762262

E-2/386/2021 DEL

APP NO: 202011024696

Docket No: 20569

01/03/2021

FORM 2

THE PATENTS ACT, 1970
(39 of 1970)

COMPLETE SPECIFICATION

(See section 10 and rule 13)

1. TITLE OF THE INVENTION

Novel Hardware Accelerator Circuit for Bit-Level Operations in a Microcontroller

2. APPLICANT(S)

1.

(a) NAME : Gulzar Singh
(b) NATIONALITY: India
(c) Correspondence Address: Electronic Science Department, Kurukshetra University, Kurukshetra, Haryana- 136119.
(d) Permanent Address: 39, Sec-2, Industrial Area, Kurukshetra, Haryana – 136118
(e) M: 9355828682
(f) E-mail: gulzarindri@gmail.com

2. (a) NAME: Anil Vohra
(b) NATIONALITY: India
(c) Correspondence Address: Electronic Science Department, Kurukshetra University, Kurukshetra, Haryana- 136119.
(d) Permanent Address: 301A, Sec-5, Kurukshetra, Haryana – 136118
(e)M:9355222388
(f) Email: vohra64@gmail.com

3. PREAMBLE TO THE DESCRIPTION

Complete Specification

The following specification describes invention

FORM 2
THE PATENTS ACT, 1970
(39 of 1970)
&
The Patents [Amendment] Rules, 2014
COMPLETE SPECIFICATION
(See Section 10 and Rule 13)

Novel Hardware Accelerator Circuit for Bit-Level Operations in a Microcontroller

Abstract: A novel hardware circuit has been invented by the authors to perform Selective Bitwise Immediate Logical Operation (herein after SBILO) on Byte sized data in a microcontroller. The unique novelity of the invented hardware circuit is that any of four logical operations, viz, (Set, Clear, Complement, No Change) can be performed concurrently with others in any combination, on bits of the register in a microcontroller. The logical operation to be performed is specified in the microcontroller instruction for each bit individually. SBILO circuit manipulates the logic of each bit concurrently as per bit operation code in the instruction. The hardware circuit invented by the authors takes only one machine cycle to perform concurrent logical operation on all eight bits of byte sized register. Firstly, the invented hardware speeds up the logical operation by using only one instruction instead of multiple bit or byte level instructions. Secondly, the invented hardware circuit reduces the program memory space consumption by taking only one instruction space for selective concurrent manipulation of up to eight bits.

Field of Invention

This invention relates to development of a novel circuit to perform logical operation on multiple bits in a single machine cycle in the microcontroller. For this, the architecture of the circuit has been designed and simulated using Verilog Hardware Description Language on the Xilinx ISE platform. The architecture has been implemented in a FPGA & practically tested. The invention relates to development of a circuit to speed-up the operations on the Bits in a microcontroller. Bit represents the Binary Logic in all digital circuits. Bit is an integral part of a larger data pack like Nibble, Byte etc. Bits are grouped to make a single larger data. Data storage element is called as a Register. During execution, Microcontroller changes or reads the Logic Level of any Bit in any Register according to the Instruction. There are three logic operations on any Bit (Set, Clear, Invert). Authors invented a circuit to speed-up the Bit Logic change. Further, the circuit designed by authors has features that the Bits involved in Logic change are selected through instruction.

Background of Invention

Microcontroller has a CPU with on chip Peripherals & Memory (Program Memory & Data Memory). The CPU has an integrated circuit namely Arithmetic Logic Unit (ALU). Numerous microcontrollers have been designed in past. Their ALU has different capabilities in terms of Arithmetic & Logical Instructions supported by it. As well, microcontrollers have difference in the number & types of peripherals implemented on the chip. Some microcontrollers have general purpose architecture, some application specific microcontrollers also have been

designed. The Mega Instructions Per Seconds (MIPS) is the performance measure unit of the execution speed of any microcontroller or microprocessor. To run any application, the program memory & data memory consumed, also, is an important factor in case of microcontrollers. Different Memory optimization techniques have been developed in past.

I. *Logic Level Operation*

The logic level in digital electronics has binary values, namely '1' and '0'. Usually, Voltage equal to Supply voltage at any node is marked as Logic '1' and voltage equal to Ground Voltage (Zero), is marked as Logic '0'. The logic at any node is either input or output of any digital component. Various digital components are Digital Gates (AND, OR, NOT, XOR, NOR, NAND), Flip-Flops (D, SR, JK, T), Multiplexer/ De-multiplexer, Counter, Shift Registers, Encoder/ Decoder, Digital Adder/ Subtractor. All the digital components have Transistors (BJT or FET) as basic Logic Level Translator/ Switcher.

II. The authors adopted the methodology to break the complex design in smaller Modules. Each Module had been created by writing a program in Verilog HDL. Top Level module encapsulated all the Modules. The Modules at same Hierarchy level had been interconnected using Digital BUS. Digital BUS also is an internal Library Component of Verilog HDL.

III. Every microcontroller has Register as digital data storage element. In any microcontroller, the Logic level of Register is changed either by Arithmetic Operation or Logical Operation. Authors have implemented the logic change by Logical operation, such that logic level of multiple bits can be changed simultaneously where logical operation code for every Bit of Register is supplied in instruction.

IV. *Logic Level Operation Types in Microcontroller*

In any micro controller, the Operations performed on any Bit are: Set, Clear or Invert. Set means the Logic is forced to '1' & Clear means Logic is forced to '0'. In a microcontroller, the operation to be performed on any Bit is specified in instruction. In traditional microcontroller, there are instructions for bit operations, which manipulate a single bit only. These instructions are called as Bit Level Instructions. Moreover, only one type of bit operation (Set, Clear, Invert) can be performed in any instruction in traditional Microcontrollers. Other type of instructions in traditional Microcontrollers is Byte Level Instructions. Though, programmers use Byte Level instructions to either Set or Clear or Invert multiple bits together, but the mixed operations are not possible with these instructions also.

Previous work done in the field of Microcontroller

Data Memory in a Microcontroller is partitioned in many banks. Variables are allocated different addresses in banks. It is essential to insert bank selection instructions in the application program. Bernhard Schools[1] developed an algorithm to optimize the number of bank selection instructions for partitioned memory for given block of instructions, in which the variables has already been allocated the bank location. The algorithm was tested to optimize for speed and size. To reduce no. of bank selection instructions, hardware based assistance is provided by implementing shared data memory locations. Frequently used variable are placed in shared memory. So, instructions are inserted in application program to allocate address in shared memory. Chunyang Gou[2] developed an algorithm to optimize number of bank selection instructions for shared memory. Many microcontrollers do not have banked data memory. To access non-banked data memory, linear addressing is required. Linear addressing needs full address storage in the instruction. The use of linear addressing increases the consumption of more Program Memory. Nash[4] developed a pseudo – linear scheme to avoid

wrap around problem in segmented & banked memory. To reduce the consumption of Program Memory, instruction width is reduced by dividing the program address range in multiple pages. Page selection is provided through page selection bits in a register. Page selection instructions are inserted in the application program. Quing[3] developed an algorithm to optimize instructions for page selection. Jinpyo[5] used control-flow directed acyclic graph (DAG) to optimize memory for embedded systems & XU Chao[6] developed an algorithm to optimize variable allocation based on block architecture.

Along with data & program memory optimization, execution speed also is an important parameter for any microcontroller. To enhance overall execution speed, some operations are carried in specific hardware circuit instead of by software instructions. Frequent write operations to data memory creates a bottleneckness to speed-up the processing. When, large data is stored in data memory, it consumes too much time to use one instruction for each write operation as well it consumes considerable program memory. So, hardware based assistance has been devised. James R[7] designed a circuit to write data less than the block size without intervention of software. To speed–up processing, pipeline architecture has been implemented in microcontrollers. In pipeline architecture, instructions are executed in parallel irrespective of category/type of instructions. Jean[8] implemented a hardware to do arithmetic & logical operations in parallel to CPU which can be broken down into multiple sequences.

When data is exchanged from microcontroller to external devices, configurable I/O ports are used. During an input instruction through PORT, data is sampled from the pin of Microcontroller. During write operations, some operations use Read-Modify-Write mechanism. Implementation of multiple ports increases the chip size. To resolve this issue, Shekhar Borkar[9] implemented a circuit to interface external bidirectional PORTS to microcontrollers on the same board. The external (Virtual) PORT worked exactly like the on chip PORT that all the ALU operations are possible on these. Charles[10] implemented a circuit to execute instruction in a single cycle. The I/O Read operation is done in the first phase of the clock. Write operation & Next Instruction Fetch is done in the second phase of the clock. So, the effective PORT I/O operation takes only one clock cycle. Stephan[11] focused on the problem of longer wait time for bulk data read / write to/from main memory by CPU or In-Out Control (IOC). The no. of cycles required for Read – Modify – Write operation were reduced. Robert[13] reduces the no. of instructions required to implement Atomic Read-Modify-Write instruction. To reduce the chip size, pin function sharing is implemented and pin function multiplexing hardware is implemented near the PORT. Further, during programming & during normal execution, same pins can be shared. Ray Allen[12] introduced a method to use the PORT pins as programming pins in microcontroller without need of special pins for programming mode. In some applications, it is required to protect the logic on any pin during write operation. Kevin[14] introduced a method to preserve the status of the bit which may be modified by the external device during RMW cycle. Peter[19] implements the method to speed up the read / write operation to/from memory so that maximum bus bandwidth is used. Mathew[23] describes memory read – modify – write speed up by using separate banks for each type of read or write. Ray Brown[24] explains the implementation of RMW functionality without using individual RMW circuit for each register & also registers to set/clear multiple bits in a register without affecting other bits.

With time, new microcontroller architectures were introduced, each architecture supporting some new instructions & operations. Kiran[20] describes about addition of new instructions to MCS-51 microcontroller & making a new family MCS-251 of microcontrollers with backward compatibility. James[21] describes the

microcontroller architecture for general – purpose embedded applications & as well communication applications. Warner[22] describes the architecture to embed multiple programmable digital & analog blocks with programmable interconnection.

The ALU of microcontroller also was enhanced by different manufacturers & researchers. Bit manipulation & Byte write techniques have been updated. Kevin [25] describes the partial bit permutation with permutation code in a control register & result in the destination register. Further, Jon[26] explains the method to improve the efficiency of an ALU. The capability of Single – Cycle ALU & Pipelined ALU are utilized. Arithmetic operations are performed in two stages: First Stage produces separate SUM & CARRY in one cycle. Second Stage produces final SUM & CARRY in one or more cycles. The usable partial results thus produced every cycle, thus maintaining effective one operation per cycle. Parallel instruction execution also proved very much useful to speed-up the performance. There are different pipeline architectures devised. Jeffrey[27] explains the implementation of pipeline architecture by dividing the instruction execution in three cycles. When operation is done on the data, write to destination of previous instruction & read of source for next instruction can be done. Thus overall performance speed of the microcontroller is increased.

Application specific support also has been embedded in many microcontroller architectures to speed up the performance. Bruce[28] provides permutation instructions to perform software based cryptographic operations on data in multimedia, encryption etc. Ruby[29] has devised a method to perform permutations based of Butterfly Networks. Some algorithm use multiple addition or subtraction in a single operation, which can be executed faster in hardware than in software. James[30] has designed an ALU for simultaneous execution of dependent or Independent addition/subtraction logic.

SUMMARY OF THE INVENTION

Authors have designed the circuit by hand. For the purpose of testing, a model was created using Verilog HDL, the design was Simulated using Xilinx ISE and for practical testing it was Implemented in a FPGA. The results of FGPA have been recorded using integrated logic analyzer in CHIPSCOPE software. Further for the purpose of testing the working of the claimed circuits, the authors designed and implemented a microcontroller architecture in FPGA.

V. *SELECTIVE BITWISE IMMEDIATE LOGIC OPERATION CIRCUIT:*
Authors have designed a circuit to perform selective concurrent logical operation on the bits of a register in the microcontroller. The block diagram 100 of the designed circuit is shown in fig 1. The signal 105 controls the passage of signals either 101 or 108 through 11 upto 110. 10 is enabled by setting 105 in active state. When 10 is enabled, the logic at 108 is controlled by 103 and 104 jointly. Signal 106 is permanently kept at logic '1' and 107 at logic '0' in the architecture. When both the signals 103 and 104 are in de-active state, the 107 is passed upto 108. When, both the signals 103 and 104 are active, logic of the signal 106 is passed upto 108. When, the signal 103 is active and 104 is de-actived, the signal 102 is passed upto 108. When, signal 103 is de-active and signal 104 is active, the signal 109 is passed to 108. The logic at 109 is always complement of signal 102. The signals 101, 102, 103, 104, 105 are all controlled by the Control Unit of the Microcontroller during the execution of instruction. The circuit block 100, comprising of five inputs and one output, has been named as SBILO circuit by the authors. The combination of 103 and 104 for selection of any of 102, 106, 107 and 109 may be altered without affecting the complexity and performance, with change in the instruction code accordingly.

Authors have designed novel circuits using the circuit 100 as basic building brick. Fig 2 is the block diagram 200 of the octal group created by packing eight circuits of 100. The authors have used 100 to implement a novel features in ALU of the Microcontroller. The novel feature implemented enabled the selective and concurrent manipulation of bits of a register during the one instruction execution time interval only. Authors classified the instructions in two categories, viz, Normal Instruction and SBILO instruction. During execution of Normal instruction, the SBILO circuit was kept disabled while during the execution SBILO instruction, SBILO circuit was enabled to function.

The circuit 300, shown Fig 3 has been designed for implementation of SBILO based ALU in a Microcontroller by authors. 305 has been implemented as the 8-bit register of microcontroller. Signals output from 200 have been connected to 305. 301 has been used as the 18 bit wide instruction bus of the microcontroller designed by the authors. The instruction bus 301 has been connected to 103 and 104 of all 100 circuits in the right to left bitwise order. The signal 302 has been the connected to 105 of all 100. 303 has been connected to 101 of all 100 circuits in right to left bit order sequentially. The data output of 200 was latched in 305 using 304 as clock. The output of 305 has been fed back to 102 of each 100 at corresponding position.

Circuit block 400 shown in Fig 4, has been designed to implement SBILO feature in data memory of a microcontroller. The data memory has been made capable to selectively and concurrently manipulate the contents of addressed location as per instruction. The data memory 408 has 304 as clock signal. Block 409 used to select the memory location address either from 401 or 407. When, SBILO type instruction executed, memory location selected by 401 for SBIL Operation. During Normal instruction, the data memory 408 functions to store data without any capability to manipulate. During SBILO instruction, the data of the addressed location fed-back to 102 bitwise to 100 of 200, manipulated according to 103 and 104 at each 100 during a single machine cycle consisting of 4 (four) clock cycles only.

The results of SBILO based ALU, as shown in Fig 5, were practically captured using Chipscope software. 0x9B result is obtained after SBILO 10~N1N~1 instruction execution on 0x70 data in the Accumulator of Microcontroller. This SBILO instruction execution took only one machine cycle to perform mixed selective logical operations on Bits of Accumulator.

The functionality of SBILO was also implemented in data memory. The results of SBILO based data memory, as shown in Fig 6, were practically captured using Chipscope software. Initially 0x21, 0x35, 0x66, 0x92 were loaded at initial four locations. Next two instructions perform (11~N0~01~N) & (101NN0~1N) on Data_memory location 1 & 0 respectively. As expected, the results 0x87 and 0xAC were obtained after SBILO. The SBILO instruction took one machine cycle to perform mixed selective logical operation on the data of addressed memory location.