(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 March 2005 (10.03.2005)

PCT

(10) International Publication Number
WO 2005/022379 A2

(51) International Patent Classification[7]: G06F 9/30, 1/32

(21) International Application Number:
PCT/IB2004/051555

(22) International Filing Date: 24 August 2004 (24.08.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/498,301 28 August 2003 (28.08.2003) US

(71) Applicant *(for all designated States except US)*: **KONINKLIJKE PHILIPS ELECTRONICS, N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventors; and

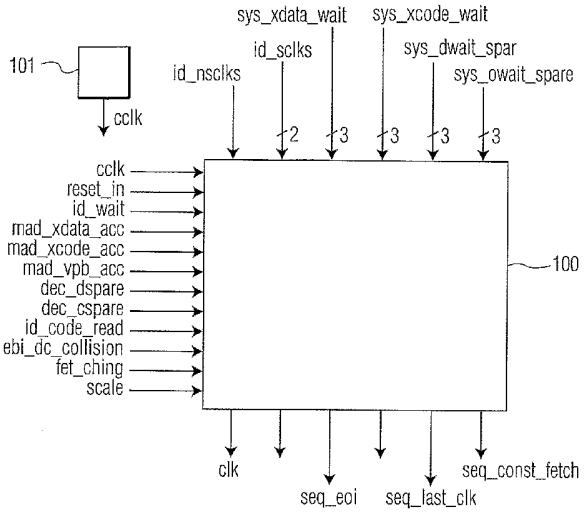
(75) Inventors/Applicants *(for US only)*: **GOODHUE, Greg** [US/US]; P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US). **KHAN, Ata** [US/US]; P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US). **SHRIVASTAVA, Pankaj** [IN/US]; P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US). **DING, Zhimin** [US/US]; P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US). **MACKENNA, Craig** [US/US]; P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US).

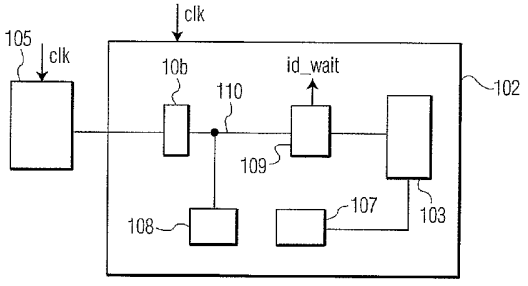
(74) Common Representative: **KONINKLIJKE PHILIPS ELECTRONICS, N.V.**; c/o Waxler, Aaron, P.O. Box 3001, Briarcliff Manor, NY 10510-8001 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM,

[Continued on next page]

(54) Title: VARIABLE INPUT PHASE FOR A MICROCONTROLLER INSTRUCTION





(57) Abstract: Typically, for instructions of any length, strobing of input data is performed during every clock cycle (201, 210) even when data is not yet available on a data bus (110). For some instructions, more than one clock cycle (201, 210) is required in order to provide the data to the data bus (110), thus, unnecessary strobing wastes electrical power. The embodiment of the invention serves to overcome this limitation by only strobing of input data when the data is ready on the input bus (110), thus providing a variable input phase. The strobing of the input data is dependent upon a characteristic of the instruction, such as instruction type, an address mode of the instruction, an internal access, an external access, and bus width for utilization by a microcontroller (102) in executing of the instruction. Thus, instructions having different characteristics have different input phases.

WO 2005/022379 A2

TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Variable Input Phase for a Microcontroller Instruction

The invention relates to the field of microcontrollers and more specifically to the field of processing of instruction data.

For reduced instruction set (RISC) processing systems, a large number of the reduced instructions require only a single clock cycle for processing thereof. For complex instruction set computer (CISC) type processing systems, many of the complex instructions require more than a single clock cycle for processing thereof. For both reduced instruction set and complex instruction set processing systems, the instructions have an input phase and an output phase. During an input phase, data is read for the instruction and during an output phase data is provided on an output bus as a result of execution of the instruction. In loading of data for the instruction, input data is strobed with a strobe signal in order to latch data from a data bus into the data input register. For typical CISC type processing systems, the input phase for instructions varies greatly from instruction to instruction. For RISC type processing systems, the input phase variation from instruction to instruction is small.

For one clock cycle instructions, during a half clock cycle thereof, the processor loads the data from a data bus by strobing the input data for latching instruction data from the data bus. During a subsequent half clock cycle the processor performs operations in accordance with the single cycle instruction data and provides output data as required. For two clock cycle instructions, such as memory access instructions, the instruction is executed during two clock cycles, where at least the first clock cycle is used to read data into the processor and the second clock cycle, or a portion thereof, is used for executing of the instruction. For the two clock cycle instruction, the input data is strobed every clock cycle, even when the data has not yet stabilized on the data bus. For a three clock cycle instruction, the input data is strobed three times. However, for instructions having various instruction lengths, data is not typically stabilized on the data bus prior to a predetermined amount of time greater than a half clock cycle. Some instructions require longer setup times than others and so repeatedly strobing results in unnecessary power consumption for implementation of latching operations.

A need therefore exists to reduce an amount of latching that takes place in processing systems for instructions that require longer than a single clock cycle to execute. It is therefore an object of the invention to provide a microcontroller architecture that overcomes the above limitations.

In accordance with the invention there is provided a method comprising the steps of: providing a clock circuit for generating a clock signal having clock signal transitions; providing an instruction to a microcontroller for processing thereof during an instruction processing time; decoding the instruction to determine at least a characteristic associated therewith; determining an input phase of the instruction in dependence upon the at least a characteristic associated therewith, the input phase having a number of clock signal transitions associated therewith; and, latching input data for use by the microcontroller upon occurrence of the number of clock signal transitions during the instruction processing time and other than latching of the input data during the input phase at or proximate each alternate clock signal transition during the instruction processing time.

In accordance with the invention there is provided a microcontroller architecture comprising: a data input bus; a clock circuit for generating a clock signal having a clock cycle denoted by clock signal transitions; an instruction decoding block for receiving the clock signal and for receiving data relating to a current instruction and for decoding of the data relating to the current instruction and for determining a characteristic of the current instruction and an instruction processing time; a data input register coupled to the data input bus for receiving input data therefrom and for storing of the input data therein for use in execution of the current instruction; a processor coupled to the data input register for processing of the input data stored therein; and, a sequencer block for receiving the clock signal and for determining an input phase of the current instruction and for generating of a strobe signal, wherein latching of the input data into the data input register is performed upon occurrence of a number of clock signal transitions during the instruction processing time and other than latching of the input data is performed proximate each alternate clock signal transition during the instruction processing time.

In accordance with the invention there is provided a storage medium having data stored thereon, the data for implementation of a processing system comprising: first instruction

data for providing an instruction to a microcontroller for processing thereof during an instruction processing time; second instruction data for decoding the instruction to determine at least a characteristic associated therewith; third instruction data for determining an input phase of the instruction in dependence upon the at least a characteristic associated therewith, the input phase having a number of clock signal transitions associated therewith; and, fourth instruction data for latching input data for use by the microcontroller upon occurrence of the number of clock signal transitions during the instruction processing time and other than latching of the input data during the input phase at or proximate each alternate clock signal transition during the instruction processing time.

Exemplary embodiments of the invention will now be described in conjunction with the following drawings, in which:
FIG. 1 illustrates a microcontroller architecture that includes a sequencer block and an instruction decoding block in accordance with a first embodiment of the invention;
FIG. 2a illustrates a timing diagram for a single clock cycle instruction that executes within a single clock cycle of a clock signal (clk);
FIG. 2b illustrates a timing diagram for a two clock cycle instruction that executes within two clock cycles of the clock signal (clk);
FIG. 2c illustrates a timing diagram for a three clock cycle instruction that executes within three clock cycles of the clock signal (clk) and also includes a single clock cycle instruction, in accordance with that shown in FIG. 2a, following the three clock cycle instruction; and,
FIG. 3 summarizes processing steps that occur for the microcontroller architecture in processing of an instruction to determine the input phase thereof.

FIG. 1 illustrates a microcontroller architecture that includes a sequencer block 100 in accordance with a first embodiment of the invention. The sequencer block 100 receives an oscillator clock signal (clk), from an oscillator circuit 101 and generates a clock signal (clk), having a clock cycle with clock signal transitions, for provision to circuitry making up a MX1 core 102. The MX1 core 102 comprises a processor 103 an address decoder (not shown). External to the MX1 core 102 is a program memory 105 for storing of instruction data. A fetching circuit 106 is disposed within the MX1 core 102 for fetching of data from the program memory 105. A very large scale integration peripheral bus bridge (VBP

bridge) 107 is coupled to the processor 103. A data input register 108 is also provided within the MX1 core 102 and coupled to the data input bus 110 for receiving input data therefrom and for being responsive to a strobe signal. Input signals and output signals provided to and from the sequencer block 100 are summarized in Table 1.

Name	Direction	Width	Function
id_wait	IN	1	Indicates when the Instruction decoding block does not have a complete instruction and the sequencer stalls during this time.
id_nsclks	IN	1	Indicates the number of clock cycles for the current instruction, with no wait states and without scaled timing.
id_sclks	IN	2	Indicates the number of clock cycles for the current instruction with scaled timing.
sys_xdata_wait	IN	3	The number of clock cycles to wait for an external data access (0 = no wait)
sys_xcode_wait	IN	3	The number of clock cycles to wait for an external code memory access (0 = no wait)
sys_dwait_spare	IN	3	Spare (un-dedicated) wait value for data.
sys_cwait_spare	IN	3	Spare (un-dedicated) wait value for code.
mad_xcode_acc	IN	1	Address decoder identifies the target address as an external code access.
mad_xdata_acc	IN	1	Address decoder identifies the target address as an external data access.
mad_vpb_acc	IN	1	Address decoder identifies the target address as a peripheral bus bridge (VPB) access.
dec_dspare	IN	1	Indicates when an access occurs that requires activating a sys_dwait_spare wait inputs.
dec_cspare	IN	1	Indicates when an access occurs that requires activating a sys_cwait_spare wait inputs.
id_code_read	IN	1	Flags the current instruction as a MOVC or EMOV A, @PRi+data2. This tells the sequencer how to construct the seq_const_fetch output.
fet_ching	IN	1	Indicates when the fetching circuit is fetching.
ebi_dc_collision	IN	1	Indicates when the External Bus Interface has a simultaneous code read and data read request (MOVX).
scale	IN	1	Tells the sequencer whether to scale instruction timing to MX1 core equivalent.
cclk	IN	1	Oscillator clock input signal, received only by the sequencer.
reset_in	IN	1	Chip reset input.
clk	OUT	1	Clock signal for circuitry making up the MX1 core.
reset	OUT	1	Processed Reset output to the rest of the core.
seq_eoi	OUT	1	End Of Instruction flag. Asserted during the last half of the last clock cycle of an instruction.
seq_last_clk	OUT	1	Identifies the last clock cycle of an instruction, true for the entire clock cycle.
seq_const_fetch	OUT	1	This signal is used for two purposes. At a ALU Source Register, it is used to capture read data at the rising edge of clk during the MOVC or EMOV A,@PRi+data2 instructions, in order to allow the sufficient code memory access time. It is also used by an address calculation unit (ACU) to distinguish between a code memory data read and a code fetch operation.

Table 1: Input and output signals provided to and from the sequencer block.

The instruction decoding block 109 is coupled to a data input bus 110 and decodes each instruction to determine a characteristic associated therewith. In dependence upon the characteristic of the instruction, the sequencer block 100 adds wait states to a strobe signal generated by the sequencer block so that the sequencer block does not strobe data for use by the processor 103 until instruction data is ready on an input bus. For a last half clock cycle of each instruction, seq_eoi is asserted, regardless of how long each instruction takes to execute and asserting of seq_eoi includes wait states. Seq_last_clk is used to identify an entire last clock cycle of an instruction.

The instruction decoding block 109 generates an instruction length in clock signal transitions for the current instruction being executed. This instruction length is determined for every instruction during an instruction decoding phase, which precedes the input phase of the instruction. The instruction length takes into account the type of instruction, its address modes, and whether the instruction is for internal access or external access, such as to the program memory 105. Bus width utilization is also taken into account when determining of the instruction length. In dependence upon the instruction length as determined by the instruction decoding block 109, the sequencer 100 generates signals designating the input phase and the output phase for the given instruction. The strobe signal is dependent upon the signals designating the input phase and the output phase for the given instruction.

A wait input signal (id_wait) from the instruction decoding block 109 allows the sequencer block 100 to stall operation thereof while additional instruction fetches are performed by the fetching circuit 106 in order to complete processing of the current instruction. If id_wait is asserted at the beginning of instruction execution, the sequencer block 100 waits until it is de-asserted before beginning to count clock signal transitions dependent upon wait times related to program memory 105 access.

For MOVC and EMOV instructions that access program memory 105, the sequencer block 100 counts external code access wait states. Additionally, the sequencer block 100 also counts external code access wait states when the fetching circuit 106 is performing instruction fetches. Seq_const_fetch denotes a difference between a code fetch and a data

access to program memory 105, as well as defining when data from program memory 105 is to be stored in the ALU Source Register (not shown).

Referring to FIGs. 2a, 2b and 2c, for each given instruction, a number of clock cycles are dynamically computed for the input phase thereof. The dynamic computation of clock cycles is a result of an instruction type and a memory latency associated with an instruction of that type. The duration of an input phase of instructions used by a microcontroller is computed for every instruction based on a latency of any memory access as well as in dependence upon the nature of the instruction. Computation of the length of the input phase is performed by the instruction decoding block 109. Although FIGs 2a, 2b and 2c illustrate a same clock cycle for cclk 210 and clk 201, this is not always the case. In some instances clk 201 is varied as determined by the sequencer block 100.

FIG. 2a illustrates a timing diagram for a single clock cycle instruction that executes within a single clock cycle 202a of clk 201. The single clock cycle comprises two clock signal transitions and ends on a third clock signal transition. During this single clock cycle 202a, the source address signal (id_src_adr) 203a and the destination address signal 204a (id_dest_adr) are generated as shown. At a halfway point 212a during the clk 201, at the second clock signal transition, the strobe signal is asserted. Upon asserting of the strobe signal data is latched and is valid during a period of time between a transition 215a in the alu_reg 205a and a transition 216a in the alu_src 206a, after the input phase. The input phase of the instruction illustrated in FIG. 2a is half a clock cycle with an output phase also having half a clock cycle. After the input phase of the instruction is over, at the halfway point 212a, seq_eoi 207a is asserted during the last half clock cycle of the single cycle instruction. A seq_last_clk 208 is asserted to identify an occurrence of an entire last clock cycle of an instruction. For the four single clock cycle instructions shown in FIG. 2a, seq_last_clk 208a is always asserted. Though the present invention is described with synchronous clocking of the input phase, asynchronous clocking of the input phase is also within the scope of the invention.

FIG. 2b illustrates a timing diagram for a two clock cycle instruction that executes within two clock cycles 202b of the clock signal 201(clk). The two clock cycles comprise four clock signal transitions and end on a fifth clock signal transition. During these two clock

cycles 202b, the source address signal (id_src_adr) 203a and the destination address signal (id_dest_adr) 204a are generated as shown. At a point in time 212b during the clk 201, the strobe signal is asserted at the end of the input phase in order to latch input data. The input data is valid during a period of time between a transition 215b in the alu_reg and a transition 216b in the alu_src. The input phase of the instruction illustrated in FIG. 2b is one and a half clock cycles with an output phase remaining unchanged, having a length of half a clock cycle. After the input phase of the instruction is over, seq_eoi 207b is asserted during the last half clock cycle of the two clock cycle instruction. For identifying an occurrence of an entire last clock cycle of an instruction, seq_last_clk 208b is asserted. For the two clock cycle instruction, the input phase is typically one and a half clock cycles in length, when a termination thereof occurs at clock signal transition 212b and the output phase is a half a clock cycle in length - unchanged from the single clock instruction. Two clock cycle instructions are of two different types, as denoted by seq_const_fetch 209. The seq_const_fetch 209 is used to denote a difference between a code fetch instruction and a data access instruction to program memory 105, as well as for defining when data from program memory 105 is to be stored in the ALU source register (not shown). When data is to be read from program memory 105, such as shown for the two clock cycle instruction 202d, seq_const_fetch is asserted and provided to the fetching circuit 106 for use in instruction data fetching operations from program memory 105.

FIG. 2c illustrates a timing diagram for a three clock cycle instruction that executes within three clock cycles 202c of the clock signal 201 (clk) 201. The three clock cycles comprise six clock signal transitions and end on a seventh clock signal transition. FIG. 2c also illustrates a single clock cycle instruction, as shown in FIG. 2a, following the three clock cycle instruction. During these three clock cycles 202c, the source address signal (id_src_adr) 203a and the destination address signal 204a (id_dest_adr) are generated as shown. At clock signal transition 212c during the clk 201, at the end of the input phase, input data is strobed with the assertion of the strobe signal and input data from the data bus 110 is latched. Data on the input data bus 110 is valid during a period of time between a transition 215c in the alu_reg and a transition 216c in the alu_src. The input phase of the instruction illustrated in FIG. 2c is two and a half clock cycles in duration with an output phase thereof having half a clock cycle. After the input phase of the instruction is over, at clock transition 212c, seq_eoi 207c is asserted during the last half clock cycle of the three

clock cycle instruction. In order to identify an occurrence of an entire last clock cycle of an instruction, seq_last_clk 208c is asserted on the fifth clock cycle transition. For the three clock cycle instruction, the input phase is two and a half clock cycles in length and the output phase is a half a clock cycle in length, unchanged from the single clock cycle and two clock cycle instructions.

FIG. 3 summarizes processing steps that occur for the microcontroller architecture in processing of an instruction to determine the input phase thereof. In a first step 301, an instruction is provided to a microcontroller for processing thereof during an instruction processing time. The instruction is then decoded in order to determine at least a characteristic associated therewith, in step 302. The input phase of the instruction is then determined, in step 303, in dependence upon the at least a characteristic associated therewith, where the input phase has a number of clock signal transitions associated therewith. Referring to step 304, input data is for use by the microcontroller upon occurrence of the number of clock signal transitions during the instruction processing time and other than latching of the input data during the input phase at or proximate each alternate clock signal transition during the instruction processing time. The processing steps as outlined in FIG. 3 are exemplified in FIGs. 2a, 2b and 2c for the one clock cycle, two clock cycle and three clock cycle instructions.

Since most instructions execute in one clock cycle, the input phase normally occurs during the first half of the clock cycle and the output phase during the second half of the same clock cycle. Data used for the output phase of the instruction is stored in flip-flops at the end of the input phase. For the cases where multiple clock cycles are used to execute instructions, such as the case where at least one of program memory access, bus multiplexing and wait states occur, which utilize several clock cycles, the instruction is considered to have a variable input phase and a fixed duration output phase. The duration of the output phase is preferably kept constant for all instructions and is the same as for a single clock cycle instruction.

Preferably, clk is suppressed during reset operations in a synchronous fashion, in order to prevent clock glitches. At least one full cycle of clk occurs before reset signal is de-asserted. Seq_const_fetch is normally asserted during the first clock cycle of a two clock

cycle instruction such as for MOVC or EMOV. Since seq_const_fetch is generated in response to a decoded input instruction, there is a possibility of glitches on this signal at the beginning of the instruction. Further preferably, if memory wait states are implemented for a given instruction being executed, then the number of clock cycles for the input phase are automatically included in the computation of the variable length input phase.

The above embodiments of the invention advantageously allow for a considerable power savings and simplify design of the sequencer block by only latching input phase data at the variable end of the input phase - strobing once, on a single clock signal transition - instead of strobing on multiple clock signal transitions occurring during the input phase. Input data is latched upon having a number of clock signal transitions occur and other than strobed during the input phase at or proximate each alternate prior clock signal transition as is the case in the prior art.

Thus, considerable power savings are realized by supporting variable length input phase. The length of the input phase is determined for each instruction and input data is only latched at the end of the input phase regardless of the number of clock cycles the input phase may require. Thus, input flip flops are only strobed once per input phase.

Numerous other embodiments may be envisaged without departing from the spirit or scope of the invention.

Claims

1. A method comprising the steps of:
providing a clock circuit (101) for generating a clock signal (201, 210) having clock signal transitions;
providing (301) an instruction to a microcontroller (102) for processing thereof during an instruction processing time;
decoding (302) the instruction to determine at least a characteristic associated therewith;
determining (303) an input phase of the instruction in dependence upon the at least a characteristic associated therewith, the input phase having a number of clock signal transitions associated therewith; and,
latching (304) input data for use by the microcontroller (102) upon occurrence of the number of clock signal transitions during the instruction processing time and other than latching of the input data during the input phase at or proximate each alternate clock signal transition during the instruction processing time.

2. A method according to claim 1 comprising the step of counting a number of alternate clock signal transitions occurring during the instruction processing time in order to determine a time to performing the step of latching.

3. A method according to claim 1 comprising the step of counting a number of clock signal transitions occurring during the instruction processing time in order to determine a time to performing the step of latching.

4. A method according to claim 3 wherein the step of counting is performed for each instruction prior to performing the step of latching.

5. A method according to claim 1 wherein the input phase is terminated (215a; 215b; 215c) upon having the number of clock signal transitions occur and upon performing the step of latching.

6. A method according to claim 1 wherein the characteristic of the instruction is at least one of an instruction type, an address mode of the instruction, an internal access and an external access.

7. A method according to claim 1 wherein the characteristic of the instruction is dependent upon a bus width for utilization by the microcontroller (102) in execution of the instruction.

8. A method according to claim 1 wherein latching is other than performed prior to having the number of clock signal transitions occur during the instruction processing time.

9. A method according to claim 1 wherein for a two clock cycle instruction (202b, 202d) the input phase of the instruction is approximately one and a half clock cycles in length.

10. A method according to claim 1 wherein for a three clock cycle instruction (202c) the input phase of the instruction is approximately two and a half clock cycles in length.

11. A method according to claim 1 wherein for instructions having different determined characteristics the number of clock signal transitions determined therefore is different.

12. A microcontroller architecture comprising:
a data input bus (110);
a clock circuit (101) for generating a clock signal (201, 210) having a clock cycle denoted by clock signal transitions;
an instruction decoding block (109) for receiving the clock signal (201, 210) and for receiving data relating to a current instruction and for decoding (302) of the data relating to the current instruction and for determining a characteristic of the current instruction and an instruction processing time;
a data input register (108) coupled to the data input bus (110) for receiving input data therefrom and for storing of the input data therein for use in execution of the current instruction;
a processor (103) coupled to the data input register (108) for processing of the input data stored therein; and,

a sequencer block (100) for receiving the clock signal (210) and for determining an input phase of the current instruction and for generating of a strobe signal, wherein latching (304) of the input data into the data input register (108) is performed upon occurrence of a number of clock signal transitions during the instruction processing time and other than latching of the input data is performed proximate each alternate clock signal transition during the instruction processing time.

13. A microcontroller architecture according to claim 1 comprising memory (105) external to the microcontroller (102) architecture, wherein the input phase of the instruction is dependent upon whether the current instruction comprises instruction data for accessing the external memory (105).

14. A microcontroller architecture according to claim 12 wherein the sequencer block (100) comprises circuitry for counting a number of clock signal transitions that occur during the instruction processing time in order to determine when to latch the input data into the data input register (108).

15. A microcontroller architecture according to claim 14 wherein the step of counting is performed for each instruction.

16. A microcontroller architecture according to claim 12 wherein the input phase is terminated upon generation of the strobe signal and upon latching of the data into the data input register (108).

17. A microcontroller architecture according to claim 12 wherein the characteristic of the instruction is at least one of an instruction type, an address mode of the instruction, an internal access and an external access.

18. A microcontroller architecture according to claim 12 wherein the characteristic of the instruction is dependent upon a bus width for utilization by the microcontroller (102) in execution of the instruction.

19. A microcontroller architecture according to claim 12 wherein for a two clock cycle instruction (202b, 202d) the input phase of the instruction is approximately one and a half clock cycles in length.

20. A microcontroller architecture according to claim 12 wherein for a three clock cycle instruction (202c) the input phase of the instruction is approximately two and a half clock cycles in length.

21. A microcontroller architecture according to claim 12 wherein for instructions having different determined characteristics the number of clock signal transitions determined therefore is different.

22. A storage medium having data stored thereon, the data for implementation of a processing system comprising:
first instruction data for providing (301) an instruction to a microcontroller (102) for processing thereof during an instruction processing time;
second instruction data for decoding (302) the instruction to determine at least a characteristic associated therewith;
third instruction data for determining (303) an input phase of the instruction in dependence upon the at least a characteristic associated therewith, the input phase having a number of clock signal transitions associated therewith; and,
fourth instruction data for latching (304) input data for use by the microcontroller (102) upon occurrence of the number of clock signal transitions during the instruction processing time and other than latching of the input data during the input phase at or proximate each alternate clock signal transition during the instruction processing time.

23. A storage medium according to claim 22 comprising fifth instruction data for providing a clock circuit (101) for generating a clock signal (201, 210) having clock signal transitions.

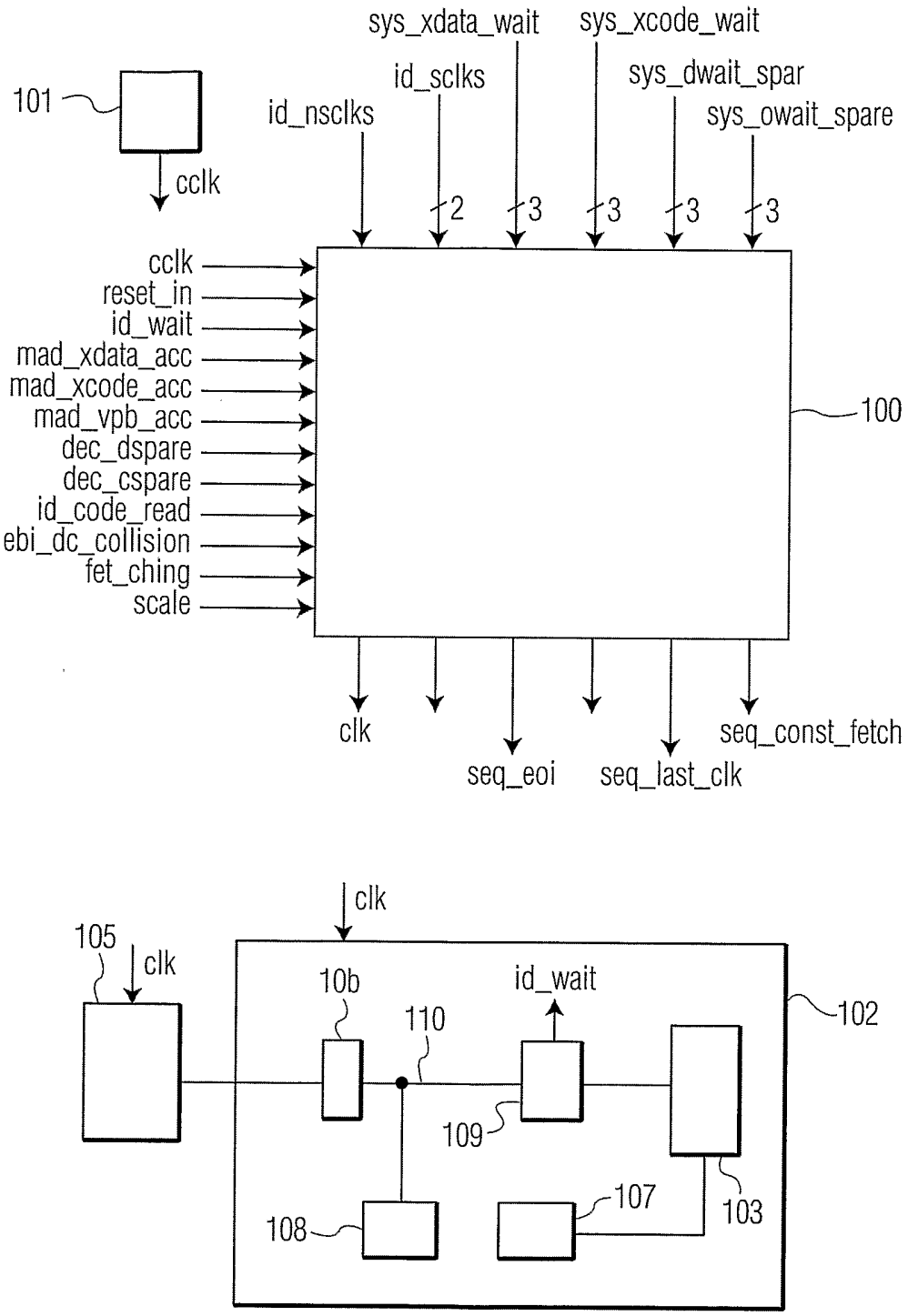

sys_xdata_wait
sys_xcode_wait
id_sclks
sys_dwait_spar
sys_owait_spare
101
id_nsclks
cclk
2
3
3
3
3
cclk
reset_in
id_wait
mad_xdata_acc
mad_xcode_acc
mad_vpb_acc
dec_dspare
dec_cspare
id_code_read
ebi_dc_collision
fet_ching
scale
100
clk
seq_eoi
seq_last_clk
seq_const_fetch
clk
105
clk
10b
id_wait
110
102
109
107
103
108

FIG. 1

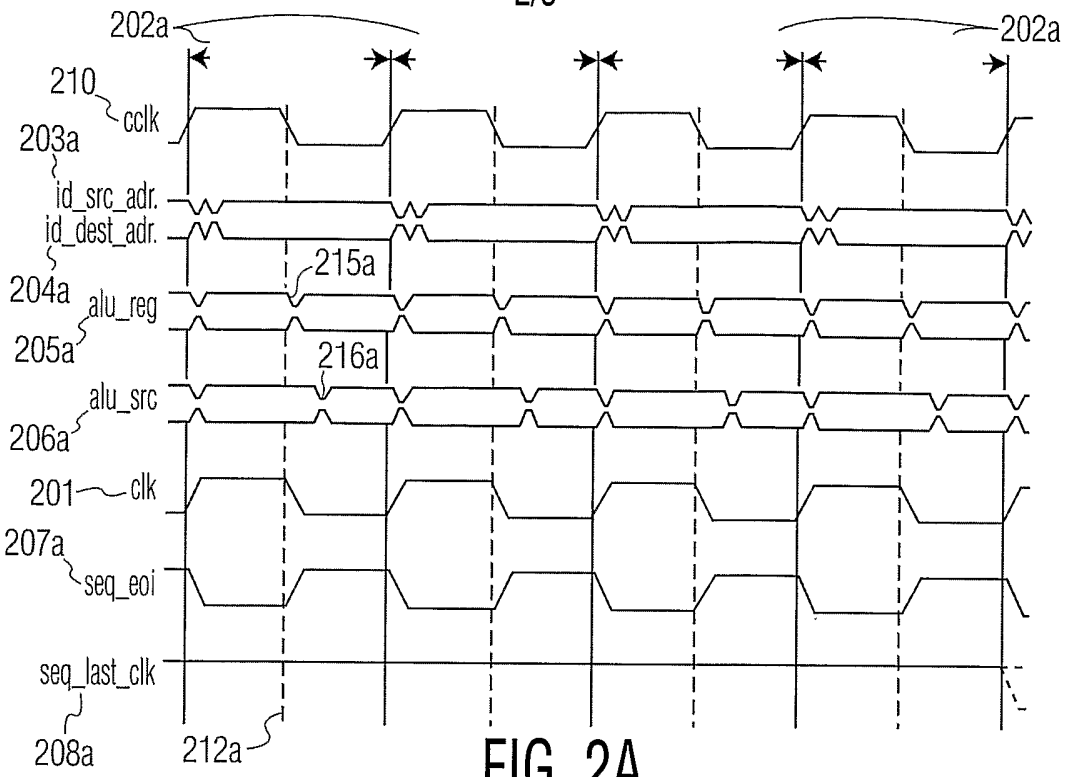

202a
202a
210
cclk
203a
id_src_adr.
id_dest_adr.
215a
204a
alu_reg
205a
216a
alu_src
206a
201
clk
207a
seq_eoi
seq_last_clk
208a
212a

FIG. 2A

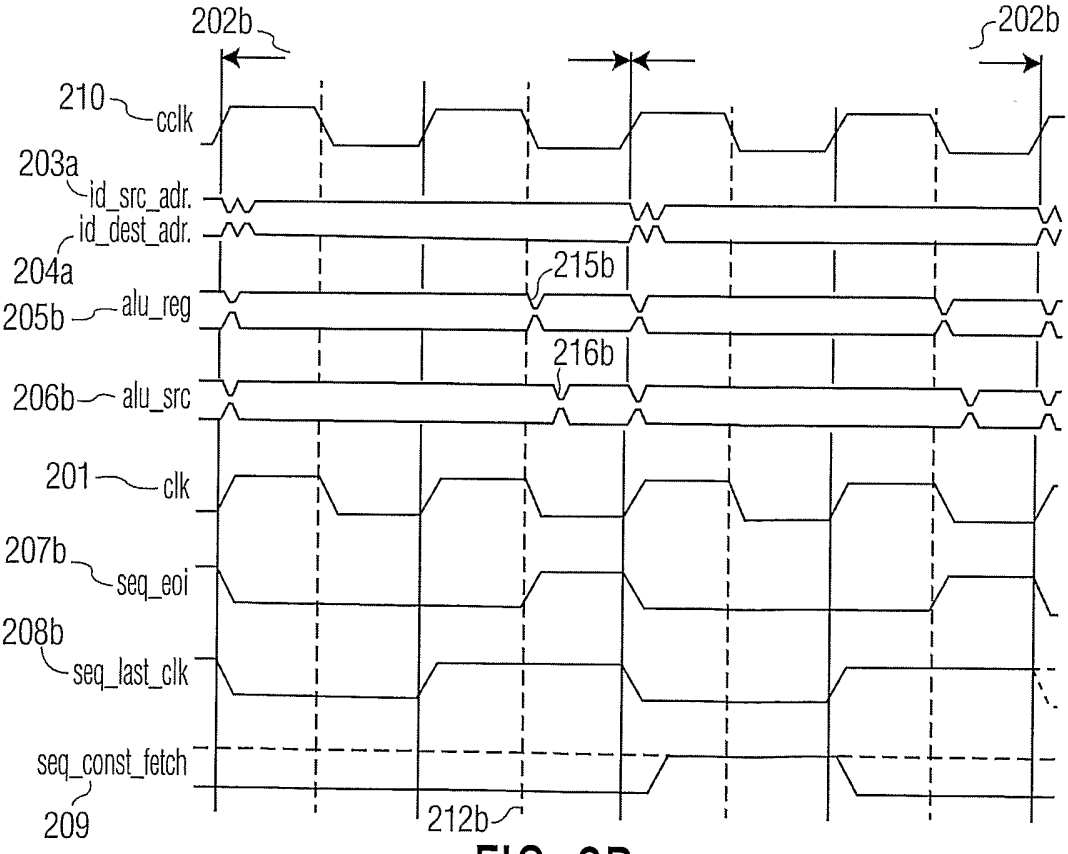

202b
202b
210
cclk
203a
id_src_adr.
id_dest_adr.
204a
215b
205b
alu_reg
216b
206b
alu_src
201
clk
207b
seq_eoi
208b
seq_last_clk
seq_const_fetch
209
212b

FIG. 2B

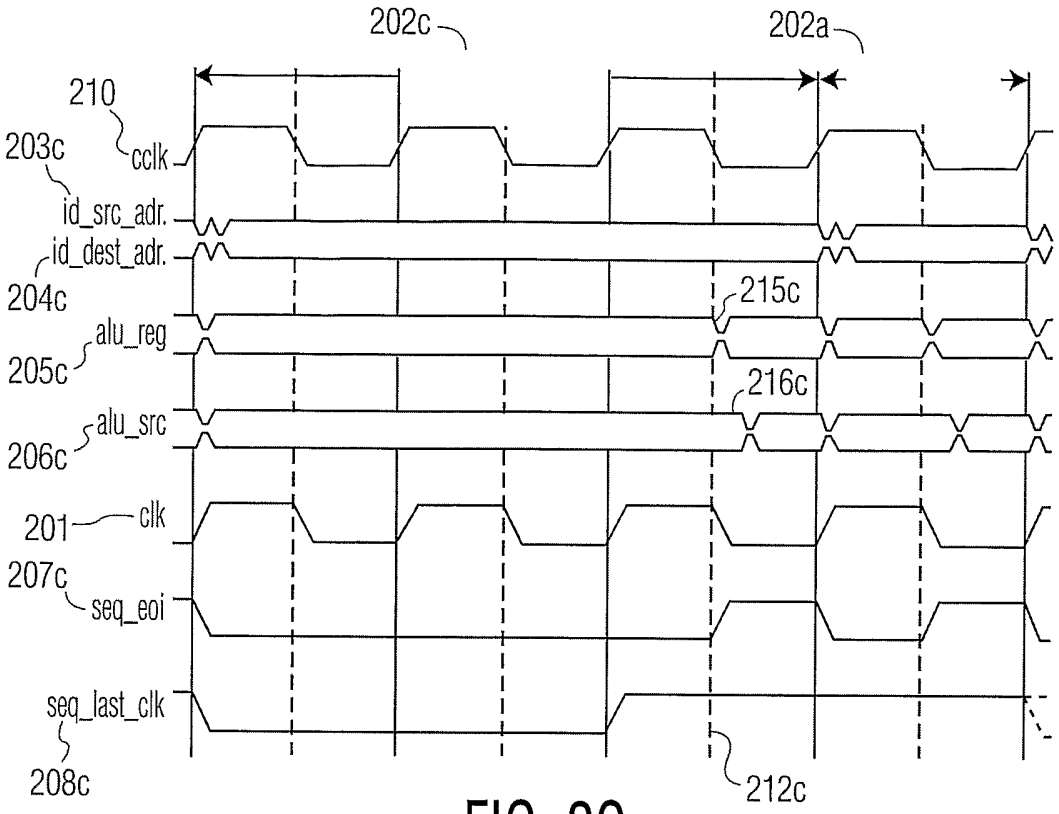

202c
202a
210
cclk
203c
id_src_adr.
id_dest_adr.
204c
215c
alu_reg
205c
216c
alu_src
206c
clk
201
207c
seq_eoi
seq_last_clk
208c
212c

FIG. 2C

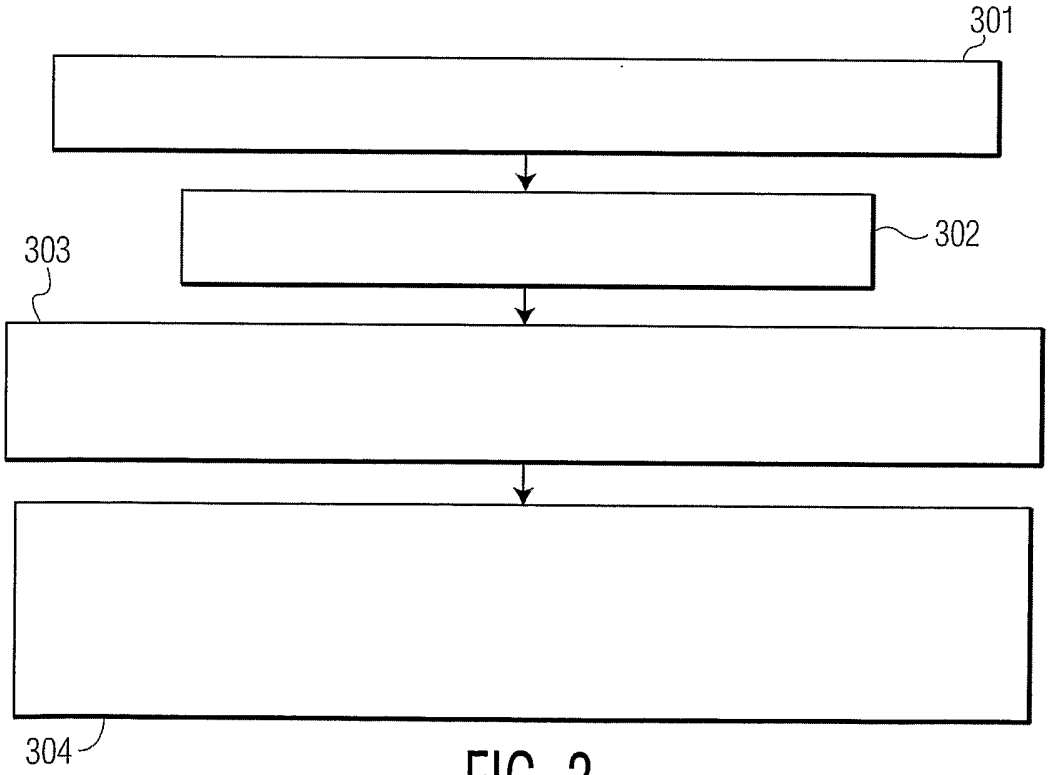

301
302
303
304

FIG. 3