POWER AND PERFORMANCE CHARACTERIZATION OF SPLASH2 BENCHMARKS ON HETEROGENEOUS ARCHITECTURE

by

Matthew Drummond

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical and Computer Engineering

Committee:

_____	Dr. Houman Homayoun, Thesis Director
_____	Dr. Jim Jones, Committee Member
_____	Dr. Brian Mark, Committee Member
_____	Dr. Monson Hayes, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: Thurs, June 18, 2015

Summer Semester 2015
George Mason University
Fairfax, VA

Power and Performance Characterization of Splash2 Benchmarks on Heterogeneous Architecture

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Matthew Drummond
Bachelor of Science
University of Notre Dame, 2011

Director: Houman Homayoun, Assistant Professor
Electrical and Computer Engineering

Summer Semester 2015
George Mason University
Fairfax, VA

DEDICATION

I dedicate this thesis to my ever supportive parents Ken and Mickey, my constantly inspiring sister Amanda, and my beautiful fiancé Meredith, whom I would not be able to live without. I would be nothing if not for the love and support of each of you.

I would like to thank my family, friends, coworkers, and professors, especially my thesis advisor Dr. Homayoun, for helping me through my graduate school journey

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

POWER AND PERFORMANCE CHARACTERIZATION OF SPLASH2 BENCHMARKS ON HETEROGENEOUS ARCHITECTURE

Matthew Drummond, M.S.

George Mason University, 2015

Thesis Director: Dr. Houman Homayoun

The computing industry has constantly struggled between speed and power. To achieve the desired performance cores are becoming larger and more complicated. This of course comes at the cost of higher area and power consumption, which is unsustainable. An optimum configuration for any application must exist for peak performance and energy efficiency. This paper shows how such an optimization can be found using Splash2 benchmarks as demonstration. Sensitivity analysis was performed on each benchmark for multiple configuration parameters, particularly frequency, core type, and thread count. The resulting data was analyzed to determine what influence the parameters had on EDP for each benchmark. Finally, the benchmarks were instrumented in such a way as to highlight individual parallel regions to determine if alternative configurations are more appropriate for these sub-regions of the applications, demonstrating the utility of a heterogeneous architecture.

1. INTRODUCTION

In its attempts to find a solution to constantly growing power and area concerns the computing industry is beginning to experiment with heterogeneous cores. While static embedded cores can be highly tuned to an individual application, in more dynamic applications this can result in large sections of the processor cores being underutilized in what is known as the Dark Silicon Problem [38]. In future designs involving many-cores homogeneous architectures simply compound this problem. The solution lies in heterogeneous architectures, ones that can be configured during an application to optimally suit the particular processing needs at any point. This paper is an exploration into the feasibility of such an effort, and will demonstrate how it might improve performance and energy consumption. To do so, three areas of investigation will be examined, each one building on the results and findings of the previous.

The first area of investigation involved determining an application's sensitivity to certain parameters. The parameters being adjusted include voltage and frequency scaling, core size (big vs little), and number of threads to individually evaluate sensitivity to each. For each sensitivity analysis the execution time, total power, Energy Delay Product (EDP), and Energy Delay^2 Product (ED2P) were recorded.

While the first area explores sensitivity to individual parameters, the investigation will proceed with a more extensive analysis, testing all permutations of the

aforementioned parameters. All possible configurations of voltage/frequency, core type, and thread count will be examined. Using the results from this experiment an optimum configuration can be found for any benchmark, both with and without power constraint considerations.

This investigation was taken further to explore the impact of a heterogeneous architecture. Previously, results for each benchmark were from the overall simulated region of interest. This region typically includes any number of parallel sub-regions amidst serial regions. It is often the case that not all of these regions have the same execution profile. As such, these sections will often benefit from a core configuration that is different from the others. This final effort involved simulations of these internal parallel sub-regions with the same permutations of parameters as were used previously. Using these results an optimal heterogeneous architecture scheme can be determined.

2. MOTIVATION

Homogenous architectures are becoming inadequate for processing needs. They are constantly over-provisioned in order to suit the most demanding application, despite needing that many resources for fractions of the time. Heterogeneous architectures leverage the fact that a simpler core can very often yield significantly better results [28]. Doing so can reduce or eliminate over-provisioning without necessarily sacrificing performance. This allows for significant gains in performance per area and per watt [31].

There are many contributing factors to the goal of high performance while focusing on energy efficiency. The diagram in Figure 1 shows many of these parameters. This paper is focusing primarily on four of these: core micro-architecture, voltage/frequency, thread count, and application phase monitoring.

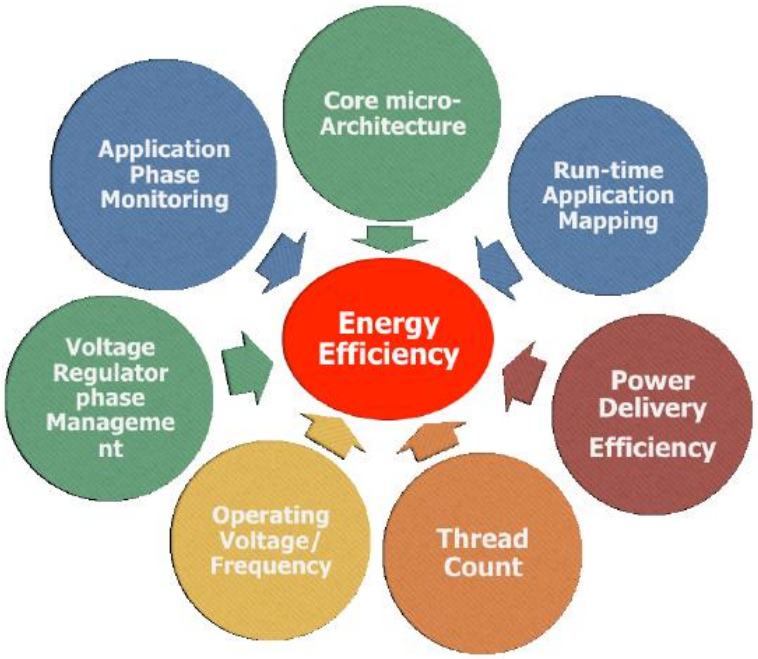

Figure 1 – Configuration factors of Energy Efficiency [32]

Heterogeneous architectures can provide more opportunity for efficient workload mappings so an application can find a better match among the various configuration parameters. In order to actually see the effects such a mapping can offer this paper compares results from simulations of all these parameters, both over full simulations as well as sub-regions within the simulation. These sub-regions show the different aspects of an application and will show that the same configuration is not necessarily optimal for all regions.

3. BACKGROUND ON HETEROGENEOUS ARCHITECTURES

The driving theory behind heterogeneous architectures is that not all cores of a processor are configured in the same manner. This allows for designs that are optimized for specific parts of an application, whereas homogeneous architectures process all portions in the same manner. Each thread can run on a core that more closely matches the needs of that thread than a single generic core. This allows for the architecture to provide more efficient workload mappings. Applications can find cores that more appropriately match their workload which improves power efficiency. These mappings are critical in exploiting heterogeneity; finding an architecture with just enough resources to match the workload needs to best improve energy efficiency. [32]

There are two different base varieties of heterogeneous architectures, static and dynamic. When an architecture is statically heterogeneous the processor cores can be different but each core is configured at the time of design. Throughout execution the configuration does not change. An architecture is dynamically heterogeneous when the actual core configurations changes to meet an application's exact needs at any particular point in execution during runtime. While a statically heterogeneous architecture that is perfectly configured to an application's needs would provide the best performance and efficiency at the lowest cost, it is not feasible without extreme application profiling. Since many applications today are enormously dynamic in nature this becomes even

more difficult. A dynamic heterogeneous architecture, though more costly to configure, provides a better chance of successful application mappings [32].

Investigations into statically heterogeneous architectures are well under way. Intel's Quick IA [5] combines a big Xeon architecture with a little Atom for high performance. TI OMAP 5 and NVIDIA Tegra combine small ARM-8 and 9 with big ARM-15 processors for embedded designs. There have also been efforts to provide dynamically heterogeneous architectures as seen in Core Fusion [2], TFlex [7], and WiD-GET [8].

One of the biggest issues involved in heterogeneous architectures is the fact that the number of configuration options is overwhelming. There are so many different pieces to a processor, all of which have a large impact on performance and energy efficiency. To test all options for all parameters would be a monumental task. Though this means that there is incredible potential for improvement, the investment required to capitalize on it would be very large.

4. RELATED WORK

Attempting to map application profiles to core configurations has been examined in a variety of instances. The Thread Reinforcer work examined the mapping between number of threads and number of cores to find the best number of threads to execute PARSEC applications, particularly in many core architectures [9]. Similarly, the ElasticCore effort took a closer look at voltage and frequency scaling [3]. ElasticCore went a step further to actually scale voltage and frequency at runtime, demonstrating a partially dynamic heterogeneous design. In the many-core realm an effort was undertaken to map biomedical applications to domain-specific accelerators, dealing with a 128-core accelerator and trying to appropriately map a task's voltage and frequency assignments to cores on the accelerator [35].

Another work examined the potential for power reduction in single-ISA heterogeneous multi-core architectures [23]. This work used the SPEC benchmarks and the SMTSim simulator to investigate the impact of choosing the most appropriate core to meet specific performance and power requirements during an application's execution. Additional work dealing with tiled multiprocessors created a programming model and applied this model to application specifications [25], resulting in a similar form of application mapping as would be required for heterogeneous architectures. Though that particular work is in the context of tiled embedded systems, the theory of application

mapping holds for all heterogeneous architectures.

Many other works have touched on the need for task/application mapping particularly for Multiprocessor Systems on a Chip (MPSoC) [36-37]. Having multiple processors available on a single chip lends itself perfectly to heterogeneity, and finding a good mapping of applications to the different processors is critical.

Many of these works focused primarily on an individual characteristic, such as thread count or voltage/frequency. This paper shows that these parameters individually, while important, do not make a truly optimum configuration. Only when combining the parameter variations can the absolute best configuration be found for an application.

5. METHODOLOGY

The Sniper Multi-Core Simulator [10] was used to collect performance metrics from simulating benchmarks. McPAT [11] integrates with Sniper and was used to obtain power results for the simulations. Sniper provides versions of various benchmarks already tailored and instrumented to work with Sniper. This work utilized selections from the included Splash2 [12] suite of benchmarks. Sniper uses Region of Interest (ROI) markers to start and stop recording processing metrics, avoiding the less critical initialization and cleanup phases [34]. Custom simulation markers were added to the benchmarks for the purpose of gathering statistics on sub-regions of the overall benchmark.

Simulations were performed on the ARGO Computing Cluster at GMU [33]. The cluster was essential for performing the many simulations in a timely fashion, but proved to be troublesome in other aspects. Due to being a shared, university controlled resource access to certain system settings and libraries were restricted. Sniper and the associated benchmarks required modifications to these settings. Some of these obstacles were overcome, but others persisted requiring the scope of this work to be limited to only Splash2 benchmarks. Though this was unexpected, the suite proved to be sufficiently representative for our purposes.

The Splash2 benchmarks are a widely used suite of parallel applications with various different profiles [12]. The actual details of the benchmarks were less of a concern than whether or not they displayed decent variation between them, to give a better look at varying profiles for different configurations. Table 1 below shows a breakdown of the Splash2 benchmark technical details.

Table 1 – Breakdown of Splash2 Benchmarks [12]

Code	Problem Size	Total Instr (M)	Total FLOPS (M)	Total Reads (M)	Total Writes (M)	Shared Reads (M)	Shared Writes (M)	Barriers	Locks	Pauses
Barnes	16K particles	2002.79	239.24	406.85	313.29	225.05	93.23	8	34648	0
Cholesky	tk15.O	539.17	172.00	111.86	28.03	75.87	23.31	3	54054	4203
FFT	64K points	34.79	6.36	4.07	2.88	4.05	2.87	6	0	0
FMM	16K particles	1250.02	423.88	226.23	38.58	217.84	30.10	20	28088	0
LU	512 × 512 matrix, 16 × 16 blocks	494.05	92.20	104.00	48.00	93.20	44.74	66	0	0
Ocean	258 × 258 ocean	379.93	101.54	81.89	18.93	80.26	17.27	364	2592	0
Radiosity	room, -ae 5000.0 -en 0.050 -bf 0.10	2832.47	---	499.72	284.61	261.08	21.99	10	231190	0
Radix	1M integers, radix 1024	50.99	---	12.06	7.03	12.06	7.03	10	0	124
Raytrace	car	829.32	---	208.90	79.95	159.97	22.22	0	94471	0
Volrend	head	754.77	---	152.19	59.57	81.93	3.07	15	28934	0
Water-Nsq	512 molecules	460.52	98.15	81.27	35.25	69.07	26.60	10	17728	0
Water-Sp	512 molecules	435.42	91.50	72.31	32.73	60.54	22.64	10	353	0

The baseline configurations used for simulation were the Atom Silvermont [15] and Xeon 5500 Series (known as Gainestown) [17] cores. The Silvermont architecture is a 22nm Tri-Gate System on a Chip (SoC). The purpose of this architecture is to provide a much greater increases in performance, but more importantly in energy efficiency. Silvermont was designed specifically for low power applications. The intended targets for Silvermont range from small hand-help appliances, like personal tablets, to large data

center computing [16]. This shows that the architecture was intended as a more generic solution rather than a very specifically designed one, which aligns very well with the idea of heterogeneous architectures.

Of particular importance is Silvermont's Uncore event set. This is a set of architectural performance monitoring events. They allow for insight into the actual behavior of the platform. This can be used for platform characterization, performance debugging and optimization, and most importantly application characterization and tuning.

The Xeon Gainestown was created as an answer to many of the IT infrastructure challenges, particularly issues with power. Similar to Silvermont, Gainestown is designed to deliver greater performance more efficiently [18]. Again, similar to the Uncore event set of Silvermont, Gainestown architectures also provide what is referred to as Intelligent Performance [19]. The intelligent performance characteristic of Gainestown is designed to adapt performance and power usage to more closely suites the needs of the applications and workloads.

Both the Uncore event set of Silvermont and the Intelligent Performance of Gainestown show essential aspect of heterogeneous architectures and again supports the driving theory of this paper. These show that trends are already in place to adapt to applications, changing cores to improve performance and efficiency.

Table 2 below shows the key differences in these architectures as seen in the baseline simulation configurations.

Table 2: Differences in Baseline Simulation Configurations

Characteristic	**Atom Silvermont**	**Xeon Gainestown**
Dispatch Width	2	4
Window Size	32	128
Levels of Cache	2	3
L2 - Cache Assoc	16	8
L2 - Cache Size	1024	256
L2 - Data Access Time	12	8
L2 – Shared Cores	2	1
L3 – Cache Present	No	Yes
D-TLB Size/Assoc	48/48	64/4
I -TLB Size/Assoc	48/48	128/4
S-TLB Size/Assoc	128/4	512/4

6. SENSITIVITY ANALYSIS

The purpose of benchmark applications is to test a wide variety of processing scenarios. As such, not all benchmarks will respond the same way to changes in parameters. To demonstrate this, and to determine what benchmarks are more sensitive to certain parameters, sensitivity analysis was performed for each benchmark being studied. The specific parameters include sensitivity to frequency, core type, and number of threads.

The entire set of results is quite m extensive, so results have been combined into a single set of graphs, seen in Figures 2-11. In these figures there are four graphs for each benchmark that show normalized data for the overall execution time, total power, EDP, and ED2P. They are normalized to a base configuration of a Xeon core running at 3.2GH, 1V at each different possible thread count. Each benchmark and frequency/voltage combination is listed in the legend for the graph. For each simulation values for execution time and performance were taken from the Sniper output and McPAT was run to collect power results. From these results, calculations were made to determine the total power, EDP, and ED2P.

6.1 Frequency Sensitivity

The first of the three main parameters subject to sensitivity analysis was voltage and frequency. All benchmarks were simulated using a baseline Xeon-like core type

running with only one thread. The frequency was changed between 2, 2.4, 2.8, and 3.2 GH with the voltage ranging changing between 0.7, 0.8, 0.9, and 1V, respectively. Note that this section is focused on the Xeon points in the first column, one thread, in the figures below.

With the results in hand, each benchmark's sensitivity to frequency can be seen. It is not surprising that the trend is constant across benchmarks. As the frequency increases, the performance increases. Similarly, higher frequencies use more power. EDP results show that the higher frequency gives higher EDP. However, the ED2P results are lower for higher frequency processors. Since ED2P puts more emphasis on delay (performance) the higher frequencies yield better results.

6.2 Core Type Sensitivity

Core type was the next parameter to be analyzed to determine benchmark sensitivity. In this case the baseline configuration was a core running a single thread at a frequency of 3.2 GH and voltage of 1V. The changing parameter was the core type, which change between an Atom-like core and one resembling the Xeon architecture. On the graphs below this section is only concerned with the xeon_3.2GH and atom_3.2GH entries again in the first column (one thread).

Core type did show some variation with regards to EDP. Typically there was a definite gap between the Xeon and Atom cores, with Xeon giving better results. There were a few cases, such as barnes and raytrace, where the smaller Atom core had the lower EDP. In these cases the energy savings of the small core outweigh the performance

benefit of the larger core. For ED2P, as before, since more emphasis is placed on delay the larger, more powerful core excels.

6.3 Sensitivity to Number of Threads

Finally, each benchmark is run with varying numbers of threads. A single Atom core was used, and each simulation was done at the same frequency of 3.2 GH and voltage of 1V. The number of threads was changed to all values between one (1) and eight (8), though some benchmarks would only run with a number of threads that was a power of two.

Since the results presented below are normalized, the trends are harder to discern. However, the results from increasing the thread count are very much as expected. More threads yield better performance at the cost of higher power. EDP and ED2P also increase as thread counts rise. The trend in EDP and ED2P stands to reason, since adding X threads does not increase performance by a factor of X, though that is very nearly the case for power.

The problem with looking at each parameter individually is that it does not come close to telling the entire story. It is only when looking at the full spectrum of results that we can really see where certain architectures excel when compared to the others. It is for this reason that the result set is presented as a full set of data normalized to a baseline configuration.

There are a variety of patterns throughout the benchmarks. The above trends continue, with bigger, faster cores running more threads yield better performance and higher power. What we are really concerned with are the variations in the trends. In the

radiosity benchmark we see that the lower frequency Xeon cores are more efficient in terms of EDP from even as low as one thread. As the number of threads increase the baseline core is actually the least efficient. The ED2P graph shows the same trend, though the less powerful cores are not more efficient at the lower thread counts. The raytrace benchmark shows this same trend to a higher degree.

In the fmm, radix, and water.nsq benchmark there is a clear delineation between the different core types. The Xeon core proves to be more efficient across the board, at all thread counts. For fmm in particular we can also see that lower frequencies provide better EDP and nearly equivalent ED2P as the higher frequencies.

The important take away from this range of results is that the metrics show clear variations across configurations. Though many of the overarching trends are similar, different benchmarks exhibit very different responses to certain configuration changes. This is very important motivation for the subsequent sections of this paper. If the benchmarks react differently at even top level simulations then lower level instrumentation should prove promising.

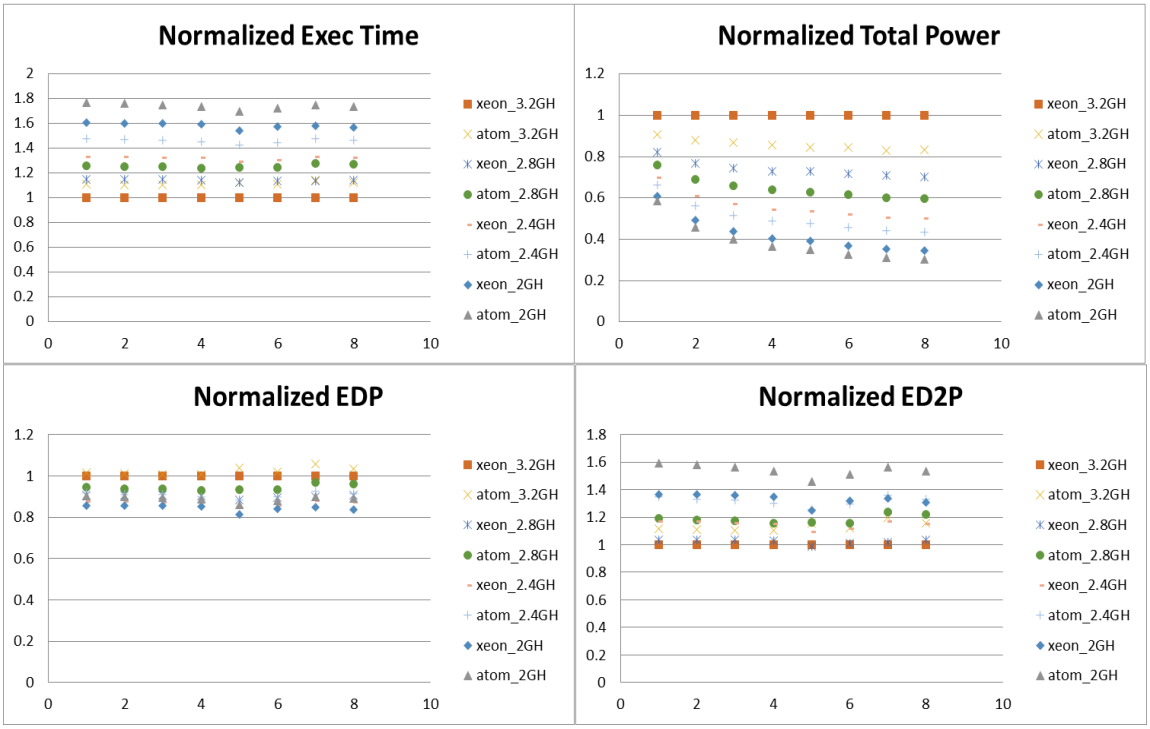

Figure 2 - Results for barnes simulations.

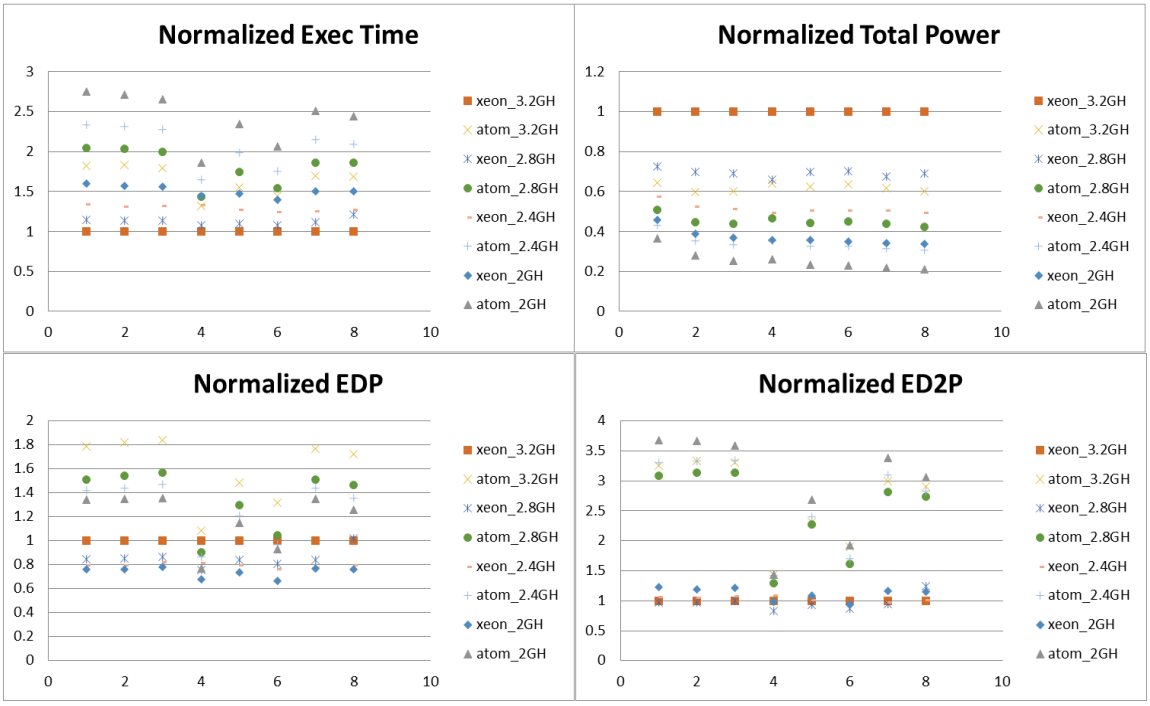

Figure 3 - Results for cholesky simulations.

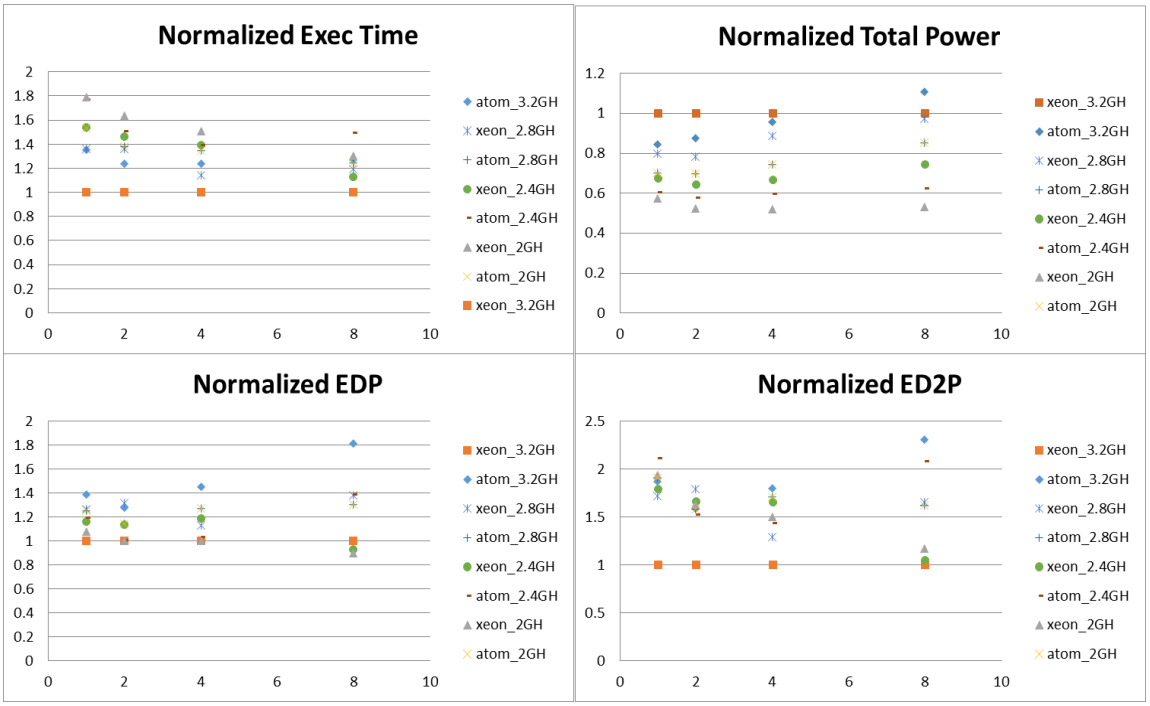

Figure 4 - Results for fft simulations.

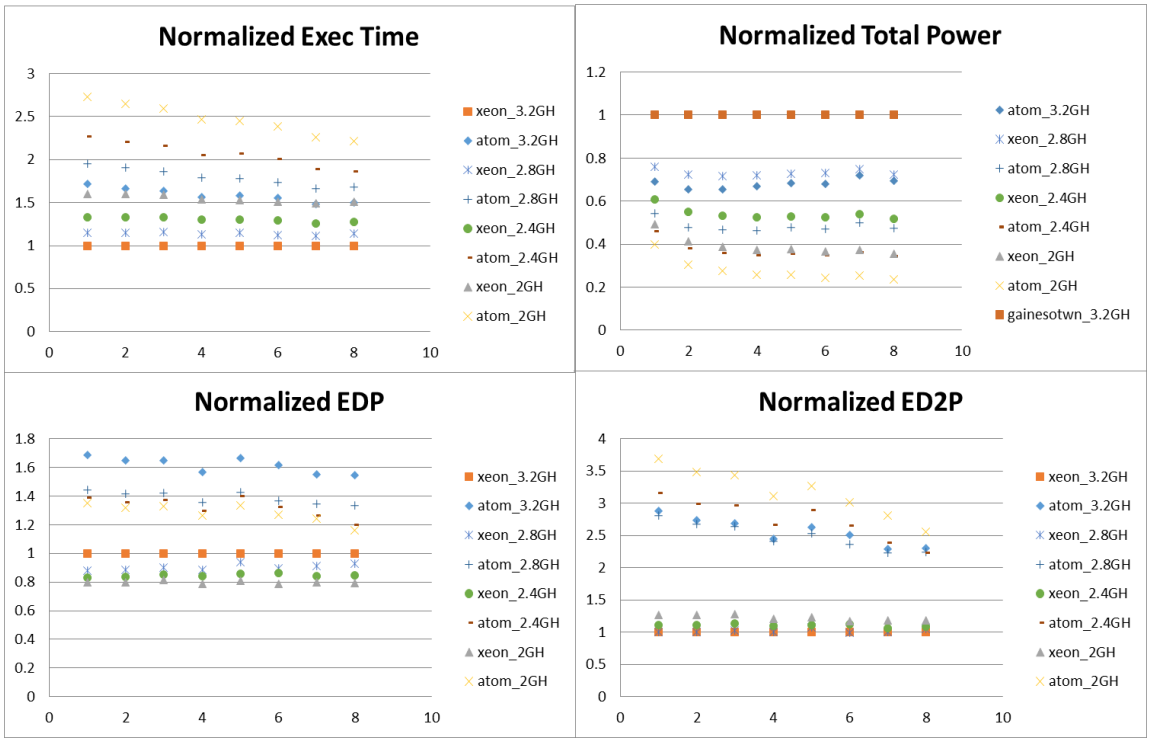

Figure 5 - Results for fmm simulations.

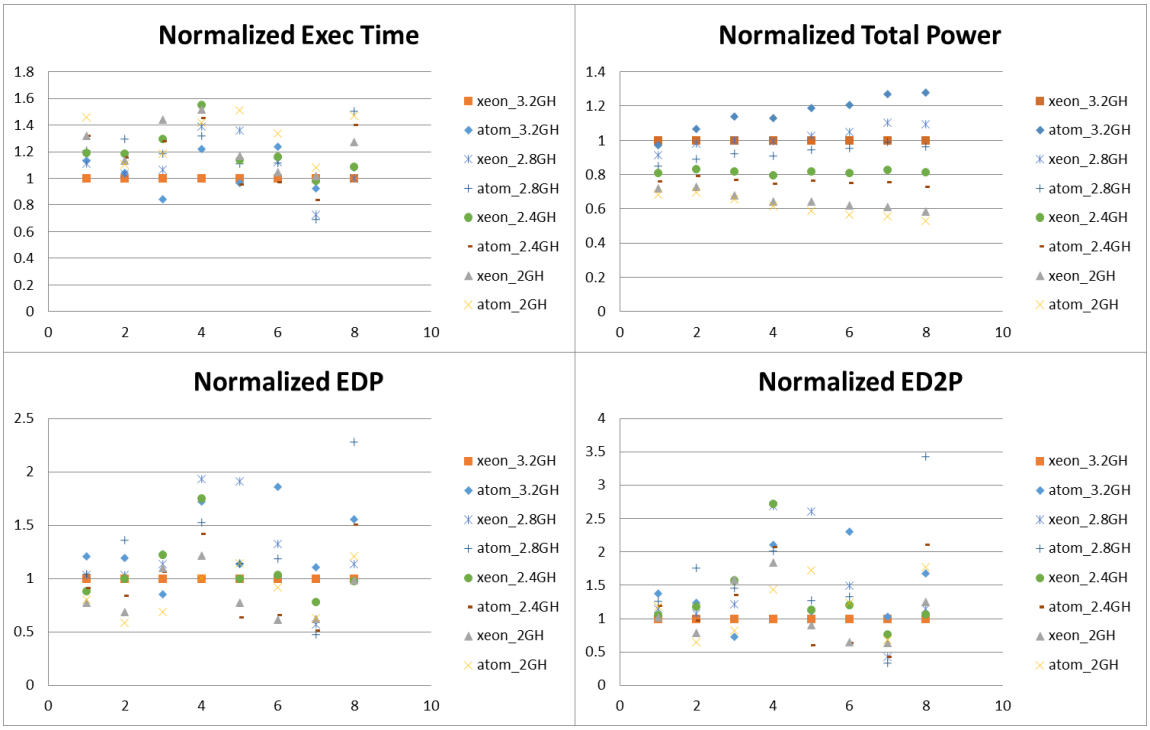

Figure 6 - Results from lu.cont simulations.

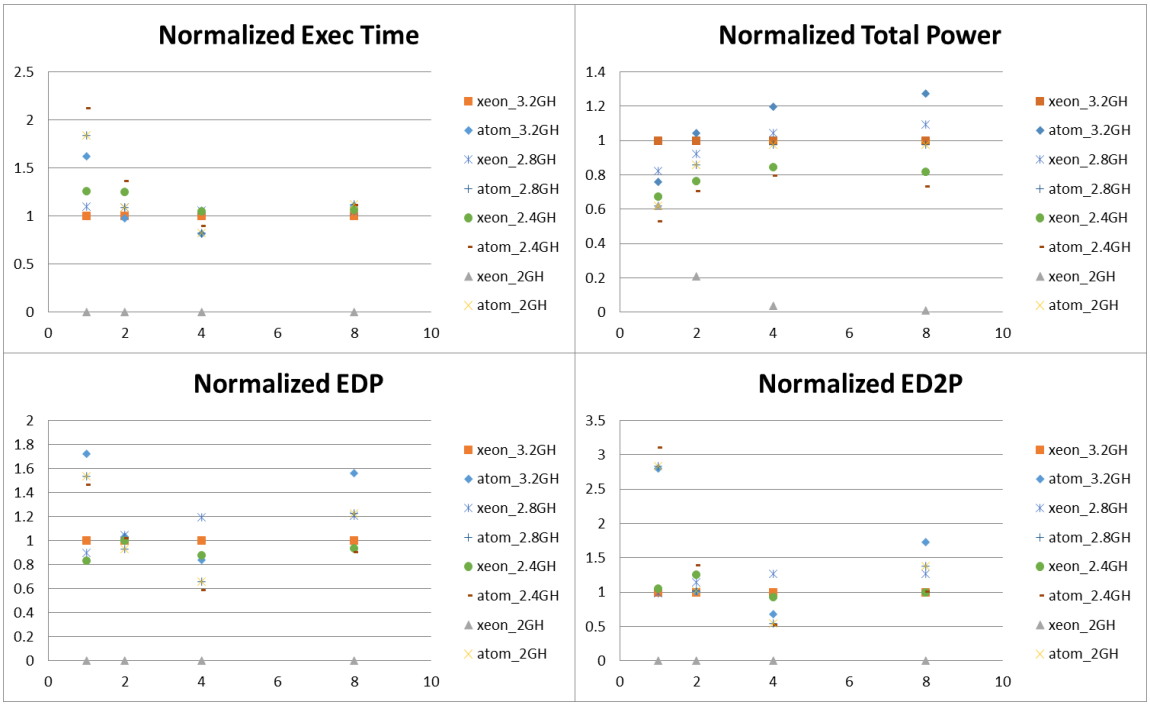

Figure 7 - Results from ocean.cont simulations.

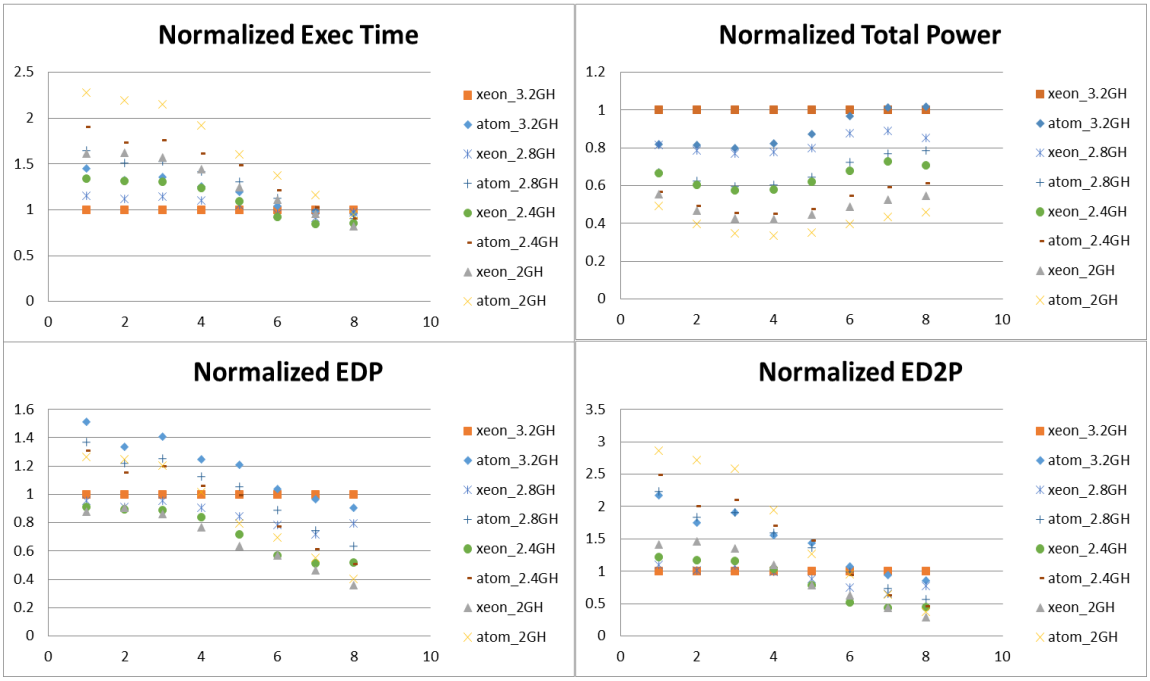

Figure 8 - Results from radiosity simulations.

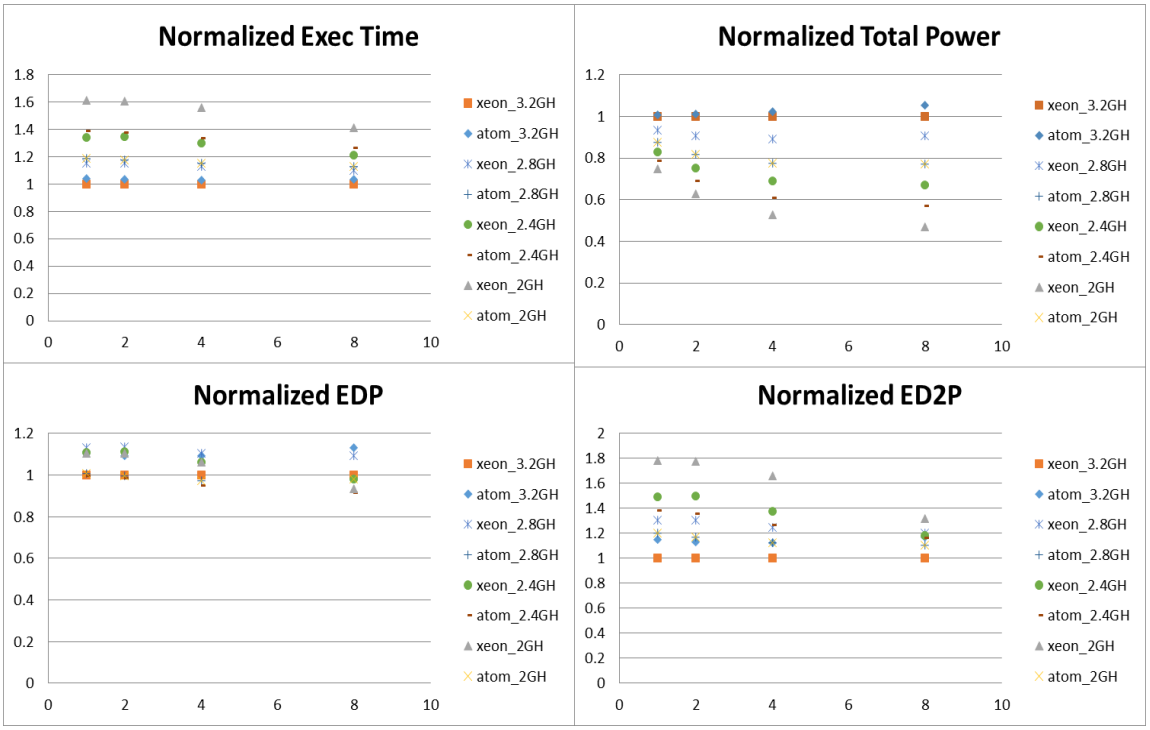

Figure 9 - Results from radix simulations.

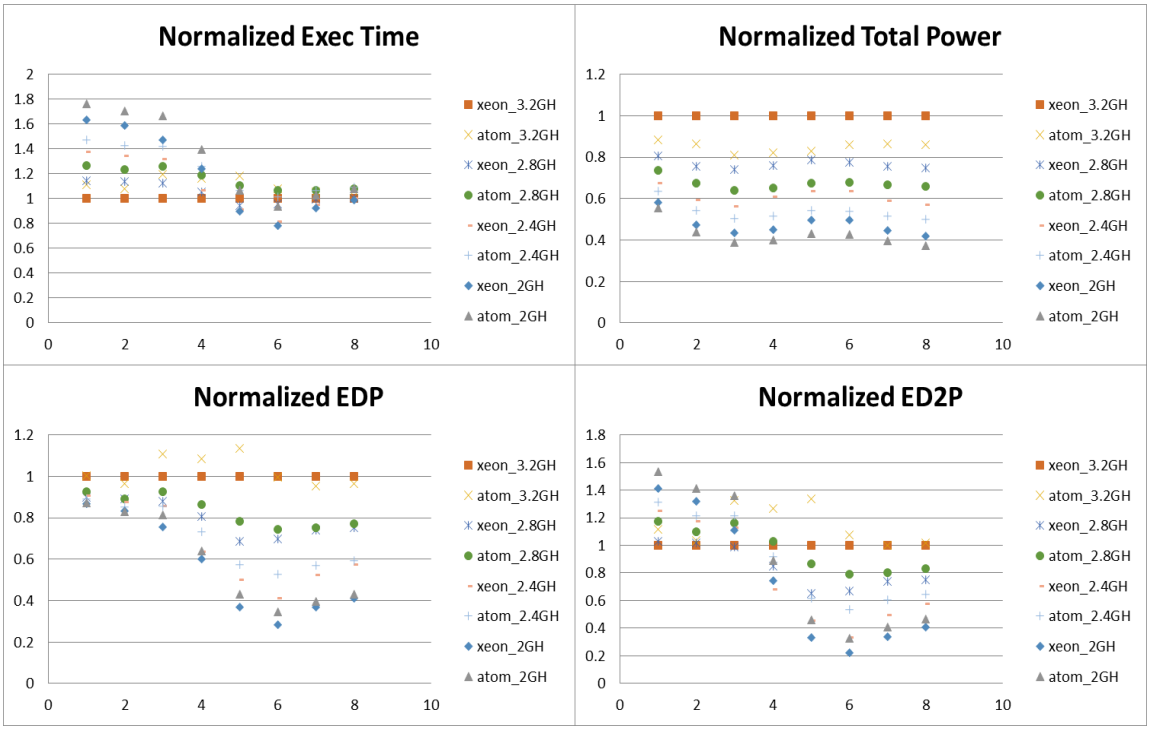

Figure 10 - Results from raytrace simulations.

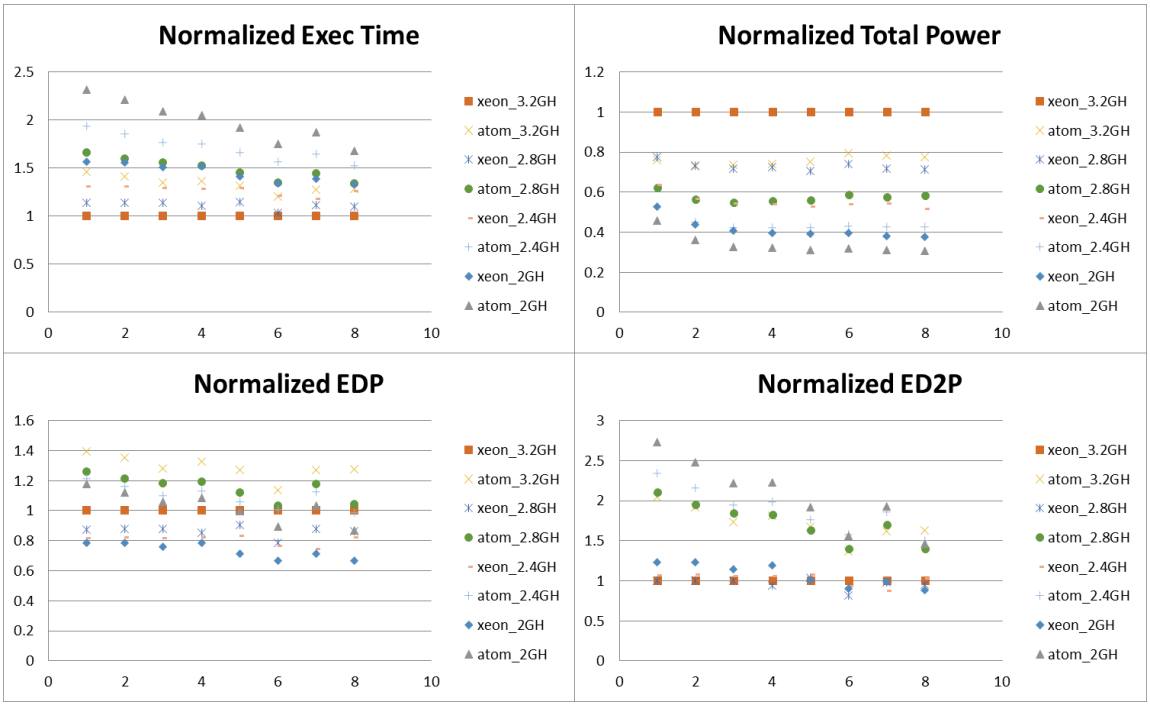

Figure 11 - Results from water.nsq simulations.

7. COMBINED ANALYSIS – INFLUENCE OF FREQUENCY, CORE TYPE, AND THREAD COUNT ON EDP AND ED2P

Section four analyzed the effect of individual parameters on the performance, EDP, and ED2P of the Splash2 benchmarks. The resulting information provided a good starting point and theoretical background for the next step, but did not address the problem on a more comprehensive level. Changing a parameter individually can help tweak performance and power, but the goal of this section was to find an optimum configuration taking into account all of the parameters: core type, frequency, and number of threads.

7.1 Algorithm

In order to find a true optimum configuration all permutations of the parameters were simulated. A simple brute force algorithm seen in Figure 12 was used to check every configuration option and find the minimum EDP and ED2P.

```
for each core in <core_types>
{
    for each frequency in <frequency_set>
    {
        for each threadCount in <thread_counts>
        {
            nextConfig = Configuration(core, frequency, threadCount);
            if (POWER_CONSTRAINT > 0) and (nextConfig.power > POWER_CONSTRAINT) then
                continue;
            if (nextConfig.edp < optimumConfig.edp) then
                optimumConfig = nextConfig;
        }
    }
}
```

Figure 12 – Algorithm for finding optimal core configuration.

7.2 General Results

Though the algorithm above includes a check against a power constraint, the first set of results presented did not perform this check. These results are simply the optimum configurations for maximizing EDP (Table 3) or ED2P (Table 4) with no constraints applied. Each table shows each benchmark, followed by the optimum core configuration parameters (core type, frequency, thread count), and the value of the optimization goal and total power for the corresponding simulation.

It should be noted that all results in sections 7.2 and 7.3 are looking at numbers of threads from 1 to 128. The results are nearly identical, save for the few instances where the number of threads is greater than 8. In each of these cases the result set with only 1 to 8 threads had on optimum configuration that was the same as below, except that the number of threads was 8.

Table 3 – Optimum configurations with optimization target EDP.

Benchmark	Core Type	Frequency	Thread Count	EDP CoV	Power CoV
barnes	xeon	2	32	79.40%	38.13%
cholesky	xeon	2	8	178.06%	64.00%
fft	xeon	3.2	2	208.82%	28.41%
fmm	xeon	2	32	51.62%	28.47%
lu.cont	xeon	2	1	312.82%	31.90%
ocean.cont	xeon	2	1	226.44%	30.47%
radiosity	xeon	2.4	6	138.05%	45.60%
radix	atom	2	8	169.70%	32.82%
raytrace	xeon	2	4	143.66%	30.25%
water.nsq	xeon	2	3	282.68%	26.41%

Table 4 – Optimum configurations with optimization target ED2P.

Benchmark	Core Type	Frequency	Thread Count	ED2P CoV	Power CoV
barnes	xeon	2	32	190.61%	38.13%
cholesky	xeon	3.2	8	193.80%	64.00%
fft	xeon	3.2	2	256.34%	28.41%
fmm	xeon	2.4	32	131.19%	28.47%
lu.cont	xeon	3.2	1	341.03%	31.90%
ocean.cont	xeon	2.8	1	246.28%	30.47%
radiosity	xeon	2.4	6	133.92%	45.60%
radix	xeon	3.2	8	177.30%	32.82%
raytrace	xeon	2.4	4	145.57%	30.25%
water.nsq	xeon	2.8	3	321.86%	26.41%

The figures above show a result that is not entirely surprising. For an optimization target of EDP we see that the vast majority of configurations have a low frequency and a low thread count. EDP gives equal weight to energy efficiency, which is found at lower frequencies and fewer threads. This is similar to current technology

trends, such as the NVIDIA TEGRA 4 and Intel Xeon Phi, both of which have max processor speeds of less than 2GH [39][40].

Looking at an optimization target of ED2P shows slightly different results. Since ED2P puts a higher emphasis on performance over power it is again not surprising that the optimal configuration show higher frequencies and higher thread counts, each of which improves performance.

7.3 Application of Power Constraint

The subsequent results, seen in Tables 5 and 6, did include the check against a power constraint. The constraint varied between 50%, 25%, and 12.5% of the maximum power for a given benchmark. The maximum power used was derived from the simulation result that had the highest total power. In addition to the same values as in section 7.2, each section will show the actual constraint value included in the results.

Table 5 - Optimum configurations with power constraints applied, part 1.

Benchmark	Factor	Constraint	EDP			ED2P		
			Core Type	Frequency	Thread Count	Core Type	Frequency	Thread Count
barnes	100.00%	102.72	xeon	2	32	xeon	2	32
	50.00%	51.36	xeon	2	8	xeon	3.2	8
	25.00%	25.68	Constraint not met.			Constraint not met.		
	12.50%	12.84	Constraint not met.			Constraint not met.		
cholesky	100.00%	240.81	xeon	2	8	xeon	3.2	8
	50.00%	120.41	xeon	2	8	xeon	3.2	8
	25.00%	60.20	xeon	2	8	xeon	2.4	8
	12.50%	30.10	Constraint not met.			Constraint not met.		
fft	100.00%	69.67	xeon	3.2	2	xeon	3.2	2
	50.00%	34.84	atom	2	8	atom	2	8
	25.00%	17.42	Constraint not met.			Constraint not met.		
	12.50%	8.71	Constraint not met.			Constraint not met.		
fmm	100.00%	86.93	xeon	2	32	xeon	2.4	32
	50.00%	43.47	xeon	2	8	xeon	2	8
	25.00%	21.73	Constraint not met.			Constraint not met.		
	12.50%	10.87	Constraint not met.			Constraint not met.		
lu.cont	100.00%	83.67	xeon	2	1	xeon	3.2	1
	50.00%	41.84	xeon	2	1	xeon	3.2	1
	25.00%	20.92	Constraint not met.			Constraint not met.		
	12.50%	10.46	Constraint not met.			Constraint not met.		

Table 6 - Optimum configurations with power constraints applied, part 2.

Benchmark	Factor	Constraint	EDP			ED2P		
			Core Type	Frequency	Thread Count	Core Type	Frequency	Thread Count
ocean.cont	100.00%	75.03	xeon	2	1	xeon	2.8	1
	50.00%	37.52	xeon	2	1	xeon	2.4	1
	25.00%	18.76	Constraint not met.			Constraint not met.		
	12.50%	9.38	Constraint not met.			Constraint not met.		
radiosity	100.00%	141.13	xeon	2.4	6	xeon	2.4	6
	50.00%	70.57	xeon	2.4	6	xeon	2.4	6
	25.00%	35.28	Constraint not met.			Constraint not met.		
	12.50%	17.64	Constraint not met.			Constraint not met.		
radix	100.00%	79.59	atom	2	8	xeon	3.2	8
	50.00%	39.80	atom	2	8	atom	2.8	8
	25.00%	19.90	Constraint not met.			Constraint not met.		
	12.50%	9.95	Constraint not met.			Constraint not met.		
raytrace	100.00%	86.07	xeon	2	4	xeon	2.4	4
	50.00%	43.04	xeon	2	4	xeon	2.4	4
	25.00%	21.52	Constraint not met.			Constraint not met.		
	12.50%	10.76	Constraint not met.			Constraint not met.		
water.nsq	100.00%	76.83	xeon	2	3	xeon	2.8	3
	50.00%	38.42	xeon	2	3	xeon	2	5
	25.00%	19.21	Constraint not met.			Constraint not met.		
	12.50%	9.60	Constraint not met.			Constraint not met.		

When including a power constraint there are a number of instances where the configurations changed in order to meet the applied constraint. In each case we see a more powerful configuration reducing to a lower power core. For instance changing from the larger Xeon core to the smaller Atom option (fft, radix), lowering the frequency (cholesky, fft, fmm, ocean.cont, radix, water.nsq), or lowering thread counts (barnes, fmm).

Some of these configurations showed tradeoffs between parameters. There are cases where the frequency was lowered, but then the thread count increased (fft,

water.nsq). There is even an instance of a large decrease in thread count resulting in an increase in frequency (barnes).

Such variability in optimal configuration is further evidence in favor of a heterogeneous design. Even if core configurations cannot be changed at runtime, it is clear that a single configuration cannot provide the required performance or efficiency for all benchmarks. Even more so, depending on the optimization goal the configuration can change even more.

8. DIVING DEEPER: OPTIMIZING APPLICATION REGION BY REGION

Up to this point this paper has focused on architectures that do not change throughout application processing. While the results are very useful for static homogenous architectures they are only supplementary to the idea of a heterogeneous architecture. It has been previously mentioned that many of the benchmarks being studied include a number of parallel sub-regions. Typically, all of these regions are included in the overall benchmark simulation and they are often separated by serial regions of processing. A static architecture has no option but to process all of these regions in the same manner, though not all regions may have the same profile. Some of these regions may benefit from different architectures than the others.

The ability to change the architecture dynamically to suit specific profiles within an application is the core motivation for this work. Such an architecture would be able to change to best suit the specific profile of each application sub-region. Doing so will result in the optimal set of configurations for the entirety of a benchmark yielding the best performance and efficiency possible. This section presents results of simulations based on the heterogeneous approach.

To illustrate the improvements offered by heterogeneous architectures a set of the Splash2 benchmarks were modified. Simulation markers were placed at various sections of the benchmarks that would be simulated as individual parallel regions. All simulations

were then run again using all permutations of the configuration parameters, voltage/frequency, core type, and thread count. The results from these simulations have been compiled an analyzed and are presented below in Table 7. Each table shows, for the given benchmark, the optimal configuration in terms of first EDP, and second ED2P, for each of the sub-regions listed under the "Marker" heading.

Table 7 - Optimum configurations for individual sub-regions.

cholesky									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	3.2	1	42.43%	atom	2	1	63.81%	118.98%
2	atom	2.4	8	121.93%	atom	2.4	8	24.79%	171.14%
3	xeon	2.8	8	178.36%	xeon	2.8	8	25.50%	226.51%

fft									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	3.2	8	264.55%	atom	3.2	8	25.75%	335.22%
2	xeon	2.8	8	45.44%	atom	2	1	46.28%	75.66%
3	atom	3.2	1	46.80%	atom	2	1	39.22%	85.31%
4	atom	2.8	8	180.77%	atom	2.8	8	34.87%	220.30%
5	atom	2.4	8	189.67%	atom	2.4	8	23.66%	233.79%

fmm									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	xeon	3.2	6	99.26%	xeon	3.2	6	32.33%	157.48%
2	atom	3.2	1	57.47%	atom	2	1	57.01%	148.37%
3	atom	2.4	8	141.64%	atom	2.4	8	29.05%	194.62%

lu.cont									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	2.4	8	134.77%	atom	2.4	8	27.34%	166.62%
2	xeon	3.2	8	116.56%	atom	2	8	23.16%	150.29%
3	xeon	2.4	8	177.58%	xeon	2.4	8	26.30%	244.32%

ocean.cont									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	xeon	2.8	8	197.50%	xeon	2.8	8	24.07%	279.13%
2	xeon	3.2	8	225.18%	xeon	3.2	8	26.14%	302.01%
3	atom	3.2	8	163.04%	atom	3.2	8	26.39%	175.25%
4	xeon	2.8	8	41.93%	xeon	2.8	8	34.39%	54.64%

radiosity									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	3.2	1	60.76%	atom	2	1	42.09%	152.23%
2	xeon	3.2	2	57.93%	xeon	3.2	2	43.49%	109.11%

radix									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	3.2	8	44.56%	atom	2	1	48.87%	86.44%
2	xeon	2.4	8	157.48%	xeon	2.4	8	31.42%	197.52%
3	atom	2.8	8	111.43%	atom	2.8	8	57.32%	139.00%
4	xeon	3.2	8	51.50%	atom	2	1	38.10%	86.87%

water.nsq									
Region	Core	Freq	Threads	EDP CoV	Core	Freq	Threads	ED2P CoV	Power CoV
1	atom	3.2	1	43.65%	atom	2	1	43.49%	116.09%
2	atom	3.2	7	195.24%	xeon	2.4	8	30.66%	253.46%
3	xeon	3.2	8	59.12%	xeon	3.2	8	35.41%	83.42%
4	atom	3.2	1	44.97%	atom	2	1	45.90%	117.58%
5	xeon	3.2	8	194.64%	xeon	3.2	8	25.26%	257.58%

The table above shows much what we expected. It can be easily seen that every marker does not benefit from the same configuration. Here we will look at an example, the radix benchmark, more in-depth. The radix benchmark was instrumented with four separate sub-regions. Over the four sub-regions all four have a different set of configuration options for optimizing EDP, and three different configurations for

optimizing ED2P. The core types vary between large and small, and the frequencies span a wide range.

Note the included metrics for CoV, which are the Coefficients of Variation of the EDP, ED2P, and Power. These show the amount of variability over all of the simulation data for their respective metrics and can be seen as an indication of just how much performance or efficiency may be gained. Though in some cases there is indeed a low CoV, meaning there was not a large difference in the results, for the most part the CoV is quite high. This specifies that not only is a particular configuration optimal, but it is far better than other configurations and the possible benefits are greater.

9. CONCLUSIONS

9.1 Novel Contributions

This paper has first demonstrated a few of the many different parameters that factor in to optimal processor configurations. Using the Splash2 benchmarks we have shown how different applications are sensitive to the core type, voltage/frequency, and thread count parameters. Configuring even just these few parameters yields a wide variance in performance and power efficiency.

Next, analysis of simulations run with all permutations of the previously mentioned parameters was performed. This effort used a simple brute force exhaustive search algorithm to determine what configuration yields the best EDP or ED2P. The results were very much as expected and coincided with metrics from devices in product and on the market today [39-40].

Finally, the true purpose of this paper was to show the benefits of a heterogeneous architecture. By instrumenting Splash2 benchmarks to record statistics on multiple sub-regions of the overall application the full impact of the heterogeneous architecture could be realized. Each of the different sub-regions were simulated with all permutations of configuration parameters which allowed for determination of the optimum parameters for each region individually. The results clearly showed that each region does benefit from a different configuration, in some cases a radically different configuration. Given the

ability to optimize each region at runtime would provide great benefit to the overall application for whatever optimization goal that is trying to be achieved

9.2 Future Work

This work has shown the utility of heterogeneous architectures through profiling a set of Splash2 benchmarks. Even on this sampling the results are extremely promising. To provide a more comprehensive look at the benefit of heterogeneous architectures and to further shows the benefit a wider range of applications could be profiled. Other benchmark suites, such as the PARSEC [12] suite, could be similarly instrumented and simulated.

In this exploration we modified the benchmarks, simulated them fully with a single configuration, and then combined the results from different configurations to show the best combination. The next step in the exploration of heterogeneous architecture is to modify the simulator itself. Custom markers can be added to the benchmarks signifying that the next regions benefits from a specified core configuration. The simulator can be modified to find these markers and, when encountering one, dynamically change the core configuration to the one specified.

This work, and the possible future work mentioned, would provide a very good foundation from which to explore an actual implementation of heterogeneous architecture. Currently we have only been concerned with simulations, finding theoretical results to prove that there would be substantial benefits. With sufficient proof provided, actual implementation can begin.

REFERENCES

[1] A. Lukefahr, S. Padmanabha, R. Das, F.M. Sleiman, R. Dreslinski, T.F. Wenisch, and Scott Mahlke, "Composite Cores: Pushing Heterogeneity Into a Core." In Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO-45), pp. 317-328. 2012. doi=10.1109/MICRO.2012.37

[2] Engin Ipek, Meyrem Kirman, Nevin Kirman, and Jose F. Martinez. "Core Fusion: Accomodating Software Diversity in Chip Multiprocessors." In ISCA 34, June 2007.

[3] Mohammad Khavari Tavana, Mohammad Hajkazemi, Divya Pathak, Ioannis Savidis, Houman Homayoun. "ElasticCore: Enabling Dynamic Heterogeneity with Joint Core and Voltage/Frequency Scaling." *ACM/IEEE 52TH Design Automation Conference.* (DAC 2015).

[4] Pusukuri, K., Gupta, R., and Bhuyan, L. 2011a. "Thread reinforcer: Dynamically determining number of threads via os level monitoring." In *Workload Characterization (IISWC), 2011 IEEE International Symposium on*. IEEE Computer Society, Austin, Texas, USA, 116 –125.

[5] Nagabhushan Chitlur, Ganapati Srinivasa, Scott Hahn, PK Gupta, Dheeraj Reddy, David A Koufaty, Paul Brett, Abirami Prabhakaran, Li Zhao, Nelson Ijih, *et al*., "QuickIA: Exploring heterogeneous architectures on real prototypes."

[7] C. Kim et al. "Composable lightweight processors." In MICRO, 2007.

[8] Y. Watanabe et al. "WiDGET: Wisconsin decoupled grid execution tiles." In ISCA, June 2010.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout. "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations." In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2011.

[10] S. Li, A. J. Ho, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures." In *Proceedings of the Annual*

IEEE/ACM International Symposium on Microarchitecture (MICRO), pages 469-480, 2009.

[11] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *Proceedings of the 22nd International Symposium on Computer Architecture (ISCA '95)*, Santa Margherita Ligure, Italy, Jun. 1995, pp. 24–36.

[12] C. Bienia, S. Kumar, et al. "The parsec benchmark suite: Characterization and architectural implications." Tech. Rep. TR-811-08, Princeton University, 2008

[15] Intel Newsroom. "Intel Launches Low-Power, High-Performance Silvermont Microarchitecture." Web. 6 May 2013. http://newsroom.intel.com/community/intel_newsroom/blog/2013/05/06/intel-launches-low-power-high-performance-silvermont-microarchitecture?wapkw=silvermont

[16] Intel Developer Zone. "Silvermone SoC Uncore Performance Monitoring Guide." Web. 10 April 2014. https://software.intel.com/en-us/articles/SLM-SoC-uncore-performance-monitoring-guide?language=it&wapkw=silvermont

[17] Intel. "Intel Xeon Processor 5500 Series." http://www.intel.com/content/www/us/en/processors/xeon/xeon-5500-brief.html?wapkw=xeon+5500

[18] Intel. "Intel Xeon Processor 5500 Series." http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-5500-brief.pdf

[19] "Enabling Dynamic Heterogeneity Through Core on Core Stacking". (Special Session Talk) Vasileios Kontorinis, Mohammad Khavari Tavana, Mohammad Hajkazemi, Dean Tullsen, Houman Homayoun. ACM/IEEE 51TH Design Automation Conference. (DAC 2014).

[20] "Adaptive Bandwidth Management for Performance-Temperature Trade-offs in Heterogeneous HMC+DDRx Memory". Mohammad Hossein Hajkazemi, Michael Chorney, Reyhaneh Jabbarvand Behrouz, Mohammad Khavari Tavana and Houman Homayoun. 25th ACM International Conference of the Great Lakes Symposium on VLSI, 2015.

[21] "Heterogeneous Memory Management for 3D-DRAM and External DRAM with QoS" Le-Nguyen Tran, Houman Homayoun, Fadi Kurdahi, Ahmed Eltawil 18th Asia and South Pacific Design Automation Conference.

[22] “Dynamically heterogeneous cores through 3D resource pooling” Houman Homayoun, Vasileios Kontorinis, Ta-Wei Lin, Amirali Shayan and Dean M. Tullsen. International Symposium on High-Performance Computer Architecture, HPCA 2012. New Orleans, Louisiana.

[23] R. Kumar, K.I. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen, “Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction,” in Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36), Dec. 2003.

[24] B. Urgaonkar, P. Shenoy, and T. Roscoe. “Resource overbooking and application profiling in shared hosting platforms.” In *Proceedings of the Fifth symposium on operating systems design and implementation (OSDI)*, pages 239 – 254, December 2002.

[25] L. Thiele, I. Bacivarov, W. Haid, and K. Huang. “Mapping Applications to Tiled Multiprocessor Embedded Systems.” In *ACSD 07*, pages 29–40, 2007.

[26] A. A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. L. Wang, “Heterogeneous computing: Challenges and opportunities,” *IEEE Comput.* 26, 6 (June 1993), 18 - 27.

[27] A. Ghafoor and J. Yang, “Distributed heterogeneous supercomputing management system,” *IEEE Comput.* 26, 6 (June 1993), 78-86.

[28] Kim, J. M., Seo, S. K., Chung, S. W. 2014. “Looking into heterogeneity: when simple is faster.” In *The 2nd International Workshop on Parallelism in Mobile Platforms.*

[29] Adarsh Reddy Ashammagari, Hamid Mahmoodi, Tinoosh Mohsenin, Houman Homayoun. "Reconfigurable STT-NV LUT-based functional units to improve performance in general-purpose processors." *Proceedings of the 24th edition of the great lakes symposium on VLSI.* ACM, 2014.

[30] Vasileios Kontorinis, Mohammad Tavana, Mohammad Hajkazemi, Dean Tullsen, Houman Homayoun. "Enabling dynamic heterogeneity through core-on-core stacking." *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE.* IEEE, 2014.

[31] Mark D Hill and Michael R Marty, “Amdahl’s law in the multicore era,” *Computer*, Vol. 41, No. 7, pp. 33–38, 2008.

[32] Houman Homayoun. “Heterogeneous architecture to address energy-efficiency crisis”, Proposal submitted to National Science Foundation, 2015.

[33] George Mason University, Office of Research Computing. "ARGO – Hardware Specs." Web. 1 June 2015.

[34] Trevor Carlson, Wim Heirman. "The Sniper User Manual." 13 November 2013.

[35] Abbas Rahimi Tinoosh Mohsenin M. Khavari Tavana, Amey Kulkarni and Houman Homayoun. "Energy-efficient mapping of biomedical applications on domain-specific accelerator under process variation," *The International Symposium on Low Power Electronics and Design, ISLPED.* IEEE, 2014.

[36] Andreas Schranzhofer, Jian-Jian Chen, and Lothar Thiele, "Dynamic power-aware mapping of applications onto heterogeneous mpsoc platforms," *Industrial Informatics, IEEE Transactions on,* Vol. 6, No. 4, pp. 692–707, 2010.

[37] Ewerson Carvalho, Ney Calazans, and Fernando Moraes, "Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs," *Rapid System Prototyping, 2007.* RSP 2007. 18th IEEE/IFIP International Workshop on. IEEE, pp. 34–40, 2007.

[38] Esmaeilzadeh, Hadi, et al. "Dark silicon and the end of multicore scaling." *Computer Architecture (ISCA), 2011 38th Annual International Symposium on.* IEEE, 2011.

[39] Intel. "Intel Xeon Phi Coprocessor 5110P." http://ark.intel.com/products/71992/Intel-Xeon-Phi-Coprocessor-5110P-8GB-1_053-GHz-60-core

[40] NVIDIA. "Tegra 4." http://www.nvidia.com/object/tegra-4-processor.html

BIOGRAPHY

Matthew Drummond is a candidate for Master of Science in Electrical and Computer Engineering, with a specialization in Microprocessors and Embedded Systems, from George Mason University. He received his Bachelor of Science from the University of Notre Dame, Notre Dame, Indiana, in 2011. At Notre Dame he worked as a research student in the field of biometric software as well as working as a Software Engineer for the United States Air Force. Since graduation from Notre Dame he has been employed as a Software Engineer for the Boeing Company in Springfield, Virginia.