Performance Measurement For Various Memory Architecture Of Superscalar Pipelined Processor

Kritika Purohit
M.Tech. Scholar
Arya College of Engineering & IT, Jaipur (Raj.)
Jaipur, India

Dr.Vibhakar Pathak, MIEEE
Professor CS/IT
Arya College of Engineering & IT, Jaipur (Raj.)
Jaipur, India

***Abstract*— There are various ways to improve the throughput of a system. One of the key factors is processor. Series of so many analyses have been carried out and developments have been made for the same viz. pipelining, parallel processing, multi-core etc. There are various parameters viz. superscalar factor, reservation station architecture, number of different functional units, reservation station size, memory architecture etc. which influence the performance of the superscalar processor. The average Instruction fetch efficiency factor of superscalar architecture controls the performance aspect. The purpose of this work is to obtain the optimal configuration of the memory for superscalar architecture by conducting a qualitative and quantitative study.**

Keywords— superscalar factor, reservation station architecture, number of different functional units, reservation station size, memory architecture component.

I. INTRODUCTION

The performance of superscalar architecture depends on the various parameter like superscalar factor, reservation station architecture, number of different functional units etc. In this dissertation impact of various memory parameters on performance of superscalar processor is measured individually, by conducting the simulation using superscalar simulator PSATSim/EdSATSim v2.1, and the performance in terms of execution time is compared.
The superscalar processor improves the performance by executing multiple instructions in parallel. The basic design choice for superscalar architecture includes the superscalar factor and reservation station architecture.

II. PERFORMANCE PARAMETERS FOR SUPERSCALAR ARCHITECTURE

The performance of superscalar architecture mainly depends on the following parameters:

a)Superscalar Factor: The superscalar degree is determined by the issue parallelism *n* i.e. the maximum number of instructions that can be issued in every cycle. A superscalar machine of degree *n* can be viewed as having *n* pipelines or a pipeline that is *n* times wider in the sense of being able to carry *n* instructions in each pipeline stage instead of one.

b) Reservation Station Architecture: Prior to execution of an instruction all its operands must be available. Register operands are fetched from the register files during decoding but it may be that some of these operands are not yet ready, because earlier instructions that update these registers have not finished their execution. This situation stalls the decoding stage until all register operands are ready

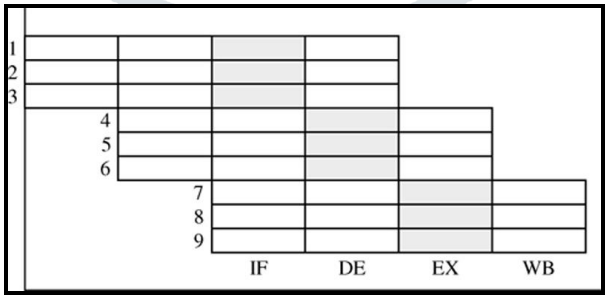

Figure 1.1 Superscalar processor with Superscalar factor=3

To avoid this stall we can fetch those register operands that are ready and go ahead. To advance these instructions into a separate buffer to await those register operands that are not ready. When all register operands are ready, those instructions can then exit this buffer and be issued into the functional units for execution. Such a temporary instruction buffer is known as a *reservation station*

Two types of reservation station implementations are possible

- *Centralized reservation station*: A single buffer is used at the source side of dispatching. One reservation station with many entries feeds all the functional units. Instructions are dispatched from this centralized reservation station directly to all the functional units to begin execution

- *Distributed reservation stations*: Multiple buffers are placed at the destination side of dispatching. Each functional unit has its own reservation station on the input side of the unit. Instructions are dispatched to the individual reservation stations based on instruction type. These instructions remain in these reservation stations until they are ready to be issued into the functional units for execution.

c) Number of Execution Unit: Scalar pipelined processors have only one functional unit and all instruction types are executed by the same functional unit. This processor consists of the ALU and MEM stages. First-generation superscalar processors are parallel pipelines with two diversified functional units, one executing integer instructions and the other executing floating-point instructions.

Current superscalar processors can employ multiple integer units. Some have multiple floating-point units. These are the two most fundamental functional unit types. An integer unit can be used for generating memory addresses and executing branch and load/store instructions. In most recent designs separate branch and load/store units have been incorporated. The branch unit is responsible for updating the PC. The load/store unit is directly connected to the D-cache. Other specialized functional units have emerged for supporting graphics and image processing applications

The best mix of functional units for a superscalar pipeline depends on the application domain. Typical programs have 40% ALU instructions, 20% branches, and 40% load/store instructions. We can have a 4-2-4 rule of thumb i.e. for every four ALU units, we should have two branch units and four load/store units. Many of the current leading superscalar processors have four or more ALU-type functional units including both integer and floating-point unit. Most of them have only one branch unit, they are able to speculate beyond one conditional branch instruction. Most of these processors have only one load/store unit. Some are able to process two load/store instructions in every cycle with some constraints

In real superscalar pipeline designs, the total number of functional units exceeds the actual width of the parallel pipeline. The width of a superscalar pipeline is determined by the number of instructions that can be fetched, decoded, or completed in every machine cycle

III. PROBLEM DEFINITION

In this work we analyzed the impact of different memory configuration on performance of Superscalar Pipelined Architecture in terms of execution time and power consumption. We simulated the design for different reservation station architecture i.e. distributed, centralized and memory architecture (L1, L2 and System). The study includes the results for model architecture (base architecture Pentium IV of family of Core Processors) and It's extensions i3, i5, and i7 architecture.

IV. PLATFORM OF RESEARCH AND EXPERIMENTAL SETUP

The simulation is quick and easy method to analyze any complex system. Simulation also helps to evaluate subtle design trade-offs in an experimental environment. It reduces the cost and time by allowing to quickly evaluating different processor implementations without having to fabricate a chip each time. A simulator also allows the processor designer / architect to easily determine the expected performance improvement of a new compiler based or micro-architectural mechanism.

As it is imperative to simulation techniques for super scalar architecture therefore for analyzing the work of this dissertation, simulation technique is used. The simulator used is PSATSim simulator. By running this simulator on various programs for different processor architecture, all the experiments are carried out.

a) PSATSim Simulator

The PSATSim is a graphical simulator that can be configured to a speculative out-of-order execution as well as power consumption of superscalar processor. Thus it supports to see the effect of the design on both power and performance. This simulator helps to understand the flow of instructions within the pipeline, dynamic scheduling, speculative execution, and the data dependencies between instructions.

PSATSim simulator models the dynamic power consumption of superscalar processor architectures. It incorporates a speculative execution model which gives relative accuracy to the power values. PSATSim allows students to experiment with a wide range of architectural parameters and to observe both the power and performance results of the configuration.

same can be adapted to support the architectural flexibility of PSATSim. Because of the large number of differences between the two simulators' architectures, power values cannot be directly compared between Watch and PSATSim [2].

b) Model processor architecture

The processor architecture used for simulations is shown in figure 1.2. The Rename Entries and Reorder Entries are 16 and 20 respectively. These values are the default values used in PSATSim simulator. Execution architecture is taken as standard i.e. We can specify Number of Integer, floating, branch, and memory Execution Units separately.

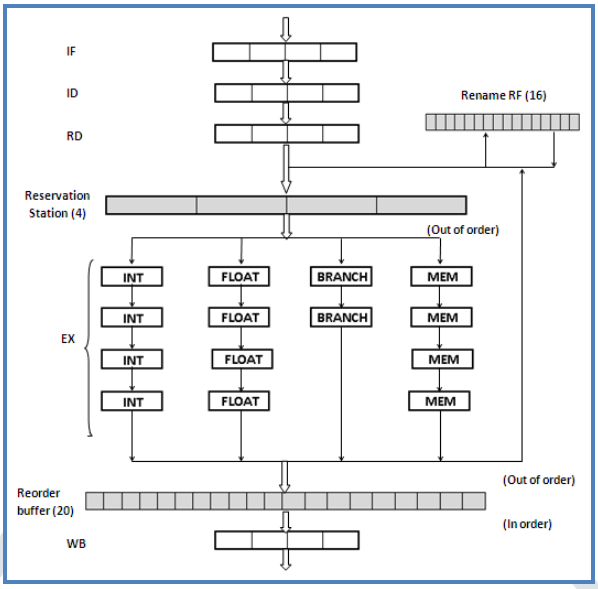

Figure 1.2 Model Processor Architecture

The best mix of these functional units for a superscalar processor depends on the application domain. Typical programs have 40% ALU instructions, 20% branches, and 40% load/store instructions. According to this 4-2-4 rule of thumb [1], for every four ALU units (four integers and four floating point); two branch units, and four load/store units shall be optimum. Therefore Number of Integer Execution Unit, floating point Execution Unit is taken as four. All other parameters are used as default in PSATSim simulator.

V. RESULT ANALYSIS

The simulation has been perform in two different segments. First segment is focused to observe the relation of IPC with memory architecture for various reservation station types and different memory architecture. Second segment is focused for observing the power consumption for various memory architectures.

Results for Intel i3, i5 and i7 Processor

This section presents the results derived for intel i3,i5 and i7 processors. We derived these results based on the experiments and standard formulas. Figure 1.3, 1.5 and 1.7 shows the Impact of Memory architecture on Speed up for Centralized Reservation Architecture for i3, i5, and i7 Processor respectively. From these graph it is clear that the speed up is considerably high for memory system L2. Figure 1.4, 1.6 and 1.8 shows the Impact of Memory architecture on Speed up for Distributed Reservation Architecture for i3, i5, i7 Processor respectively. Derailed results are shown following graphs.

Table 1.1 Results for Intel i3 processor

architecture	memory system	file	f=ipc/sf	For 2+1 Core		
				speedup (f,n)=1/((1-f)+(f/n)) n= 2.5	speedup (f,n)=1/((1-f)+(f/n)) n= 2.6	speedup (f,n)=1/((1-f)+(f/n)) n= 2.8
distributed	L1	applu	0.483605	1.408774127	1.423696442	1.451144876
distributed	L1	applu	0.481425	1.406182987	1.420982445	1.448199707
distributed	l2	applu	0.480035	1.404535808	1.419257357	1.446328057
distributed	l2	applu	0.4821625	1.407058509	1.421899439	1.449194726
distributed	system	applu	0.4811675	1.405877552	1.420662553	1.447852615
distributed	system	applu	0.4838675	1.409086778	1.424023941	1.45150032
centralized	L1	applu	0.2851725	1.206423239	1.212842698	1.224477569
centralized	L1	applu	0.2862275	1.207345247	1.213798461	1.225495292
centralized	l2	applu	0.2852675	1.206506205	1.2129287	1.224569143

centralized	l2	applu	0.36215	1.277612398	1.286771984	1.303459288
centralized	system	applu	0.35435	1.270018669	1.278873018	1.294995306
centralized	system	applu	0.2862275	1.207345247	1.213798461	1.225495292
distributed	L1	compress	0.614735	1.584386819	1.608488862	1.653403028
distributed	L1	compress	0.6312875	1.609716247	1.635281835	1.683013436
distributed	l2	compress	0.5958475	1.556440825	1.578969103	1.620863289
distributed	l2	compress	0.6111425	1.578994323	1.602789347	1.647113566
distributed	system	compress	0.6173625	1.588354187	1.612683133	1.658033543
distributed	system	compress	0.6335225	1.613198545	1.638968106	1.687093045
centralized	L1	compress	0.3262875	1.243429254	1.251239208	1.265432343
centralized	L1	compress	0.3293025	1.246232483	1.254150757	1.268543673
centralized	l2	compress	0.3257075	1.242891438	1.250680659	1.264835561
centralized	l2	compress	0.3292575	1.246190551	1.254107201	1.268497123
centralized	system	compress	0.3262575	1.243401424	1.251210305	1.265401461
centralized	system	compress	0.3293025	1.246232483	1.254150757	1.268543673
distributed	L1	epic	0.492195	1.419077798	1.434492256	1.462867461
distributed	L1	epic	0.4946425	1.422041212	1.437598282	1.466242259
distributed	l2	epic	0.49047	1.416996591	1.432311177	1.460498218
distributed	l2	epic	0.494315	1.421643961	1.437181885	1.465789775
distributed	system	epic	0.49187	1.418685219	1.43408082	1.462420494
distributed	system	epic	0.4946425	1.422041212	1.437598282	1.466242259
centralized	L1	epic	0.217998	1.150481615	1.154937946	1.162982145
centralized	L1	epic	0.2191653	1.151409351	1.155896879	1.163997935
centralized	l2	epic	0.2178273	1.150346027	1.154797803	1.1628337
centralized	l2	epic	0.2191525	1.151399209	1.155886396	1.163986829
centralized	system	epic	0.21792	1.150419673	1.154873923	1.16291433
centralized	system	epic	0.2191653	1.151409351	1.155896879	1.163997935
distributed	L1	fpppp	0.4156225	1.332220485	1.343666473	1.364602551
distributed	L1	fpppp	0.4187525	1.335561941	1.347153051	1.368359759
distributed	l2	fpppp	0.4129125	1.32934088	1.340662287	1.361366131
distributed	l2	fpppp	0.4168275	1.333504911	1.345006611	1.366046571
distributed	system	fpppp	0.4166525	1.333318222	1.344811819	1.365836669
distributed	system	fpppp	0.419325	1.336174932	1.34779273	1.36904922
centralized	L1	fpppp	0.2149313	1.148051255	1.152426079	1.160321761
centralized	L1	fpppp	0.2157068	1.148664858	1.153060231	1.160993352
centralized	l2	fpppp	0.2150558	1.148149719	1.152527839	1.160429527
centralized	l2	fpppp	0.215704	1.148662681	1.153057981	1.160990969
centralized	system	fpppp	0.2152153	1.14827589	1.152658234	1.160567618
centralized	system	fpppp	0.2157068	1.148664858	1.153060231	1.160993352

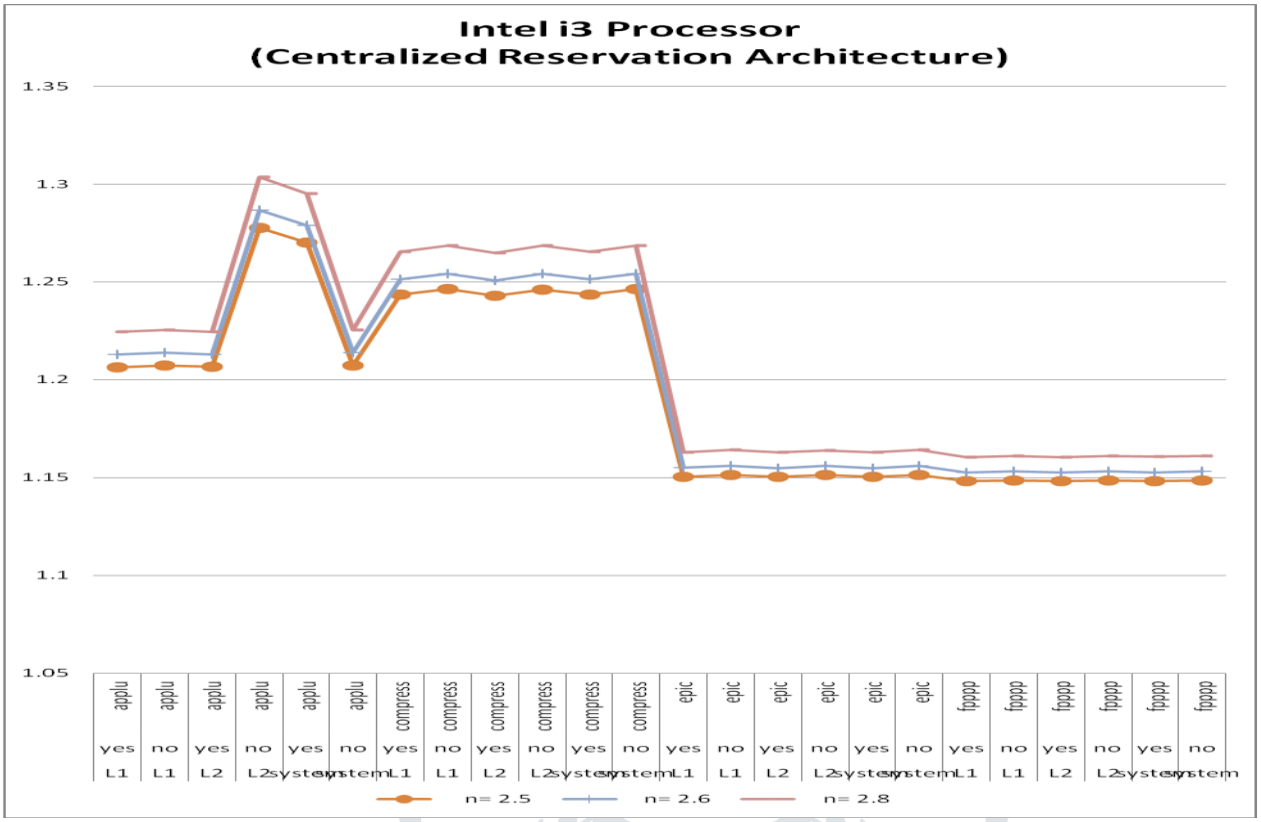

Figure – 1.3 Impact of Memory architecture on Speed up for Centralized Reservation

Architecture for i3 Processor

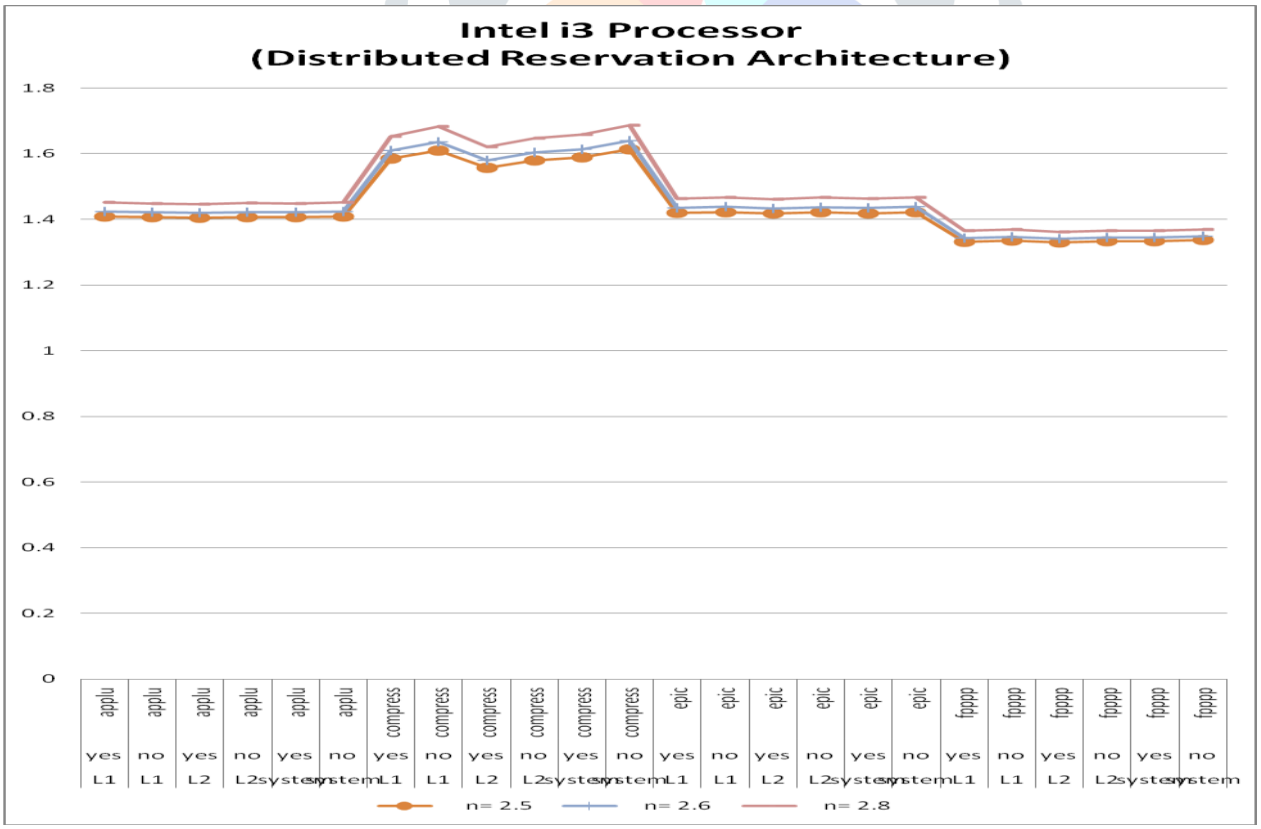

Figure – 1.4 Impact of Memory architecture on Speed up for Distributed Reservation Architecture for i3 Processor

Table 1.2 Results for Intel i5 processor

architecture	memory system	file	f=ipc/sf	For 4+1 Core		
				speedup (f,n)=1/((1-f)+(f/n)) n= 2.5	speedup (f,n)=1/((1-f)+(f/n)) n= 2.55	speedup (f,n)=1/((1-f)+(f/n)) n= 2.75
distributed	L1	applu	0.483605	1.408774127	1.416342285	1.444561979
distributed	L1	applu	0.481425	1.406182987	1.413689084	1.441672865
distributed	l2	applu	0.480035	1.404535808	1.412002548	1.439836749
distributed	l2	applu	0.4821625	1.407058509	1.414585556	1.442648966
distributed	system	applu	0.4811675	1.405877552	1.413376346	1.441332368
distributed	system	applu	0.4838675	1.409086778	1.416662436	1.444910647
centralized	L1	applu	0.2851725	1.206423239	1.20968739	1.221707408
centralized	L1	applu	0.2862275	1.207345247	1.210626524	1.222710287
centralized	l2	applu	0.2852675	1.206506205	1.209771897	1.221797648
centralized	l2	applu	0.36215	1.277612398	1.282265638	1.299476075
centralized	system	applu	0.35435	1.270018669	1.274517277	1.291148007
centralized	system	applu	0.2862275	1.207345247	1.210626524	1.222710287
distributed	L1	compress	0.614735	1.584386819	1.596583187	1.642562068
distributed	L1	compress	0.6312875	1.609716247	1.622648998	1.671481674
distributed	l2	compress	0.5958475	1.556440825	1.567844915	1.610761645
distributed	l2	compress	0.6111425	1.578994323	1.591036165	1.636417124
distributed	system	compress	0.6173625	1.588354187	1.600664747	1.647085688
distributed	system	compress	0.6335225	1.613198545	1.626233895	1.675464761
centralized	L1	compress	0.3262875	1.243429254	1.247398578	1.26204862
centralized	L1	compress	0.3293025	1.246232483	1.250256716	1.265111978
centralized	l2	compress	0.3257075	1.242891438	1.246850252	1.261461018
centralized	l2	compress	0.3292575	1.246190551	1.250213961	1.265066147
centralized	system	compress	0.3262575	1.243401424	1.247370204	1.262018214
centralized	system	compress	0.3293025	1.246232483	1.250256716	1.265111978
distributed	L1	epic	0.492195	1.419077798	1.426894528	1.456059757
distributed	L1	epic	0.4946425	1.422041212	1.429929962	1.459369334
distributed	l2	epic	0.49047	1.416996591	1.424762882	1.453736168
distributed	l2	epic	0.494315	1.421643961	1.429523042	1.458925607
distributed	system	epic	0.49187	1.418685219	1.426492426	1.455621411
distributed	system	epic	0.4946425	1.422041212	1.429929962	1.459369334
centralized	L1	epic	0.217998	1.150481615	1.152749164	1.161070693
centralized	L1	epic	0.2191653	1.151409351	1.153692748	1.162072909
centralized	l2	epic	0.2178273	1.150346027	1.152611263	1.16092423
centralized	l2	epic	0.2191525	1.151399209	1.153682433	1.162061952
centralized	system	epic	0.21792	1.150419673	1.152686166	1.161003783
centralized	system	epic	0.2191653	1.151409351	1.153692748	1.162072909
distributed	L1	fpppp	0.4156225	1.332220485	1.338031222	1.35959536
distributed	L1	fpppp	0.4187525	1.335561941	1.341446101	1.363287233
distributed	l2	fpppp	0.4129125	1.32934088	1.335088582	1.356414995
distributed	l2	fpppp	0.4168275	1.333504911	1.339343836	1.361014302
distributed	system	fpppp	0.4166525	1.333318222	1.339153047	1.360808048
distributed	system	fpppp	0.419325	1.336174932	1.342072594	1.363964673
centralized	L1	fpppp	0.2149313	1.148051255	1.150277399	1.158445761

centralized	L1	fpppp	0.2157068	1.148664858	1.150901441	1.159108415
centralized	l2	fpppp	0.2150558	1.148149719	1.150377538	1.158552093
centralized	l2	fpppp	0.215704	1.148662681	1.150899227	1.159106064
centralized	system	fpppp	0.2152153	1.14827589	1.150505854	1.158688347
centralized	system	fpppp	0.2157068	1.148664858	1.150901441	1.159108415

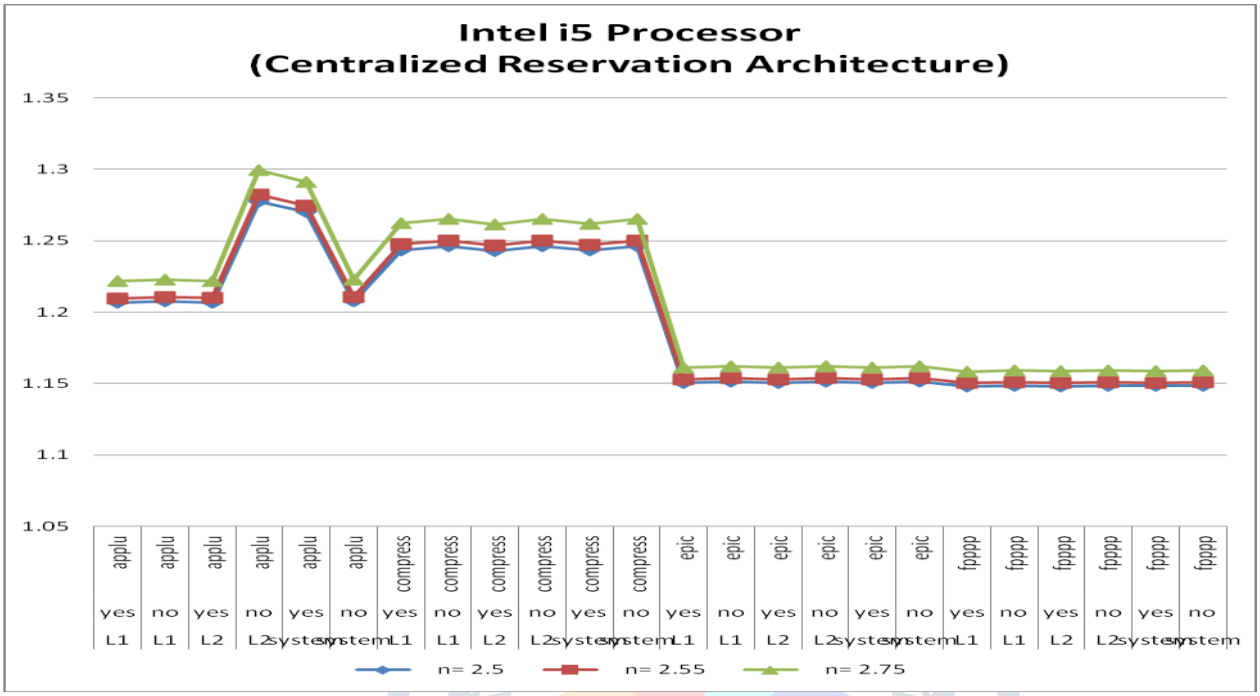

Figure – 1.5 Impact of Memory architecture on Speed up for Centralized Reservation Architecture for i5 Processor

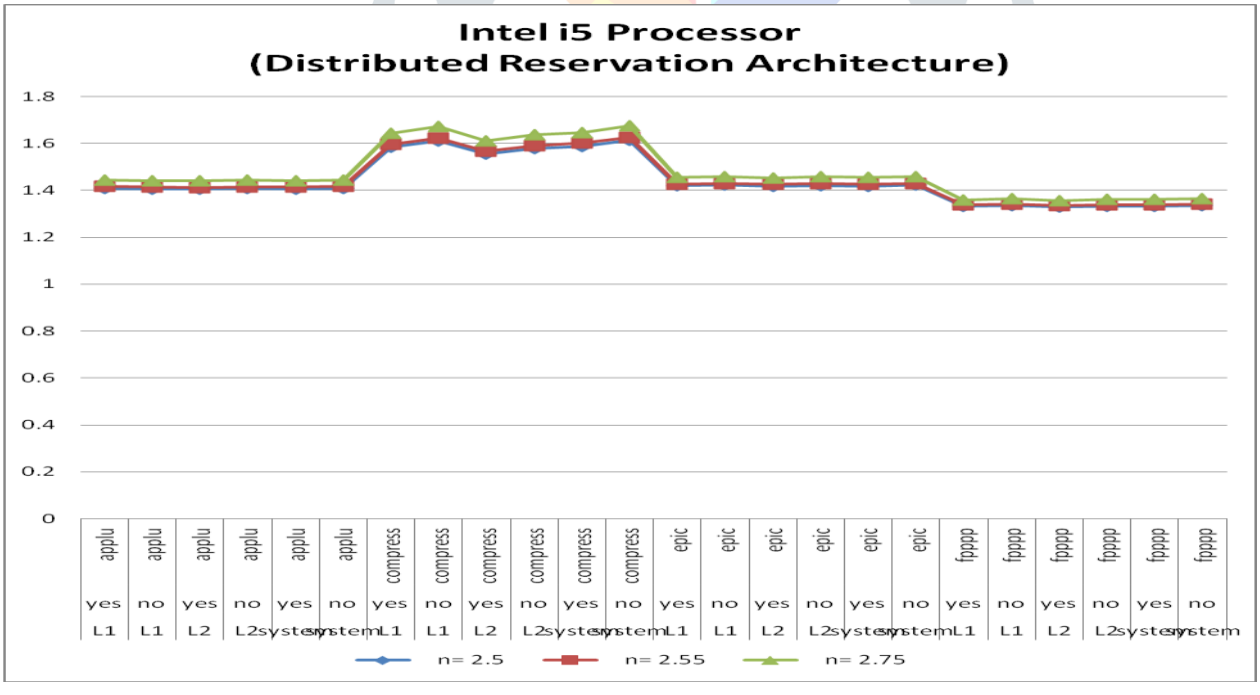

Figure – 1.6 Impact of Memory architecture on Speed up for Distributed Reservation Architecture for i5 Processor

Table 1.3 Results for Intel i7 processor

architecture	memory	file	f=ipc/sf	For 6+1 Core

	system			speedup (f,n)=1/((1-f)+(f/n)) n= 6.45	speedup (f,n)=1/((1-f)+(f/n)) n= 6.55	speedup (f,n)=1/((1-f)+(f/n)) n= 6.7
distributed	L1	applu	0.483605	1.690981517	1.694261027	1.699019247
distributed	L1	applu	0.481425	1.685730779	1.688975214	1.693682375
distributed	l2	applu	0.480035	1.682399822	1.685622088	1.690296976
distributed	l2	applu	0.4821625	1.687503465	1.690759724	1.695484097
distributed	system	applu	0.4811675	1.685112718	1.688353035	1.693054201
distributed	system	applu	0.4838675	1.691615981	1.694899739	1.699664145
centralized	L1	applu	0.2851725	1.317453121	1.318625757	1.320322768
centralized	L1	applu	0.2862275	1.319002187	1.320181936	1.321889258
centralized	l2	applu	0.2852675	1.317592461	1.318765737	1.320463674
centralized	l2	applu	0.36215	1.440927823	1.442709828	1.445290886
centralized	system	applu	0.35435	1.427372467	1.429083374	1.431561224
centralized	system	applu	0.2862275	1.319002187	1.320181936	1.321889258
distributed	L1	compress	0.614735	2.080850394	2.087169935	2.096363576
distributed	L1	compress	0.6312875	2.143225176	2.150110983	2.160132735
distributed	l2	compress	0.5958475	2.013969061	2.019705932	2.028048056
distributed	l2	compress	0.6111425	2.067789265	2.073993026	2.083017416
distributed	system	compress	0.6173625	2.09050806	2.096913835	2.106233548
distributed	system	compress	0.6335225	2.151935038	2.158901673	2.169041673
centralized	L1	compress	0.3262875	1.380643934	1.382117692	1.384251403
centralized	L1	compress	0.3293025	1.385517173	1.387015088	1.389183846
centralized	l2	compress	0.3257075	1.379710391	1.381179536	1.383306556
centralized	l2	compress	0.3292575	1.385444185	1.386941737	1.389109969
centralized	system	compress	0.3262575	1.380595616	1.382069135	1.3842025
centralized	system	compress	0.3293025	1.385517173	1.387015088	1.389183846
distributed	L1	epic	0.492195	1.71199369	1.71541512	1.720379968
distributed	L1	epic	0.4946425	1.718076499	1.721539478	1.72656483
distributed	l2	epic	0.49047	1.707732334	1.71112478	1.716047425
distributed	l2	epic	0.494315	1.717260054	1.720717445	1.725734657
distributed	system	epic	0.49187	1.711189201	1.714605151	1.71956202
distributed	system	epic	0.4946425	1.718076499	1.721539478	1.72656483
centralized	L1	epic	0.217998	1.225790425	1.226566241	1.227688277
centralized	L1	epic	0.2191653	1.227274168	1.22805603	1.229186823
centralized	l2	epic	0.2178273	1.225573678	1.226348612	1.22746937
centralized	l2	epic	0.2191525	1.227257941	1.228039738	1.229170434
centralized	system	epic	0.21792	1.225691404	1.226466817	1.227588269
centralized	system	epic	0.2191653	1.227274168	1.22805603	1.229186823
distributed	L1	fpppp	0.4156225	1.541271165	1.5436117	1.547004074
distributed	L1	fpppp	0.4187525	1.547579476	1.549957023	1.55340319
distributed	l2	fpppp	0.4129125	1.53585073	1.538159642	1.541506057
distributed	l2	fpppp	0.4168275	1.543693666	1.546048387	1.54946138
distributed	system	fpppp	0.4166525	1.543341378	1.545694034	1.549104025
distributed	system	fpppp	0.419325	1.548738903	1.551123277	1.554579367
centralized	L1	fpppp	0.2149313	1.221909187	1.222669244	1.22376846
centralized	L1	fpppp	0.2157068	1.222888326	1.223652351	1.224757312

centralized	l2	fpppp	0.2150558	1.222066274	1.222826967	1.223927104
centralized	l2	fpppp	0.215704	1.222884851	1.223648862	1.224753803
centralized	system	fpppp	0.2152153	1.222267581	1.22302909	1.224130408
centralized	system	fpppp	0.2157068	1.222888326	1.223652351	1.224757312

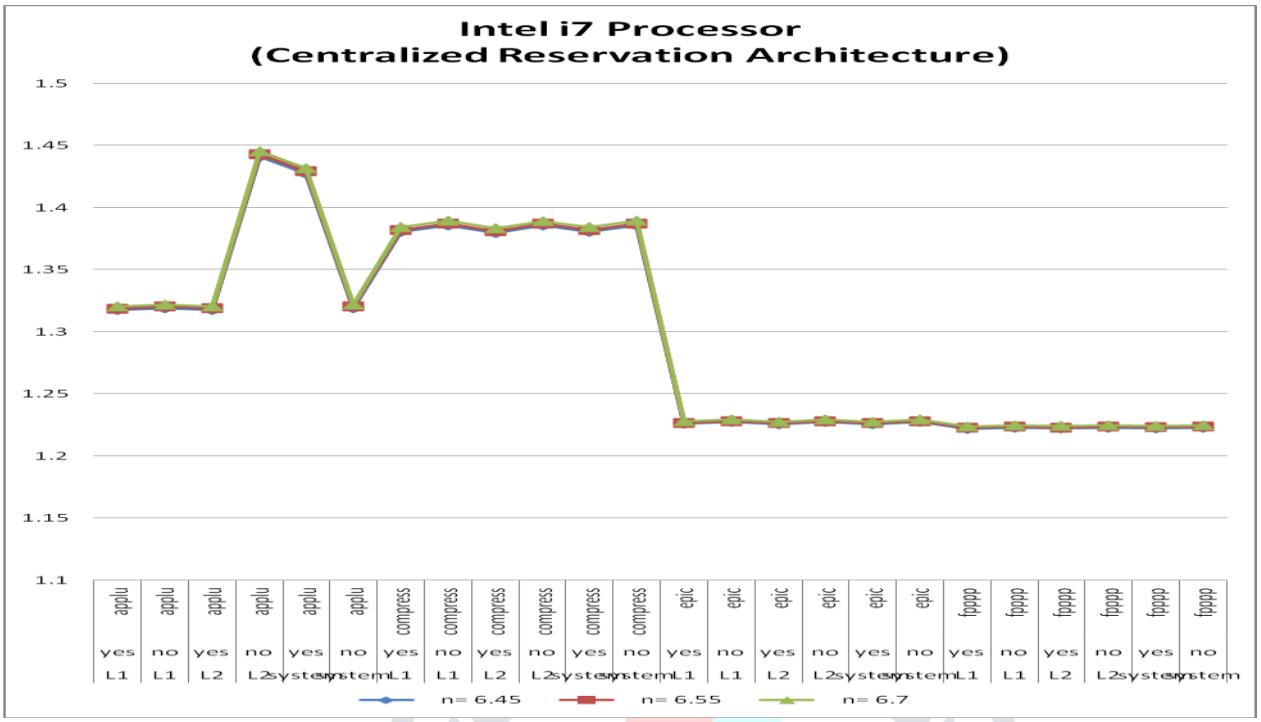

Figure – 1.7 Impact of Memory architecture on Speed up for Centralized Reservation Architecture for i7 Processor

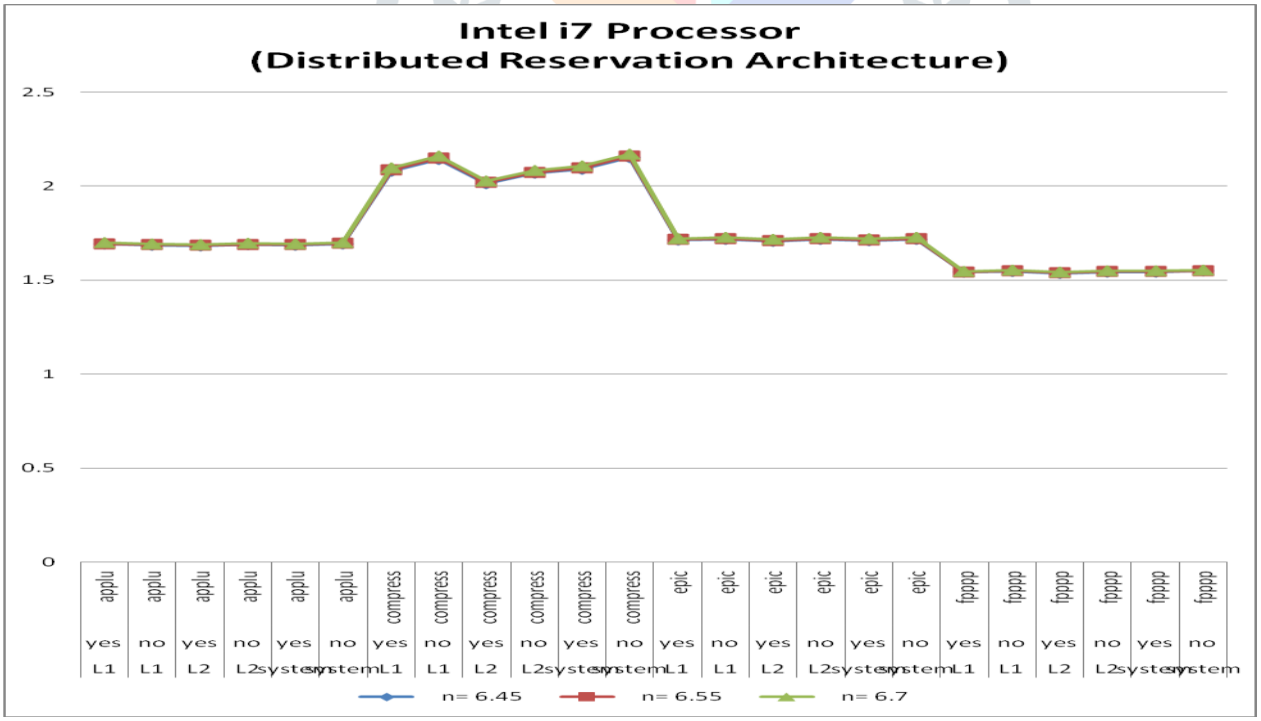

Figure – 1.8 Impact of Memory architecture on Speed up for Distributed Reservation Architecture for i7 Processor

If we combine the finding from pervious section, then on averaging the L2 memory system is found optimal then the L1 and System memory architecture.

VI. CONCLUSION

This work aims at finding optimal values for the superscalar processor by conducting a qualitative and quantitative study. This is achieved by using PSATSim simulation tool which provides the right platform for evaluating the superscalar processors by varying the parameter values.

The basic purpose of this work is conceptually based on two different segments viz. :

- The first part has been conducted for power analysis considering the super scalar factor already established.
- The second part has been conducted for IPC analysis considering the super scalar factor and power.

The impact of superscalar factor and power analysis is analyzed on performance of Superscalar Pipelined Architecture in terms of IPC.

This analysis has been performed for distributed and centralised reservation architecture with superscalar factor 4 and three memory architectures viz.

- Case 1 when the system has only L1 cache Memory
- Case 2 when the system has L1 & L2 Cache Memory
- Case 3 when the system has system memory

After performing simulation, It is found that for distributed reservation architecture and superscalar factor 4, the power consumption for processor architecture having only L2 cache memory is highest, and architecture having L1 & system is lowest.

If we consider the above result with the results derived for i3, i5 and i7 processor then on averaging the L2 memory system is found optimal then the L1 and System memory architecture.

REFERENCES

[1] Shen John, Lipasti Mikko, Modern Processor Design: Fundamentals of Superscalar Processors, First edition, TMH, 2004

[2] Clint W. Smullen, IV, Tarek M. Taha, "PSATSim: An Interactive Graphical Superscalar Architecture Simulator for Power and Performance Analysis", 33nd International Symposium on Computer Architecture, 2006

[3] Vladimir Lazarov, Maria Marinova, "Dependencies Evaluation in Superscalar Processors", CompSysTech '04 Proceedings of the 5th international conference on Computer systems and technologies, 2004, Pages 1 – 4

[4] Hwang Kai, Jotwani Naresh, Advanced Computer Architecture, Second edition, TMH, 2010

[5] Hartstein, A.; Puzak, T.R., "The optimum pipeline depth for a microprocessor," Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on , vol., no., pp.7,13, 2002

[6] Jangid Deepak, Jha Suresh, Purohit Rajesh. "An Improved Register Data Flow Technique in Superscalar Architecture", UGC National Conference on Emerging Trends in Computer Communication and Networks, ETCN-2012

[7] J. Yi and D. Lilja, "Effects of Processor Parameter Selection on Simulation Results," Minnesota Supercomputer Inst. Report 2002/146, 2002.

[8] Arpad Gellert, Horia Calborean, Lucian Vintan, Adrian Florea, 'Multi-Objective Optimizations for a Superscalar Architecture with Selective Value Prediction', IET Computers & Digital Techniques, Vol. 6, Issue 4, ISSN: 1751-8601, pp. 205-213, 2012

[9] James E. Bennett , Michael J. Flynn, " Performance factors for superscalar processors", Tech. Report CSL-TR-95-661, Computer Systems Laboratory, Stanford University, February 1995

[10] Wallace, S.; Bagherzadeh, N., "Performance Issues of a Superscalar Microprocessor," Parallel Processing, 1994. Vol. 1. ICPP 1994. International Conference on, vol.1, no., pp.293,297, 15-19 Aug. 1994

[11] Taha, T.M.; Wills, D.S., "An Instruction Throughput Model of Superscalar Processors," Computers, IEEE Transactions on , vol.57, no.3, pp.389,403, March 2008

[12] Taiwo O. Ojeyinka, Olusola Olajide Ajayi "Performance Analysis of Dual Core, Core 2 Duo and Core i3 Intel Processor", International Journal of Computer Applications (0975 – 8887) Volume 120 – No.10, June 2015

[13] Rebaya, K. Gasmi, I. Amari and S. Hasnaoui, "Performance analysis of an efficient technique for ordering programs into multiple processors architectures," *2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, Gafsa, 2017, pp. 124-128.

[14] S. Dhotre, P. Patil, S. H. Patil and R. Jamale, "Analysis of scheduler settings on the performance of multi-core processors," *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, Tirunelveli, 2017, pp. 687-691.

[15] C. Damman, G. Edison, F. Guet, E. Noulard, L. Santinelli and J. Hugues, "Architectural performance analysis of FPGA synthesized LEON processors," *2016 International Symposium on Rapid System Prototyping (RSP)*, Pittsburgh, PA, 2016, pp. 1-8.

[16] X. Zhang, B. Yin and H. Shi, "Performance analysis of multi-server based on processor-sharing queue," *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Pyeongchang, 2016, pp. 843-848.

[17] Vitale, G. Rizzo, B. Rengarajan and V. Mancuso, "An Analytical Approach to Performance Analysis of Coupled Processor Systems," *2015 27th International Teletraffic Congress*, Ghent, 2015, pp. 89-97.

[18] Y. C. Chan, J. Guo, E. W. M. Wong and M. Zukerman, "Performance analysis for overflow loss systems of processor-sharing queues," *2015 IEEE Conference on Computer Communications (INFOCOM)*, Kowloon, 2015, pp. 1409-1417.