Modify the µCS-51 Architecture to SIMD, VLIW and Superscalar µC

Ph.D: Abdelmoneim M. Fouda [1], Assem Badr [2] and Prof: Abdelsamie kodb [3]

**[1] Computer Eng. Department, Modern Academy for Engineering & Technology
Cairo, Egypt**

**[2] Computer Eng. Department, Modern Academy for Engineering & Technology
Cairo, Egypt**

**[3] Electrical Eng. Department, Faculty of Engineering, Al-Azhar University
Cairo, Egypt**

Abstract

Computer architects have always strived to increase the performance of general purpose computers and special functional CPUs. The high performance may depend on the hardware chips manufacturing, however this trend has constrains, because there are limits of the physical technology, the other trend to increase the performance depend on the parallelism (multiple-processing). Recently the superscalar processing is the latest up-to-date technique of innovations aimed for producing ever-faster microprocessors, they are capable of executing more than one instruction in one clock cycle, in addition to perform multiple independent operations simultaneously.

In this paper we modify the architecture of the conventional microcontroller (µC-51) using VHDL, also we develop some innovated special programmed instructions which utilize parallelism (via multiple processing units). Thus we present a modified superscalar processor using single instruction multiple data (SIMD) with very long instruction word (VLIW) architectures. The proposed modified microcontroller (MµC) has impact in reducing number of clock cycles per instruction, improvement of overall µC performance, and increasing transmission/reception data rate of the µC serial/parallel ports.

Keywords: *µCS-51, ISA, VHDL, SIMD, ILP, Superscalar, VLIW, Data rate, Iron law.*

1. Introduction

Embedded applications are becoming ever more diverse and complex; processors targeting such applications have an increased tendency to attain a desirable performance via highly specialized instructions tailored for the needs of their targets[1].

Consequently, conventional microcontroller architectures vary widely, offering the possibility for implementing various calculating platforms, capable to execute applications in different fields such as signal processing, or communication, command and control. The limitations introduced by the standard architectures become a problem when a high performance and low cost implementation is demanded (for a specified application to a standing context). So it is required to develop the conventional µC (CµC) in such a way to satisfy a trade-off between reaching the specified application and costs[2]. Parallelism is one of the best solutions to achieve high speed of processing, and lowest power consumption for overall speeding application. Parallel processing reduces the execution time taken by any program. The execution time taken by any program is determined by three factors: First, the number of instructions executed, second, number of clock cycles needed to execute each instruction and the third is the length of each clock cycle [3].

There are two forms of parallelism that can be exploited by modern computing machinery to achieve higher performance, the first form of parallelism is called Thread Level Parallelism or TLP, it is software capability to execute independent programs simultaneously using different flows of execution, called threads. The illusion of multiple thread execution is often achieved on a single conventional processor by running each thread successively for short intervals. The second form is Instructional Level Parallelism (ILP), it is hardware architecture to reduce program runtime by overlapping the execution time, it mean that execute many instructions in same machine cycle[4].

The conventional 8051 is scalar processor at which each instruction manipulates single data items at a time, while vector processor manipulates many data simultaneously. The superscalar processor is mixture of both types, its CPU architecture implements a form of parallelism (ILP) within a single processor. Moreover the superscalar processor contains multiple redundant functional units within each CPU, which allows multiple instructions processing on separate data items concurrently.

The ILP techniques can be achieved either by superscalar processor or multi-core processor. The superscalar processor executes more than one instruction during a single clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor, each functional unit is not a separate CPU core but an execution resource within a single CPU (such as an arithmetic logic unit, a bit shifter, or a multiplier). The P5 Pentium was the first superscalar x86 processor; the Nx586, P6 Pentium Pro and AMD K5 were among the first designs which decode x86-instructions [5]. While as multi-core processor has two or more independent actual processors called "cores". Therefore the superscalar processor allows faster CPU throughput than would otherwise be possible at a given clock rate.
Practically it is found that most of the µCs 8051 family transmit their data in the low data rate (LDR) and medium data rate (MDR) which less than 3Mbit/s, therefore it isn't suitable to interface with higher data rate equipments (such as Ethernet, fast Ethernet, and gigabit Ethernet which have data rates of 10Mbit/s, 100Mbit/s and 1Gbit/s respectively). So we develop and modify the internal architecture for the CISC (complex instruction set computer) microcontroller family µCS-51 and their instruction set architecture (ISA) by using SIMD type of superscalar techniques and with Very Large Instruction Word (VLIW) architectures to improve their performance and increase the transmission rate.
The paper is organized as follows; the next section-2 describes the overall design steps of the developed µC, including the ISA instruction set modification in section-2-1, and the processor architecture modification in section 2-2. While as section-3 discuss the different numerical results and processor performance. Finally, section 4 presents an overall conclusion.

2. Design Processor

To develop a novel µC based on a conventional one, different modifications are required including instruction set modification, and architecture modification. This section introduces the necessary basic principles to design modified VHDL code for the internal architecture of the conventional CISC µC 8051. The developed VHDL code is obtained by adding the so called "modified instruction decoder" (MID) in addition to the conventional instruction decoder (CID). We can alternate between either MID or CID using the so called "selected instruction decoder flag (SIDF)". Another modification is obtained by inserting two advanced VHDL codes represent two individual programmable input/output data sequencer, the first one called programmable output digital parallel/serial sequencer (PODPSS), the other one called programmable input digital parallel/serial sequencer (PIDPSS), both sequencers depend on single instruction and multiple data (SIMD).

2.1 Instruction set modification

The ISA can be adapted or extended to meet the modern application requirements [7]; this section is dedicated to discuss the processor's ISA modification. Many methodologies can used to increase the number of instructions, either increasing the number of bits assigned for op-codes, or using two/more successive locations to store instructions in the µC program memory. The selected methodology must keep the characteristics of the µC's memory, internal data and address bus avoid from any change.
From literature survey, and guided with Intel µCS-51 family's data sheets and its manual of instructions set [6], it's found that the machine code "A5H" is a reserved code which is not used for any operations or tasks, as shown in Fig.1. Utilizing this reserved code, we can expand the number of µCS-51family's ISA and adding sophisticated instructions customized for specific multiple data applications. The suggested assembly syntax code "InovConv" uses the reserved machine code "A5H", this code will have toggle alternative action (TAA), namely, when the "InovConv" code is written by the user it is compiled into "A5h", and alternate as toggling action between either CID (when SIDF = "0") or MID (when SIDF = "1") as shown in Fig.2a. Now the developed µC can be used in two states either in its conventional mode of operation (256 instructions) or in the modified mode of operation (255 instructions) therefore the number of instructions for the CµC-51 will be expanded from 255 to 511 instructions.

Hex Code	Number of Bytes	Mnemonic
A0	2	ORL
A1	2	AJMP
A2	2	MOV
A3	1	INC
A4	1	MUL
A5	1	Reserved
A6	2	MOV

Fig.1 Partial of conventional µCS-51 instruction set.

Mnemonic	Hex code	SIDF	Enabled instruction decoder
InovConv	A5	0	Conventional Instruction Decoder (CID)
		1	Modified Instruction Decoder (MID)

Fig.2a Toggling stages of A5h.

HEX code	conventional (ISA)		modified (ISA)		
	No. of Bytes	Assembly code	No. of Bytes	Assembly code	SIDF
A5	1	Reserved code	1	InovConv ***	0*
A5	1	Reserved code	1	InovConv ***	1**
A6	2	MOV @R0,Direct	6	OUTSeqB N,B,P,S	N.C
A7	2	MOV @R1,Direct	6	INSeq N,B,P,S	N.C
A8	2	MOV R0,Direct	1	HLTSeqB	N.C
A9	2	MOV R1,Direct	1	LdBKifDon	N.C

*** Boolean alternative action (BAA) ** Causes innovation codes
N.C ..... Not Care * Causes conventional codes

Fig.2b Partial of modified µCS-51 instruction set (ISA).

A sample of modified instruction set of the developed µC-51 is shown in Fig.2b; both modified instructions "OUTSeqB" and "INSeqB" are SIMD instructions with programmable process they can be executed simultaneously. For illustrating the execution of the new instruction "OUTSeqB N,B,P,S", it is assigned for output sequence of data bytes (maximum 8 bytes per sequence) retrieved from address 10h to 17h in the µC's main memory (RAM). The generated sequence may be transmitted throughout serial/parallel port according to the flag B (1-bit), if B="1" then the serial port will transmit data, while if B="0" the parallel port will transmit data. The operand "P" (3-bits) is assigned to allocate the number of transmitted data bytes per sequence, if "P" equal "000" it means that sequence has only one data byte (the contents of RAM location 10h), else if "P" equal "001" it means that the sequence has 2 data bytes (the contents of RAM location 10h, 11h), and so on, if "P" equal "111" it means that sequence has 8 data bytes (the contents of RAM location from 10h to 17h). The generated sequence may be repeated if the instruction "NLOPOUTS" is de-activated, and the repeated loop is terminated immediately if the instruction "HLTSEQB" is activated.
Moreover to control the transmission rate of the manipulated data and with the help of using simulator "Modelsim SE 6.5" we derived the following empirical relation (1):

$$AdjustedData\ Rating_{parallel} = \frac{f_{cp}}{1+N_{operand}} \quad [MB/S] \quad (1)$$

Where f_{cp} is the frequency of the global clock pulses for the VHDL µC, to calculate the data rate performance, assume that $f_{cp} = 50\ MHz$ and the operand "N" has its minimum value N= "000000000h" then the maximum data rating equal to 50-MBps (i.e. output byte each 20 ns), else if operand "N" increased to (000000001), then the data rate is 25-MB/S (i.e., output byte each 40 ns) and so on, if N has its maximum value "7FFFFFFFFh" then very low data rate is obtained (Byte each 11.45 minutes).
Similarly, the instruction "INSeqB N,B,P,S" is developed for receiving sequence of consecutive 8-data bytes (stored in RAM data memory from address 20h to 27h) through the PIDPSS unit, and is controlled by the status of the 4 operands "B", "P", "S", "N" as previously mentioned. As a conclusion we described a sample of two innovated instructions, they have full duplex mode of transmitting/receiving data while the µC process other instructions simultaneously. In the next section we will explain the VHDL realization for the mentioned instruction modifications.

2.2 Processor's Architecture modification

Adapting the old architecture to a more flexible implementation empowering further developing[2], High-level design tools and field-programmable gate arrays (FPGAs) significantly reduce the effort, cost and risk of hardware implementation. These technologies can be incorporated into a manageable and affordable prototyping framework a VLSI-scale "breadboard" for exploring and evaluating new microprocessor designs[8]. For this reasons a new architecture will be delivered to modify the CµCs-51 by the VHDL over the FPGA technique. The basic block diagram for the internal architecture of the MµC-51 is shown in Fig.3. The shaded blocks represent the traditional units of the CµC-51, and the other blocks represent the created units for the MµC-51.
The 1st modification is obtained by adding MID-unit and modified instruction register (MIR-unit) beside conventional instruction register (CIR-unit) and the CID-unit, all of them is controlled by SIDF.
Initially the SIDF is set to '0', so the CID is active, so the traditional 8051 instruction set will be fetched, decode, and then executed. When using the code "InovConv" (denominated by user's program) the SIDF is set to '1', so the MID is activated and the modified 8051 instruction set will be fetched, decode, and then executed. Furthermore, when the "InovConv" code is written again by user, it toggles the SIDF to logic '0' causes enable for CID again, and so on.

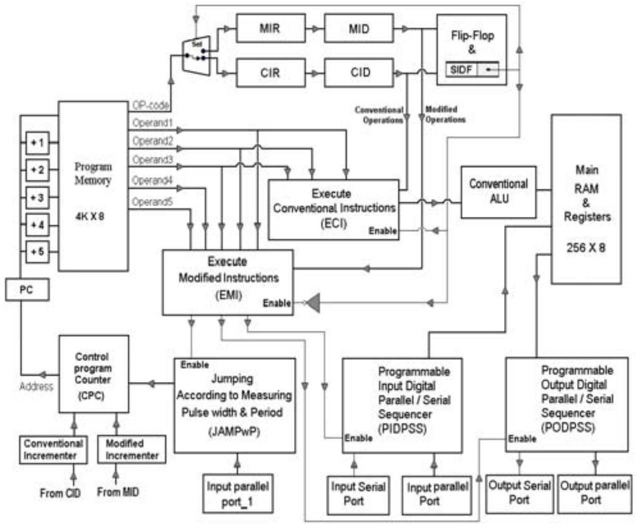


Fig.3 The basic of internal architecture for the MµC-51

The 2nd modification is particular for the program memory's interface; it is performed by increasing the number of data bus associated with program memory (from single bus to 6 data bus). The instructions of the CμC-51 represented by maximum 4 bytes in the program memory as show in Fig.4a, each byte fetched one by one through single bus, while the instructions of the modification version needs maximum 6 sequencing bytes stored in the program memory as show in Fig.4b. Such developed construction is capable for fetching all adequate bytes simultaneously. The recommended memory location fetched starting from the value stored in the program counter (PC), and the desired number of fetched bytes specified by the modified unit designated as control program counter (CPC), the increment in the program counter related to the absolute values stored in the CID or MID.

Conventional OP-Code	Operand-1	Operand-2	Operand-3

Conventional Instruction format

Modified OP-Code	Operand-1	Operand-2	Operand-3	Operand-4	Operand-5

Modified Instruction format (maximum No. operands)

Fig.4a The conventional and modified VLIW instruction formats

Bit No.	1 8	9 43	44	45 47	48
	OP-Code (OUTSeqB) (8 bits)	Operand-N (35 bits)	Operand-B (1 bit)	Operand-P (3 bits)	Operand-S (1 bit)

Format of instruction (OUTSeqB) consist of op-code(byte) and 4 operands(5bytes)

Operand	B = 0	Enable parallel port
	B = 1	Enable serial port

Operand	S = 0	Prevent start & stop bits
	S = 1	Permit start & stop bits

Fig.4b The format of modified instruction (OUTSeqB)

In addition to the original RAM data bus, the 3rd modifications adds additional 8-input data bus particular for the RAM locations (with address 10h to 17h), and adding additional 8-output data bus for the RAM locations (with address 20h to 27h). Such modifications allow RAM-data bus to read/write its specific memory locations concurrently according to the specified address.

The 4th modification includes the creation of so the called "execute modified instruction EMI" which is responsible for controlling both two modified data sequencer units "PODPSS" and "PIDPSS". They are providing response for executing the other logical, arithmetic, and transferring modified operations. Each sequencer attached with μC internal/external ports for handling a generated sequence of data with the μC's peripherals. The PODPSS is particularized for transmitting data through specific μC external output port, while as the PIDPSS is particularized for transmitting data through specific μC external input port. Each sequencer handles data with different data rates and adequate number of data bytes.

Another modification includes different types of adjustable counters to establish the operation of PODPSS or PIDPSS. The adjustable counter_1 is responsible for adjusting the data rate (related to the detected operand "N"). The adjustable counter_2 is used to adjust the number of bytes per each generated sequence of pulses (according to the operand "P"). Moreover; the adjustable counter_3, is allocated for serial port transmission, which is responsible for inserting start/stop bits for the data sequence (according to the operands "B" and "S").

Firstly; as show in Fig.5 the adjustable counter_1 consists of logical counter, register and logical comparator. The basic idea of the adjustable counter is loading operand "N" value into the register, and counting the received clock pulses, so if the counting value of the counter equal to the stored operand "N" value, the comparator becomes active output to clear the counter, and the two comprised values becomes not equal, so the counter will restarting again and so on.

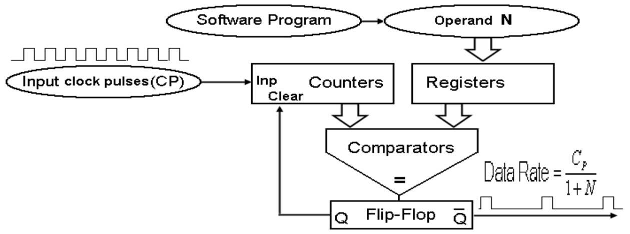


Fig.5 The block diagram for the adjustable counter_1

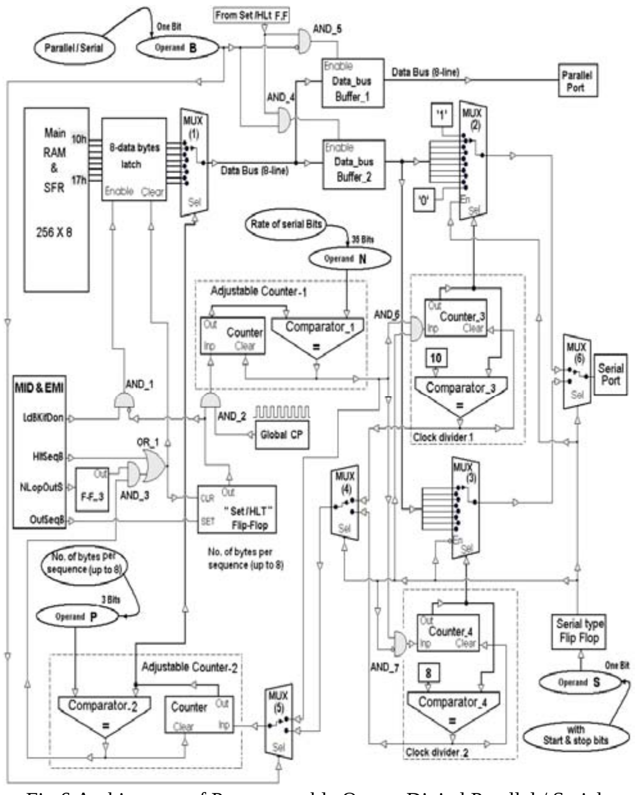


Fig.6 Architecture of Programmable Output Digital Parallel / Serial Sequencer (PODPSS)

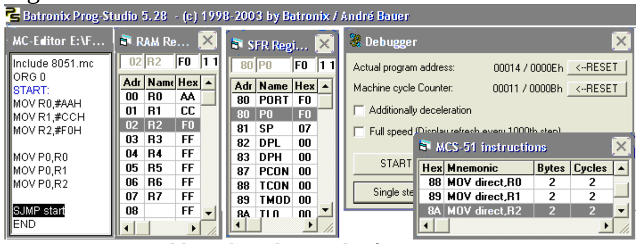
The internal structure of PODSS is shown in Fig.6 for more illustrations assume the following scenario; it is required to transmit (through parallel port) a sequence of 3-data bytes with values AAh, CCh, and F0h, stored in RAM locations with addresses 10h, 11h, and 12h respectively. The data transmission rate will equal to the global frequency of the µC, and. Initially, the "Set/HLt " Flip-Flop has output '0' for halting the PODPSS by blocking the gate AND-2 (block global clock pulse Cp), and permit the 1st modified instruction "LdBkifDon" for passing through AND_1 to latch the contents of 8 RAM locations (starting from address 10h) to the multiplexer MUX_1 via 8-output RAM bus. The modified instruction "NlopOutS" set FF-3 to "1" thus it's ANDED with the output of comparator_2 (via AND-3) to prevent looping of generated data sequence (allow only one sequence of data bytes). For more clarifications, the SIMD instruction "OutSeqB 0,0,2,0" is used to set the PODPSS, the operand "N=0" causes maximum data rate, the operand "B=0" drive "data_bus_buffer_1" through AND_5, so the 1st data byte (AAh) will be transmitted via the parallel port. The operand "P" adjusts the number of transmitted data bytes per one sequence, in case "P=2" sequence has 3 data bytes (AAh, CCh, F0h) respectively. The 1st data byte (AAh) still on the parallel port until the rising edge from comparator_1 increment adjustable counter_2 through MUX_5, thus causing change in the selection pins of MUX_1 to select the 2nd data byte (CCh). Moreover; the next rising edge from comparator_1 increments counter_2 to select the 3rd data byte (F0h) , the next rising edge from comparator_1 will increment the counter_2, so the comparator_2 will drive '1' which will AND with the logic '1' from "FF_3" to halt the PODPSS. Note that the operand "S" not effected in the parallel port.

Now for serial data transmission, another SIMD instruction "OutSeqB 0,1,2,1" is used. The operand "B=1" enables "data_bus_buffer_2" through AND_4. The 1st data byte (AAh) will be transferred to the serial port through the MUX_2 which enabled by operand "S=1". Each rising edge of the adjusted data rate signal from comparator_1 will increment the counter_3 causes transmission of start bit and the consequence 8-data bits (starting with least significant bit LSB) and terminated with the stop bit consecutively. After 10 rising edge of the adjusted data rate signals, the comparator_3 increment the counter_2 through the MUX_4 and MUX_5 causes select the next data byte to be transmitted (CCh) of the input of MUX_1 and so on.

Note that the both modified instruction "HLtSeqB", "NlopOutS" can used at any time by the programmer, the "HLtSeqB" for immediate halting for the transmitting data, but the "NlopOutS" for halting the transmitting data after finishing the current sequence (don't loop again)

3. Numerical results

Form the simulation point of view; assume the previously described task for transmitting data bytes (retrieved from 3-RAM locations) through serial and parallel ports. To compare the behavior of both modified and CµCs assuming the measuring parameters including data rate, time processing. Firstly; using the CµC-AT89C52, and Proteus 7.5 sp3 simulator package (Labcenter.com) to draw its schematic circuit, then guided with simulator package Prog-Studio-5 (Batronix.com) to edit and compile program code for particular task as shown in Fig.7. With aid of the Proteus digital timing analyzer we can follow up the behavior of AT89C52 for the desired task as shown in Fig.8 and Fig.10. Secondly; using package "Xilinx ISE 9.2i" to edit and synthesize in VHDL code the MµC. With aid of the Modelsim-SE 6.5 (Mentor-graphic) we can follow up the behavior of MµC, as shown in Fig.9 and Fig.11.



```
Include 8051.mc
ORG 0
START:
MOV R0,#AAH
MOV R1,#CCH
MOV R2,#F0H

MOV P0,R0
MOV P0,R1
MOV P0,R2

SJMP start
END
```

Adr	Name	Hex
00	R0	AA
01	R1	CC
02	R2	F0
03	R3	FF
04	R4	FF
05	R5	FF
06	R6	FF
07	R7	FF
08		FF

Adr	Name	Hex
80	PORT	F0
80	P0	F0
81	SP	07
82	DPL	00
83	DPH	00
87	PCON	00
88	TCON	00
89	TMOD	00
8A	TL0	00

Actual program address: 00014 / 0000Eh <--RESET
Machine cycle Counter: 00011 / 0000Bh <--RESET
Additionally deceleration
Full speed (Display refresh every 1000th step)

Hex	Mnemonic	Bytes	Cycles
88	MOV direct,R0	2	2
89	MOV direct,R1	2	2
8A	MOV direct,R2	2	2

Fig.7 Assembly and machine codes for 3 sequence patterns

In the case of parallel transmission, for the CµC, as shown in Fig.7, the left form indicate the editor of the assembly code to transfer 3 data bytes (AAh, CCh, F0h) from 3-RAM locations (R0,R1,R2) respectively via the parallel port (P0). The two forms in the middle indicate the contents of data memory and the contents of ports during execution. The top right form indicate the control form illustrate the address of program counter and the number of machine cycles. In the bottom right form partial of µCs-51 instruction set. It is found that; the output data instructions occupy two bytes in the program memory and it is executed in two machine cycles.

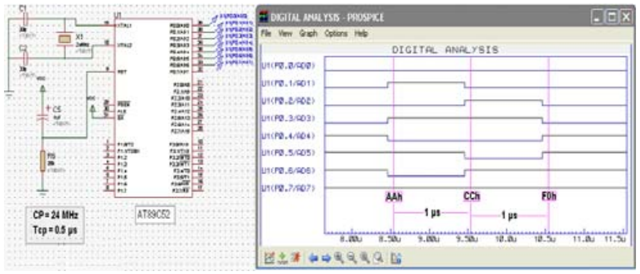


Fig.8 µC AT89C52 circuit diagram and output timing diagram (1us between bytes)

In Fig.8 the parallel output timing diagram of CµC's port P0, it is found that the smallest difference time between any two successive output data bytes is 1µs, this is very large time and causes low data rate transmission. It is concluded that, this low data rate is due to the number clock pulses and number of cycles taken for µC-51 instructions which response for transmitting the parallel bytes (as MOV P0,R1), it has 2-machine cycles each cycle has 12-Cps for execution. Since that the maximum Cp for µC-51 equal to 24MHz, then the delay between any two successive output data bytes or the minimum duration of the conventional parallel output (Tc_{PP}) is determined by "Iron Law" (2) [9][10]

$$\frac{Time}{program} =$$

$$\frac{No.of\ instructions}{program} \times \frac{No.of\ cycles}{instruction} \times \frac{time}{cycle} \qquad (2)$$

From 8051 data sheet, the duration time for one machine cycle T_{MC} is equal to 12 of clock pulse (Cp) duration and defines by the equation (3)

$$T_{MC} = 12\ T_{cp} = \frac{12}{f_{cp}} = \frac{12}{24 \times 10^{6}} = 0.5\ \mu s \qquad (3)$$

(3)
From equation (2) and (3) then

$$Tc_{PP} = 1 \times 2 \times 0.5 = 1\ \mu s \qquad (4)$$

Now applying Iron law to calculate the execution time (Tc_{exe}) for the 3 conventional instructions (MOV Port, Memory) which response to out the 3 data byte (AAh, CCh, F0h) from 3-RAM locations (R0,R1,R2) as observed in the digital analyzer in Fig.8 we get eq.(5)

$$Tc_{exe} = 3 \times 2 \times 0.5 = 3\ \mu s \qquad (5)$$

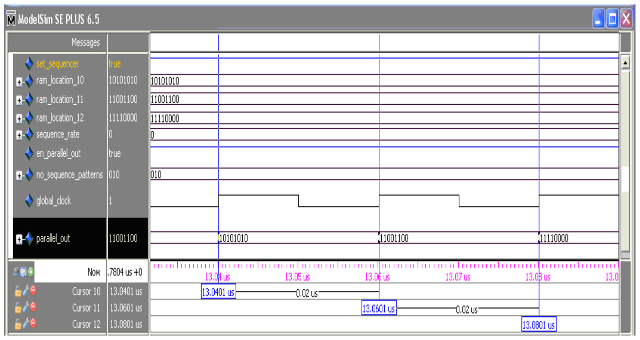


Fig.9 Timing diagram "Modelsim SE 6.5" showing the sequences of 3 parallel patterns

Consequently, for the MµC as shown in Fig.9, the SIMD instruction "OutSeqB 0,0,2,0" (as we explained before) will transmit 3-bytes via parallel port and with maximum data rate. Our MµC execute the instruction in one Cp for each output. The time difference between any two successive data bytes (AAh, CCh, F0h) is (Tm_{PP}), which equal to the duration time of modified machine cycle (T_{MMC}), and equal to one Cp duration, assume that the operating frequency of MµC is f_{Mcp} equal to 50 MHz, so the Tm_{PP} will define as eq. (6):

$$Tm_{PP} = T_{MMC} = T_{Mcp} = \frac{1}{f_{Mcp}} = \frac{1}{50\times10^{6}} = 0.02\ \mu s \qquad (6)$$

Therefore the modified executed time (Tm_{exe}) for the 3-data bytes using the SIMD "OutSeqB 0,0,2,0" takes 3 clock pulses as observed in the digital analyzer in Fig.9 is determined by "Iron Law" defined in eq. (7)

$$Tm_{exe} = 1 \times 3 \times 0.02 = 0.06\ \mu s \qquad (7)$$

It is noted that the execution time of MµC outperform the traditional one, the speed up of the MµC will obtained as equation (8) [9]:

$$Speed\ \ up(M_{Parallel}) = \frac{\left(time \big/ program \right)_{C}}{\left(time \big/ program \right)_{M}}$$

$$= \frac{Tc_{exe}}{Tm_{exe}} = \frac{3\ \mu s}{0.06\ \mu s} = 50\ times \qquad (8)$$

Moreover, for CµC-51, the data rate of parallel transmission (parallel channel signaling rate PCSR) can be determined by the following eq. (9) [10]:

$$PCSR = \sum_{i=1}^{m} \frac{\log_2 n_i}{T_i} \qquad [Mb/s] \qquad (9)$$

Where m is the number of parallel channels, n_i is the number of significant conditions of the modulation in i-th channel, and T_i is the unit interval, expressed in seconds, for i-th channel. For parallel transmission with equal unit intervals and equal numbers of significant conditions (two- condition modulation) on each channel, the conventional parallel channel signaling rate (PCSR_{C}) will reduces to eq. (10):

$$\mathrm{PCSR}_{C} = \frac{8}{Tc_{PP}} = \frac{8}{1\times10^{-6}} = 8\,Mb/s = 1\,MB/s \qquad (10)$$

the parallel transmitting data rate for MµC-51 determine by parallel channel signaling rate (PCSR) law, and the modified parallel channel signaling rate (PCSR_{M}) will reduces to eq. (11):

$$\mathrm{PCSR}_{M} = \frac{8}{Tm_{PP}} = \frac{8}{0.02\times10^{-6}}$$

$$= 400\,Mbps = 50\,MB/s \qquad (11)$$

PCSR_{M} Is meeting with our empirical parallel data rating in equation (1), in case of operand (N=0), the maximum data rate is defined as eq. (12):

$$Adjusted\ Data\ Rating_{parallel} = \frac{f_{cp}}{1+N}$$

$$= \frac{50\times10^{6}}{1+0} = 50\,MB/s \qquad (12)$$

Comparing between the results of eq.(10) and eq.(11) or eq.(12), the parallel data rate of the modified type µC is superior than that of conventional type. The data rate enhancement factor can be obtained as relation (13):

$$PDRE = \frac{\mathrm{PCSR_M}}{\mathrm{PCSR_C}} = \frac{50MBps}{1MBps} = 50\ times \qquad (13)$$

Furthermore the second task is output group of 3 data bytes through the serial port for both conventional and MµC, the Fig.10 indicates the editor of the assembly code designed for output 3 data bytes (AAh, CCh, F0h) from 3 RAM locations (R0, R1, R2) with start and stop bits, and simulated in simple AT89C52's circuit diagram using "Proteus 7.5" as show in Fig.11. The maximum serial data rate (mode1) of CµCs-51 is direct proportional with Cp ($f_{cp} = 24MHz$) in relation (14) [6].

$$Maximum\,Data\,Rate_{mod\,e1} = \frac{f_{cp}}{192}$$

$$= \frac{24\times10^{6}}{192} = 0.125\ \ [Mb/s] \qquad (14)$$

Moreover, for CµC-51, the data rate of serial transmission (serial channel signaling rate SCSR) can be determined as follows in eq. (15) [10]:

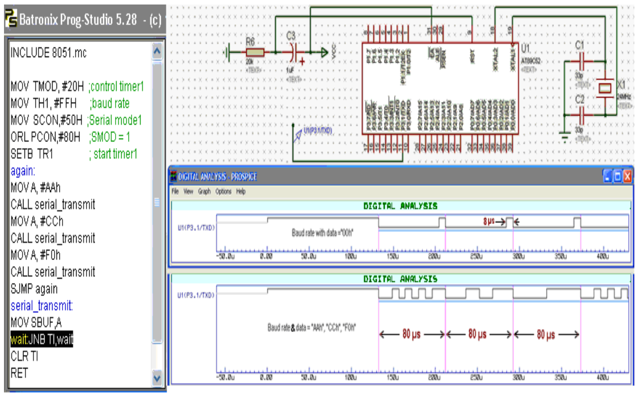
$$SCSR = \frac{\log_2 n}{T} \qquad [Mb/s] \qquad (15)$$

As shown in Fig.11 the minimum duration for one bit ($Tc_{SP} = 8\,\mu s$), and with a two-condition modulation (n=2) the $SCSR_C$ will reduce to eq. (16):

$$SCSR_C = \frac{1}{Tc_{SP}} = \frac{1}{8\times10^{-6}} = 0.125\,Mb/s \qquad (16)$$

We notice that the simulation result of the CµC-51 data rates (mode1) is meet with the $SCSR_C$. The total time to transmit start bit, 8 data bits, and stop bit is ($10\,Tc_{SP} = 80\,\mu s$), so the conventional executed time (Tc_{exe-S}) for out the 3 data bytes (AAh, CCh, and F0h) serially is define by eq. (17):

$$Tc_{exe-S} = 30 * Tc_{SP} = 240\,\mu s \qquad (17)$$


```
INCLUDE 8051.mc

MOV  TMOD, #20H  ;control timer1
MOV  TH1, #FFH    ;baud rate
MOV  SCON,#50H  ;Serial mode1
ORL PCON,#80H   ;SMOD = 1
SETB  TR1           ; start timer1
again:
MOV A, #AAh
CALL serial_transmit
MOV A, #CCh
CALL serial_transmit
MOV A, #F0h
CALL serial_transmit
SJMP again
serial_transmit:
MOV SBUF,A
wait:JNB TI,wait
CLR TI
RET
```
Batronix Prog-Studio 5.28 - (c)
DIGITAL ANALYSIS
U1(P3.1/TXD)
Baud rate with data ="00h"
8µs
Baud rate & data = "AAh", "CCh", "F0h"
80 µs
80 µs
80 µs

Fig.10 Circuit diagram, timing diagram and assembly code of AT89C52 for 3 sequence serial bytes

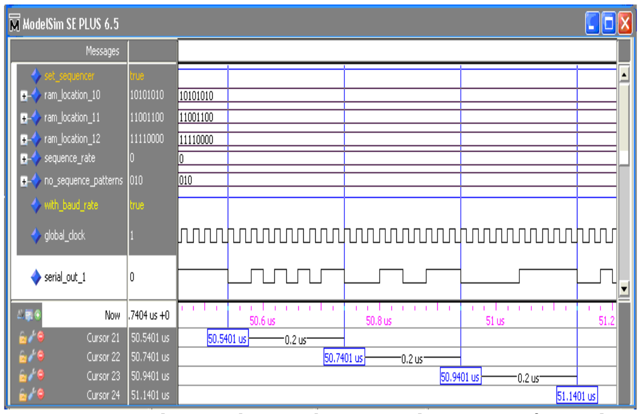


Fig.11 Timing diagram showing the generated sequences of 3 serial bytes with start and stop bits

Consequently the simulation using "Modelsim SE 6.5" as show in Fig.11, for the SIMD instruction "OutSeqB 0,1,2,1" (as we explained before) will transmit 3 bytes in serial transmitting with maximum data rate, we can observe that the minimum duration of the modified the total time for transmit any byte (AAh, or CCh, or F0h) is 0.2 µs, the duration time for any serial output bit (Tm_{SP}) or the time difference between any two successive bits is only 0.02 µs, which equal to one of clock pulse duration (assume $f_{Mcp} = 50$ MHz) is define by eq. (18)

$$Tm_{SP} = T_{Mcp} = \frac{1}{f_{Mcp}} = \frac{1}{50\times10^{6}} = 0.02\,\mu s \qquad (18)$$

Therefore the modified executed time (Tm_{exe-S}) for SIMD "OutSeqB 0,1,2,1" which response to out the 3 data bytes (AAh, CCh, F0h) serially (with start and stop bits) from 3 RAM locations (R0, R1, R2), each bit need one clock pulse (totally 30 Cps) as observed in the digital analyzer in Fig.11 is determined by “Iron Law” in equation (19).

$$Tm_{exe-S} = 1\times30\times0.02 = 0.6\,\mu s \qquad (19)$$

Comparing results of eq. (17) and eq. (19) it is found that the MμC is more superior to that the traditional type in the case of serial transmissions. The speed up of the modified superscalar μC for the modified SIMD "OutSeqB" in the case of serial data transmission is obtained from relation (20) [9]:

$$Speed\ up(M_{Serial}) = \frac{Tc_{exe-S}}{Tm_{exe-S}} = \frac{240\ \mu s}{0.6\ \mu s} = 400\ times \quad (20)$$

Moreover the serial transmitting data rate for MμC-51 determine by serial channel signaling rate (SCSR) law, so the modified serial channel signaling rate ($\mathrm{SCSR}_{\mathrm{M}}$) will reduces to eq. (21):

$$\mathrm{SCSR}_{\mathrm{M}} = \frac{1}{Tm_{SP}} = \frac{1}{0.02\times10^{-6}} = 50\,Mbps = 5.0\,MB/s \quad (21)$$

The $\mathrm{SCSR}_{\mathrm{M}}$ will equal to $6.25\ MB/s$ in case of transmitting serially without both start and stop bits.

The $\mathrm{PCSR}_{\mathrm{M}}$ is meeting with our empirical parallel data rating in eq.(1), in case of operand (N=0), the maximum serial data rate (without both start and stop bits) will define in equation (22) , and for maximum serial data rate (with start and stop bits) will define in equation (23).

$$Adjusted\ Data\ Rating_{Serial\_8bit} = \frac{f_{cp}}{8\times(1+N)} = \frac{50\times10^{6}}{8\times(1+0)} = 6.25\ MB/s \quad (22)$$

$$Adjusted\ Data\ Rating_{Serial\_10bit} = \frac{f_{cp}}{10*(1+N)} = \frac{50\times10^{6}}{10\times(1+0)} = 5.0\ MB/s \quad (23)$$

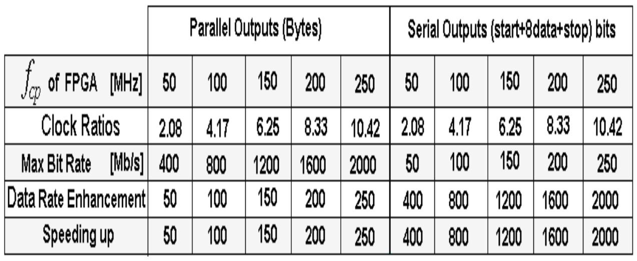
Moreover, in all cases the modified type is superior to the traditional μC, the serial data rate enhancement (SDRE) of the modified superscalar μC as follows:

$$SDRE = \frac{\mathrm{SCSR}_{\mathrm{M}}}{\mathrm{SCSR}_{\mathrm{C}}} = \frac{5MBps}{0.0125MBps} = 400\,times \quad (24)$$

Finally, for the purpose of comparisons between the traditional μC-51 and the developed type shows the maximum data rate, enhancement data rate, speed up, at different applied clock frequencies of FPGA chips as shown in Fig.12.

4. Conclusions

Recently the family of μC-51 has numerous applications in different fields. Utilizing the reserved bits of the traditional μC-51 we extended its internal architecture and expanded its instruction set along the main dimensions of parallelism. By using the VLIW and SIMD types on superscalar architecture processor we can enhance the performance of the conventional μCs. we can execute multiple instructions in same clock cycle, and we can perform multi-operations like fast Fourier transform, matrix multiplication, finite/infinite impulse response filters. The modified μC is more superior to the traditional μC from many points of view of performance parameters including data rate, execution time, and speed up.



	Parallel Outputs (Bytes)					Serial Outputs (start+8data+stop) bits				
f_{cp} of FPGA [MHz]	50	100	150	200	250	50	100	150	200	250
Clock Ratios	2.08	4.17	6.25	8.33	10.42	2.08	4.17	6.25	8.33	10.42
Max Bit Rate [Mb/s]	400	800	1200	1600	2000	50	100	150	200	250
Data Rate Enhancement	50	100	150	200	250	400	800	1200	1600	2000
Speeding up	50	100	150	200	250	400	800	1200	1600	2000

Fig.12 most comparison levels for several clock frequencies

References

[1] Jonghee M. Youn, Sechul Shin, "A New Addressing Mode for the Encoding Space Problem on Embedded Processors", 7th Symposium on Application Specific Processors (SASP), IEEE, 2009.
[2] Elena Roxana Buhus, "A System-On-Chip Approach in Designing a Dedicated RISC Microcontroller Unit Using the Field-Programmable Gate Array", 5th International Conference on Systems, computer society IEEE, 2010.
[3] M.Suaib, A.Palaty, K.Sambhav, "Architecture of SIMD Type Vector Processor ", International Journal of Computer Applications, Volume 20, April 2011.
[4] Website,www.realworldtech.com/page.cfm, last accessed on the 18th of Aug 2011, hour 16:30
[5] Web site, "wikipedia.org/wiki/Superscalar" last accessed on the 10th of Sep 2011, hour 13:40
[6] Web site, www.Philips.com, "80C51 family programmer's guide and instruction set" last accessed on the 17th of Aug 2011, hour 02:03
[7] Sajjan G.Shiva, "Computer Organization, Design and Architecture", Marcel Deker,NY, third edition, revised and expanded, pp.181-450,2000
[8] Joydeep Ray, "High-Level Modeling and FPGA Prototyping of Microprocessors ", IEEE, 2003.
[9]A.R.Alameldeen, D.A.Wood, "IPC Considered Harmful for Multiprocessor Workloads", computer society IEEE, Aug 2006.
[10] H. Weik Martin, "Fiber Optics Standard Dictionary", Hall book, 3rd edition, 1997.