

Efficient Architecture for Computer Hardware Based On RISC & Harvard Architecture

Ankit Birle¹; Purnima Khandelwal²

Swami Vivekanand College of Engineering, Indore ^[1]; Oriental University, Indore^[2]

I. Abstract

Now a day's computer is a UN divisible part of our life. A computer is not just a computer its part of our life there are so many gadgets by which we are start our daily life like electrical tooth brush mobile alarm clock they all having some processor and a proper architecture for process there work our aim to increase their efficiency and speed so its save our important time. In this paper we show the simulation of 16 bit RISC processor with pipelining and different memory allocation for data and instruction. The design architecture is written in VERILOG Hardware Description Language (VERILOGHDL) code using Xilinx ISE tool for synthesis and simulation Processor is basically the brain of our computer. The term processor generally replaced the term Central Processing Unit (CPU). Today microprocessors can be found in almost every digital system. The decision to include the microprocessor has been made to transform the design efforts from a logic design into a software design. Our processor is RISC (Reduced instruction set computer) based and here we use a Harvard architecture for memory. The RTL views and verified simulation results of processor are shown in this paper. The synthesis report of the design is also described the timing of output after input is given to them.

Keywords:-Pipelining, RISC, VERILOGHDL, Xilinx, Harvard architecture.

II. INTRODUCTION

Our original goal was to create an 8-bit ALU, but as the design progressed, it was decided to produce a 16-bit ALU, which is more useful in computation. This larger design included the data path for a CPU containing Program Counter (PC), Memory Address Register (MAR), Memory Data Register (MDR), Instruction Register (IR), Register File (RF) and other various Components to tie them all together. The processor of the CPU was implemented using Verilog. It's a 16 bit RISC based processor which is pipelining and different space for data and instruction which is useful for execution of any instruction and Pipelining is used for increases the speed of execution because in pipelining according their stages instruction run simultaneously and we get output after every cycle. RISC instruction set help user to program an efficient designing for any architecture. 8 Bit processor is low speed and used only in low cost

Applications but 16 Bit processor so high speed and also consume low power ad used in many applications. FPGA is used for The implementation of such an instruction set would take up less real estate on the chip (or FPGA) and will allow more peripherals to be fabricated on a single chip making it ideal for a System-On-Chip (SOC) implementation of an application. It will also be beneficial in embedded system design where a custom processor core implementation is required with tight instruction requirements so that it takes less space on a FPGA. The first part of paper show the instruction set which we are used to design for the processor and then we show about simulation and RTL schematic of that processor and finally the waveform and resulting time for any execution of instruction set.

III. DESIGN ARCHITECTURE

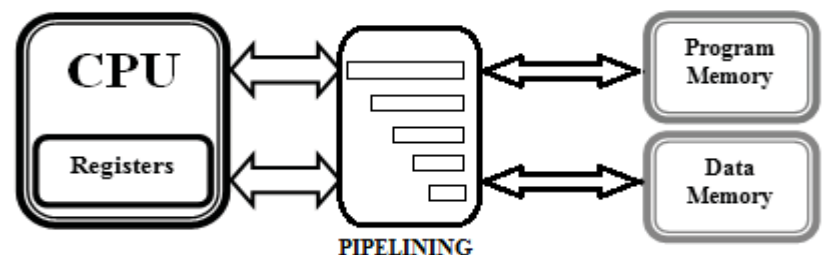


Fig. 1 ARCHITECTURE

The processor is based on Harvard architecture for memory storage and access in which data and instruction use a different location for storage so it's the time of execution it's also avoid overlapping condition and increases performance of processor here we also use a concept of pipelining means output is come after every cycle we need not to wait for execution of any instruction next instruction will come for execution just after a one cycle so also increases speed and reduce the timing of execution and also nullify delay.

IV. INSTRUCTION SET

RISC type instruction set are used here for reduce instruction size or in RISC type instruction set one instruction is run in a single cycle. It's a user friendly means user use a simple type instruction for any operation and future work is done by processor itself.

Name	Instruction	Effect(s)
nop	nop	Do nothing (instruction word = 0)
add	add rd, rs	$R[rd] = R[rd] + R[rs];$
sub	sub rd, rs	$R[rd] = R[rd] - R[rs];$
addu	add rd, rs	$R[rd] = R[rd] + R[rs];$
subu	sub rd, rs	$R[rd] = R[rd] - R[rs];$
mov	mov rd, rs	$R[rd] = R[rs];$
and	and rd, rs	$R[rd] = R[rd] \text{ and } R[rs];$
or	or rd, rs	$R[rd] = R[rd] \text{ or } R[rs];$
nand	nand rd, rs	$R[rd] = R[rd] \text{ nand } R[rs];$
nor	nor rd, rs	$R[rd] = R[rd] \text{ nor } R[rs];$
xor	xor rd, rs	$R[rd] = R[rd] \text{ xor } R[rs];$
not	not rd, rs	$R[rd] = \text{not } R[rs];$
jr	jr rs	$pc = R[rs];$ (<i>rd=R0, i.e. no writeback</i>)
jlr	jlr rd, rs	$R[rd] = (pc+1); pc = R[rs];$ (<i>normally rd=R7</i>)
lw	lw rd, rs, immed	$R[rd] \leftarrow \text{memory}(\text{sign_ext}(\text{immed})+R[rs]);$
sw	sw rd, rs, immed	$R[rs] \rightarrow \text{store word}(\text{sign_ext}(\text{immed})+R[rd]);$
lwi	lwi rd, immed	$R[rd] = \text{sign_ext}(\text{immed});$ (<i>rs is unused</i>)
addi	addi rd, rs, immed	$R[rd] = R[rs] + \text{sign_ext}(\text{immed});$
beq	beq rd, rs, immed	if $(R[rd] == R[rs])$ $pc += \text{sign_ext}(\text{immed}) + 1;$
bne	bne rd, rs, immed	if $(R[rd] != R[rs])$ $pc += \text{sign_ext}(\text{immed}) + 1;$
blt	bgtz rd, rs, immed	if $(R[rd] < R[rs])$ $pc += \text{sign_ext}(\text{immed}) + 1;$
bgt	bgtz rd, rs, immed	if $(R[rd] > R[rs])$ $pc += \text{sign_ext}(\text{immed}) + 1;$

TABLE 1: INSTRUCTION SET FOR THE PROCESSOR

The uP16 is a very basic 5-stage pipelined processor. It has 2048 bytes of 18-bit instruction memory, 2048 bytes of 16-bit data memory and an 8 x 16-bit register file. Only full word accesses are supported (byte accesses are not supported, although it would be fairly easy to allow for byte accesses). The program counter (PC) is 16 bits and indicates the next memory instruction to execute. As each instruction is executed, the PC increments by 1 (as only word accesses are allowed). On reset the PC is set to 0. There is no interrupt mechanism in the uP16 CPU.

Field	4 bits	3 bits	3 bits	8 bits
R-format	OpCode			Function/Immed8/Displacement8
J-format	OpCode	Target address (14 bits)		

TABLE 2: INSTRUCTION FORMAT

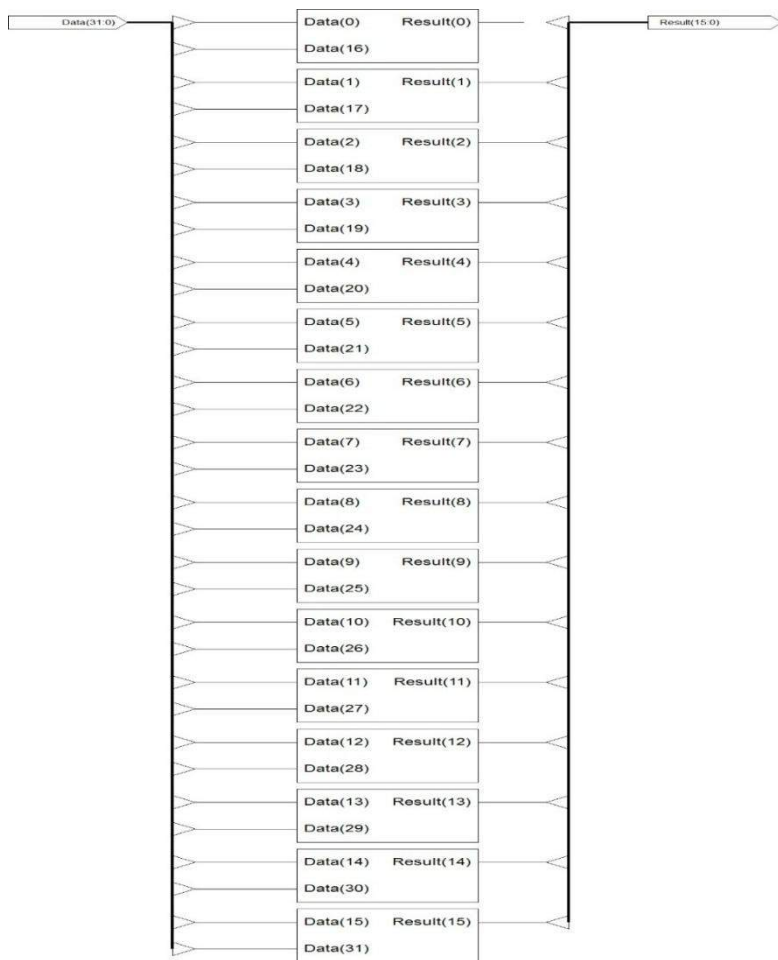
V. METHODOLOGY

There are several methodology use to design a processor here we use a reduced instruction set rather than a complex one so it's to program. Basically there are four stages for taking a output of any instruction first is fetching a data from the input side secondly decode it's into machine language execute them into proper way and finally now we are able to get a result. This are the main procedure for any instruction. Here we also use a concept of pipelining that means we get output after any cycle here we use 5 stage pipelining so all four stages done simultaneously and we get output continuously after fifth cycle.

VI. DESIGNING

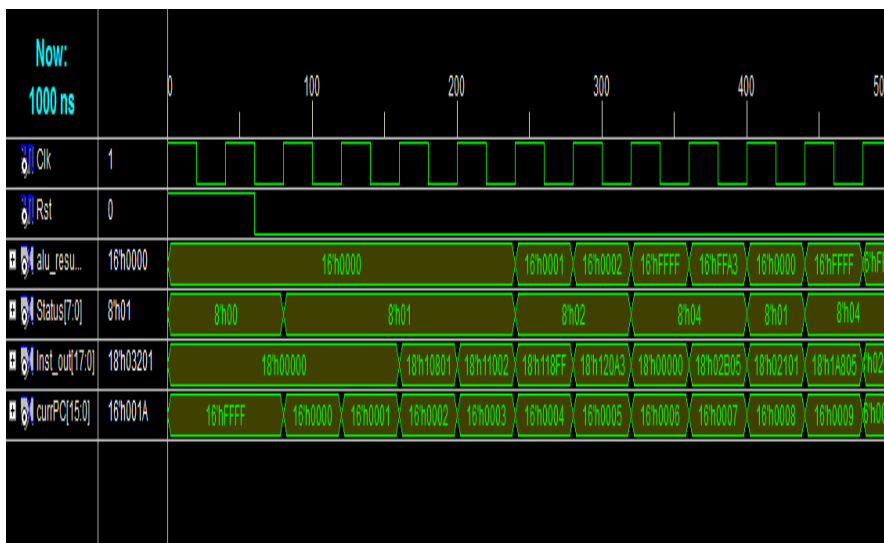
- **System operations and interaction between different units**

Here we use a Harvard architecture for memory so there are two different storage element and the final stage is shown by program counter. Program counter is store a next address which are going to execute and also wait and store an address till the previous instruction is going on. There are a set of resistor which are use to store a data at run time and also store a result these resistor are called as general purpose resistors. And a main block of ALU is also there which are perform all the arithmetic operation like addition subtraction etc. Here one block is separate xorblock. Apart from this there is rom and ram block are also use in this processor storage and use at run time now come to the implementation part here we use a FPGA. Field programmable gate array is made by logical devices and use as a blank chip for storage any program it is better than CPLD because in CPLD there are so many PLA types block and each having so many programmable gates so it so much complex for programing purpose but having advantage that CPLD is a nonvolatile devices means it store a data even after power is off. So for the processor we use a FPGA rather than complex CPLD.



VII. RESULT AND WAVEFORMS

For taking a result and simulate it there is a software called Xilinx is used here and for simulation Xilinx ISE simulation tool was used at the output just saw a waveform of status of resistor clock reset program counter and ALU performance according to input instruction.



Because of use of Harvard architecture and pipelining for instruction and mostly use of RISC instruction set for programing, response time was so less compare to normal processors timing result as shown as follow these are come under syntheses report of processor.

Timing Detail

All values displayed in nanoseconds (ns)

Timing constraint:

Default period analysis for Clock 'Clk'

Clock period: 12.069ns (frequency: 82.857MHz)
 Total number of paths / destination ports: 49620 / 408

Delay: 12.069ns (Levels of Logic = 11)
 Source: TOP_2/EX_Sel_ALUSrc2 (FF)
 Destination: TOP_3/ALU_Status_0 (FF)
 Source Clock: Clk rising
 Destination Clock: Clk rising

Total 12.069ns (7.737ns logic, 4.332ns route)
 (64.1% logic, 35.9% route)

In fig. 3 diagram show the all four stages like fetching data, decode data, execution and store in memory location it also show the ALU result status of program counter.

VIII. CONCLUSION

Processor are used to perform a give task within a given time period or as soon as possible now a days there are so many processor are available in a market. So many processor with so many configuration even better than this 16 bit but it is too costly but efficient 16 bit processor are also use in so many application which are short time application no so much complex like toys some mobiles etc. It is good choice for small application because 8 bit processor is not so much efficient in concern of timing and 32 bit is costly so this one is better choice for small size application.

IX. FUTURE WORK

For optimizing a more sophisticated result there is need to implemented more efficient processor by applying an Interrupt handling mechanism and increases the compatibility with other modules so it's directly use with other modules.

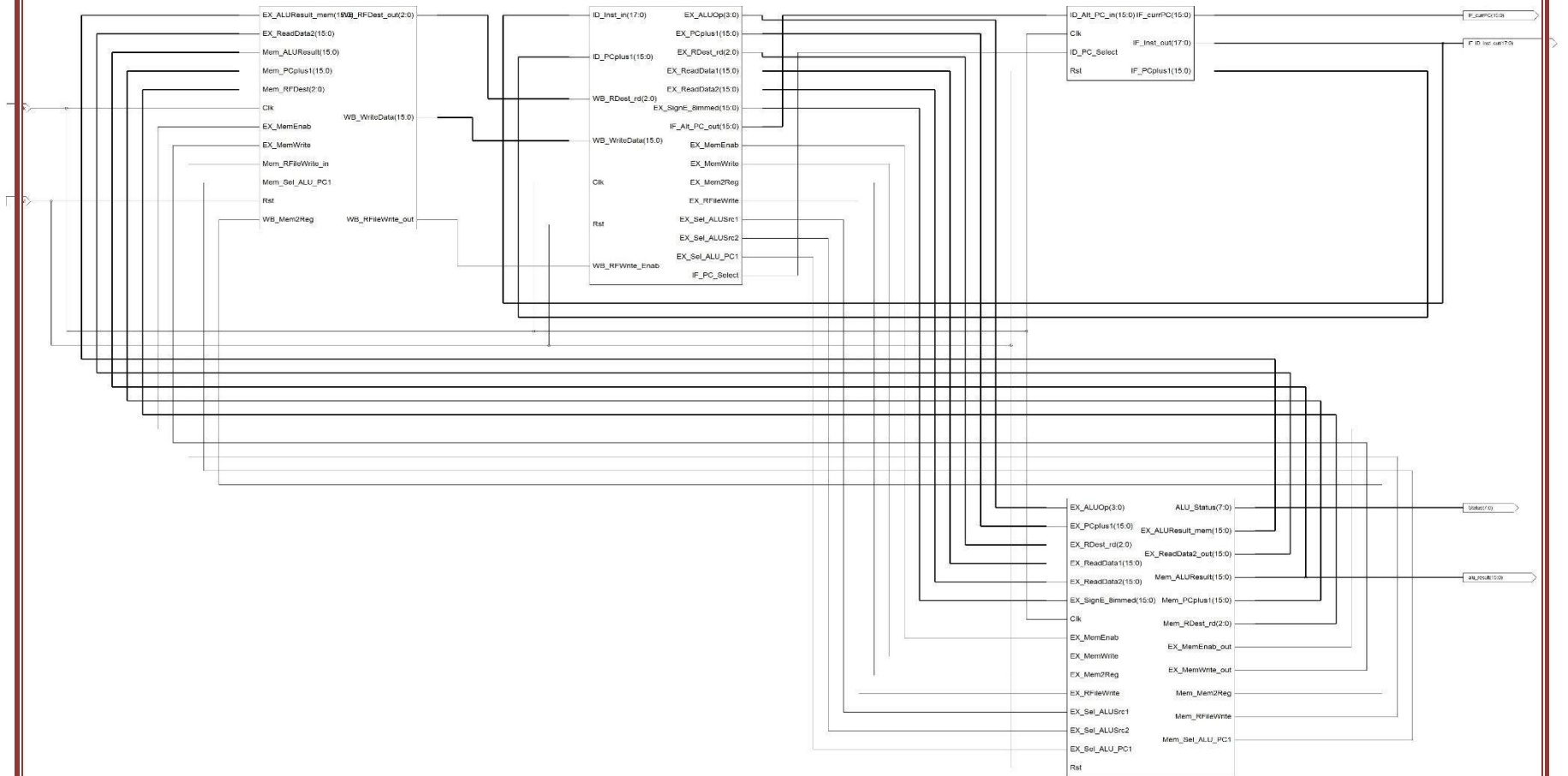


Fig.3 ALL 4 STAGES

X. REFERANCE

1. Antonio H. Zavala, Jorge Avante R., Quetzalcoatl Duarte R., J David Valencia P., "RISC-Based Architecture for Computer Hardware Introduction" IEEE Journal.
2. Anuruddh Sharma and MuktiAwad" A 16-BIT RISC PROCESSOR FOR COMPUTER HARDWARE INTRODUCTION",An International Journal (ESTIJ), ISSN: 2250-3498, June 2012.
3. DimitrisMandalidis, PanagiotisKenterlis, John N. Ellinas,"A computer Architecture Educational System based on a 32-bit RISC Processor", International Review on Computers and Software (I.RE.CO.S.).
4. SagarBhavsar, Akhil Rao, AbhishekSen, Rohan Joshi." A 16-bit MIPS Based Instruction Set Architecture for RISC Processor", International Journal of Scientific and Research Publications, Volume 3, April 2013.
5. R. Uma," Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool", International Journal of Engineering Research and Applications (IJERA) Vol.2, Issue 2, Mar-Apr 2012,pp.053-058