JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN ELECTRONICS AND COMMUNICATION ENGINEERING
ISSN: 0975 – 6779| NOV 16 TO OCT 17 | VOLUME – 04, ISSUE – 02

DESIGN OF PROCESSOR FOR IMPROVEMENT IN SPEED USING RECONFIGURABLE HARDWARE

T. M.DUDHANE[1], J. J.BANDA1[2], S.B.PATIL[3]

Department of electronics and telecommunication engineering

Chhatrapati ShivajiRaje College of Engineering, Savitribai Phule University of Pune, India

tmdudhane@yahoo.com,
bandal864@gmail.com
sbpatil@gmail.com

ABSTRACT:
The desire to enhance the processors speed, RISC design philosophy is the solution using FPGA. To improve the speed for execution of instructions and to increase the number of instructions for implementation by the use of FPGA as the gate capacity growing day by day. The PIC16F84 microcontroller with its 33 instructions and additional 15 more instructions designed , simulated and implementation using VHDL Code, downloading on FPGA –SPARTAN-III, there is improvement in speed of the execution of instruction .The core is designed module wise, VHDL code is developed for each module and downloaded on FPGA. While designing, parallelism of digital design hardware approach is considered that is using maximum number of sequential circuits and minimum number of combinational circuits to avoid delay. The designed RISC-core with additional instructions may be used as IP core.

KEYWORDS: RISC, FPGA, VHDL, SPARTAN-III,

1. INTRODUCTION

In today's word, we see many industrial and domestic products like remote controllers; telephone bill printing machines, automobiles, mobile phones, oven, and automation is required [1]. This is required to facilitate the process of mechanism for its operation and control. Data storage and processing is an integral part of any automatic control system. So there is a need to have a device called, "Microcontroller", which helps to carry out the function of atomization.

While designing, the improvement in speed and having implementation of maximum instruction near about 50 instructions, are the goals of the designing. For the achievement of this goal, parallelism approach is used. The PIC16F84,RISC CPU has 35 instructions which are single word and single cycle except program branching instruction which are two cycle. The present operating speed is 20MHz and clock input is DC. Program memory is 1024 words, Data RAM is 68 bytes. Data E2PROM is 64 bytes. Core has 8-bit data size and 14-bit wide instruction words [5].

The present PIC is developed module wise at gate level and VHDL code is developed module wise. The four states T1,T2,T3 and T4 are developed, which are op code, fetch, decode, execution and write back respectively. For implementing purpose SPARTAN-III is used, because of low cost, high volume and high performance and consumer oriented applications [2].

2. METHODOLOGY

For the designing and implementation of the RISC processor, the design methodology is discussed below.

2.1MODULEWISE DEVELOPMENT:-

The PIC16F84 Microcontroller is partitioned into number of modules as instruction decoder, Arithmetic unit, Logic unit, Rotate/shifter unit, Bit-set clear unit, Generation of T-states and combination of all above units. The VHDL code is written for individual module using Xilinx ISE simulator. It features optimized direct compile for the fastest compile times and competitive simulation performance.

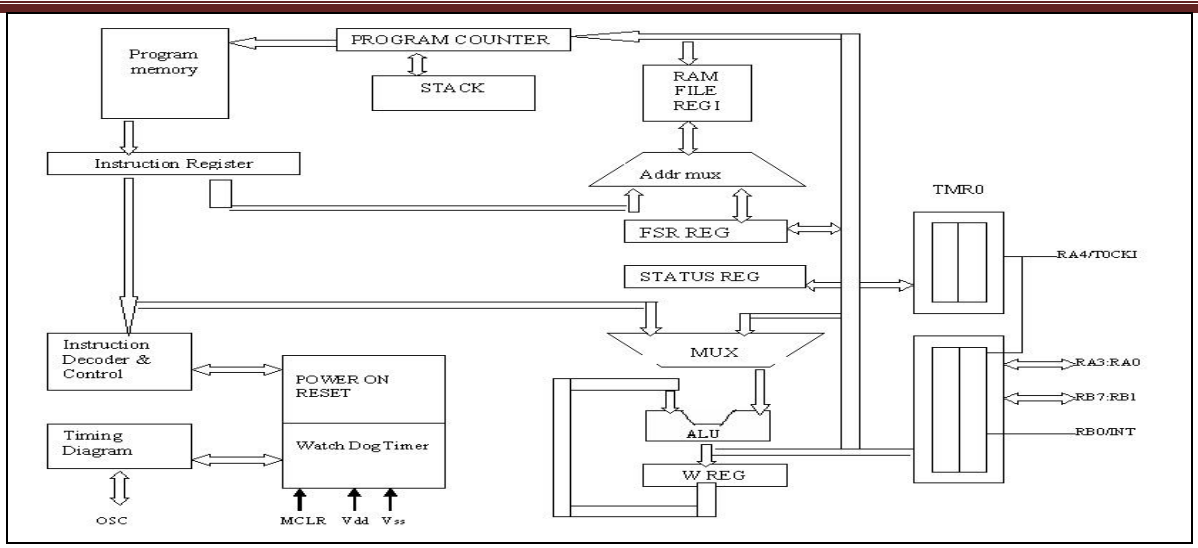

Figure1.The Block Diagram of RISC 16F84

2.2 REGISTERS AND INSTRUCTION SET:

The instruction set is listed in Table-I. Along with these instructions additional instructions that can be implemented are,BC,BN,BNC,BNN,BNOV,BNZ,BOV,BZ,POP,PUSH,MULWF,RLCF,RRCF,SHL,SAL,SHR,SAR.

Total instructions, can be implemented are near about 50.The instructions are

- Byte oriented
- Bit oriented and
- Literal and control oriented.

For bit oriented instruction b is 3-bit address, specifies the no. of bits affected by the operation.
For byte oriented instruction, f is 7-bit file register address and 'd' is the destinator which specifies where the result is to be placed i.e. if d=0, result is in W (8-bit) and if d=1, result is in f i.e. file register. for literal and control instruction, k is 8-bit or 11-bit constant or literal value.

Table I. Instruction

Mnemonics	Operands	Opcode	Description
Single Bit Manipulation			
bcf	f,b		Clear bit b of register f, where b=0 to 7
bsf	f,b		Set bit b of register f, where b=0 to 7
Data Transfer and Clear			
clrw			Clear W
clrf	f		Clear f
movlw	k		Move literal value to W,putting result into W
movwf	f		Move W to f
movf	f,F		Move f to F
movf	f,d		Move f to W or F depending on d bit
swapf	f,d		Swap nibbles of f, putting result into F or W
Increment / Decrement / Complement			
incf	f,d		Increment f, putting result in F or W reg
decf	f,d		Decrement f, putting result in F or W reg
comf	f,d		Complement f, putting result in F or W reg
Logical			
Andlw	k		AND literal value into W
Andwf	f,d		AND W with f, putting result in F or W reg
Iorlw	k		Inclusive-OR literal value into W
Iorwf	f,d		Inclusive-OR W with f, putting result in F
Xorlw	k		Exclusive-OR literal value into W
Xorwf	f,d		Exclusive-OR W with f, putting result in F
Arithmetic			
Addlw	k		Add literal value into W
addwf	f,d		Add w and f, putting result in F
sublw	k		Substract W from literal value, putting result in W
subwf	f,F		Substract W from f, putting result in F
Rotate			
rlf	f,F		Copy f into F, rotate F left through carry bit
rrf	f,F		Copy f into F, rotate F right through carry bit
Conditional Branch			
btfsc	f,b		Test bit b of register f,where b= 0 to 7; skip if clear
btfss	f,b		Test bit b of register f,where b= 0 to 7; skip if set
decfsz	f,d		Decrement f, putting result in F or W reg, skip if zero
incfsz	f,d		Increment f, putting result in F or W reg, skip if zero

Set.

The register for instruction is 14 bit wide.

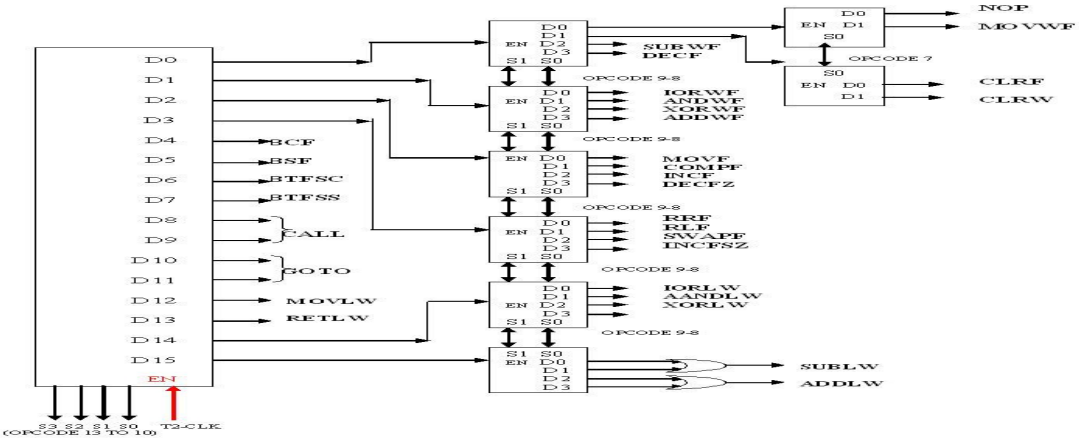

Figure2.Diagram of instruction Decoder.

2.3 Implementation of instructions:-

2.3.1 Decoder Unit.

Upper byte of the instruction is used for decoding purpose. All the instructions are decoded into T2 state. For the instruction decoder unit, upper 4-bits of the
instruction register i.e. 13-10 are used as select lines for 4:16 decoder, next two bits i.e.9-8 are used as select lines for2:4 decoder, next one bit i.e.7th bit is used as select lines of 1:2 decoder. Overall, first 4 bits, next 2 bits and next 1 bit, of opcode of the instruction are used as select lines of decoder and instructions are generated. This gives unique address of the instruction.T2-state is ANDed with ORing all instructions, to enable the decoder.

2.3.2. Arithmetic Unit:-

The arithmetic unit implements 8 instructions like ADDLW, ADDWF, and SUBWF etc. are ORed together and ANDed withT3-state to enable 3, 16:1 multiplexer. Two 16:1 multiplexer are used, one gives 8-bit value of w i.e. 'a' and 2ndmultiplexer gives,8 bit value of f.3rdmux.is used to add the carry, if required.

Eight bit byte adder is used to add the values of 8-bit w register and 8-bit value of Literal i.e .f, with
carry if required. O/p of byte adder is also available in 2's complement form. Opcode bits 11-8 are used as select lines of multiplexer and upper 2-bits i.e. 13-12 are used as select lines of 2:1.

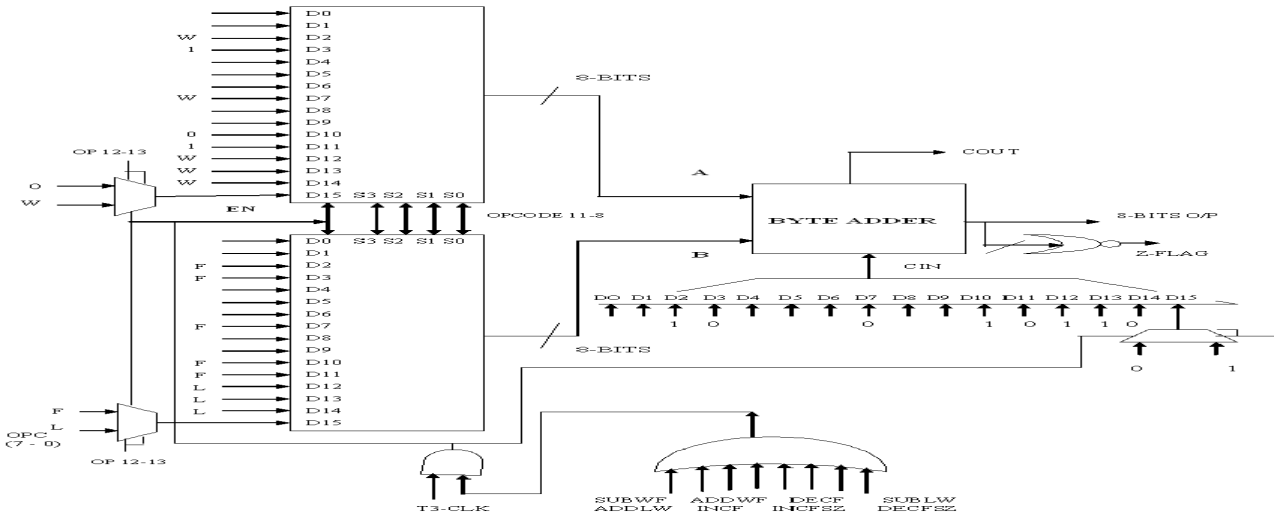

Figure3.Block Diagram of arithmetic unit

2.3.3 Logic Unit:-

Logic Unit is responsible for implementing logical instruction, such as IORLW, COMF, ANDLW, IORWF, ANDWF, and XORLW AND XORWF. All the instructions are ORed together and ANDed with T3-state to enable the 8:1 multiplexer.10-8 bits of opcode is used as select lines of the multiplexer,8:1 for COMF and ANDLW instruction, the opcode bits 10-8 are same value, so higher 2-bits i.e. 13-12 are used as select lines of the 2:1 mux. The NOR gate is used to generate the zero flag.

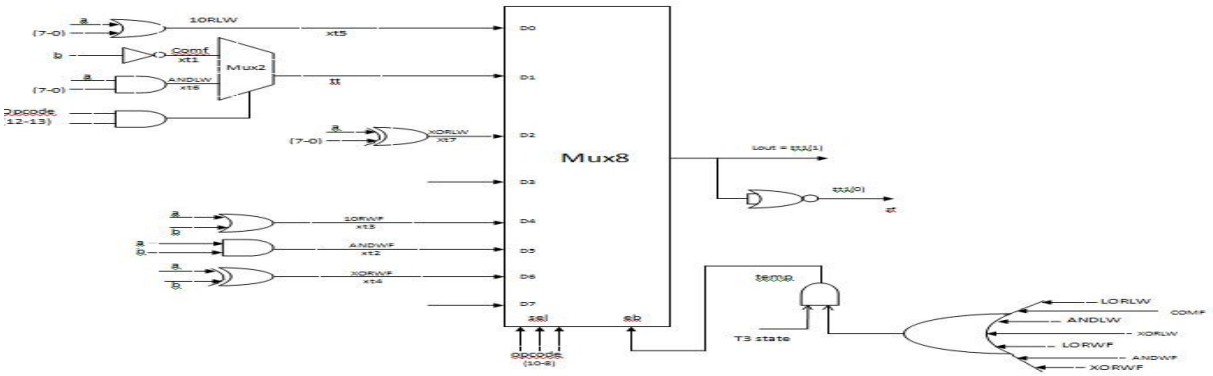

Figure4. Block Diagram of Logic Unit

2.3.4. ROTATE /SHIFT UNIT:-

This unit performs rotate and shift operations. The instructions which are implemented by using this unit are NOP, CLRW, CLRF, RRF, RLF and SWAPF. For NOP-no operation instruction, the contents of file register 'b' are transferred as it is. Opcode 10-8 are used as select lines .CLRF,CLRW has addresses 001,which are grounded pin;d_2 and d_3 are not used so d_1, d_2 , d_3 are connected to ground .For d_4 ,there is RRF instruction i.e. contents of f means b bits are rotated right through carry. So carry is in 8th multiplexer. For d_5, there is RLF i.e. contents of f register bits are rotated left, so there is carry in the lower, 1stmultiplexer.For d6, SWAP instruction, is executed, swapping bits of b d_7 is not used, so grounded. All the instructions ORed together and ANDed with T_3 –state and this signal is used as enable the multiplexer.

2.3.4 BIT SET AND BIT CLEAR OF FILE REGISTER UNIT:

In Bit set and clear unit, two instructions are implemented that is BCF and BSF.BCF is bit clear and BSF is bit set f .When BCF that is bit clear file is enabled and T_4 is high, then een1 is one, active low decoder is enabled.BSF i.e. bit set file is disabled that is een_2 is 0, active high decoder is disabled.

3. PERFORMANCE MEASURES

Following are the performance measures of the implemented RISC Processor.

1. The VHDL code for individual mode is downloaded on SPARTAN-III.SPARTAN-III is selected because, it is designed for high volume, increasing amount of logic resources, capacity of internal RAM, low cost.

2. While downloading the VHDL code using Xilinx simulator, there is improvement in speed .Present speed is 20MHz.But with 8-bit series, proposed microcontroller gives the speed of 35.6MHz

3. PIC Microcontroller so far designed gives the speed from 12MHz to 24MHz maximum, but using the FPGA i.e. SPARTAN-III, Simulation result gives the speed of execution of instruction as 35.6 MHZ.

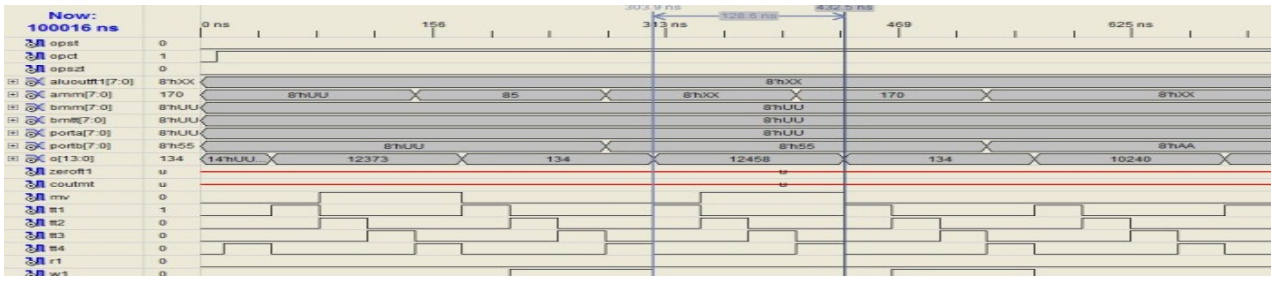

Figure5. Simulation Result of Final Module

4.The Number of instruction, which may be implemented by using this proposed microcontroller will be **5.**By using data bus width of 8-bit ,and instruction register-14 bit wide, there is increase in speed, as well as there is increase in performance to execute the more number of instruction.

4. APPLICATIONS

As the new concept of implementing the RISC processor using VHDL coding with the use of FPGA, there is a wide range for the development of boards, which implements the more number of instructions.

The main advantage of preferring VLSI design is single chip solution, which is supporting to create our own processor. VLSI has made possible to have a digital hardware implementation, which can be changed as per the requirement and application based too.

The designed core RISC PIC16F84 can be used as IP core with its own instruction set. As the architecture of the proposed processor is developed using VHDL, the architecture is flexible. Modifications are possible through coding for port change, port enhancement, and addition of instructions through assigning addresses.

5. CONCLUSIONS

The VHDL code development of individual modules and VHDL code for the combined module downloading on FPGA, simulation result gives the improvement in the speed of execution of instructions.

By running one application, VHDL code verifies the operation of the RISC processor.

REFERENCES

1. Kui YI and Yue-Hua DING, " 32 bit multiplication and Division ALU Design Based on RISC Structure", Wuhan Polytechnic University proc. 2009 International Joint Conference on ArtificalIntelligence,2009, pp. 761-764.

2. Rohit Sharma, Vivek Kumar Sehgal, Nitin, Pranav Bhasker and Ishitaverma," Design and Implementation of a 64-bit RISC Processor using VHDL, Jaypee University , proc. Uksim, 11th International Conference on Computer Modeling and Simulation,2009,pp 568-573.

3. Xia Li LongweiJi , Bo Shen, wenhong Li and Qianling Zhang,, " VLSI Implementation of a high – performance 32-bit RISC microprocessor", Fudan University,2002,pp.1458-1461.

4.Mamun Bin IbneKeaz, Md. Shabiul Islam, Mohd. S. Sulaiman, " A single Clock cycle MIPS RISC processor Design using VHDL",multimedia university, 2002 pp 199-203.

5.Data sheet of PIC Microcontroller 16F84.