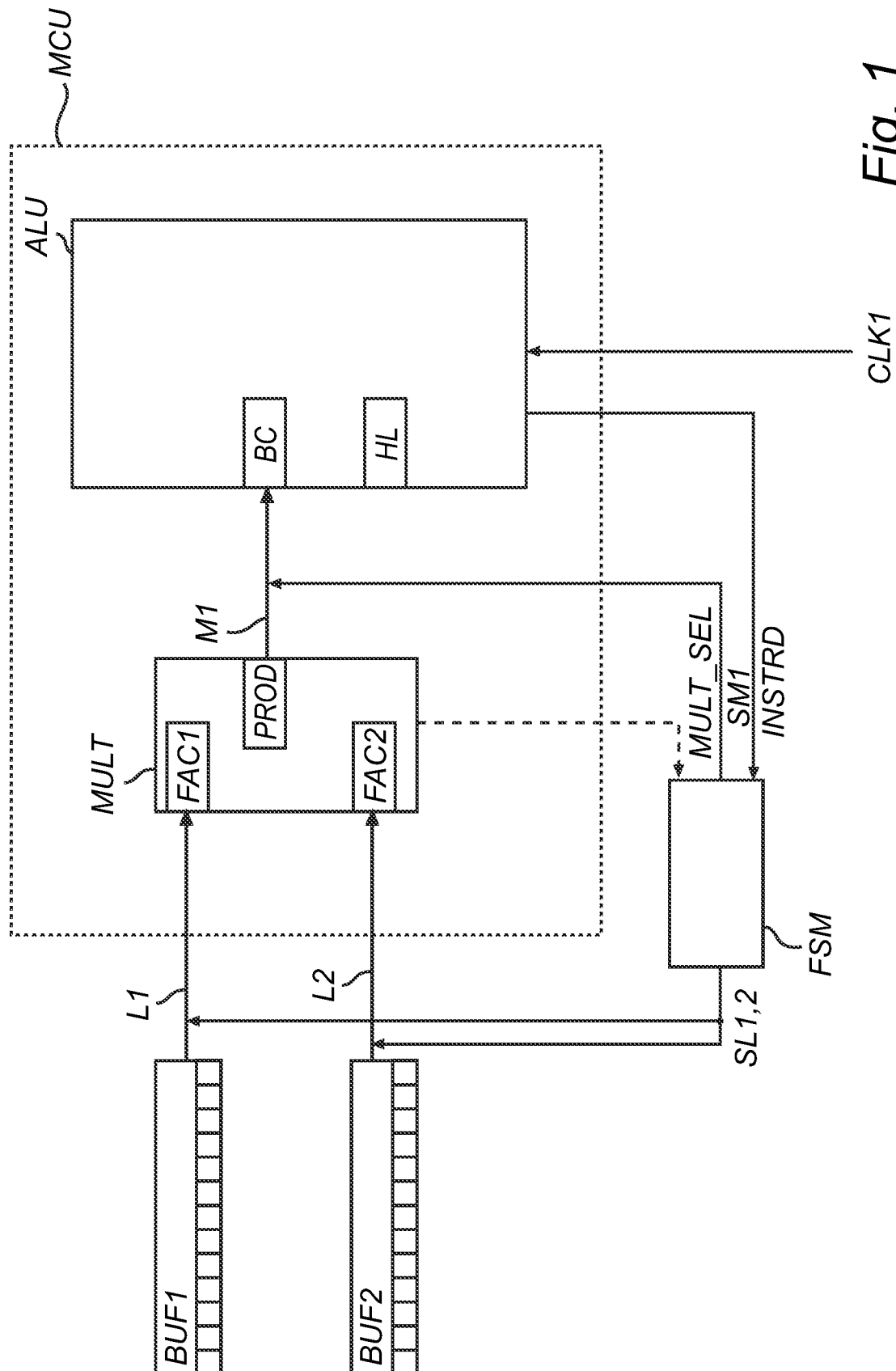
$\frac{1}{8}$ 

Fig. 1

2/8

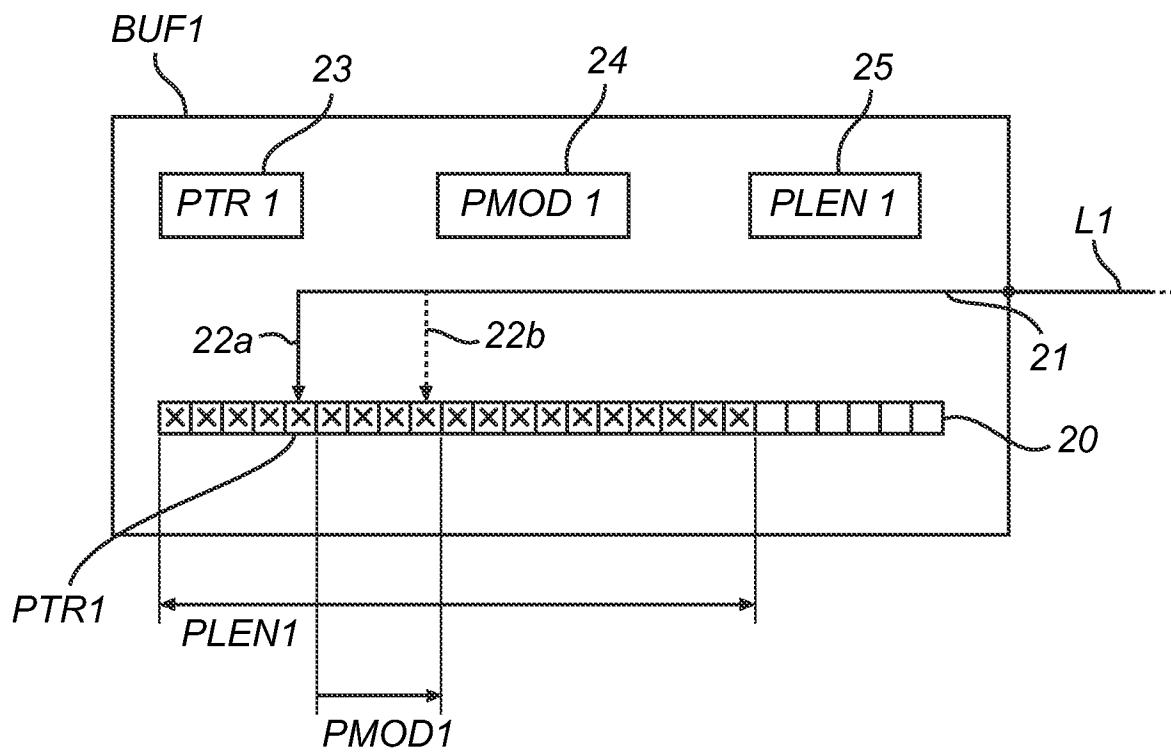


Fig. 2

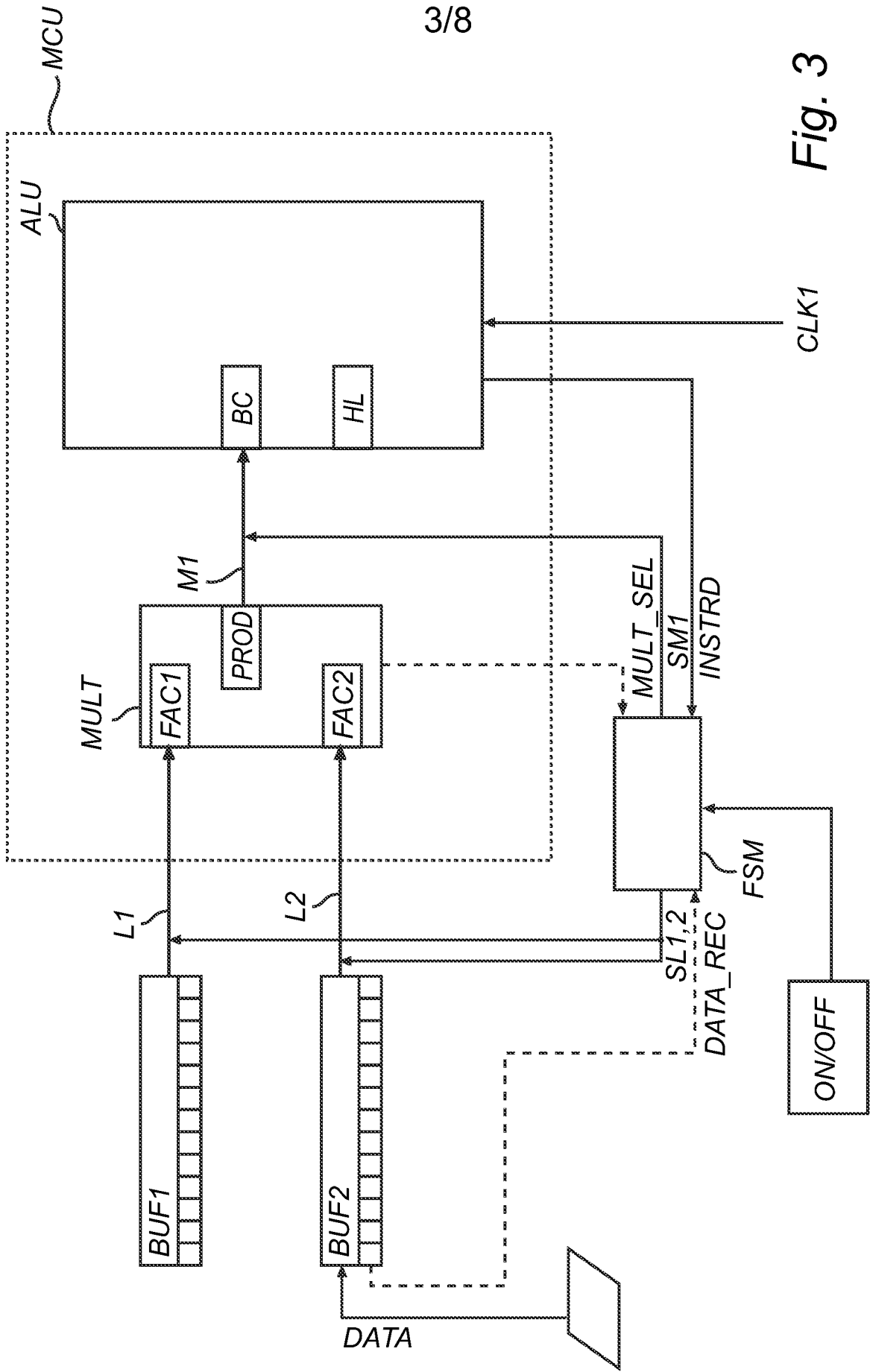


Fig. 3

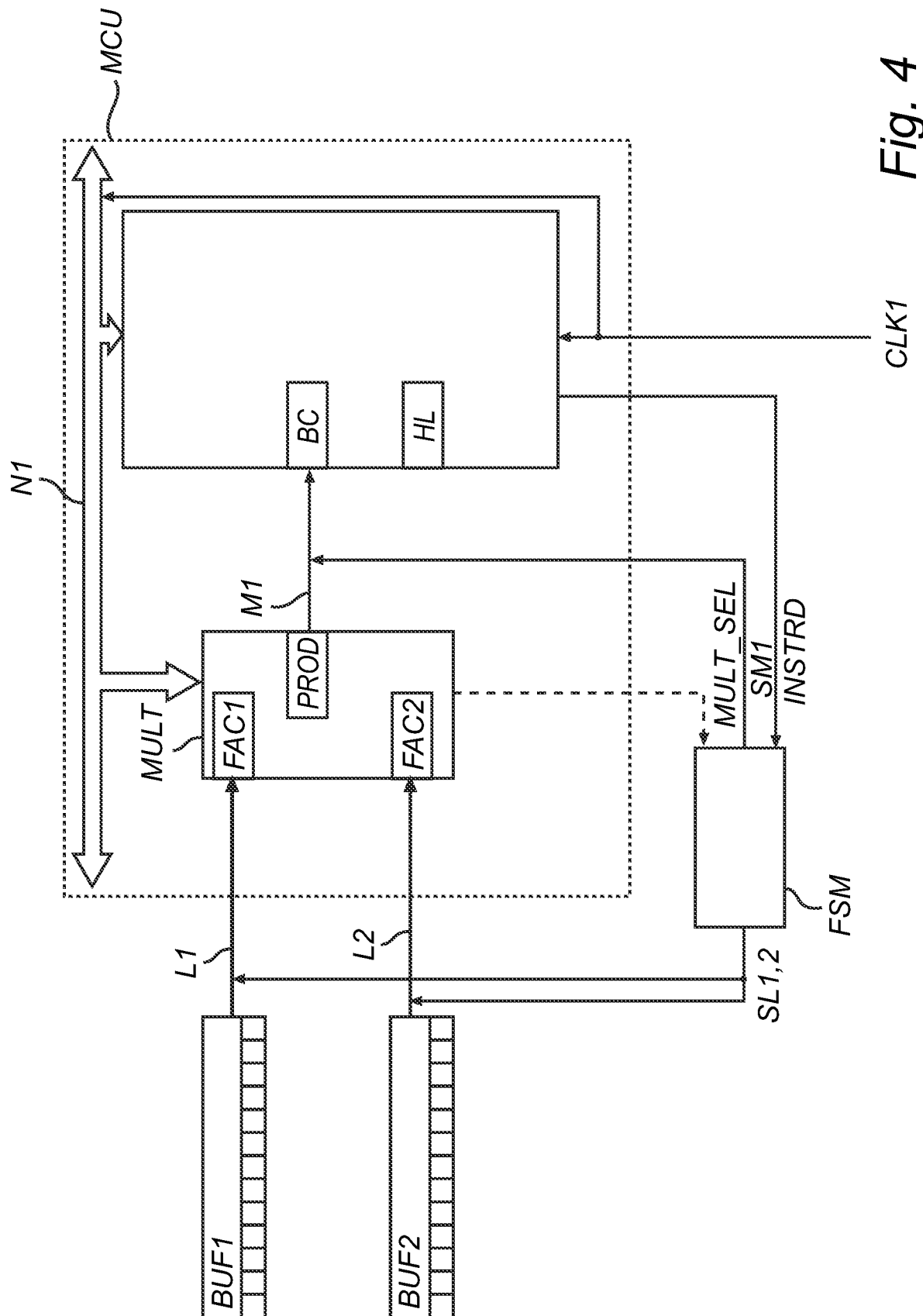
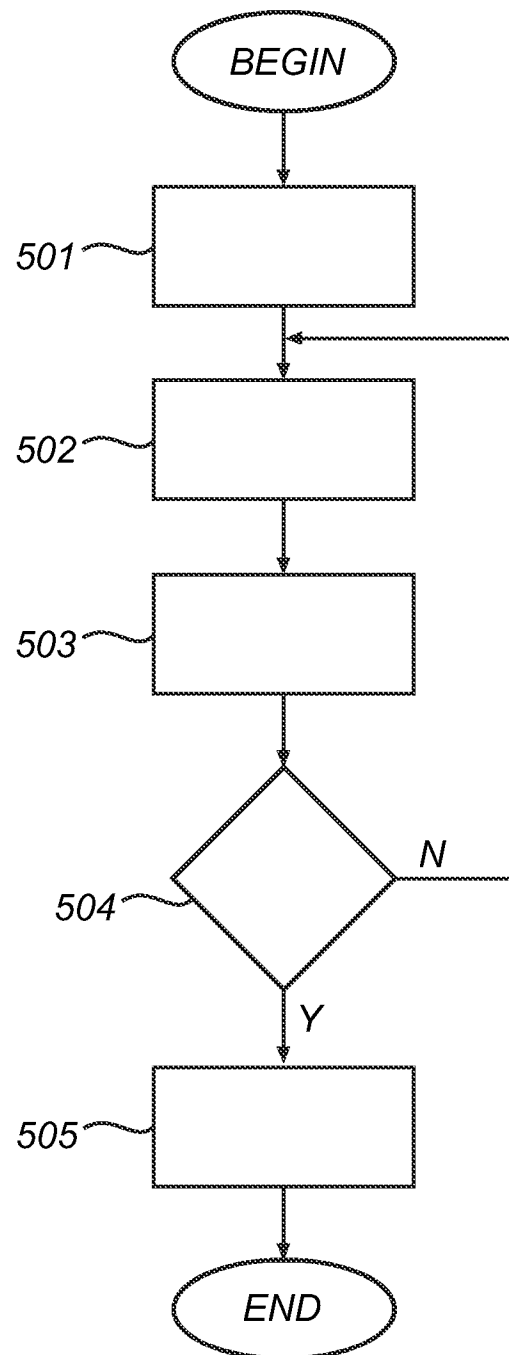


Fig. 4

5/8

*Fig. 5*

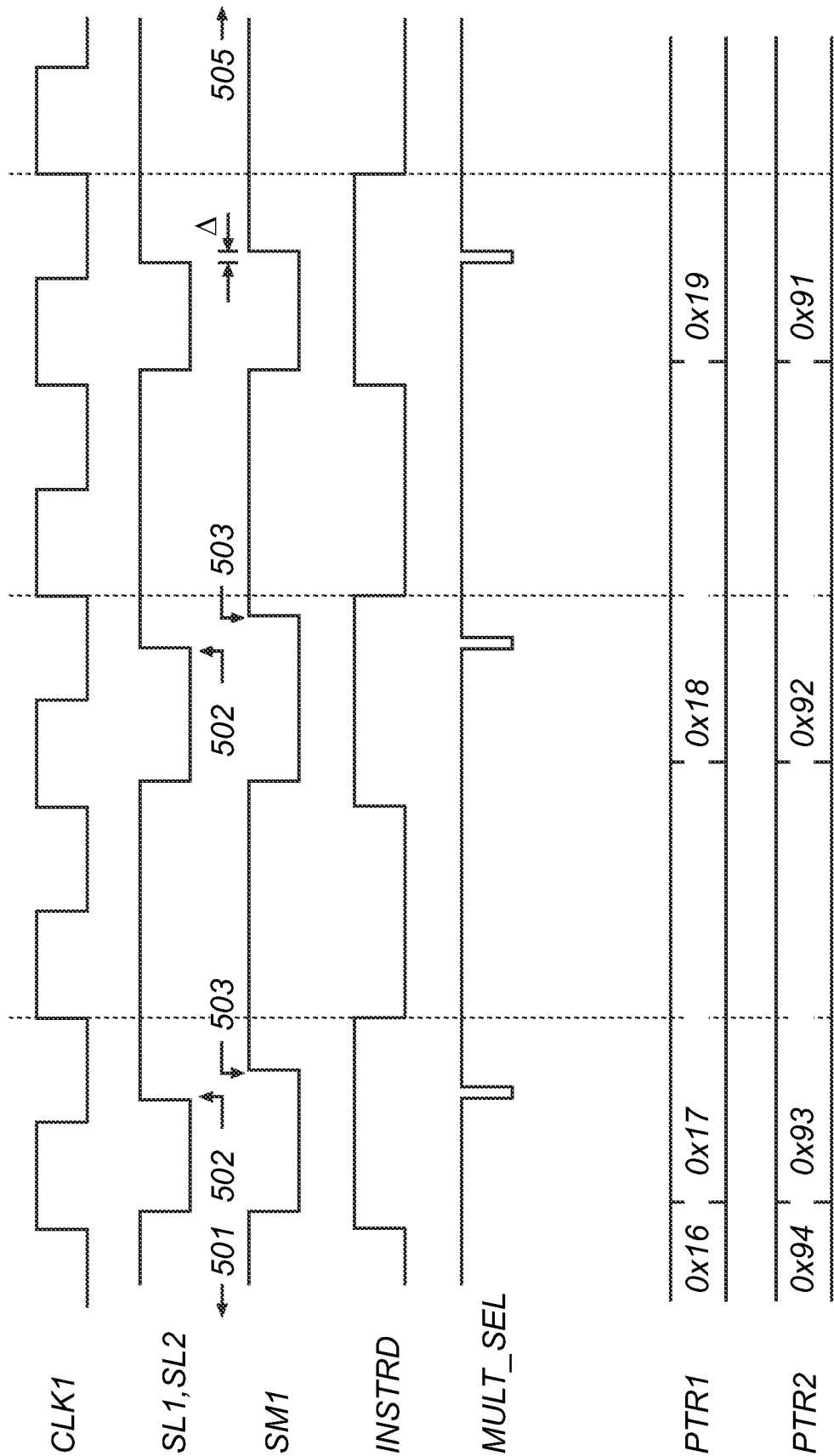


Fig. 6

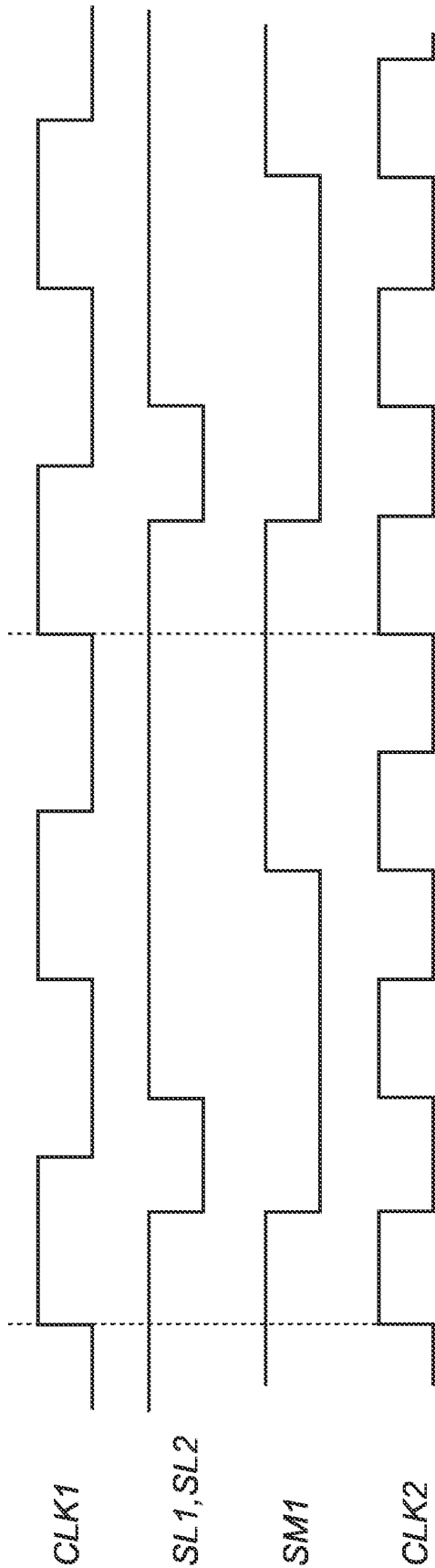


Fig. 7a ^{7/8}

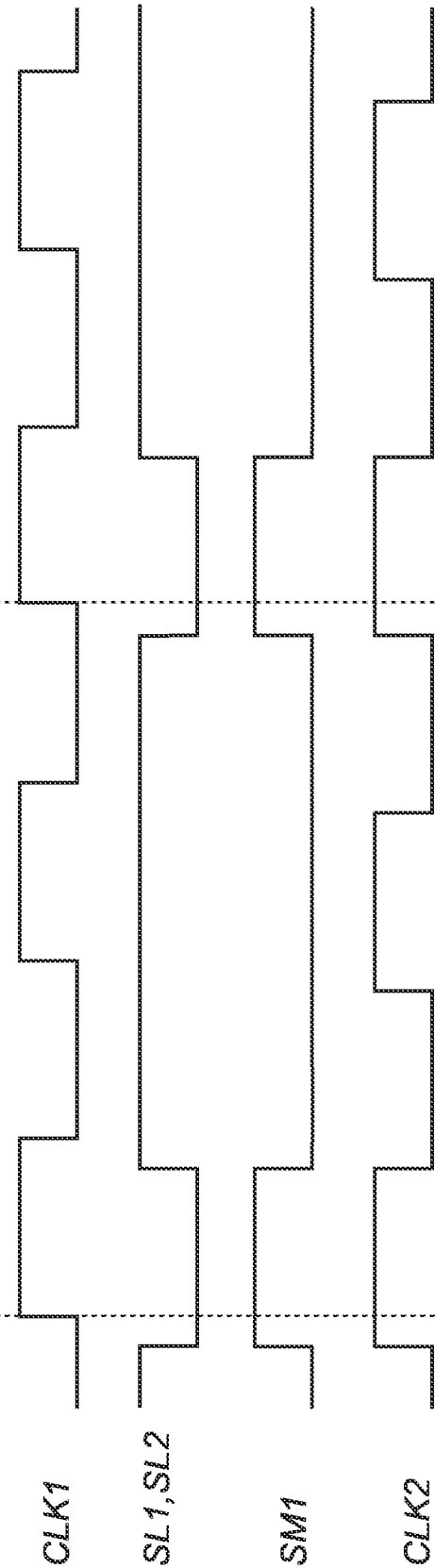


Fig. 7b

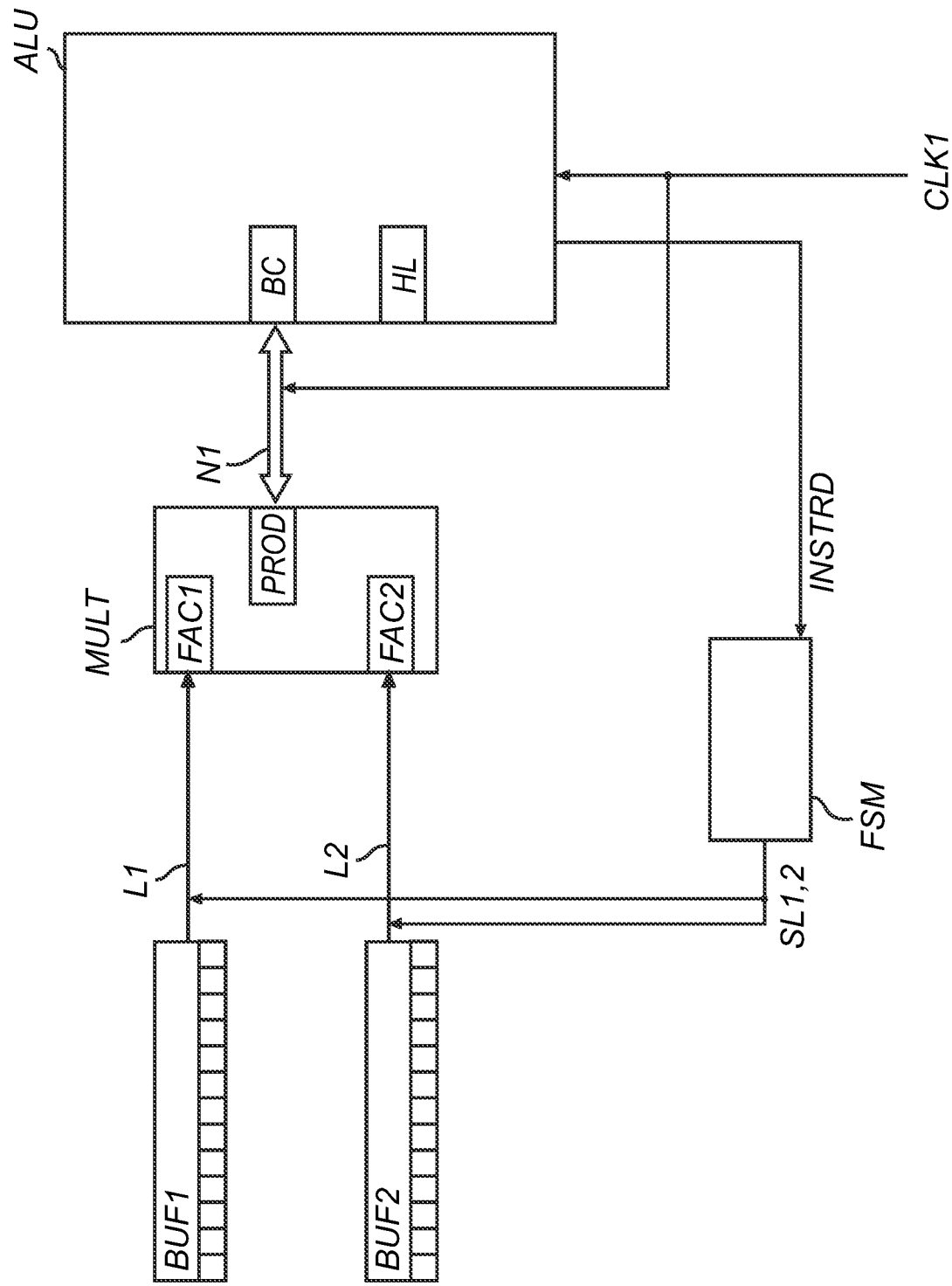


Fig. 8

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2011/055963

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F9/302 G06F7/544 G06F5/10 G06F9/38
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 698 971 A2 (HITACHI GLOBAL STORAGE TECH [NL]) 6 September 2006 (2006-09-06)	1-9, 11-17, 20-29, 31-38,41
Y	paragraph [0016] paragraph [0034] - paragraph [0035] paragraph [0037] - paragraph [0044] -----	10,18, 30,39
X	US 2006/200732 A1 (DOBBEK JEFFREY J [US] ET AL) 7 September 2006 (2006-09-07) the whole document -----	1,2,14, 20,21, 35,41
A	US 5 659 700 A (CHEN HWANG-CHUNG [TW] ET AL) 19 August 1997 (1997-08-19) abstract; figure 2 ----- -/-	9,29



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

16 January 2012

Date of mailing of the international search report

07/02/2012

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Daskalakis, T

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2011/055963

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2007/048133 A2 (QUALCOMM INC [US]; PLONDKE ERICH [US]; CODRESCU LUCIAN [US]; AHMED MUH) 26 April 2007 (2007-04-26) paragraph [0045] - paragraph [0054] -----	9,29
Y	US 6 282 631 B1 (ARBEL YGAL [US]) 28 August 2001 (2001-08-28) figures 9,12 -----	10,18, 30,39
A	Sthephen B. Furber: "VLSI RISC Architecture and Organization", 1989, Marcel Dekker, Inc., N. York, XP002667053, ISBN: 0-8247-8151-1 page 234 - page 237 page 253 - page 257 -----	10,18, 30,39

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/EP2011/055963

Patent document cited in search report		Publication date		Patent family member(s)		Publication date
EP 1698971	A2	06-09-2006	CN	1828724 A		06-09-2006
			EP	1698971 A2		06-09-2006
			US	2006195498 A1		31-08-2006

US 2006200732	A1	07-09-2006	NONE			

US 5659700	A	19-08-1997	US	5659700 A		19-08-1997
			US	5918252 A		29-06-1999

WO 2007048133	A2	26-04-2007	CA	2626684 A1		26-04-2007
			CN	101331449 A		24-12-2008
			EP	1941351 A2		09-07-2008
			JP	2009512942 A		26-03-2009
			KR	20080072852 A		07-08-2008
			US	2007094478 A1		26-04-2007
			WO	2007048133 A2		26-04-2007

US 6282631	B1	28-08-2001	NONE			

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-8, 11-17, 20-28, 31-38, 41

Apparatus and method for performing repetitive multiply-accumulate operations, wherein the transfer of intermediate products and accumulation are performed in one-to-one alternation

2. claims: 9, 29

Apparatus for performing repetitive multiply-accumulate operations with buffer logic comprising read pointer, modifier and length registers

3. claims: 10, 18, 30, 39

Apparatus and method for performing repetitive multiply-accumulate operations, wherein the Arithmetic and Logic Unit which performs accumulation is a Z80 architecture.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

Continuation of Box II.2

Claims Nos.: 19, 40

Inasmuch as independent claim 14 seeks protection for a method of operating a microprocessor such as that defined in claim 1, it appears that claims 19 and 40 seek protection for a program simulating the same processor on any computer. However, the description merely teaches a method of operating a processor with specific architectural features to support this method of operation, such as a Finite States Machine to control data flow buffers and buses with particular adaptations. The description remains silent as to how the method steps of claim 14 could manifest themselves in a processor the architecture of which does not include at least the above mentioned processor adaptations. Although simulating a processor architecture is well known in the art, it cannot be said that performing the above mentioned method steps in a general processor which does not comprise means adapted to this effect is trivially different. Nor could it be said that a skilled person aware of the teachings of the present application and endowed with the general knowledge in the art of computer architectures could carry out the invention of claims 19 and 40 without undue experimentation. Therefore, the description does not disclose the invention as claimed in claims 19 and 40 in such terms that the solution to the technical problem can be understood (Rule 5.1(iii) PCT) and thus claims 19 and 40 do not fulfil the requirements of Article 5 PCT.

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guideline C-VI, 8.2), should the problems which led to the Article 17(2) declaration be overcome.