IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 2, September 2012)

Area Optimized 32-Bit Pipeline RISC Processor in VHDL

Swati Joshi
swati_joshi20@yahoo.co.in

Puran Gour
purangour@rediffmail.com

***Abstract* — RISC architecture is a solution of recent era for complex computation in SOC. As name itself suggest that there is less complicated control mechanism because of reduced instruction. In this paper RISC design has been proposed for lesser area and balance between high speed by pipeline approach. Design is implemented using VHDL.**

***Keywords*- RISC, SOC, pipeline, VHDL**

I. INTRODUCTION

Reduced Instruction Set Computers (RISC) is a high performance computing engines. The RISC processor is solution of drawbacks encountered in the traditional CISC processor architectures. Because of its advantages, such as its simple, flexible and fixed instruction format and hardwired control logic, paves for higher clock speed, by eliminating the need for microprogramming. The combined advantages of high speed, low power, area efficient and operation-specific design possibilities have made the RISC processor ubiquitous.

RISC architecture has been developed as a result of the 801 project which started in 1975 at the IBM T.J.Watson Research Center and was completed by the early 1980s [5]. RISC architecture starts with a small set of most frequently used instructions which determine the pipeline structure of the machine enabling fast execution of those instructions in one cycle. If addition of a new complex instruction increases the “critical path” (typically 12-18 gate levels) for one gate level, than the new instruction should contribute at least 6-8% to the overall performance of the machine. One cycle per instruction is achieved by exploitation of parallelism through the use of pipelining. It is parallelism through pipelining that is the single most important characteristic of RISC architecture from which all the remaining features of the RISC architecture are derived. Basically we can characterize RISC as a performance oriented architecture based on exploitation of parallelism through pipelining.

Designing of processor using VHDL

For developing the system various techniques are available like conventional technique of developing, micro-processor and microcontroller centered style etc and now a daily VLSI style technique is mostly recommended because of its no. of benefits, such as Little measurement, Low energy dissipation, Function at high-speed, Protection up to 20 decades, Re-training is possible for more than 20,000 times, Quick contingency development, It is having online reconfigurable considering, Far away finish development is possible, Time to promote is little, Price to performance quantity is excellent, Simple up gradations to new requirements. He key for choosing VLSI style is a single processor remedy, which is assisting to make our own processor. VLSI has made possible to have electronic components execution, which can be modified as per client need. Different illustrative dialects are available with different types of style access such as VHDL [Very high-speed incorporated routine Hardware Information Language], Verilog and ABEL.

II. CONCEPT OF RISC

Overall design procedure is composed of pipeline, stage analysis, instruction execution analysis, RT-level (Register Transfer Level) functional unit composition, and control signal generation.

A. Pipeline Stage

A conventional computer executes one instruction at a time with a Program Counter pointing to the instruction currently being executed. Pipelining is analogous to an oil pipeline where the last product may have gone in before the first result comes out.

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 2, September 2012)

We can apply pipelining to the classical fetch/execute instruction processing. There are three phases to the fetch/execute cycle:

• Instruction fetch
• Instruction decode
• Instruction execute

If we assume these all take one time unit (clock cycle) to execute a three stage pipeline will look like the following.

Fetch	I_1	I_2	I_3	I_4	I_5	I_6
Decode		I_1	I_2	I_3	I_4	I_5
Execute			I_1	I_2	I_3	I_4

Time#1 time#2 time#3 time#4 time#5 time#6

The core employs 3-stage pipeline such as fetch, decode, and execute. Although most of data processing instruction is executed in a single cycle, some other instructions take multiple clock cycles. The number of cycle in the execute stage is in the range of 1 to 17 are 3 execute operations, which is address computation, memory read, and register write., This multi-cycle execution makes next instruction delayed as additional cycles cycles. The pipeline operations for Memory transfer instruction are shown in Fig. 1. In execute stage, there result appears. This provides a way to start a task before the first

Cycle	1	2	3	4	5	6	7	8
Instruction								
Add	F	D	E					
STR		F	D	E	Date			
SUB			F		D	E		
BIC				F		D	E	
AND						F	D	E
EOR							F	D

(a)

Cycle	1	2	3	4	5	6	7	8	9
Instruction									
BL	F	D	E						
X		F	D	E					
XX			F	D					
LDR				F	D	E	Data	Write	
ADD					F			D	E
SUB						F			D
MOV									F

(b)

Fig.1. Instruction pipeline operation (a) STR (b) LDR

B. Instruction execution

In order to understand the execution operation of RISC instructions, we analyzed the data-path and the movement of operand for each instruction. From this analysis, we can find out partial control signals in execute stage of each instruction. The flow of a data processing instruction in the execute stage is illustrated in Fig:2. If register and data U0 units are connected with data-path and active buses, operand movement is carried out. Each instruction can be grouped into several types, such as data processing, data transfer, branch, and the others. Each type determines the execution operation of an instruction.

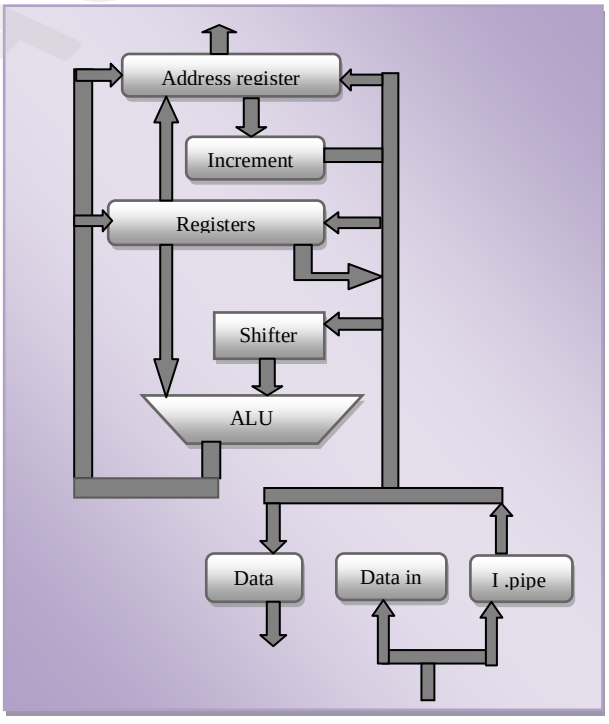

Figure2.Instruction data path activity

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 2, September 2012)

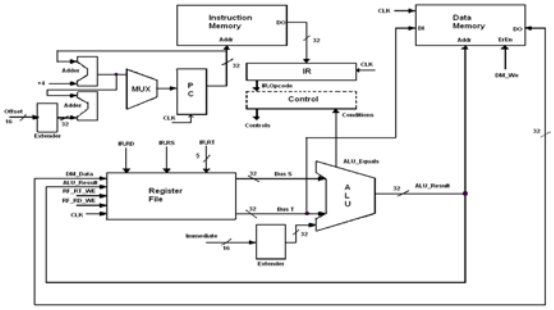

Figure.3.32-bit RISC architecture with pipeline

III. INSTRUCTION SET OF 32-BIT RISC PROCESSOR

Arithmetic Instructions			
Mnemonic	**Description**	**Byte**	**Cycle**
ADD A, Rn	Adds the register to the accumulator	1	1
ADD A, direct	Adds the direct byte to the accumulator	2	2
ADD A, @Ri	Adds the indirect RAM to the accumulator	1	2
ADD A, #data	Adds the immediate data to the accumulator	2	2
ADDC A, direct	Adds the direct byte to the accumulator with a carry flag	2	2
ADDC A, @Ri	Adds the indirect RAM to the accumulator with a carry flag	1	2
ADDC A, #data	Adds the immediate data to the accumulator with a carry flag	2	2
SUBB A, Rn	Subtracts the register from the accumulator with a borrow	1	1
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow	2	2
SUBB A, @Ri	Subtracts the indirect RAM from the accumulator with a borrow	1	2
Branch Instructions			
Mnemonic	**Description**	**Byte**	**Cycle**
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Returns from subroutine	1	4
RETI	Returns from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (from –128 to +127 locations relative to the following instruction)	2	3
JC rel	Jump if carry flag is set. Short jump.	2	3
JNC rel	Jump if carry flag is not set. Short jump.	2	3
JB bit, rel	Jump if direct bit is set. Short jump.	3	4
JBC bit, rel	Jump if direct bit is set and clears bit. Short jump.	3	4
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if the accumulator is zero. Short jump.	2	3

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 2, September 2012)

Bit-oriented Instructions			
Mnemonic	**Description**	**Byte**	**Cycle**
CLR C	Clears the carry flag	1	1
CLR bit	Clears the direct bit	2	3
SETB C	Sets the carry flag	1	1
SETB bit	Sets the direct bit	2	3
CPL C	Complements the carry flag	1	1
CPL bit	Complements the direct bit	2	3
ANL C, bit	AND direct bit to the carry flag	2	2
ANL C,/bit	AND complements of direct bit to the carry flag	2	2
ORL C, bit	OR direct bit to the carry flag	2	2
ORL C,/bit	OR complements of direct bit to the carry flag	2	2
MOV C, bit	Moves the direct bit to the carry flag	2	2
MOV bit, C	Moves the carry flag to the direct bit	2	3

Data Transfer Instructions			
Mnemonic	**Description**	**Byte**	**Cycle**
MOV A, Rn	Moves the register to the accumulator	1	1
MOV A, direct	Moves the direct byte to the accumulator	2	2
MOV A, @Ri	Moves the indirect RAM to the accumulator	1	2
MOV A, #data	Moves the immediate data to the accumulator	2	2
MOV Rn,A	Moves the accumulator to the register	1	2
MOV Rn,direct	Moves the direct byte to the register	2	4
MOV Rn,#data	Moves the immediate data to the register	2	2
MOV direct, A	Moves the accumulator to the direct byte	2	3
MOV direct,Rn	Moves the register to the direct byte	2	3
MOV direct, direct	Moves the direct byte to the direct byte	3	4
MOV direct, @Ri	Moves the indirect RAM to the direct byte	2	4
MOV direct,#data	Moves the immediate data to the direct byte	3	3
MOV @Ri, A	Moves the accumulator to the indirect RAM	1	3
MOV @Ri,direct	Moves the direct byte to the indirect RAM	2	5
MOV @Ri,#data	Moves the immediate data to the indirect RAM	2	3
XCHD A, @Ri	Exchanges the low-order nibble indirect RAM with the accumulator	1	3

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 1, Issue 2, September 2012)

IV. RESULTS

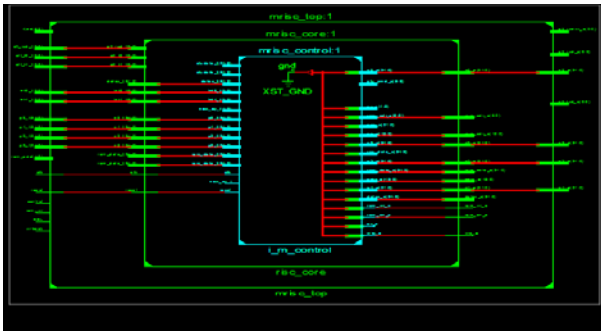

Figure.4.block diagram of RISC-32 bit processor

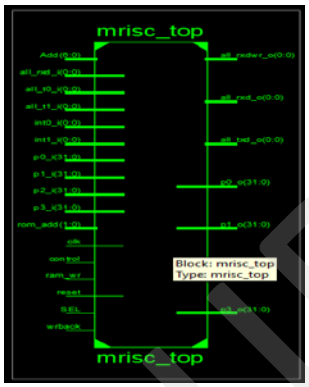

Figure.5.Pin diagram of RISC-32 bit processor

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1311	20480	6%
Number of 4 input LUTs	2429	40960	5%
Number of bonded IOBs	200	489	40%
Number of MULT18X18s	4	40	10%

Figure.6.device utilization

V. CONCLUSION

Here we design a three pipeline RISC processor of which simulation is done on modelsim and synthesis is carried out on device Xilings Xc3s2000-5fg676.

REFERENCES

[1]. Xia Li, Longwei Ji, Bo Shen, Wenhong Li and Qianling Zhang, "VLSI Implementation of a High-performance 32-bit RISC Microprocessor," International Conference on Communications, Circuits and Systems and West Sino Expositions, IEEE 2002, Vol. 2, pp.1458-1461.

[2]. Zhenyu Gu, Zhiyi Yu, Bo Shen and Qianling Zhang, "Functional Verification Methodology of a 32-bit RISC Processor," International conference on communication and circuit and systems and west Sino Expositions, IEEE 2002, Vol. 2, pp. 1454-1457.

[3]. Sivarama P. Dandamudi, "Fundamentals of Computer Organization and design" Springer-Verlag New York, Inc., 2003.

[4]. Yasuhiro Takahashi, Toshikazu Sekine, and Michio Yokoyama, “Design of a 16-bit Non-pipelined RISC CPU in a Two Phase Drive Adiabatic Dynamic CMOS Logic,” International Journal of Computer and Electrical Engineering, Vol. 1, No. , April 2009 1793-8198.

[5]. Samiappa Sakthikumaran et al., “A Novel Low Power and High Speed Wallace Tree Multiplier for RISC Processor”, 3rd International Conference on Electronics Computer Technology - ICECT 2011.

[6]. A 32-b RISC Implemented in Enhancement-Mode JFET Ga As Rasset, T.L.;Niederland,R.A.;Lane,J.H,Geideman,W.A.;Mc DonnellDouglas Astronautics Company, Huntington Beach, CA 92647 Date of Current Version: 27 March 2009

[7]. VHDL-based development of a 32-b pipelined RISC processor for educational Purposes Buhler, M. Baitinger, U.G. Stuttgart Univ. Date of Current Version: 06 August 2002