International Journal of Communication Networks and Security

Volume 2 | Issue 2 Article 7

April 2013

DESIGN SPACE EXPLORATION AND OPTIMIZATION OF SUPER SCALAR PROCESSOR

K. NAVATHA
Sumathi Reddy Institute of Technology for Women, Warangal,Andhra Pradesh,INDIA.,
dnavatha@yahoo.com

PROF.G. KRISHNA MURTHY
Sumathi Reddy Institute of Technology for Women, Warangal,Andhra Pradesh,INDIA.,
G.KRISHNA@gmail.com

DESIGN SPACE EXPLORATION AND OPTIMIZATION OF SUPER SCALAR PROCESSOR

K.NAVATHA[1], PROF.G. KRISHNA MURTHY[2]

[1,2]Sumathi Reddy Institute of Technology for Women, Warangal,Andhra Pradesh,INDIA.
E-mail:dnavatha@yahoo.com

Abstract- Designing a microprocessor involves determining the optimal microarchitecture for a given objective function and a given set of constraints. Superscalar processing is the latest in along series of innovations aimed at producing ever-faster microprocessors. By exploiting instruction-level parallelism, superscalar processors[1] are capable of executing more than one instruction in a clock cycle.The architectural design of super scalar processor involves a lot of trade off issues when selecting parameter values for instruction level parallelism.The use of critical quantitative analysis based upon the Simple Scalar simulations is necessary to select optimal parameter values for the processor aimed at specific target environment. This paper aims at finding optimal values for the super scalar processor and determines which processor parameters have the greatest impact on the simulated execution time.

Keywords- *Superscalar, Benchmarks, microprocessor optimization, simple scalar, instruction level parallelism.*

I. INRODUCTION

Designing a new microprocessor is a complex process as the processor design space is huge and the various design parameters and constraints interact with each other. These design issues typically concern performance, cycle time, power consumption, chip area, reliability, security, verifiability, etc. The task for a designer is to optimize the microarchitecture such that a given objective function is optimized. The objective function can take many forms depending on the target domain of the microprocessor under design.

Superscalar processing, the ability to initiate multiple instructions during the same clock cycle, is the latest in a long series of architectural innovations aimed at producing ever faster microprocessors. Introduced at the beginning of this decade, superscalar microprocessors are now being designed and produced by all the microprocessor vendors for high-end products. Although viewed by many as an extension of the Reduced Instruction Set Computer (RISC) movement of the 1980s, superscalar implementations are in fact heading toward increasing complexity. And superscalar methods have been applied to a spectrum of instruction sets, ranging from the DEC Alpha, the "newest" RISC instruction set, to the decidedly non-RISC Intel x86 instruction set.

II.SIMPLE SCALAR BACKGROUND

Simple Scalar is a set of tools that model a virtual computer system with CPU, Cache and Memory Hierarchy. Using the Simple Scalar tools[2], users can build modeling applications that simulate real programs running on a range of modern processors and systems. The tool set includes sample simulators ranging from a fast functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. In addition to simulators, the Simple Scalar tool set includes performance visualization tools, statistical analysis resources, and debug and verification infrastructure. It is capable of simulating binary programs on one of the several processor simulators provided.

III. EXPERIMENTAL SETUP

To determine the effect of various processor parameters and their interactions, we used sim-outorder from the SimpleScalar tool suite (version 3.0, PISA).[3] This simulator has several processor parameters that can be easily changed. substantial previous work [4][5][6][7][8] described the architecture of several 4-way issue processors that were implemented along with the values for many important parameters. Therefore, we were able to gather a set of values that accurately reflected the actual values that are used in commercially available processors, which gave us a good initial starting point. Table 1 lists the 38 parameters that we initially considered analyzing, i.e. to determine their effect on the execution time (cycles).

Number	Parameters
1-5	L1 I-Cache size,Associativity,Block size,Replacement policy, Latency
6-7	Branch predictortype and configuration
8	# instruction fetch queue entries
9-11	Decode , issue, commit widths
12	# of ROB entries
13	# of Int ALUs
14	# of FP ALUs
15	# of Integer multipliers/Dividers
16	# of FP Multipliers/Dividers
17	Functional unit latencies

18	Degree of pipelining in functional units
19	# of LSQ entries
20	# of memory ports
21-25	L1 D-Cahe Size, Associativity, Block size, Replacement policy, Latency
26-30	L2 Cache size,Associativity, Block size,Replacement policy, Latency
31-32	Memory latency, First Block; Memory latency ,Following blocks
33	Memory bus width
34-38	Instruction and data TLB entries,page size,TLB entry

Table 1: Initial List of Processor Parameters

Simultaneously trying to measure the effect of 38 different parameters and all their possible combinations is an intractable problem. Performing the set of simulations for the parameters shown in Table 1 would have required approximately 20 billion years. So, in this paper, we ultimately considered only 4 parameters; Issue width, Register Update Unit, No. of integer ALU units, L1 data Cache Associativity. A subset of SPECint95 benchmarks [9][10][11]:compress94,go,cc1,perl ,anagram were used. The no of instructions simulated for each benchmark and their corresponding inputs is tabulated in table 2

Benchmarks	input	# instructions
Compress95	Compress95.in	164M
Cc1	Cc1.ss	53M
Perl	Perl.ss	46M
Go	Go.ss 50 9	100M
Anagram	Anagram.ss	

Table 2: Benchmarks

The above four parameters are varied one by one keeping the rest constant. The simulations are run for the following values:

Parameters	Values
Issue width	4,8,16,32,64
Register Update Unit	4,8,16,32,64,128,256
#interger ALUs	1,2,4
L1 Data Associativity	1,2,4,8,16,32,64

Table 3: Initial List of Processor Parameters under test with their values.

IV. RESULTS

Simulations were run for various combinations of the input parameters. The following behavior patterns were observed:

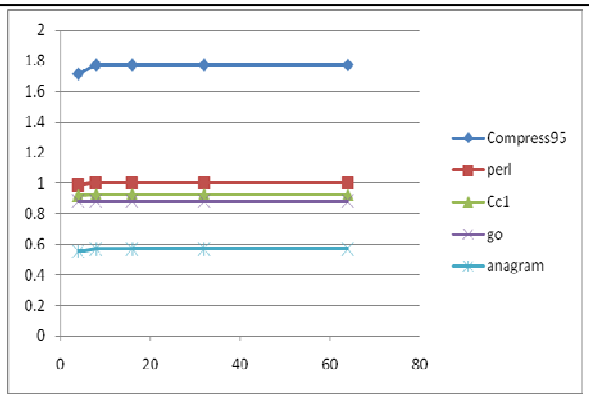

Fig 1: Issue width vs Instructions Per Cycle

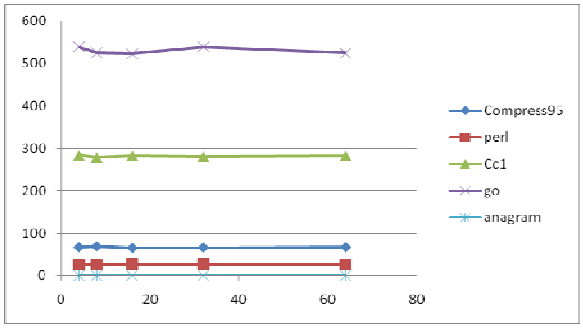

Fig 2 Issue width vs simulation time

In fig.1 we observe that IPC increases as Issue width increases and remains same after Issue Width = 8 .Figure 2 shows that the increase in the Issue width increases the simulation time linearly and for most of the benchmark programs Issue width of 8 gives a minimum simulation time.

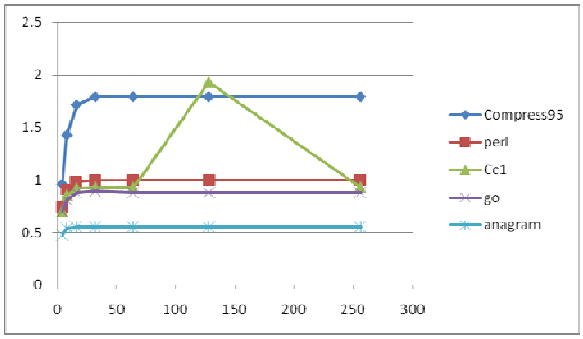

Fig3: Register Update Unit vs Instructions Per Cycle

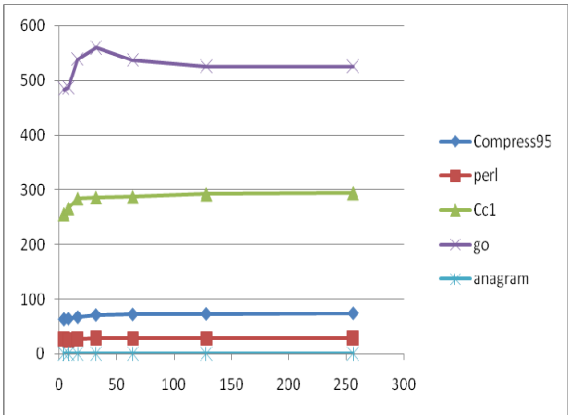

Fig 4: Register Update Unit vs Simulation time

In fig.3 we observe that the graphs hit a minimum value for RUU=8.In fig 4 IPC count increases steeply until it hits RUU=16.so for the benchmark programs under consideration RUU=8 is the appropriate configuration.

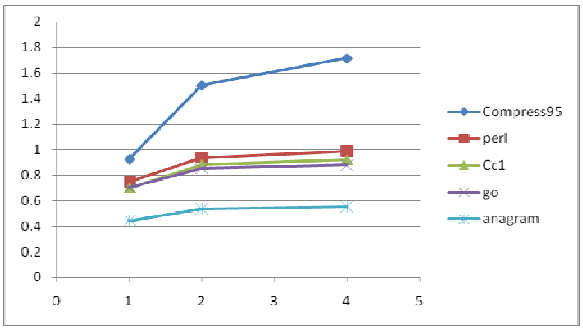

Fig 5: integerALUs vs Instructions Per Cycle

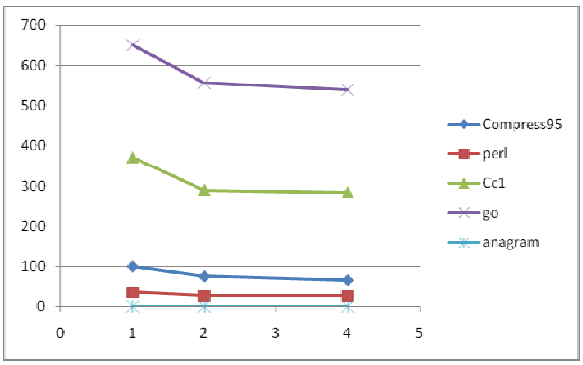

Fig 6: Integer ALUs vs Simulation time

In fig 5 IPC increases till no. of ALUs =2 and then the graph becomes less steeper and same is the case for fig 6.

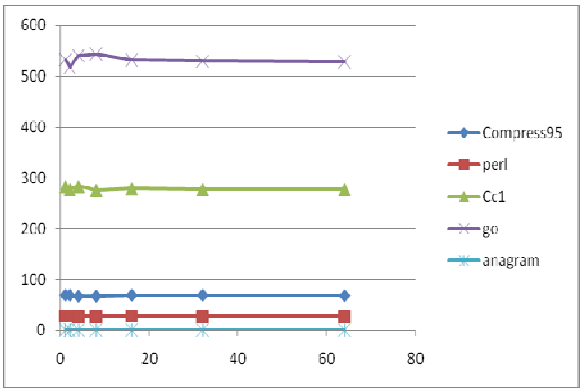

Fig 7 : L1 Data Cache Associativity vs Simulation Time

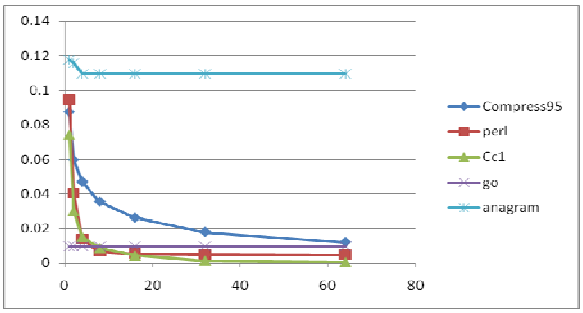

Fig 8: L1 Data Cache Associativity vs Miss rate

From the fig 7 and fig 8 we observe that for most of the benchmark programs the simulation time hits a minimum for data cache associtivity of 2.

A. Effect of Single Parameters

Figure 9 shows the effect, in percent, of each parameter on the execution time. The effect that a parameter has is what percentage of the variation of the execution time is due to that parameter. Therefore, the more effect that a parameter has, i.e. the larger percentage accounted for by the parameter, the more sensitive the execution time is to changes in that parameter.

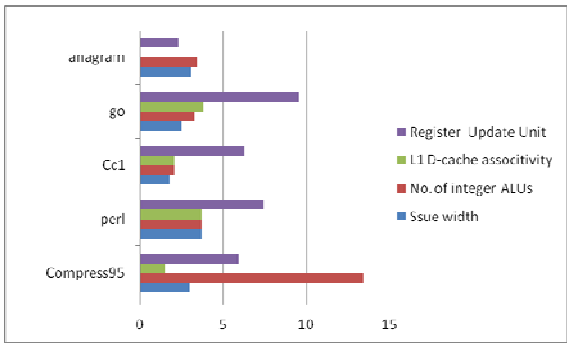

Figure 9: The Effect (Percent) of Single Parameters on the Total Variation in the Execution Time

V. CONCLUSION

Computer architects rely heavily on simulators when trying to design a new processor architecture or when evaluating the performance of new compiler-based and microarchitectural mechanisms. However, since the simulation results can change significantly based on the values of the processor parameters that are used (i.e. independent of what feature is actually under test), it is extremely important to choose reasonable parameter values. However, it is unknown how much of an impact each processor parameter (or interaction between two or more parameters) actually has. Our results has presented a methodical implementation for selecting optimal parameter values. The parameter values have been selected by making a quantitative study of the simulation results.. Based on these results, we recommend that extreme care be exercised when choosing the values for these four dominant parameters. Not only must the parameter values be reasonable ,they must also be appropriate for the architecture and be feasible to implement.

REFERENCES

[1] J. Hennessy and D. Patterson, \Computer architecture: A quantitative approach, fourth edition,"

[2] D. C. Burger and T. M. Austin. The SimpleScalarTool Set. Computer Architecture News, 1997.

[3] http://www.simplescalar.com

[4] P. Bannon and Y. Saito; "The Alpha 21164PC Microprocessor"; International Computer onference, 1997

[5] D. Christie; "Developing the AMD-K5 Architecture"; IEEE Micro, Vol. 16, No. 2, March-April 1996; Pages 16-26

[6] R. Kessler, E. McLellan, and D. Webb; "The Alpha 21264 Microprocessor Architecture"; International Conference on Computer Design, 1998

[7] D. Leiholz and R. Razdan; "The Alpha 21264: A 500 MHz Out-of-Order Execution Microprocessor"; International Computer Conference, 1997

[8] D. Papworth; "Tuning the Pentium Pro Microarchitecture"; IEEE Micro, Vol. 16, No. 2, March-April 1996; Pages 8-15

[9] Standard Performance Evaluation Corporation, SPEC CINT95 Benchmarks,available from: http://www.spec.org/cpu95/CINT95/index.html, last accessed: May 31st, 2006.

[10] http://www.specbench.org/osg/cpu95

[11] SPEC Newsletter , Fall-89, “SPEC Benchmark Suite : Designed For Today’s Advanced Systems”