VLIW Architecture for High Speed Parallel Distributed Computing System

1.T.NAVEEN KUMAR REDDY, MTECH(VLSI) ,JBREC, shastrypnvm@gmail.com@gmail.com
2.P.NAVMSASTRY,Professor,JBREC , tanveen48@gmail.com
3. Dr.D.N RAO,Principal , JBREC, principal_jbr@yahoo.com
1,2,3.Joginapally BR Engineering College ,Yenkapally, Moinabad Mandal , R.R. District, Hyderabad Telangana State

Abstract- The aim of project is to design new processor architecture for accelerating data-parallel applications based on the combination of VLIW and Instruction level parallelism. The processor utilizes VLIW architecture for processing multiple independent scalar instructions concurrently on parallel execution units. The proposed processor, which is called VLIW Architecture for high speed parallel distributed computing system, has unified register file of 64x32-bit registers in the decode stage for storing scalar data. It can issue up to six scalar operations in each cycle for parallel processing a set of operands and producing up to six results. However, it cannot issue more than one memory operation at a time, which loads/stores 128-bit scalar data from/to data cache. Six 32-bit results can be written back into VLIW register file. The complete design of our proposed VLIW processor is implemented using VHDL targeting the Xilinx FPGA Virtex-5, XC5VLX110T-3FF1136 device. The required numbers of slice registers and LUTs are 3,379 and 11,977 (11,593 for logic and 384 for memory), respectively. The number of LUT-FF pairs used is 14,311, where 10,932 for unused flip-flops, 2,344 for unused LUT, and 1,045 for fully used LUT-FF pairs.

Keywords—VLIW architecture; Instruction-level parallelism; unified data path; FPGA/VHDL implementation

I. INTRODUCTION

One of the most important methods for achieving high performance is taking advantage of parallelism. The simplest way to take the advantage of parallelism among instructions is through pipelining, which overlaps instruction execution to reduce the total time to complete an instruction sequence. All processors since about 1985 use the pipelining technique to improve performance by exploiting instruction-level parallelism (ILP). The instructions can be processed in parallel because not every instruction depends on its immediate predecessor. After eliminating data and control stalls, the use of pipelining technique can achieve an ideal performance of one clock cycle per operation (CPO). To further improve the performance, the CPO would be decreased to less than one. Obviously, the CPO cannot be reduced below one if the issue width is only one operation per clock cycle. Therefore, multiple-issue scalar processors fetch multiple scalar instructions and allow multiple operations to issue in a clock cycle.

VLIW architectures are characterized by instructions that each specify several independent operations. Thus, VLIW is not CISC instruction, which typically specify several dependent operations. However, VLIW instructions are like RISC instructions except that they are longer to allow them to specify multiple independent simple operations. A VLIW instruction can be thought of as several RISC instructions packed together, where RISC instructions typically specify one operation. The explicit encoding of multiple operations into VLIW instruction leads to dramatically reduced hardware complexity compared to superscalar. Thus, the main advantages of VLIW is that the highly parallel implementation is much simpler and cheaper to build the equivalently concurrent RISC or CISC chips.

The main objective of this project is design new processor architecture for data-parallel applications. Based on the combination of VLIW and instruction-level parallelism. It is based on VLIW architecture for processing multiple scalar instructions concurrently. Moreover, data-level parallelism (DLP) is expressed efficiently using instructions and processed on the same parallel execution units of the VLIW architecture.

II . THE ARCHITECTURE OF VLIW PROCESSOR

It supports few addressing modes to specify operands: register, immediate, and displacement addressing modes. In the displacement addressing mode, a constant offset is signed extended and added to a scalar register to form the memory address for loading/storing 192-bit data. The addressing modes that will support VLIW architecture are

i) Register mode
ii) Immediate mode
iii)Displacement Addressing mode

In case of arithmetic and logical operations register mode, for the immediate values used as immediate mode. All VLIW instructions are 6×32-bi (VLIW[191:0]), which simplifies instruction decoding. Figure 1shows the VLIW instruction formats (R-format, I-format, and J-format), which are very close to MIPS. The first 32-bit instruction (VLIW[31:0]) can be scalar/ control instruction. However, the remaining 32-bit instructions (VLIW[63:32], VLIW[95:64], VLIW[127:96], VLIW[159:128]and VLIW[191:160]) scalars.

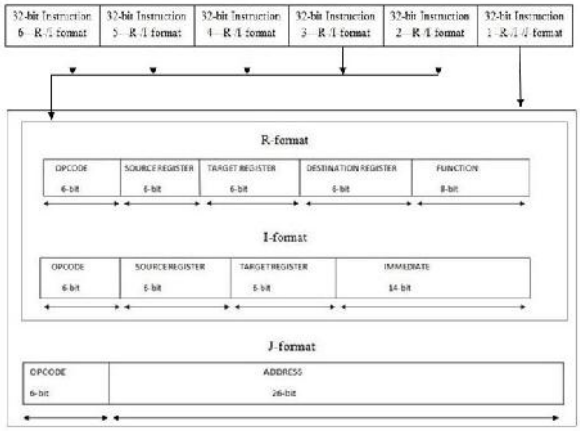

Fig 1: VLIW ISA formats

VLIW is a load-store Architecture with simple Hardware, fixed-length instruction encoding, and simple code generation model. It has common data path for executing VLIW instructions. Instruction cache stores 192-bitVLIW instructions of an application. Data cache loads/stores scalar data needed for processing scalar instructions. A single register file is used for both multi scalar elements

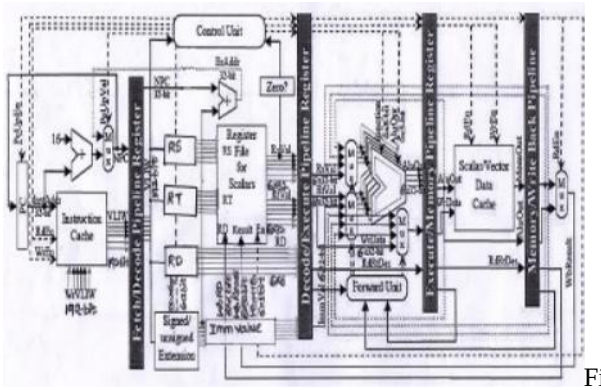

Fig 2: VLIW Architecture for high speed parallel Distributed computing system

(1) Fetching192-bit VLIW instruction,
(2) Decoding/reading operands of individual instruction
(3) Executing six scalar operations,
(4) Accessing memory to load/store 192-bit data
(5) Writing back six results

In above architecture it consist 5 main modules IF, ID, IE, MA and WB modules, It execute multi-scalar instructions, first fetching 192-bit VLIW instruction then decoding/reading operands of six individual instructions, after executing six instructions, and accessing memory to load/store 192-bit data, and writing back six results.

The VLIW instruction pointed by PC is read from the instruction cache of the fetch stage and stored in the instruction fetch/decode (IF/ID) pipeline register. The control unit in the decode stage reads the fetched VLIW instruction from IF/ID pipeline register to generate the proper control signals needed for processing multiple scalar data. The register file of the VLIW processor has sixty four 64 registers. Data are accessed from register file using source addresses 2×6×32-bit operands can be read and 6×32-bit can be written to the register file each clock cycle.

Thus, the control unit reads the RS (register source), RT (register target), and RD (register destination) of each individual instruction in the fetched VLIW to generate the sequence of control signals needed for reading/writing multi-scalar data from/to register file. The register file can be seen as 64×32-bit scalar registers.

The execute units of processor operate on the operands prepared in the decode stage and perform operations specified by the control unit, which depends on opcode1/function1, opcode2/function2, opcode3/ function3, opcode4/function4, opcode5/function5, and opcode6/function6 fields of the individual instructions in VLIW.

In Memory access stage, the registers can be loaded/stored individually using load/store instructions. And finally, the write back stage of VLIW stores the 4×32-bit results come from the memory system or from the execution units in the register file.

Six individual instructions packed in VLIW instruction are decoded and their operands are read from the unified register file (RsVal1/RtVal1, RsVal2/RtVal2 , RsVal3/RtVal3, RsVal4/RtVal4, RsVal5/RtVal5 and RsVal6/RtVal6) according to six pairs of RS/RT fields (RsVal1/RtVal1, RsVal2/RtVal2, RsVal3/RtVal3, RsVal4/RtVal4, RsVal5/RtVal5 and RsVal6/RtVal6), respectively. The execute units of VLIW operate on the operands prepared in the decode stage and perform operations specified by the control unit, which depends on opcode1/function1, opcode2/ function2, opcode3/ function3, opcode4/function4, opcode5/function5 and opcode6/function6 fields of the individual instructions in VLIW. For load/store instructions, The execute stage of VLIW has two modules: ALUs and Forward unit

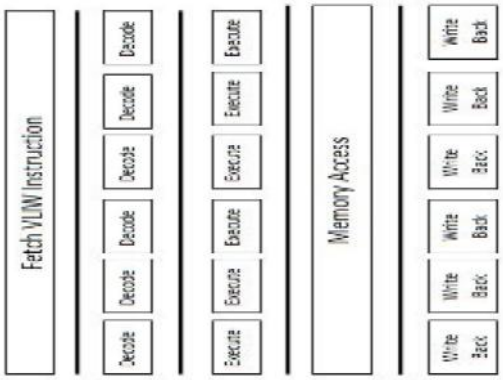

Fig 3: VLIW pipeline

III. FPGA/VHDL IMPLEMENTATION O VLIW PROCESSOR

The complete design of our proposed VLIW processor is implemented using VHDL targeting the Xilinx FPGA Virtex-5, XC5VLX110T-3FF1136 device. The required numbers of slice registers and LUTs are 3,379 and 11,977 (11,593 for logic and 384 for memory), respectively.

The first stage (Fetch Stage) has three modules: Program counter, Instruction adder, and Instruction memory as shown in fig. The 32-bit output address of the Program

counter module is sent to the input of the Instruction memory module to fetch 192-bit VLIW from instruction cache when read enable (RdEn) control signal is asserted. Note that the write enable (WrEn) control signal is used to fill in the instruction cache with the program instructions through 192-bit WrVLIW input. Depending on the PcUpdVal control signal ('1' for branch and '0' for other instructions), the input address of Program counter module is connected to the next sequential address or the branch address (BrAddr).

Decode Stage has three modules: Control unit, Register file, and Hazard detection. Depending on the control signals 6×1-bit IsUnSgn, the 6×14-bit immediate values are signed-/unsigned-extended to 6×32-bit The control unit of VLIW is based on microprogramming, where control variables are stored in ROM (control memory). Each word in the control memory specifies the following control signals: IsUnSgn (0/1 for signed/unsigned instruction), RdRtDes (0/1 if destination is RT/RD), IsImmIns (0/1 for immediate/un immediate instruction), RdEn (1 for load instruction), WrEn (1 for store instructions), Wr2Reg (1 when instruction writes result in register), and AluOpr (5-bit ALU operations).

The execute stage of VLIW has two modules: ALUs and Forward unit as shown in figure. The ALUs operate on four pairs of operands prepared in the decode stage. ALUs perform operations depending on the VLIW opcodes (6×5-bit AluOpr) and functions (6×8-bit Function). For load/store operations, the first ALU adds the operands (RsVal1 and ImmVal1) to form the effective address, however, the remaining ALUs are ideal. In all cases, the result of the ALUs (6×32-bit AluOut) is placed in the EX/MEM pipeline register. Instead of complicate the decode stage, the forward unit of VLIW is in the execute stage. The key observation needed to implement the forwarding logic is that the pipeline registers contain both the data to be forwarded as well as the source and destination register fields. In VLIW processor, all forwarding logically happens from the ALUs or data memory outputs to the ALUs inp`uts and data to be written in memory (WrData). Thus, the forwarding can be implemented by comparing the destination registers of the instructions contained in the EX/MEM or MEM/WB stages against the source registers of the instructions contained in the ID/EX registers. In addition to the comparators and combinational logic required to determine when a forwarding path needs to be enabled, the multiplexers at the ALUs inputs should be enlarged.

The VLIW registers can be loaded/stored individually using load/store instructions. Displacement addressing mode is used for calculating the effective address by adding the singed extended immediate value (ImmVal1) to RS register (RsVal1) of the first individual instruction in VLIW. In addition, the ImmVal1 register is incremented by 16 to prepare the address of the next 6×32-bit element of the data. In the first implementation of our proposed VLIW processor, four elements (192-bit) can be loaded/stored per clock cycle. The fourth stage is Memory Stage, which has one component called Data memory as shown in figure.

The write back stage of VLIW stores the 6×32-bit results come from the memory system or from the execution units in the VLIW register file. Depending on the effective opcode of each individual instruction in VLIW, the register destination field is specified by either RT or RD. The control signals 6×Wr2Reg are used for enabling the writing 6×32-bit results into the VLIW register file.

THEROTICAL CALUCLATIONS:

The complete192 bit VLIW instructions in case of the immediate instruction format as follows
(ADD_I&"000000"&"000100"&"00000000001010")&
(ADD_I&"000000"&"000011"&"00000000001111")&
(ADD_I&"000000"&"000010"&"00000000000110")&
(ADD_I&"000000"&"000001"&"00000000000100")&
(ADD_I&"000000"&"000101"&"00000000001000")&
(ADD_I&"000000"&"000110"&"00000000000111"). The 32- bit of least significant bit of 192-bit will be (001101&"000000"&"000110"&"00000000000111")
.Similarly for the remaining Instructions.

TABLES:

	Statistics	Fetch Stage	Decode Stage	Execute Stage	Memory Stage	Writeback stage
Slice Logic Utilization	Number of Slice Registers:	32	2048		192	
	Number Of Slice LUTS:	103	13128	1958	386	192
	Number Used as Logic:	103	13128	1958	2	192
	Number Used as Memory:				384	
Slice Logic Distribution	Number of LUT Flip Flop pairs Used:	103	13128	1958	386	192
	Number With an Unused Flip Flop:	71	11080	1958	194	192
	Number with an unused LUT:	0	0	0	0	0
	Number of fully used LUT-FF pairs:	32	2048	0	192	0
Timing	Maximum Frequency(MHz)	406.48	No Path found	No Path found	756.81	No Path found

Table 1: FPGA Statistics for VLIW Stages

	Statistics	
HDL Synthesis Report	2x128x192-bit single-port RAM	32-bit up loadable accumulator
	5x16x4-bit ROM	30x6-bit registers
	1x16x7-bit ROM	6x8-bit registers
	6x32x32-bit Multipliers	1x192-bit latch
	1x32-bit adder	32-bit comparator greater
	6x32-bit add sub	32-bit comparator less equal
	31x1-bit registers	36x6-bit comparator equal
	7x192-bit registers	12x32-bit 64-to-1 multiplexer
	72x32-bit registers	31x1-bit xor2
	6x4-bit registers	6x32-bit xor2
Slice logic Utilization	Number of Slice Registers: 3379	
	Number of Slice LUTS: 11977	
	Number used as Logic: 11593	
	Number Used as Memory: 384	
Slice Logic Distribution	Number of LUT Flip Flop pairs Used : 14311	
	Number with an unused Flip Flop: 10932	
	Number with an unused LUT: 2334	
	Number of fully used LUT-FF pairs: 1045	
Timing	Maximum Frequency: 100.701MHz	

Table 2: HDL Synthesis Report

SIMULATION RESULTS:

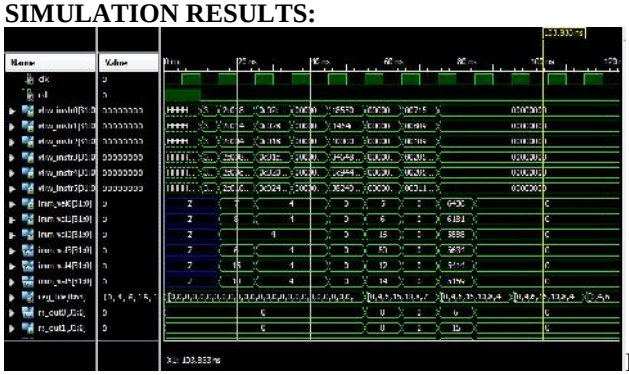

Fig: Top module of VLIW

Fig: Fetch module

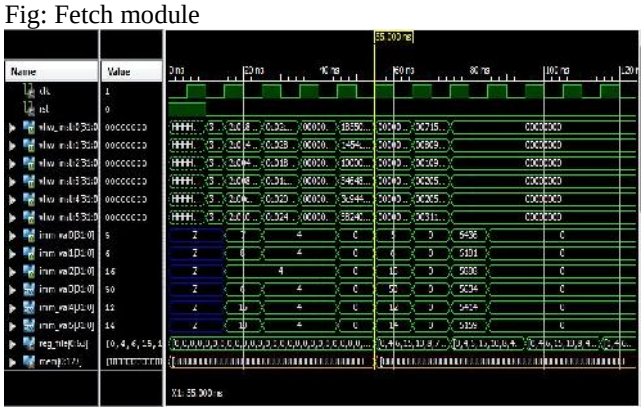

Fig: VLIW ADDI Instructions

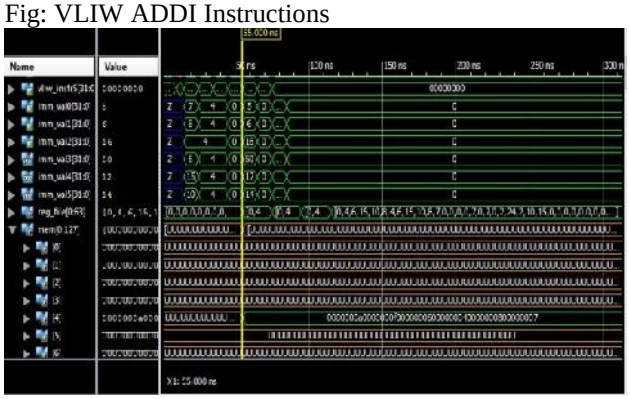

Fig: VLIW LOAD/ STORE Instructions

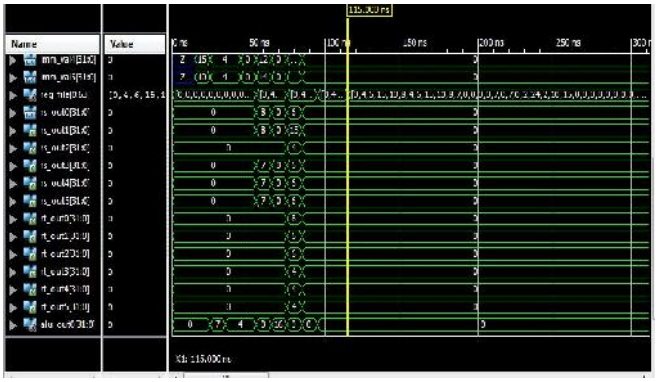

Fig: VLIW ADD/SUB/MUL Instructions

CONCLUSIONS:

The proposed new processor architecture called VLIW for accelerating data-parallel applications. VLIW executes multi-scalar instructions on the same parallel execution data path. We are achieving Executing six instructions parallel, by the increasing the parallelism. With the decrease in time to execute the instructions, these Processors are utilized in Data parallel Applications, Multimedia Applications.

[1]. Mostafa I. Soliman **,"**A VLIW architecture for executing multi-scalar/vector instructions on unified datapath" Electronics, Communications and Photonics Conference (SIECPC), 2013 Saudi International

[2.] Philips, Inc., An Introduction to Very-Long Instruction Word (VLIW) Computer Architecture, Philips Semiconductors,

[3] J. Fisher, "VLIW architectures and the ELI-512," Proc. 10th International Symposium on Computer Architecture, Stockholm, Sweden, pp. 140-150, June 1983.

[4] J. Fisher, P. Faraboschi, and C. Young, Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools, Morgan Kaufmann, 2004.

[5]. An Introduction To Very-Long Instruction Word (VLIW) Computer Architecture". Philips Semiconductors.

[6]. Fisher, Joseph A. (1983). "Very Long Instruction Word architectures and the ELI-12"(PDF). Proceedings of the 10th annual international symposium on Computer architecture. International Symposium on Computer Architecture.

[7]. MIPS32 Architecture For Programmers Volume II: The MIPS32 Instruction Set, MIPS, 2011. Available at: http://www.cs.cornell.edu/courses/cs3410/2011sp/MIPS Vol2.pdf.

[8]. J. Smith and G. Sohi, "The microarchitecture of superscalar processors," Proceedings of the IEEE, vol. 83, no. 12, pp. 1609-1624, December 1995.

T.NAVEEN KUMAR REDDY, Pursuing M.TECH in VLSI from JBREC college Affiliated from JNTUH. Moinabad, Rangareddy Dist, Hyderabad, Telangana..he got 77.6% in her Mtech 1st year. he has received the Btech degee in Sri Kottam Tulasi Reddy Memorial college of Engineering in 2011.

Prof.P.N.V.M Sastry Currently is working with a Capacity of **Dean- IT EDA Software – R&D CELL & ECE** DEPARTMENT, He Did Master Degree In Science- M.Sc Electronics, AU -1998.Did PG Diploma In VLSI Design From V3 Logic Pvt Ltd B'Lore-2001, Did M.Tech (ECE) From IASE Deemed University-2005. Currently Pursuing (PhD)-ECE , JNTU Hyderabad -2012 , Over **Past 16 years of Rich Professional** Experience with Reputed Software MNC's, Corporates – INFOTECH, ISiTECH as a **Team Leader(Level 6) Eng-Eng- HCM Electronics Vertical, Program Lead – Embedded & VLSI& Delivery Manager – IT Semiconductor Software Engineering Vertical** , also worked with **Govt R&D, Industrial Organizations, Academic Institutions** of Comparative Designations

Dr. D.N Rao B.Tech,M.E,Ph.D, principal of JBREC, Hyderabad. His carrier spans nearly three decades in the field of teaching, administration, R&D, and other diversified in-depth experience in academics and administration. He has actively involved in organizing various conferences and workshops. He has published over 11 international journal papers out of his research work. He presented more than 15 research papers at various national and international conferences. He is Currently approved reviewer of IASTED International journals and conferences from the year 2006. He is also guiding the projects of PG/Ph.D students of various universities