JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

EFFECT OF VECTORIZATION AND BRANCHING ON PROCESSING SPEED OF VLIW PROCESSOR

[1]Bharat Sharma, [2]Jai Parkash Savelia
[1]Assistant professor, [2]Assistant Professor
[1]Department Of Computer Science and Engineering,
[1]Gulzar Group of Institutes, Ludhiana, India

Abstract : Since the microprocessor was invented in 1970's from that time the major work researchers are doing is to improve the performance of the microprocessor. Especially for high performance computing (HPC) applications like video conferencing, speech processing, weather forecasting and military applications where huge amount of data is needed to process very fast. Simply says "two heads are better than one" means two powerful concepts will forms a better concept that will be application specific but not general purpose and very powerful at its place, So we are implementing a processor having both instruction level parallelisms (ILP) by VLIW design and data level parallelism (DLP) by vector design is a good idea for such applications. First objective of this research is to measure and compare the performance of a statically multi issue processor can be called as an "array based VLIW processor" with respect to simple VLIW processor. Secondly we show the effect of branching on the processor clock cycles for both with and without branching using TRIMARAN framework 4.0(a compiler and simulator infrastructure for research in embedded system) on ILP+DLP approach.

IndexTerms - Vectorization, Branching, VLIW, ILP, DLP.

INTRODUCTION

one important aspect of vector processor architecture basic data types a specific word processor word size, where the N word, conventional scalar processors .in multiple data sets (SIMD) architecture on the same instructions, and the instruction set is made up of different Words that work Instructions. Now the vector architecture, it uses vector data types, and Vector N-bit vector array is a collection of length, and Cray was invented in architecture, which is a vector file is not registered. Superscalar processor composed of several operations similar to these processors by implementing a long instruction word similarity reflects the level of education. This very long instruction word too many math, logical and control operations which included multi-ops, as the saying goes. Each of these functions can be executed on perhaps simple RISC processor that person is involved in the operation. VLIW composed of several operations by executing a long instruction word of instruction-level parallelism (ILP) tries to achieve that is a type of microprocessor architecture.

I. OBJECTIVE

The objective of this paper is to compare and find the benefits and shortcomings between two successful architectures of processors. So we are implementing a processor having both instruction level parallelisms (ILP) by VLIW design and data level parallelism (DLP) by vector design is a good idea for such applications. First objective of this research is to measure and compare the performance of a statically multi issue processor can be called as an "array based VLIW processor" with respect to simple VLIW processor. Secondly we show the effect of branching on the processor clock cycles for both with and without branching using TRIMARAN framework 4.0(a compiler and simulator infrastructure for research in embedded system) on ILP+DLP approach.

II. RESEARCH METHODOLOGY

Trimaran is an integrated compiler and simulation infrastructure for research in computer architecture and compiler optimizations. Trimaran is highly parameterizable, and can target a wide range of architectures that embody embedded processors, high-end VLIW processors, and multi-clustered architectures. Trimaran also facilitates the exploration of the architecture design space, and is well suited for the automatic synthesis of programmable application specific architectures. It allows for customization of all aspects of an architecture, including the datapath, control path, instruction set, interconnect, and instruction/data memory subsystems. The modular nature of the compiler and the hierarchical intermediate program representation used throughout makes the construction and insertion of new compilation modules into the compiler especially easy. Trimaran is already populated with a large number of existing compilation modules, providing leverage for new compiler research as well as education in advanced compiler topics. The Trimaran Graphical User Interface (GUI) makes the configuration and use of the system surprisingly easy. Among the rich suite of compiler analysis and optimizations are: • Advanced region formation algorithms (e.g., superblocks and hyperblocks) to expose instruction level parallelism with speculation and predication, • Various backend

instruction partitioning and mapping algorithms for automatically distributing parallelism in a multi-clustered architecture, • A first of its kind back-end vectorizer that extracts and exploits data level parallelism using short vector instructions (SIMD), • Various register allocation heuristics, • Instruction scheduling algorithms including software pipelining with modulo scheduling.

1. ARRAY BASED ARCHITECTURE

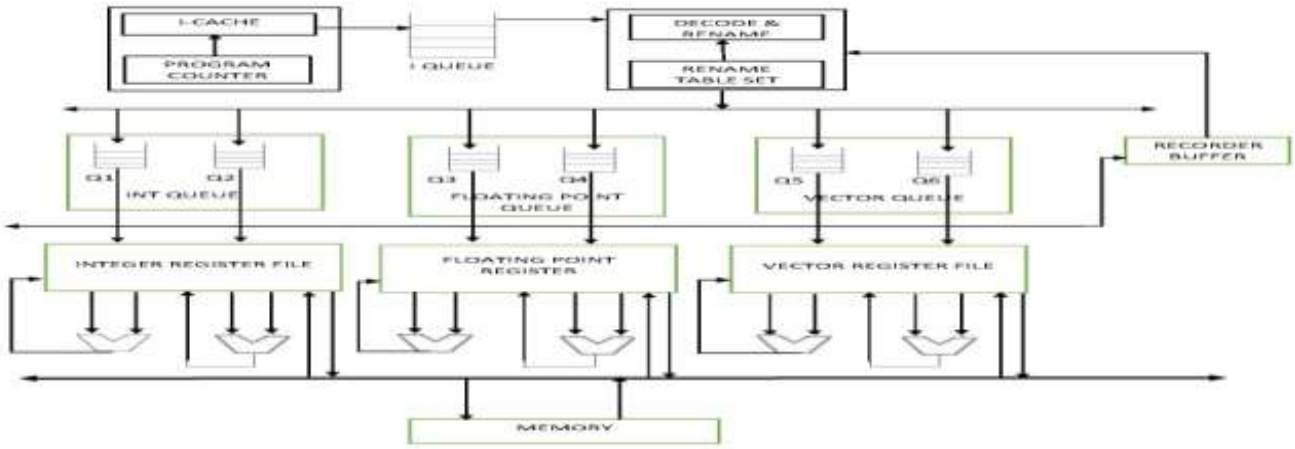

Figure1: Proposed array based architecture

Some main points of this architecture are as follows:1. multiple functional units are attached with a single global register file.2. compiler itself dynamically issues the long instruction word.3.each instruction consists of multiple parallel operations which are independent.4.each operation requires statically known number of cycles to execute. Architecture shows the execution and flow of the data and instructions for all four operations: I-fetch, Decode and rename, execution and write-back.

2. VECTORIZATION

Vectorization displays and SIMD architecture, with support for the efficient execution of a program which is a technique to exploit the data-level parallelism. Trimaran as the technique is called selective vectorization. This is in order to improve resource utilization and performance scalar and vector processing units by distributing computing between educations creates highly efficient schedule. This type of technology is mostly applied to multimedia applications. Vectorization by default, one at a time on a single thread application program, a series of actions that are modified to act, which is a special case of parallelization. Vectorization a vector processing instruction in both traditional computers and modern supercomputer is a key feature where a process conducted over several operands. Vectorization a vectorized compiler without human assistance programs that allow a process to convert scalar functions. Exploiting data-level parallelism from the application program.

Multi-thread processor vectorization for high performance computing applications is yet to find out. We are array-based VLIW processor architecture VLIW infrastructure in order to realize the many standards vectorize. In this type of architecture known as vector directions of each instruction operates on the array of numbers where a single cycle are issued. Array-based VLIW processor vectorizations 2lc used for the purpose we combine the advantages of VLIW processor and vectorization process. Open space of the impact of automatic vectorization stage S2lc. vectorization function of a vector instruction set processor that utilizes a parallel version is to convert a sequential loop. The shift operation is a major rearrangement.

3. MACHINE DESCRIPTION FOR ARRAY BASED VLIW PROCESSOR

Number of clusters	4
Issue slots	6
Per-cluster Functional Units:	
Integer units	1
Float units	1
Memory units	1
Branch units	1
Vector length	4

TCC total compute cycles are the number of cycles taken by a processor to execute one single program.

4.1 Total Compute Cycles (TCC)

In the Table shown below there are seven bench marks taken to compute the total cycles taken in processing the same bench mark with three different processors. With small code and less jumping statements like fact2 and fib benchmarks can be executed in less number of cycles but in comparison to performance they also have increase in performance between 70-80% if compared array based VLIW processor with single issue scalar processor and multiple issue VLIW processor without vectorization.

S.NO.	BENCHMARK	SCALAR	VLIW	ARRAY BASED VLIW	PERFORMANCE INCREASES WITH COMPARE TO	
					SCALAR (%)	VECTOR (%)
1.	Ifthen	4112	3216	3103	24.53	3.51
2.	Fact2	63	55	10	88.33	81.81
3.	Fib	46	46	12	73.91	73.91
4.	Nested	22020	16041	3884	82.36	75.57
5.	Sqrt	3171	2950	3189	0.056	0.81
6.	Strcpy	21468	17461	15434	28.10	11.60
7.	eight	4344	4336	4013	7.61	7.44

Table 1: Total Compute Cycles (TCC)

4.2 IPC Measurements

Now taking in consideration of IPC measures. The table below is the instruction per cycle count of the benchmarks.

S.NO.	BECNHMARK	SCALAR	VECTOR	ARRAY BASED VLIW
1.	Ifthen	1.03	1.63	0.87
2.	Fact2	1.17	1.64	3.10
3.	Fib	2.50	2.50	2.90
4.	Nested	2.04	1.57	1.52
5.	sqrt	2.60	2.77	3.00

Table 2: IPC Measurements

4.3 Branching Effect

S.NO.	BENCHMARK	SCALAR	VLIW WITHOUT VECTOR	VLIW WITH VECTOR
1.	Fact2	15	15	1
2	Fib	16	16	1
3.	Ifthen	901	902	901
4.	Nested loop	1214	4906	1214
5.	Sqrt	1	18	3
6.	Strcpy	23	19	29
7.	Eight	735	668	735

Table 3: Measurements after Branching effect

In the above table it is shown that the benchmarks are processed on different processors and in processing the branches are taken by the address pointer are different in three processors in most of cases. Branching effect is very efficiently shown in case of VLIW processor due to advance Branching mechanism but in Array based VLIW processor where large amount of data to be processed it comes again depending upon the code and compiler that the compiler is able to predict the branch before it is taken. In benchmark “sqrt” branches are 18 in case of VLIW but it is increased by 2 units become 3 with vectorization. In benchmark “strcpy” branches are 19 in case of VLIW but it is increased by 10 units become 29 with vectorization. In benchmark “eight” branches are 668 in case of VLIW but it is increased by 67 units become 735 with vectorization.

So here the conclusion is that with increase of data operands in code the problem of branching may arise as seeing in these examples. There should be an advance branching mechanism be embedded in new Array based VLIW architecture. The above Table shows the branch taken values in the program execution of same benchmarks. Branch taken causes the delay in program execution. Jump statements are the kind of branches that moves the address pointer from one location to another location. If one branch is taken then the instructions which are followed by that branch will be made useless and they will wait that when the pointer will come back to that instruction.

III. RESULTS

Array based VLIW processor is feasible to be made and should be more efficient to execute the programs in less time by using two types of parallelism ILP and DLP. Array based VLIW processor have which takes advantages of both VLIW and Vector processor architecture.
However, the compiler is critical for achieving maximum benefit because complete benefit of this processor cannot be realized unless the compiler can schedule the instructions in parallel efficiently. So Complier needs to more complex and should able to schedule the multiple instructions in an effective way. Branching problem is still remained as it is, not resolved like VLIW processor branch prediction mechanism.

IV. FUTURE WORK

This array based VLIW processor has a lot of possibilities of improvements in many areas like efficient branch prediction and less access time on-chip memory. It can be a very good and efficient architecture for multimedia programs.

V. REFERENCES

[1] Bona, M.Samit, D.Sciutot, C.silvanog 2006, “energy estimation and optimization of embedded VLIW processor based on instruction clustering”, ACM, page: 886-891.
[2] Mohammad Suaib, Abel Palaty, Kumar SambhavPandey 2011, “Architecture of SIMD Type Vector Processor”, International journal of computer applications.
[3] Andrea Lodi, Mario Toma, Fabio Campi, Andrea Cappelli, Roberto Canegallo, and Roberto Guerrieri 2003, “A VLIW Processor With Reconfigurable Instruction Set for Embedded Applications”, IEEE.
[4] Anju S. Pillai, Isha T.B. 2013, “Factors Causing Power Consumption in an Embedded Processor - A Study”, IJAIEM.
[5] Christoforos E., Kozyr David A. pattersonakis 2003, “scalable vector processors for embedded systems”, IEEE.
[6] Christos Kozyrakis, David Patterson “Overcoming the limitations of conventional vector processors”, Department of Electrical Engineering Stanford University.
[7] ChristoforosKozyrakis, David Patterson 2002, “Vector Vs. Superscalar and VLIW Architectures for Embedded Multimedia Benchmarks”, IEEE.
[8] Carlos Carvalho 2012, “The Gap between Processor and Memory Speeds”, ICCA.