Volume : 4 | Issue : 1 | January 2015 • ISSN No 2277 - 8179

Research Paper

FPGA Based Implementation of 16 bit RISC Microcontroller

Engineering

KEYWORDS : 16 bit microcontroller, RISC, VHDL,FPGA,Xilinx.

Nilam Patel	Dept. Of E&TC ,D.N.Patel Collage Of Engineering, Shahada, NandurbarIndia.
Prof. J.H.Patil	Dept. Of E&TC ,D.N.Patel Collage Of Engineering, Shahada, Nandurbar India.

ABSTRACT

This project describes the design and implementation of some of the internal hardware components of a microcontroller. The subsystems designed are the fundamental hardware components necessary to create the 16-bit timer's input capture and output compare modes, the index register, and other systems. The subsystems designed were the clock controller, the 16-bit timer, the register controller, the register, and the comparator. The project engineers implemented their system using Xilinx to test their logic and VLSI to construct the gates. VHDL code was written in order to implement the project onto an FPGA board.

INTRODUCTION

When the controller design become more complex inCISC and the performance was also not up to expectations,people started looking on some other alternatives. It hadbeen found that when a processor talks to the memory thespeed gets killed. So the one improvement on CPI was tokeep the instruction set very simple. Simple in not the wayit works but the way it looks. That's why there are veryfew instructions in any typical RISC architecture whereprocessor asks data from memory probably not other thanLoad and Store. At the end the pipelining added a newdimension in the speed just with the help of some Addition nal registers, which increases throughput byreducing CPI. Hence the instruction can be executedeffectively in one clock cycle[1].

A common misunderstanding of the phrase "ReducedInstruction Set Computer" is the mistaken idea thatinstructions are simply eliminated, resulting in a smallerset of instructions. In fact, over the years, RISC instructionsets have grown in size and today many of them have alarger set of instructions than many CISC CPUs. The term"Reduced" in that phrase was intended to describe the factthat the amount of work any single instructionaccomplishes is reduced at most a single data memorycycle compared to the "complex instructions" of CISCCPUs that may require number of data memory cycles inorder to execute a single instruction[2]. Mostmicroprocessors in today's market are based on eitherRISC or CISC architectures. Research has shown thatRISC architecture greatly boosts computer speed by usingsimplified machine instructions for frequently usedfunctions. The following features typically found in RISCbased systems.

1) Pre-fetching: The process of fetching nextinstruction or instructions into an event queue before thecurrent instruction is complete is called pre-fetching.
2) Pipelining: Pipelining allows issuing an instructionprior to the completion of the currently executing one.
3) Superscalar operation: Superscalar operation refers toa processor that can issue more than one instructionsimultaneously.

ARCHITECTURE

The objective of the project is to design a 16-bit RISCprocessor which utilizesminimum functional units. The architecture of proposed16-bit Processor is shown in Fig.1. The processorincorporates 16-bit ALU capable of performing 11arithmetical and logical operations, 16-bit programcounter, 24-bit Instruction register, Sixteen 16-bit generalpurpose registers, 3-bit flag register to indicate carry, zeroand parity.

The processor has four states idle, fetch, decode andexecute. The control unit provides necessary signalinteraction to perform expected function in all the states.

The main objective of this project is to design a RISC microcontroller usingVHDL and implement it in an FPGA. The microcontroller instruction set and featuresare based on Atmel AVR AT90S1200 RISC microcontroller.

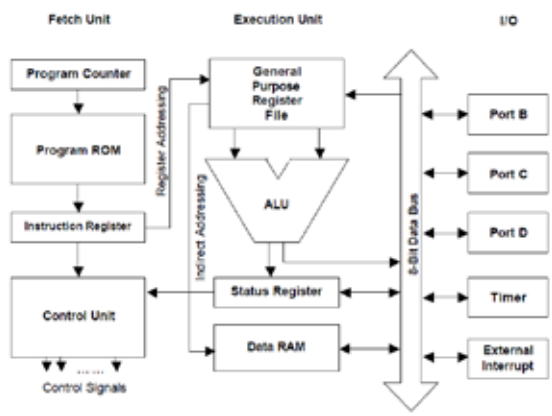

Fig. 1. Architecture Overview

shows the top-level block diagram of the design, the bus structure hasbeen simplified, but every block represents a module to be designed. At first glace, thereare 11 modules in the top-level, with the 3 ports sharing the same module. These 11modules are to be design separately using the top down design approach. Some moduleslike the instruction register and status register are easy to design, but modules like ALUand the control unit require a lot of understanding. The overall dataflow and busstructure between all the modules must be understand before designing the modulesindividually.

There are basically two kinds ofbuses, direct bus and common bus. Direct bus connects two modules directly and is usedspecifically by the connected modules. There are many direct buses, such as theconnection between program counter and program ROM, between program ROM andIR, between register file and ALU, etc. No control signals are required for direct buses.

The data bus is the onlycommon bus in this design. The data bus provides connection between the generalpurpose register file, ALU, status register, SRAM and all the I/O features. The registerfile can only receive data from the data bus. All others modules can receive and senddata to the data bus. Since there are so many possible data flows, control signals arerequired to control the correct flow direction. Only one source to the data bus is allowedat a time. If not, logic contentions will happen and the value of the data bus will beinvalid. Tri-state bus is used to implement the common data bus. The impedance isso high that it can be seen as unconnected to the bus system. If the ALU is

Volume : 4 | Issue : 1 | January 2015 • ISSN No 2277 - 8179

the datasource, the data bus will be flooded with the result of the ALU and is available to all theconnected modules. Control logic will generate an enable signal for the real destinationto receive the data.The system can be divided into3 units, the fetch unit, execute unit and I/O unit. Fetch unit is in charge of fetching thenext instruction and the execute unit is in charge of executing the current instruction. I/Ounit provide a connection with the outside world. The fetch unit and execute unit formthe CPU of the microcontroller.The first module of the fetch unit is the program counter (PC). The PC containsthe address of the next instruction to be executed. It points to the program ROM tolocate the instruction. The instruction from the ROM is then latched into the instructionregister (IR). The control unit takes the content of the IR and decodes it. It then assertthe appropriate control signals to execute the instruction. All modules are connectedwith direct buses.

The execute unit in charge of executing most instructions. Normally, to executean instruction, 2 operands are output from the register file to the ALU. The ALU thenperform the operation and send the result to the data bus. Contents of the data bus (theresult) is then stored back to the register file. The ALU also evaluate the status registerflags and send them directly to the status register (SR). The whole execution process isdone in a single cycle. The ALU perform many operations - include passing the contentsof a general register to the data bus. SR also has a direct bus connection to the controlunit required for branch evaluation. The register file isaddressed directly by some bits in IR.

INSTRUCTION SET

The operation of the CPU is determined by the instruction it executes, referred toas machine instructions or computer instructions. The collection of different instructionsthat the CPU can execute is referred to as the CPU's instruction set. Since the instructionset defines the datapath and everything else in a processor, it is necessary to study itfirst.

Table 1 shows the instruction set summary of the designed microcontroller,while the instruction set summary of the origi-

Mnemonic	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS			
ADD	Add Two Registers	S,Z,C,N,V,H	1
ADC	Add with Carry Two Registers	S,Z,C,N,V,H	1
SUB	Subtract Two Registers	S,Z,C,N,V,H	1
SUBI	Subtract Constant from Register	S,Z,C,N,V,H	1
SBC	Subtract with Carry Two Registers	S,Z,C,N,V,H	1
SBCI	Subtract with Carry Constant from Register	S,Z,C,N,V,H	1
AND	Logical AND Registers	S,Z,N,V	1
ANDI	Logical AND Register and Constant	S,Z,N,V	1
OR	Logical OR Registers	S,Z,N,V	1
ORI	Logical OR Register and Constant	S,Z,N,V	1
EOR	Exclusive OR Registers	S,Z,N,V	1
COM	One's Complement Register	S,C,Z,N,V	1
NEG	Negate (2's Complement) Register	S,C,Z,N,V,H	1
SBR	Set Bit(s) in Register	S,Z,N,V	1
CBR	Clear Bit(s) in Register	S,Z,N,V	1
INC	Increment	S,Z,N,V	1
DEC	Decrement	S,Z,N,V	1
TST	Test for Zero or Minus	S,Z,N,V	1
CLR	Clear Register	S,Z,N,V	1
SER	Set Register	None	1

Addressing Modes

There are 7 addressing modes in the microcontroller. Rd and Rr are devoted tothe destination register and soure register.

1. Direct Single Register AddressingThe operand is in Rd.
2. Direct Double Register AddressingThe operands are in Rd and Rr. Result is stored back to Rd.
3. I/O Direct AddressingFirst operand is one of the I/O registers. The address is contained in 6 bits ofthe instruction word. The second operand is either Rd or Rr. Used by IN andOUT instructions to read from or write to the I/O registers.
4. Data Indirect AddressingOperand address is the contents of the Z-register. Used when accessing theSRAM with LD and ST instructions.
5. Data Indirect Addressing with Pre-DecrementZ-pointer is decremented by 1 before the operation. Operand address is thedecremented contents of the Z-register. Used when accessing the SRAMwith LD and ST instructions.
6. Data Indirect Addressing with Post-IncrementThe Z-register is incremented by 1 after the operation. Operand address is theoriginal content of the Z-register before increment. Used when accessing theSRAM with LD and ST instructions.
7. Relative Program Memory AddressingProgram execution continue at address PC + offset. The offset is contains inthe instruction word. Unconditional branch instructions (RJMP, RCALL) canreach the entire program memory from every location. However, conditionalbranch instructions can only reach –64 to 63 locations away from the currentaddress.

FINITE STATE MACHINE STATES

Fig.2 shows the state diagram of the finite state machine (FSM). The 8states are EXE (execute), SLEEP, BRANCH1, BRANCH2, SBICS (skip if bit in I/Oclear/set), CBISBI (clear/set bit in I/O), ST and LD.

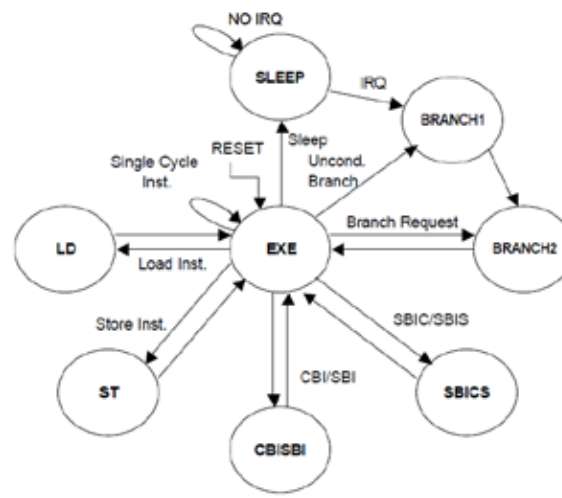

Fig.2. State Diagram

The state diagram shows the state flow but does not clearly show the inputs. Theinputs to the FSM are the 46 output lines of the instruction decoder, timer IRQ, externalIRQ, skip request and branch request. Branch request is generated by the branchevaluation unit when the condition of the conditional branch instruction is fulfilled.

SIMULATION RESULTS

The fig. 3 shows the simulation results for MVI instruction.The instruction MVI R1 0005 is written at address 0000hof instruction memory. In the decode process destinationregister 'Rz' is assigned with R1 and 'immediate_value' isassigned with 0005. At the next positive edge of the clockcycle when 'reg_wr' signal goes high, the value 0005indicated by 'reg_wr_data' is written into the register R1.

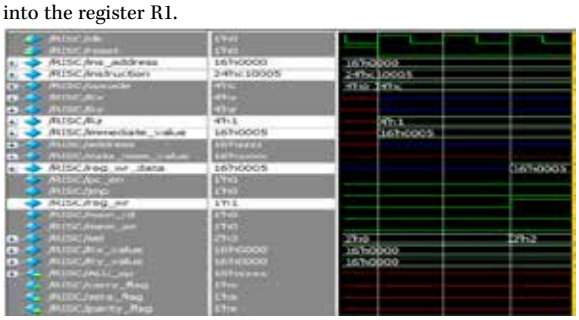

Fig. 3 MVI

CONCLUSION AND FUTURE WORK

As a conclusion, this project has been completed successfully fulfilling are theobjectives and scopes specified. The author has used his extra time to optimized thespeed of the design until 12 MHz. The data RAM that is not specified in the scope of theproject has also been included. Hardware stack is enlarged to 4-level instead of 3 and atotal of 24 I/O lines are available.

The design can be improved in number of ways. Toachieve a more sophisticated design more features can beadded to the current design. The number of instructionsthat the processor supports can be increased. Pipeliningcan be added to improve the performance of the proposeddesign.

REFERENCE

[1]. Mamun B, Shabiul I. and Sulaiman S,"A SingleClock Cycle MIPS RISC Processor Designusing VHDL" | [2]. Hamblen J." Synthesis, Simulation, andHardware Emulation to Prototype a PipelinedRISC Computer System" | [3]. Zainalabedin N,"VHDL for Modeling and Designof Processing Units" | [4]. Takanori M, Satoshi A and Masaaki I, "AMultithread Processor Architecture Based on theContinuation" | [5]. Kasuga-Koen, Kasuga, Fukuoka, " The InnovativeArchitecture for Future Generation HighPerformanceProcessors and Systems" | [6]. Virendra S. and Michiko I,"Instruction-BasedSelfTesting of Delay Faults in PipelinedProcessors", IEEE Transaction on VLSI systems,vol. 14, no.11,pp.1203-1215. | [7]. Patterson A. and Hennessy J,"ComputerOrganization & Design", Morgan KaufmannPublishers, 1999 | [8]. Peter J Ashenden,"Digital Design, An embeddedsystems approach using Verilog", MorganKaufmann Publishers,2010 | [9]. Ramesh Gaonkar,"Microprocessor architecture,programming and applications with the 8085",Penram International Publishing,1989 |