Amdahl's Law in the Era of Process Variation

Gayathri Ananthanarayanan

School of Information Technology

Geetika Malhotra

Computer Science and Engineering

M. Balakrishnan

Computer Science and Engineering

Smruti R. Sarangi

Computer Science and Engineering
Indian Institute of Technology
Hauz Khas, New Delhi, India
E-mail: {gayathri,mcs122798,mbala,srsarangi}@cse.iitd.ac.in

Abstract: In this paper we propose to extend Amdahl's law for modelling multicores with process variation using simple mathematical techniques. We consider three major families of multicore processors – symmetric, asymmetric, and dynamic. We consider a conservative operating mode for setting the target frequency (*plain*), and a more optimised method (*opt*). Subsequently, we propose three separate corollaries to the standard Amdahl's Law to model the performance of different multicore configurations with different modes of operation. We observe that most of the major trends published in prior work or the ones that we observe through Monte Carlo simulations can be explained by a simple hypothetical concept called an *equivalent core*. The crux of our approach is to look at a set of equivalent cores with no variation that have the same performance as the target system. It is much simpler and much more intuitive to reason in terms of equivalent cores. Along with being an effective analytical tool, it can be used to speed up a lot of heuristics, and can be exposed to higher level software for advanced scheduling decisions. Lastly, we validate our models with experiments on a real system, and the maximum error is limited to 8%.

Keywords: Process variation, Amdahl's law, Equivalent core

Biographical notes: Gayathri Ananthanarayanan is a Ph.D student in the School of Information Technology, IIT Delhi. Geetika Malhotra is a Master's student in the Computer Science and Engineering Department, IIT Delhi
Prof. M. Balakrishnan is a Professor in the computer science and engineering department at IIT Delhi. His primary interests are in embedded systems and assistive devices. He is a member of the IEEE and ACM.
Prof. Smruti R. Sarangi has received a Ph.D in computer architecture from the University of Illinois. He has subsequently worked in Synopsys and IBM Research Labs. Currently, he is an Assistant Professor in the Computer Science Department at IIT Delhi. His primary interests are in computer architecture, and parallel algorithms. He is a member of both the IEEE and ACM.

1 Introduction

It is getting increasingly difficult to accurately fabricate transistors in the nanometer era. Because of the inherent limits introduced by manufacturability and the underlying device physics, designers are finding it difficult to scale the resolution limit of their processes to a couple of nanometers. Consequently, it is not possible to accurately guarantee the dimensions of transistors. This leads to variability in a transistor's power consumption and switching frequency. This phenomenon is known as *process variation* (refer to Borkar et al. (2003)). There are two direct consequences of such limitations in the context of traditional conservative designs namely increased power consumption and reduced frequency (Sarangi et al. (2008b)). Consequently, it is not possible to sustain traditional gains in performance when we consider the effect of process variation.

In this paper, we take a look at a seminal law in computer architecture namely Amdahl's Law, in the

context of process variation. In 1967, Gene Amdahl proposed a famous law (Amdahl (1967)) named after him, which relates the speedup to the amount of available parallelism and the number of processors. Amdahl's law gives an upper bound on the expected speedup; however, it totally ignores some aspects of multiprocessors such as interconnect bandwidth and contention. Nevertheless, it has stood the test of time, and can give designers and researchers a broad estimate of the range of the expected speedup. It has proved its worth as an analytical tool, as well as a generalised predictive indicator for estimating trends in computer architecture.

We observe in this paper that this seminal law needs to be extended when we consider process variation. The law implicitly assumes homogeneous cores. We believe that it is rather difficult to succinctly model the phenomenon of variable performance per core. Consequently, we try to model the variable performance per core by effectively expanding or contracting the set of cores. We try to mathematically map a world with process variation to a world without it. For example, our model will help us make conclusions of the form – a 128 core machine with a 6% variation in the threshold voltage has similar performance as a 100 core machine with no variation.

We complicate this line of reasoning further. We observe that due to process variation, the transistor switching time varies across the length and breadth of a die. In some areas it might be lower than the nominal value, whereas in some other areas it might be higher than the nominal value. Since we need to ensure that a design is free of timing faults, a reduction in the switching frequency due to variation will require us to reduce the frequency. However, it is possible in future processors to run the sequential part of a parallel program in the faster areas of a die, and effectively speed it up by running it at a faster frequency as suggested by Miller et al. (2012). In this case, variation is beneficial for the sequential part, whereas it is detrimental for the parallel part. Our model explicitly takes this optimisation into account. Secondly, prior work on extending Amdahl's law by Woo and Lee (2008); Hill and Marty (2008) has looked at different kinds of processors – symmetric, asymmetric, and dynamic. In this paper, we also propose corollaries to the general law for different processor configurations.

We provide a background of the problem in Section 2. Subsequently, we propose our performance model, and extensions to Amdahl's law in Section 3, and show some experimental results in Section 4. Lastly, we survey related work in Section 5 and conclude in Section 6.

2 Background

2.1 Amdahl's Law

Let the serial fraction of a parallel program be $1-p$ and the parallel fraction be p. The speedup, S, is given by Equation 1. This is the classical form of the Amdahl's Law.

$$S = \frac{1}{1-p+\frac{p}{n}} \tag{1}$$

We first propose a generalised version of Amdahl's law. Let us assume that the relative performance of the sequential core is given by η_{seq} and the relative performance of each parallel core is given by η_{par}. We then arrive at Equation 2.

$$S = \frac{1}{\frac{1-p}{\eta_{seq}}+\frac{p}{n\times\eta_{par}}} \tag{2}$$

2.1.1 Corollaries of the Amdahl's Law

Here, we list three corollaries proposed by Hill and Marty for several futuristic processors (Hill and Marty (2008)). They start out by defining the term *base core equivalent(BCE)*, which refers to the die area occupied by one core. They consider a setup with n simple cores (1 core = 1 BCE). Let $Perf(r)$ be the performance of a core consisting of r BCEs relative to a core containing 1 BCE.

In the future, we will have chips with possibly hundreds of cores. It might not be possible to achieve large scale parallelism for most benchmarks. Consequently, it might make sense to replace a group of r cores with one larger core at the design stage itself. Hill and Marty call this a *symmetric* configuration. Here one core consists of r base core equivalents, and the resultant speedup is given by Equation 3. The authors conclude that for programs with a high sequential component, we need to choose relatively higher values for r.

$$S_{symmetric}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r)}+\frac{pr}{Perf(r)n}} \tag{3}$$

It is not necessary that all the cores have to be of the same size. We can envision an asymmetric architecture (Aater S. et al. (2010); Kumar et al. (2004)), which has one large core and several smaller cores. We can use the large core for speeding up critical sections (Aater S. et al. (2010)). If the size of the large core is equal to r BCEs (base core equivalents), then the speedup is given by Equation 4.

$$S_{asymmetric}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r)}+\frac{p}{Perf(r)+n-r}} \tag{4}$$

Lastly, we need to consider the case of dynamic multicore architectures (Gupta et al. (2010); Ipek et al. (2007)), where r cores can be dynamically fused to form a large sequential core, or can also work separately

as independent parallel cores. There have been a spate of recent proposals, which advocate this idea of reconfigurability at the level of a core. In this case, the sequential core has performance equal to $Perf(r)$, and each parallel core has a normalised performance of 1. The speedup is given by Equation 5.

$$S_{dynamic}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r)} + \frac{p}{n}} \tag{5}$$

2.2 Process Variation

In this section, we explain the basics of process variation, and show how it can be modelled mathematically. We will explicitly use this mathematical model in Section 4 to create test cases.

2.2.1 Model

Process Variation is defined as the variability in transistor dimensions and electrical characteristics due to the inherent limits of manufacturability. These parameters are typically, the threshold voltage(V_{th}) and the channel length (L_{eff}). Both these parameters are related to each other by a simple equation proposed by Sarangi et al. (2008a). L^0_{eff} and V^0_{th} refer to the nominal values.

$$L_{eff} = L^0_{eff}\left(1 + \frac{V_{th} - V^0_{th}}{2V^0_{th}}\right) \tag{6}$$

Consequently, prior work has only considered the variation in the threshold voltage, V_{th}.

There are two components of the threshold voltage variation – systematic and random. *Systematic* variation can happen because of lithographic aberrations or due to optical diffraction. This type of variation is typically modelled as a multivariate normal distribution. There is a good amount of spatial correlation between values of V_{th} in regions with close proximity. This correlation is modelled as an approximately linear function of the distance between two points, and it is mostly independent of the direction(isotropic).

Along with the systematic component, there is a *random* component of variation. This is caused by fluctuations in dopant density and due to the jagged nature of lines on silicon because of physical processes such as etching. This is modelled as a normal distribution at the gate level. Random variation is not known to show any measurable amount of spatial correlation.

We can summarise the effect of both the components using Equation 7.

$$V_{th} = V^0_{th} + \Delta V_{th-sys} + \Delta V_{th-rnd} \tag{7}$$

2.2.2 Impact on Clock Frequency

In Section 2.2.1 we presented a model to compute the spatial distribution of the threshold voltage, V_{th}. However, to make the model useful to our analyses, we need a method to convert a spatial map of V_{th} to a spatial map representing the switching frequencies of transistors. We can then use this map to find the peak frequency of different cores in a large multicore die.

We start out by computing the access time of a typical logic gate(T_g) as given by the Alpha power law (refer to Sarangi et al. (2008a)). Here, V is the supply voltage, and α is a constant (1.3).

$$T_g \propto \frac{V(1 + V_{th}/V^0_{th})}{(V - V_{th})^\alpha} \tag{8}$$

Given, the fact that V_{th}/V is around 15%, we can simplify this equation using Taylor expansion (see Equation 9).

$$T_g \propto V^{1-\alpha}\left(1 + \frac{V_{th}}{V^0_{th}} + \frac{\alpha V_{th}}{V}\right) \tag{9}$$

Using results from Equation 7, we can split the gate delay into three components.

$$\begin{aligned} T_g &\propto T_0 + \underbrace{V^{1-\alpha}\left(\frac{\Delta V_{th-sys}}{V^0_{th}} + \frac{\alpha \Delta V_{th-sys}}{V}\right)}_{T_{sys}} \\ &\quad + \underbrace{V^{1-\alpha}\left(\frac{\Delta V_{th-rnd}}{V^0_{th}} + \frac{\alpha \Delta V_{th-rnd}}{V}\right)}_{T_{rnd}} \\ &\propto T_0 + T_{sys} + T_{rnd} \end{aligned} \tag{10}$$

T_0 is the nominal value of the gate delay. T_{sys} and T_{rnd} are the systematic and random components of the gate delay respectively. Since ΔV_{th-sys} and ΔV_{th-rnd} are normally distributed with zero mean, T_{sys} and T_{rnd} are also normally distributed. Like prior work (Sarangi et al. (2008a,b)), we assume that the systematic component of T_g within a pipeline stage is constant. This is due to the high degree of spatial correlation. Across stages it is distributed as per the multivariate normal distribution described in Section 2.2.1.

Now let us assume that a pipeline stage consists of κ FO4 delays (delay of a representative transistor having a fanout of four). In this case, the delay of a pipeline stage, T_{stage}, is given by Equation 11.

$$\begin{aligned} T_{stage} &= \sum_{i=0}^{\kappa} T^i_g = \kappa \times T_0 + \kappa \times T_{sys} + \sum_{i=0}^{\kappa} T^i_{rnd} \\ &= \kappa \times T_0 + \kappa \times T_{sys} + T^\kappa_{rnd} \end{aligned} \tag{11}$$

Here, T^κ_{rnd} is the sum of κ instances of the random variable T_{rnd}. It has zero mean and its variance is κ times the variance of T_{rnd}. Since both T_{sys} and T^κ_{rnd} are distributed normally, T_{stage} also has a normal distribution. We can finally compute f_{stage} as $1/T_{stage}$. Using some simplifying assumptions and Taylor's expansion, it is possible to prove that f_{stage} is also normally distributed with reasonable error bounds. However, this is beyond the scope of the paper.

We are thus in a position to generate samples of the random variable T_{stage} and hence compute the distribution of core frequencies across a large multicore die. We will use this result to generate samples for evaluation in Section 4.

In this paper, we do not consider advanced techniques to mitigate variation such as body biasing and timing speculation (Sarangi et al. (2008b)) because of the resultant complexity and for lack of brevity. However, it is possible to incorporate their effects by considering an effective σ/μ of V_{th} variation after these techniques are applied.

3 Variation Aware Amdahl's Law

3.1 Performance Model

Let us denote the relative performance of a processor consisting of r base core equivalents, and running at frequency, f, as $Perf(r, f)$. Note that the performance is relative to 1 BCE running at the nominal frequency. In some cases, we use the term $Perf(r)$, which refers to the relative performance at the nominal frequency. We use two important empirical rules and two basic results from textbooks in elementary computer architecture in our analyses. They have been used by prior work in extending Amdahl's Law (see Hill and Marty (2008)).

The first of the rules is Pollack's rule, which states that performance is proportional to the square root of die area (refer to Hill and Marty (2008)) (see Equation 12). Here, α_1 is the constant of proportionality.

The second is a rule regarding cache miss rates (Hartstein et al. (2006); Gluhovsky and O'Krafka (2005)). It says that the miss rate, m_r, of a cache is approximately inversely proportional to the square root of the size of the cache. We further assume that the size of the cache is proportional to the area of the processor. We thus arrive at Equation 13. α_2 is another constant of proportionality.

Now, we look at two more basic results in architecture. Equation 14 shows a simple relationship between performance and IPC (instructions per cycle). It says that the performance is IPC multiplied by the clock frequency.

Lastly, in Equation 15, we show a relation between CPI (clock cycles per instruction), the miss rate (m_r), and the miss penalty to main memory, m_p. Note that m_p is the miss penalty at the nominal frequency, f_0. It is measured in terms of cycles. Since the miss penalty to main memory is independent of the frequency of the cpu, m_p needs to be scaled for a different frequency by a factor equal to $\frac{f}{f_0}$.

Secondly, note that α_1, α_2, m_p, m_r, and CPI_{ideal} are constants.

$$Perf(r, f_0) = \alpha_1 \sqrt{r} \tag{12}$$

$$m_r = \frac{\alpha_2}{\sqrt{r}} \tag{13}$$

$$Perf(r, f) = IPC(r, f) \times f \tag{14}$$

$$CPI(r, f) = CPI_{ideal} + m_p \times \frac{f}{f_0} \times m_r \tag{15}$$

Using these four equations, we derive an equation for the relative performance $Perf_{rel}(r, f)$ given by Equation 16. In this case, our baseline system runs at the nominal frequency and has a size equal to r BCEs. Let us define $Perf_{rel}(r, f) = Perf(r, f)/Perf(r, f_0)$. From Equation 12, we deduce that $Perf(r, f_0) = \alpha_1\sqrt{r}$. Let us further define f_{rel} as f/f_0, and let us consider that $\alpha_1 = \alpha_3 \times f_0$. We have:

$$\begin{aligned}
&Perf_{rel}(r, f) \\
&= Perf(r, f)/Perf(r, f_0) \\
&= \frac{f}{CPI(r, f) \times \alpha_1\sqrt{r}} \\
&= \frac{f}{(CPI_{ideal} + m_p f_{rel} m_r) \times \alpha_1\sqrt{r}} \\
&= \frac{f}{(CPI(r, f_0) - m_p m_r + m_p f_{rel} m_r) \times \alpha_1\sqrt{r}} \quad (Eqn\ 15) \\
&= \frac{f}{(\frac{f_0}{Perf(r,f_0)} + m_p m_r (f_{rel} - 1)) \times \alpha_1\sqrt{r}} \\
&= \frac{f}{(\frac{f_0}{\alpha_1\sqrt{r}} + m_p m_r (f_{rel} - 1)) \times \alpha_1\sqrt{r}} \\
&= \frac{f}{f_0 + m_p m_r (f_{rel} - 1) \times \alpha_1\sqrt{r}} \\
&= \frac{f}{f_0 + m_p (f_{rel} - 1) \frac{\alpha_2}{\sqrt{r}} \times \alpha_1\sqrt{r}} \quad (by\ Equation\ 13) \\
&= \frac{f}{f_0 + m_p (\alpha_3 f_0) \alpha_2 (f_{rel} - 1)} \\
&= \frac{f_{rel}}{1 + m_p \alpha_3 \alpha_2 (f_{rel} - 1)}
\end{aligned} \tag{16}$$

Hence, we can conclude that the relative performance $Perf_{rel}$ is a function of the relative frequency (f_{rel}), the miss penalty (m_p), and two constants of proportionality – α_2 and α_3.

3.2 Amdahl's Laws

When we consider a processor with process variation, the frequency will vary randomly across the entire die. To maximise performance, we need to maximise frequency. Let us first look at parallel cores. If we set the frequency of one parallel core to 3 GHz and the other to 4 GHz, then the entire process will still effectively run at 3 GHz. This is because most parallel programs try to equally divide the work among the threads. If one thread is running faster, then it will reach a synchronisation point like a barrier sooner, and wait for the slower threads. Consequently, to save power, it makes sense to run all the parallel cores at the same frequency. This should be the maximum frequency that the slowest core can support without suffering from timing faults. We can denote this

frequency as f_{min}. If a parallel core consists of multiple BCEs, then we choose the minimum frequency that each BCE can support.

Handling the sequential part of the execution is slightly more tricky. The first or rather default option is that we stick with f_{min}. We call this the *plain* configuration. However, this is suboptimal. Secondly, future processors are expected to have sophisticated voltage-frequency scaling capabilities. Hence, we can have a smarter scheme in which we run the sequential portion of a program on the fastest set of cores as suggested by Miller et al. (2012). During the execution of the sequential part, we can increase the frequency of the sequential core to the maximum that it can support. If the sequential core contains r BCEs, then we can run it at frequency, f_r. Here, f_r is the minimum frequency of the r constituent BCEs. We call this the *opt* configuration.

We now discuss the procedure to derive the corollaries to the Amdahl's Law for different types of multiprocessors for the *opt* configuration. To get the equations for the *plain* configuration, we just need to replace all instances of f_r by f_{min}. We omit this case for the sake of brevity. Let us now look at the three configurations considered in Section 2.1.1 and by Hill and Marty (2008) – symmetric, asymmetric, dynamic. For the symmetric case, we can choose to run the sequential part of the execution(critical section) on the fastest core. The fastest core can be determined after a chip has been fabricated, and tested. Likewise for dynamic multicores, we can choose to combine the r cores in such a way that f_r is maximised. Given the systematic nature of variation, we expect the fastest r BCEs to be possibly co-located. Unfortunately, it is hard to do such optimisations for the case of an asymmetric multicore. This is because it is not possible to know in advance, which region of a chip will be the fastest. However, in some cases especially if the prime cause of systematic aberration is defects in the imaging process, it might be possible to predict the faster regions of a die by analysing the pattern of variation across different dies. In this case, designers can appropriately modify the design and place the asymmetric core such that its frequency is maximised.

We can thus modify the corollaries to the Amdahl's Law presented in Section 2.1.1.

$$S_{symmetric}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r,f_r)} + \frac{pr}{n \times Perf(r,f_{min})}} \quad (17)$$

$$S_{asymmetric}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r,f_r)} + \frac{p}{Perf(r,f_{min}) + (n-r) \times Perf(1,f_{min})}} \quad (18)$$

$$S_{dynamic}(p,n,r) = \frac{1}{\frac{1-p}{Perf(r,f_r)} + \frac{p}{n \times Perf(1,f_{min})}} \quad (19)$$

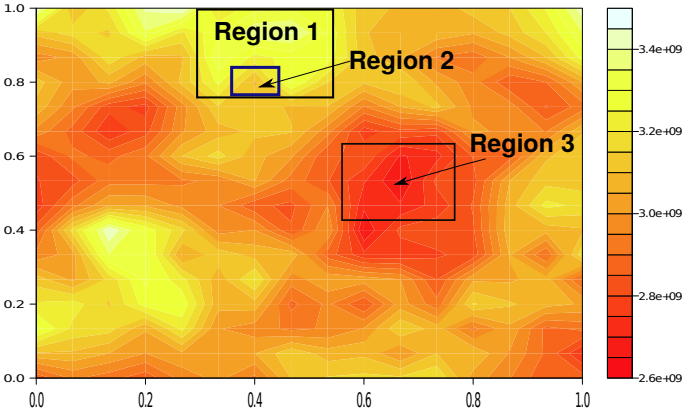

Figure 1 Sample frequency variation map

3.2.1 Approximating $Perf(r, f_r)$, $Perf(r, f_{min})$

We observe that $Perf(r, f_r) = Perf_{rel}(r, f_r) \times Perf(r, f_0)$. Here, $Perf_{rel}(r, f_r)$ is dependent on the variation map of a die. Figure 1 shows the variation of peak frequency on a sample die. For the sequential portion of the execution, we can choose a core in Region 1. Note that the transistors in *Region 1* are the fastest in the entire die. The physical core represented by *Region 1* will run at the slowest peak frequency of any point in that region. This is shown as *Region 2* in the figure. This frequency is denoted by f_r in Section 3.2. Note that f_r is a function of the number of total BCEs, size of each BCE, and the variation map. We define $\mathcal{X}(r,n)$ as:

$$\begin{aligned} \mathcal{X}(r,n) &= Perf_{rel}(r, f_r) \\ &= Perf(r, f_r)/Perf(r, f_0) \end{aligned} \quad (20)$$

We shall see in Section 4.2 that $\mathcal{X}(r,n)$ is approximately linear in r for a wide set of variation maps and constants used by our model. We are in the process of investigating the reasons for this behavior.

Likewise, we can define another function $\mathcal{Y}$ for $Perf(r, f_{min})$.

$$\begin{aligned} \mathcal{Y}(r,n) &= Perf_{rel}(r, f_{min}) \\ &= Perf(r, f_{min})/Perf(r, f_0) \end{aligned} \quad (21)$$

f_{min} represents the minimum frequency. In other words, f_{min} is the smallest peak frequency across all the regions of the die. The slowest region in the die is *Region 3* as shown in Figure 1. The value of $\mathcal{Y}$ is mostly independent of both n and r as observed empirically. Hence, we can approximate $\mathcal{Y}(r,n)$ as just a constant $\mathcal{Y}$, which is dependent only on the frequency variation map.

3.3 Equivalent Configurations

We observe:

$$\begin{aligned} Perf(r, f_r) &= \mathcal{X}(r,n) \times Perf(r, f_0) \\ &= \mathcal{X}(r,n) \times \alpha_1 \sqrt(r) \qquad (by\ Equation\ 12) \\ &= \alpha_1 \sqrt{r\mathcal{X}(r,n)^2} \\ &= Perf(r\mathcal{X}(r,n)^2, f_0) \qquad (by\ Equation\ 12) \end{aligned} \tag{22}$$

Similarly, we have:

$$Perf(r, f_{min}) = Perf(r\mathcal{Y}^2, f_0) \tag{23}$$

Let us now try to map a world with variation to a world without it. We wish to replace a processor with variation by another processor, which does not suffer from process variation and has the same speedup. Let $config$ denote any configuration (symmetric or asymmetric or dynamic) with variation. Let $config'$ denote the same configuration without variation. We want a mapping that satisfies Equation 24. S represents speedup.

$$S_{config}(p, n, r) = S_{config'}(p, n', r') \tag{24}$$

With some algebraic manipulations we can derive the values of n' and r' as shown in Table 1. We show the derivation for the symmetric *opt* case in Equation 25. The rest of the entries in Table 1 follow the same pattern.

With this mapping, we have thus arrived at a variant of Amdahl's Law without the effects of process variation. None of the results are dependent on the parallel fraction – p. We can derive some quick insights that can serve as a sanity check.

Insight 1:
All three *opt* configurations have a larger number of sequential BCEs as compared to *plain*. We thus expect a good degree of sequential acceleration in the *opt* configuration. Whereas, in the *plain* configuration, the number of sequential BCEs are lesser than r. We thus expect the sequential portion to get slowed down.

Insight 2:
As compared to the no-variation case, we expect slowdowns in *plain* because of the reduction in the total BCEs, n' ($\because \mathcal{Y} < 1$).

There are several ways to interpret the results in Table 1 and Equation 24.

Interpretation 1: (Area)
We can interpret the results as a contraction or expansion in area and the total number of cores. For example, in the symmetric *plain* case, we can assume that every core with variation shrinks by a factor of $\mathcal{Y}^2$ and the total number of cores stays the same. For the dynamic *opt* case, we can assume that every core increases in area by a factor of $\mathcal{X}(r,n)^2$, and the total number of cores is equal to $n\mathcal{Y}/\mathcal{X}(r,n)^2$.

Interpretation 2: (Configuration)
We can assume that the area remains the same; although, the configuration in terms of n and r changes. For example, in the case of dynamic *plain* , r gets multiplied by a factor of $\mathcal{Y}^2$, and n gets multiplied by a factor of $\mathcal{Y}$. This formulation can be utilised to explain speedups as shown in Section 4.3.

4 Evaluation

4.1 Setup

We simulate a core with typical values of IPC, L2 miss rates, and memory latency. We use an ideal IPC of 1, memory latency of 200 cycles, and 0.25 percent global L2 miss rate, and a nominal frequency of 3 GHz. Using these values, we get a value of $m_p\alpha_2\alpha_3$ in Equation 16 as $\frac{1}{3}$.

We generated sample variation maps in R using the geoR package as described by Sarangi et al. (2008a) and superimposed an example layout on it using the approach mentioned in Sarangi et al. (2008b). A sample variation map is shown in Figure 2. The σ/μ variation is 9% and the spatial correlation factor is 0.5. This means that a value at a certain point is correlated with values in a radius equal to half the width of the die. We generated 100 sample dies and consider the geometric mean of performance using the method suggested by Sarangi et al. (2008b).

The next step is to calculate the values of $\mathcal{X}(r,n)$ and $\mathcal{Y}$ values such that they can be used in Equations 17, 18, and 19. From the variation map, we observe that due to high spatial correlation, if we divide a die into a large number of cores, then each core has a homogeneous timing profile. However, if we divide a die into a smaller number of cores, then each core has some degree of heterogeneity. A core is expected to have at least 10-20 functional units(FUs). Consequently, each FU has a more or less homogeneous timing profile, and the frequency of a core can be modelled as the minimum frequency that a constituent FU can support. Thus $\mathcal{Y}$ was found to be fairly independent of both n and r. However, $\mathcal{X}$, is a function of both n and r for a smaller number of cores. Beyond 128 cores, it is dependent on only r/n as the values tend to converge because of spatial locality. Figure 3 plots the values of $\mathcal{X}$ and $\mathcal{Y}$. We further observe an approximate linear dependence between $\mathcal{X}$ and r.

4.1.1 Variation of $\mathcal{X}$ and $\mathcal{Y}$

Table 2 lists the different values of $\mathcal{X}_{max}$ and $\mathcal{Y}$ for different values of σ/μ for the threshold voltage, V_{th}. For obvious reasons $\mathcal{X}_{min}$ is equal to $\mathcal{Y}$. Based on the results in Figure 3, we can safely assume that $\mathcal{X}$ varies linearly for different values of r with an insignificant amount of error.

$$
\begin{aligned}
S_{symmetric}(p,n,r) &= \frac{1}{\frac{1-p}{Perf(r,f_r)} + \frac{pr}{n \times Perf(r,f_{min})}} \\
&= \frac{1}{\frac{1-p}{Perf(r\mathcal{X}(r,n)^2,f_0)} + \frac{pr}{n\mathcal{Y} \times Perf(r,f_0)}} \quad (Equations\ 22\ and\ 23) \\
&= \frac{1}{\frac{1-p}{Perf(r\mathcal{X}(r,n)^2,f_0)} + \frac{pr\mathcal{X}(r,n)^2}{n\mathcal{X}(r,n)\mathcal{Y}\mathcal{X}(r,n) \times Perf(r,f_0)}} \\
&= \frac{1}{\frac{1-p}{Perf(r\mathcal{X}(r,n)^2,f_0)} + \frac{pr\mathcal{X}(r,n)^2}{n\mathcal{X}(r,n)\mathcal{Y} \times Perf(r\mathcal{X}(r,n)^2,f_0)}} \\
&= S_{symmetric'}(p, n\mathcal{X}(r,n)\mathcal{Y}, r\mathcal{X}(r,n)^2)
\end{aligned}
\tag{25}
$$

	# Sequential BCEs	# Total BCEs	# Cores
	r'	n'	n'/r'
		plain	
symmetric	$r\mathcal{Y}^2$	$n\mathcal{Y}^2$	n/r
asymmetric	$r\mathcal{Y}^2$	$n\mathcal{Y} - r\mathcal{Y} + r\mathcal{Y}^2$	$((n\text{-}r)/\mathcal{Y} + r)/r$
dynamic	$r\mathcal{Y}^2$	$n\mathcal{Y}$	$n/r\mathcal{Y}$
		opt	
symmetric	$r\mathcal{X}(r,n)^2$	$n\mathcal{X}(r,n)\mathcal{Y}$	$n\mathcal{Y}/r\mathcal{X}(r,n)$
asymmetric	$r\mathcal{X}(r,n)^2$	$n\mathcal{Y} - r\mathcal{Y} + r\mathcal{X}(r,n)^2 - (\mathcal{X}(r,n) - \mathcal{Y})\sqrt{r}$	$n\mathcal{Y}/r\mathcal{X}(r,n)^2$ - $\mathcal{Y}/\mathcal{X}(r,n)^2 + 1$ - $(\mathcal{X}(r,n) - \mathcal{Y})/\sqrt{r}\mathcal{X}(r,n)^2$
dynamic	$r\mathcal{X}(r,n)^2$	$n\mathcal{Y}$	$n\mathcal{Y}/r\mathcal{X}(r,n)^2$

Table 1 Effective number of BCEs for different configurations

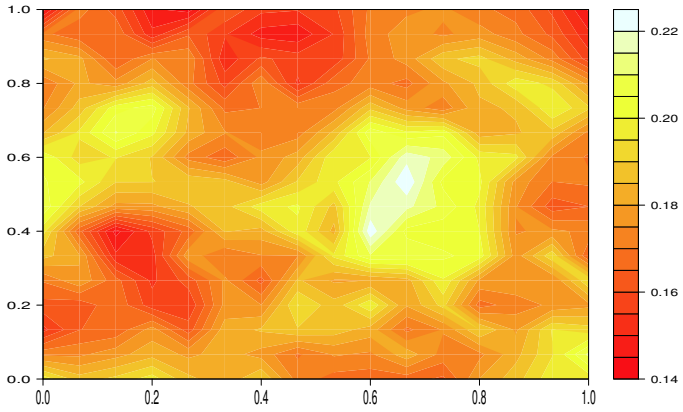

Figure 2 Sample V_{th} variation map

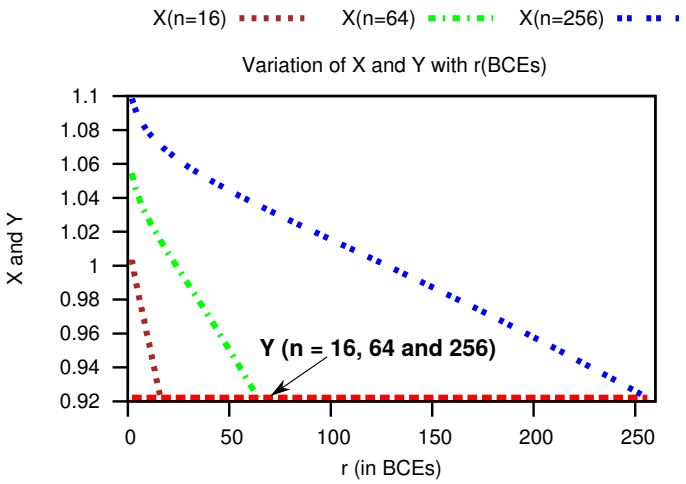

Figure 3 $\mathcal{X}$ and $\mathcal{Y}$

4.2 Performance Results

In Section 4.2.1, 4.2.2, and 4.2.3, we plot the speedups of different schemes. In all our experiments we consider a futuristic system with 256 base core equivalents ($n = 256$). We first plot the speedups across different values of r for the *no-var* (no variation) configuration. We subsequently plot the values for the *opt* and *plain* configurations normalised to *no-var*. We plot the results for three parallel fractions (p) – 0.75, 0.90, and 0.99. The plots for *no-var* are similar to the results derived by Hill and Marty (2008). We use the prefixes *sym*, *asym*, and *dyn*, for symmetric, asymmetric, and dynamic respectively.

4.2.1 Symmetric Cores

Figure 4 plots the speedups for different values of r, and Figure 5 plots the ratios. Figure 4 is normalised to the performance of 1 core ($n = 1, r = 1$).

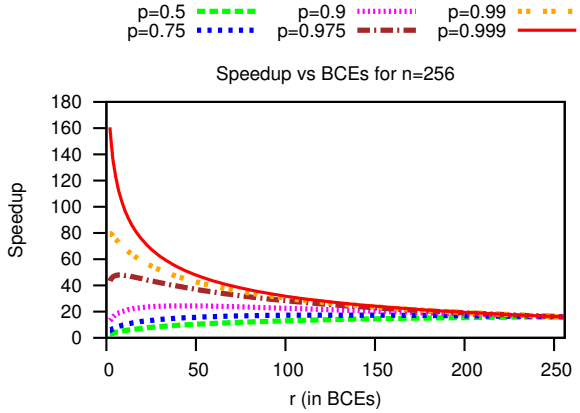

Figure 4 *sym-no-var* versus r

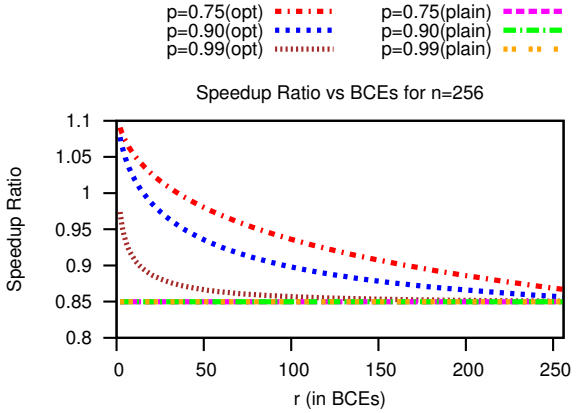

Figure 5 *sym-opt* and *sym-plain* versus r

$\sigma/\mu\%$	$\mathcal{X}_{max}$	$\mathcal{Y}$	$\frac{f_r}{f_0}$	$\frac{f_{min}}{f_0}$
1	1.017	0.996	1.026	0.995
2	1.028	0.987	1.042	0.980
3	1.039	0.977	1.059	0.966
4	1.049	0.967	1.076	0.952
5	1.061	0.958	1.094	0.938
6	1.073	0.949	1.112	0.925
7	1.084	0.940	1.131	0.912
8	1.096	0.931	1.151	0.900
9	1.108	0.922	1.171	0.889
10	1.121	0.914	1.193	0.876
11	1.134	0.905	1.215	0.864
12	1.147	0.897	1.238	0.853

Table 2 Different values of $\mathcal{X}$ and $\mathcal{Y}$

Result 1 – *plain*
The *plain* configuration has a constant speedup ratio, 0.85, irrespective of r. This means that unless the amount of process variation is reduced through either timing speculation, body biasing, or changes in the manufacturing process, the *sym-plain* configuration is not the preferred option.

Result 2 – *opt*
The *opt* configuration has speedups as compared to *no-var* for relatively small values of r (< 30). This means that to take advantage of variation, we should not have very large symmetric cores. The optimal number of cores considering different values of p, should be between 10-20 for a system with 256 BCEs. To summarise, a symmetric core should be roughly 5% of the core die area.

Result 3 – *opt*
For large values of p (> 0.90), the *opt* configuration does not have any speedups. This is because the speedups come from an accelerated sequential section. Consequently, we can conclude that the symmetric configuration is not the best choice for high values of p. Operating systems should schedule applications with moderate values of p on symmetric multicores in large datacenter scale systems.

4.2.2 Asymmetric Cores

Figures 6 and 7 show the speedups for asymmetric multicores across a range of values for the configuration granularity, r. We take cognisance of the fact that the baseline speedups for *no-var* are higher than the symmetric case.

Result 4 – *plain*
As compared to the symmetric case, asymmetric multicores show some degree of variability in the performance for *plain* . Nonetheless there is always a slowdown as compared to *no-var* . Since there is no sequential acceleration, we get the best results with the highest value of the parallel fraction – p. We will further observe in Section 4.2.3, that dynamic multicores are not very helpful in this case. Consequently, keeping the complexity of dynamic cores in mind, we can conclude that it is best to schedule embarrassingly parallel benchmarks on asymmetric multicores for the *plain* configuration.

Result 5 – *opt*
As compared to *asym-plain*, there are speedups over *asym-no-var* in *asym-opt* till r is equal to about 50 for $p < 0.90$. The asymmetric configuration is more resilient to changes in r. This is because the loss of parallel cores with increasing r is partially made up by the increasing size of the sequential core. Since the sequential core runs at a frequency, which is higher than nominal, the effects cancel out.

Result 6 – *opt*
High values of p, imply lower values of r for optimal speedups in the *opt* case. This is because we require more cores for parallel acceleration for higher values of p. Setting an r value between 20-40 BCEs can prove to be useful for a wide range of benchmarks. This means that the large asymmetric core should neither be very big, nor should it be very small. It should occupy about 10% of the core die area. Researchers should focus on designing cores of this size and ensure that they scale for subsequent technology nodes. A similar conclusion was derived by Eyerman and Eeckhout (2010) with albeit a more complex analysis.

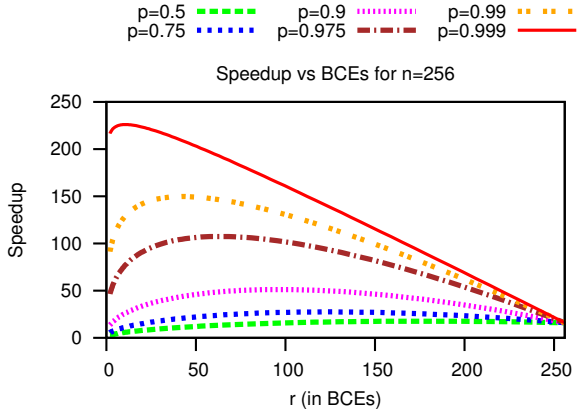

Figure 6 *asym-no-var* versus r

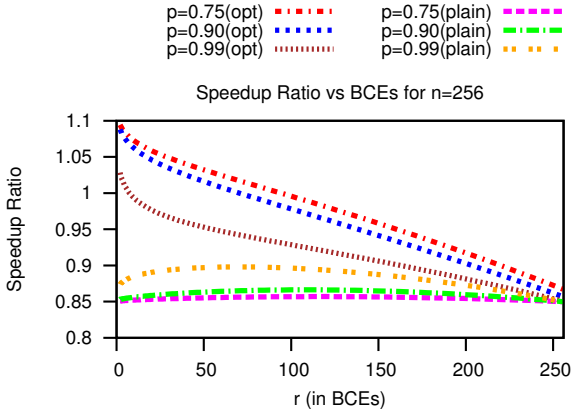

Figure 7 *asym-opt* and *asym-plain* versus r

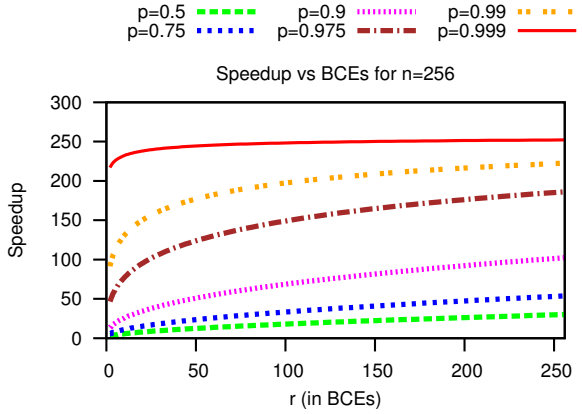

Figure 8 *dyn-no-var* versus r

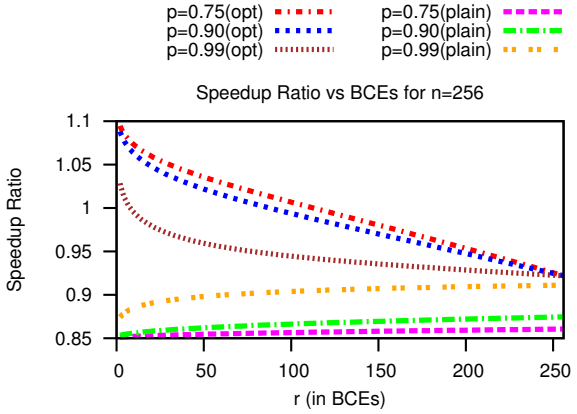

Figure 9 *dyn-opt* and *dyn-plain* versus r

4.2.3 Dynamic Cores

Figure 8 and 9 show the speedups for the dynamic configuration. We observe fairly constant speedups in Figure 8 across a wide range of values for r.

Result 7 – *plain*

Like the asymmetric multicore, *plain* has a fairly constant speedup across a wide range of values for r. The *plain* dynamic multicore is definitely more versatile, and should be used when we cannot make any assumptions about the target workload or when we do not have any knowledge of the variation map.

Result 8 – *opt*

As compared to asymmetric multicores, the performance does not degrade significantly with high values of r. It nonetheless does degrade because by increasing r we are losing the benefit of spatial locality, and consequently it will be necessary to reduce the frequency to ensure that there are no timing faults. Nevertheless, *opt* is always better than *plain*. Secondly, the dynamic *opt* configuration is very good for benchmarks with large sequential portions. Hence, designers should focus on designing dynamic multicores for a large range of r. For smaller values of r, the dynamic core can be used for sequential acceleration of parallel programs by leveraging process variation information. For mostly sequential programs, we would need a large value of r because performance increases as $\sqrt{r}$.

4.3 Notion of Equivalent Cores

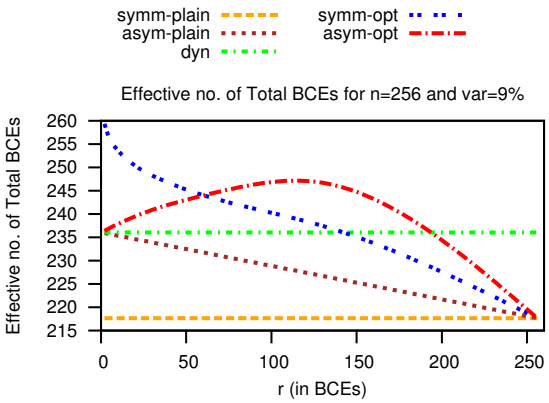

Figure 10 Effective cores for *opt* and *plain*

In Section 4.2, we looked at some of the major design decisions that need to be taken in the presence of process variation. In this Section, we look at how we can leverage the knowledge of equivalent cores(Equation 16

and Table 1) to explain some trends. We plot the effective number of total BCEs, n', in Figure 10.

We observe that *sym-plain* is the worst configuration because it has the least number of effective cores(217). *Sym-opt* fares much better because it has a higher number of effective cores. In fact, for very low values of r, the effective number of cores are more than 256. However, n' rapidly reduces with r to 217. We see a similar reducing trend in *asym-plain*. In comparison, *asym-opt* is better because it has a higher value of n' across a large range of r. In this case, as r increases to high values, the number of parallel cores reduce, and thus we do not get commensurate speedups. The most versatile and effective configurations are *dyn-opt* and *dyn-plain*. Please note that the number of effective cores is equal for both *dyn-opt* and *dyn-plain* (Table 1). The *dyn* configurations have 237 effective cores across all values of r.

We can use the values of $\mathcal{X}$ and $\mathcal{Y}$ (see Table 2) to explain the effectiveness of different schemes to mitigate process variation. For example, some approaches such as body biasing or supply voltage scaling (Sarangi et al. (2008b)) change the voltage of the substrate or the supply voltage to effect a change in the threshold voltage. This reduces the amount of variation at the cost of power. Let us say that we reduce the amount of variation from 9% to 4% using a combination of such techniques. We can quickly evaluate the consequent change in performance by looking at the number of extra equivalent cores we gain using the results in Table 1 and Table 2.

The notion of equivalent cores mentioned in Table 1 is much more than a mere modelling tool. We believe that it can be used by architects, and circuit designers to port their results from systems with no variation to systems with variation. We can use our modified version of Amdahl's law for predicting trends, explaining results (as we do for Eyerman and Eeckhout (2010)), and as mathematical models for multicore systems. Furthermore, we can use the notion of equivalent cores to abstract away the notion of physical cores. An operating system or virtual machine can just see the number of abstract cores and schedule a parallel application. A lower level layer can dynamically split the tasks if required and map them to actual physical cores.

4.4 Experiments with Real World Benchmarks

In this section, we try to validate our models with experiments on real systems. For our experiments we used a Dell R810 server. It had two 6-core Intel Xeon (X5650) processors running at 2.65Ghz. The processors had the hyperthreading mode enabled, and thus 24 threads were available to software. Each processor had 12 MB of L2 cache, and the system had 24 GB of main memory. We used Ubuntu Linux v12.04, and the version of the kernel was 3.2.42. We wrote all our benchmarks in C, and used gcc (v. 4.6.3) to compile the benchmarks.

4.4.1 Nature of the Workload

We use a synthetic benchmark that has a sequential and a parallel section. The sequential section repeatedly adds a pair of 500x500 matrices in a *for* loop. The number of iterations is equal to $iter_{seq}$. This is a configurable parameter. In the parallel section, each thread adds a pair of matrices in parallel. The number of iterations in the parallel section is equal to $iter_{par}$ per thread. $iter_{par}$ and the number of threads running in parallel is configurable. We try to minimise the effects of operating system jitter in our baseline system by ensuring that there are no other user processes in parallel, and by setting the priority of the processes to *real time*.

We simulate two configurations – *baseline* and *equivalent*. The *baseline* configuration assumes that we have process variation. Secondly, the sequential and parallel portions can possibly run at different frequencies. The *equivalent* configuration assumes a system without variation. Here, both the sequential and parallel portions run at the same frequency. Table 1 shows the mapping between both the configurations using the concept of an *equivalent core*. The aim of this experiment is to show that using this concept of equivalent cores, we have an *equivalent* configuration whose performance is the same as the *baseline* configuration.

To simulate the equivalent configuration, we run both the sequential and parallel sections at a frequency equal to f_0. However, in this configuration, the number of cores ($c' = n'/r'$) might be different than the number of cores in the *baseline* configuration($c = n/r$). This will happen when we are using any *opt* configuration. To incorporate this change, we need to scale the number of iterations in the parallel section by c/c'. Secondly, in both the *plain* and *opt* configurations, the BCEs per core (r') change. We only evaluate the case of symmetric configurations because asymmetric and dynamic multicore processors have not been commercialised yet. In the symmetric configuration, $r' = r \times \lambda^2$. $\lambda = \mathcal{Y}$ for *plain* , and $\lambda = \mathcal{X}$ for *opt* . In either case, the performance is expected to change by a factor of λ (according to Equation 12). Hence, to incorporate this change, we need to further multiply the number of iterations of both the sequential and parallel sections by a factor of $1/\lambda$.

To simulate the *baseline* configuration, we need to assume that the number of cores is equal to n/r. Since $r = 1$ in this case, the number of cores is equal to n ($n = 24$). The number of iterations in the sequential and parallel sections are $iter_{seq}$ and $iter_{par}$ respectively. However, the frequencies of both the sections need to be different. The frequency of the sequential section needs to be equal to $f_0 \times f_{min}$ for *plain*, and $f_0 \times f_{rel}$ for *opt* . Likewise, the frequency of the parallel section needs to be equal to $f_0 \times f_{min}$ for both *plain* and *opt* (according to Equations 20, and 21).

4.4.2 Results for the plain *Configuration*

Based on our discussion above, we set the appropriate values of $iter_{seq}$, $iter_{par}$, and the frequencies of the different sections for both the *baseline* and *equivalent* configurations.

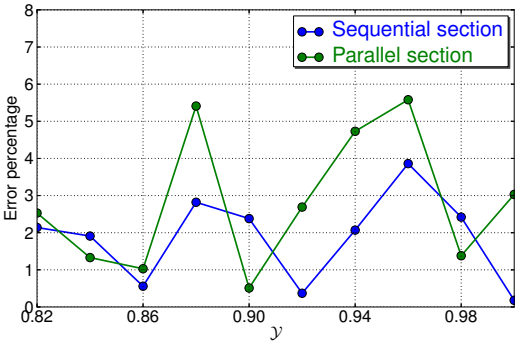

Figure 11 Error percentage versus $\mathcal{Y}$ for *plain* configuration

Figure 11 shows the results for the *plain* configuration. We compute the relative error as the difference in the execution time of both the configurations divided by the execution time of the *baseline* configuration. For the sequential section, the error is mostly limited to 3%. Whereas, for the parallel section the error varies from 0.5-5.5%.

4.4.3 Results for the opt *Configuration*

In a similar manner, we set the values of $iter_{seq}$, $iter_{par}$, and the frequencies of different sections for the *opt* configuration. Figure 12 shows the error in the sequential section as a function of $\mathcal{X}$(r,n). We found the error to be independent of $\mathcal{Y}$. The error varies from 0-2.5%. Figure 13 shows the error in the parallel section as a function of $\mathcal{Y}$ and $\mathcal{X}$(r,n). The error is contained within 1-6% for a majority of values. For low values of $\mathcal{Y}$, and high values of $\mathcal{X}$, the error approaches 8%.

To summarise, the error for most of the configurations is less than 5-6%. We believe that the results are compelling because, the operating system jitter is typically of the same order (also observed by De et al. (2009); Petrini et al. (2003); Jones et al. (2003)). Hence, we believe that it would not have been possible to obtain significantly better results because of the inherent noise in the system.

5 Related Work

The original paper (Amdahl (1967)) on Amdahl's law was published by Gene Amdahl in 1967. It has subsequently appeared in most major computer architecture textbooks. We focus on some recent augmentations to the original formulation, which have primarily looked at the law from the point of energy, temperature, and performance. To the best of our knowledge, researchers have not looked at Amdahl's law based formulations for chips with process variation as we do in this paper.

Hill and Marty (2008) extended Amdahl's law to model multicores. They proposed a set of performance equations for symmetric, asymmetric, and dynamic multicores. We use their formulations as a basis for our technique. Along with the basic equations, they looked at a series of trends and made a set of design recommendations. This work was extended by Yao et al. (2009) to calculate the optimal configuration for different types of multicores using analytical techniques. Eyerman and Eeckhout (2010) explore Amdahl's law exclusively in the context of critical sections. As compared to other papers, the authors argue against asymmetric multicores since they don't find them helpful in accelerating the performance of barriers and critical sections. They favor symmetric multicores.

Woo and Lee (2008); Cho and Melhem (2010) extended the Hill and Marty model to take energy into account. They proposed a formulation that considers parameters dependent on the dynamic power and leakage power. They use a different core for sequential computation and a different set of cores for parallel computations and sum up the power. Subsequently, they model the power for this ensemble of cores, and propose an additional optimisation framework in Cho and Melhem (2010). Cassidy and Andreou (2012) propose a more extensive optimisation framework based on similar extensions to Amdahl's law for both performance and energy.

Huang et al. (2010) proposed a thermal extension to Amdahl's law. They added some temperature related constraints and parameters to the Hill and Marty model. They observed a sizeable loss in performance due to thermal constraints. However, they were able to conclude that a properly configured asymmetric multicore is the best choice.

6 Conclusion and Future Directions

In this paper, we have tried to create a new formulation of Amdahl's law that takes process variation into account. It tries to intuitively explain the broad trends in processor performance by a novel mathematical widget namely an *equivalent core*. The crux of our approach is to replace a system with variation with a system without variation by mapping the actual number of cores to a set of equivalent cores. We subsequently reason on only the set of equivalent cores. We observe that our method of reasoning is potent enough to explain broad trends in performance, previous results, and to evaluate the soundness of different design methodologies and architectures. We further use our model to comment on different architectural design styles and their suitability for running a multitude of parallel programs.

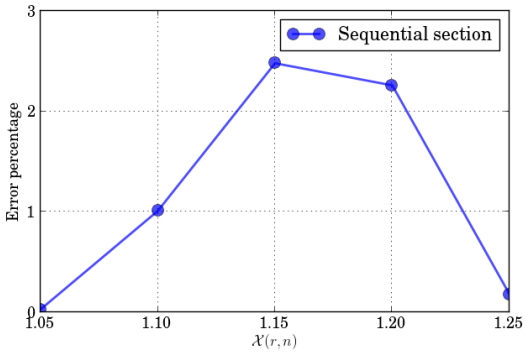

Figure 12 Error percentage versus $\mathcal{X}(r,n)$ in the sequential section – *opt* configuration

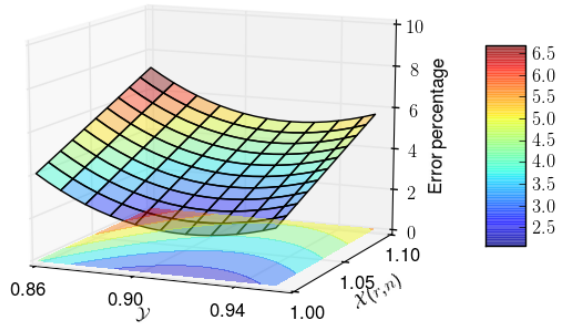

Figure 13 Error percentage versus $\mathcal{X}(r,n)$ and $\mathcal{Y}$ in the parallel section – *opt* configuration

Skeptics might argue that our model is simplistic and needs a non-trivial number of parameters for tuning it to get exact values. However, we have empirically observed that the conclusions we derive and the methods we suggest are fairly independent of the exact choice of parameters. Secondly, generic laws like Amdahl's Law have always been used for zeroth order design decisions, and for guiding research at a high level. We believe that our method of reducing a real system to a hypothetical system with equivalent cores belongs to the same class of solutions. This is a small step in the journey to mathematically model complex architectures of the future.

References

Aater S., M., Mutlu, O., Qureshi, M., Patt, Y., 2010. Accelerating Critical Section Execution with Asymmetric Multicore Architectures. Micro, IEEE 30 (1), 60 –70.

Amdahl, G. M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS.

Borkar, S., Karnik, T., Narendra, S., Tschanz, J., Keshavarzi, A., De, V., 2003. Parameter variations and impact on circuits and microarchitecture. In: Proceedings of the 40th annual Design Automation Conference. DAC '03. pp. 338–342.

Cassidy, A. S., Andreou, A. G., 2012. Beyond Amdahls Law: An Objective Function That Links Multiprocessor Performance Gains To Delay and Energy. IEEE Trans. on Computers 61 (8).

Cho, S., Melhem, R. G., 2010. On the Interplay of Parallelization, Program Performance, and Energy Consumption. IEEE Trans. Parallel Distrib. Syst. 21 (3), 342–353.

De, P., Mann, V., Mittal, U., 2009. Handling OS jitter on multicore multithreaded systems. In: IPDPS.

Eyerman, S., Eeckhout, L., 2010. Modeling critical sections in Amdahl's law and its implications for multicore design. SIGARCH Comput. Archit. News 38 (3), 362–370.

Gluhovsky, I., O'Krafka, B., 2005. Comprehensive multiprocessor cache miss rate generation using multivariate models. ACM Trans. Comput. Syst. 23 (2), 111–145.

Gupta, S., Feng, S., Ansari, A., Mahlke, S., 2010. Erasing core boundaries for robust and configurable performance. In: Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO '43. pp. 325–336.

Hartstein, A., Srinivasan, V., Puzak, T. R., Emma, P. G., 2006. Cache miss behavior: is it $\sqrt{2}$? In: Proceedings of the 3rd conference on Computing frontiers. CF '06. pp. 313–320.

Hill, M. D., Marty, M. R., 2008. Amdahl's Law in the Multicore Era. IEEE Computer 41 (7), 33–38.

Huang, W., Skadron, K., Gurumurthi, S., Rib, R. J., Stan, M. R., Huang, W., 2010. Exploring the Thermal Impact on Manycore Processor Performance. In: Semitherm.

Ipek, E., Kirman, M., Kirman, N., Martinez, J. F., 2007. Core fusion: accommodating software diversity in chip multiprocessors. SIGARCH Comput. Archit. News 35 (2), 186–197.

Jones, T., Dawson, S., Neely, R., Tuel, W., Brenner, L., Fier, J., Blackmore, R., Caffrey, P., Maskell, B., Tomlinson, P., Roberts, M., 2003. Improving the scalability of parallel jobs by adding parallel awareness to the operating system. In: SC.

Kumar, R., Tullsen, D. M., Ranganathan, P., Jouppi, N. P., Farkas, K. I., 2004. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In: ISCA.

Miller, T., Pan, X., Thomas, R., Sedaghati, N., Teodorescu, R., feb. 2012. Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips. In: High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on. pp. 1 –12.

Petrini, F., Kerbyson, D. J., Pakin, S., 2003. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In: SC.

Sarangi, S., Greskamp, B., Teodorescu, R., Nakano, J., Tiwari, A., Torrellas, J., feb. 2008a. VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects. Semiconductor Manufacturing, IEEE Transactions on 21 (1), 3 –13.

Sarangi, S. R., Greskamp, B., Tiwari, A., Torrellas, J., 2008b. EVAL: Utilizing processors with variation-induced timing errors. In: MICRO. pp. 423–434.

Woo, D. H., Lee, H.-H., dec. 2008. Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era. Computer 41 (12), 24 –31.

Yao, E., Bao, Y., Tan, G., Chen, M., 2009. Extending Amdahl's law in the multicore era. SIGMETRICS Perform. Eval. Rev. 37 (2), 24–26.