Moore's Law goes Multicore: The economic consequences of a fundamental change in how computers work

Neil Thompson

Sloan School of Management, MIT[1]

[1] I would like to thank my PhD advisors David Mowery, Lee Fleming, Brian Wright and Bronwyn Hall for excellent support and advice over the years. Thanks also to Philip Stark for his statistical guidance. This work would not have been possible without the help of computer scientists Horst Simon (Lawrence Berkeley National Lab) and Jim Demmel, Kurt Keutzer, and Dave Patterson in the Berkeley Parallel Computing Lab, I gratefully acknowledge their overall guidance, their help with the Berkeley Software Parallelism Survey and their hospitality in letting me be part of their lab. Andrew Gearhart and David Sheffield also provided great feedback and help beta-testing the survey. The participants of the UPCRC Parallel Computing Lab retreat, and the Haas Business and Public Policy professors and students made suggestions that improved this paper, as did numerous industry contacts that let me interview them. I am grateful to them all. Andrew Fang also provided excellent research assistance. Finally, particular thanks to Nick Bloom for all his advice and encouragement on this paper.

Abstract

"Computing performance doubles every couple of years" is the popular re-phrasing of Moore's Law, a 50-year old prediction by one of the founders of Intel. Over that time, it has described the 500,000-fold increase in the number of components on modern computer chips that forms the basis for virtually all information technology used today. But what impact has this vast expansion of the technological frontier of computing had on the productivity of firms?

This paper tries to answer this question by focusing on the mid-2000s, at a moment when the manifestation of Moore's Law changed to 'multicore' chips. Unlike earlier chips that increased processor speed faster with each generation, these chips stayed the same speed but added more-and-more processors, or 'cores,' on each chip (e.g. Intel CORE Duo). But, taking advantage of multiple processors requires that software can split work amongst them. That is, the software must be *parallelized*. Software that is not designed that way cannot use the extra processors and thus its performance stagnates. Thus the switch to multiple-processors chips grows the technological frontier asymmetrically, disproportionately benefiting parallelized software and firms that use it.

This paper estimates the productivity impact of this differential expansion on a panel of Swedish firms from 2001 to 2007 using a remarkably detailed dataset on Swedish I.T. usage and a novel survey of computer scientists carried out for this paper. It shows that prior to the switch to multicore, productivity growth was consistent with an equal expansion of the technological frontier across all firms, but that afterwards those that could take advantage of Moore's Law in its multicore form grew productivity faster, becoming more productive over the 2005-2007 period.

The paper then argues that this should be interpreted as a broad-based, causal estimate of the effect of information technology (I.T.) on firm productivity because the changeover to multicore chips was a surprise, and because of the similarity of the treatment and control groups prior to the changeover.

Real-world examples are also presented that support the description of the phenomenon and the mechanism being described.

1 Introduction

In spring 2002, Intel got it wrong. At the Intel Developer Forum they announced that the processors on their computer chips would continue to speed-up, as they had for decades, saying: “What you have seen is a public demonstration of 4 GHz silicon straight off our manufacturing line. We have positive indications to be able to take [this technology] to the 10 GHz space.”[2] Less than two years later, engineering problems forced Intel to first delay and then cancel plans to introduce a 4 GHz chip to the public.[3] Instead they, and other chip-makers, turned to producing computer chips with multiple processors – the so-called ‘multicore’ chips. This was an enormous change, which Patrick Gelsinger, the Chief Architect of Intel’s 486 processors, later called the single greatest shift to date in microprocessor architecture.[4]

For users of information technology, this was a shock whose ramifications took time to appreciate. In 2005, the Chair of an important programming technical committee wrote, “[the automatic performance improvements from processor speed-up] has already been over for a year or two, only we’re just now noticing.”[5]

Dreamworks was one company that experienced and later wrote about their efforts to adapt to this change. At the time of Intel’s abandonment of processor speed-ups Dreamworks was in the throes of what they called “Shrek’s Law” – the doubling of computing power used in each new incarnation of the Shrek movies (the second of which was just coming out). By 2008, after several years without processor speed-ups providing performance improvements for them, Dreamworks made the expensive decision to re-write large parts of their animation code to *parallelize* it. This allowed them to split up the work that the software was doing amongst the multiple processors on their multicore chips, which their previous *unparallelized* code had been largely unable to do. This was a success, providing a 70% overall improvement, with particular gains in areas like fluid modelling that lent themselves to parallelization.[6]

This paper focuses in on the period from 2001-2007, which covers Intel’s abandonment of processor speed-ups (2004), the launch of multicore processors as the standard hardware for PCs (2005) and the initial period before software developers, like those at

[2] De Gelas (2005).
[3] Sutter (2005).
[4] Brock (2006), pp 102-104.
[5] Sutter (2005).
[6] Dreamworks ([*confirm date*]) and interviews.

Dreamworks, could adapt to multicore. Analyzing this period allows this paper to focus on how firm productivities changed when an unexpected technological shock hit virtually all users of I.T., but where the effect was strongest for users whose existing code had less parallelism.

This paper shows that before the release of multicore chips in 2005, having more parallelism in your code was *not* predictive of having greater firm productivity, consistent with the unresolved debate of the benefits of it in computer science discussions at the time. Furthermore, the distribution of productivity amongst those with the most parallelism and those with the least parallelism is very similar, suggesting that pre-treatment the groups were alike.

After Intel's unexpected switch to multicore chips, however, those firms with more parallelism in their software grew productivity more rapidly. By 2007 their total factor productivity was growing 0.75pp faster per standard deviation in software parallelism. A placebo test and various robustness checks affirms that this effect does not arise out of long-term trends in the underlying industries, nor other obvious candidates for differences between firms (size, technological sophistication, etc.).

To put the estimated magnitude in context, if one divides firms into terciles by the amount of parallelism that they had in 2001 (well before the shock), the top tercile grows productivity 2.5pp more than the bottom tercile over the 2001-2007 period, with all of the gain coming after the 2005 introduction of multicore.

This estimate provides a lower-bound[7] for the impact of Moore's Law on firm productivity over this time period. It suggests that Moore's Law-fueled improvements in computing performance have substantially improved the productivity of firms, which is consistent with the increasing share of firm revenues expended on I.T. over the past decades.[8]

To the author's knowledge, this estimate also provides the most broad-based causal identification of the effect of I.T. on firm productivity since it uses an unexpected technological shock that impacts virtually all I.T.-using firms.

The argument for this paper is laid out as follows. Section 2 outlines previous findings about the importance of computing to firms. Section 3 discusses how Moore's Law and

[7] This is a lower bound since all firms continued to benefit from other effects of Moore's Law, e.g. increasing on-chip memory (cache).

[8] IDC ([*confirm date*]

computer hardware have evolved.[9] Section 4 shows how these hardware changes translate into software performance, and why some software benefits more from multicore than others. Section 5 outlines the central hypothesis of this paper: that these differences in software performance impact firm productivity. It also presents the data used to evaluate this hypothesis. Section 6 discusses the methodology, and Section 7 presents the results, and their robustness. Section 8 places these findings in a larger context, and Section 9 concludes.

2 Firms and Information Technology

The use of I.T. by firms is extensive; databases organize their data, enterprise resource systems coordinate their supply-chains, manufacturing systems, and customer relationships, and desktop software helps them create and manage the email and other documents of a modern office. This permeation of I.T. into the way firms work is evidenced in their spending decisions. In 2002, market-research company IDC estimated that firms spent ~4% of gross revenues on I.T. Globally, this is estimated to total $1.5 trillion.[10]

A variety of approaches have been used to estimate the impact of I.T. on firm productivity. Macroeconomists have looked at this question through a growth-accounting lens, for example Gordon (2000), Jorgenson, Stiroh, Gordon and Sichel (2000). Many of these have suggested positive impacts from I.T. expenditures.

Of particular relevance for this work is the growth decomposition work by Jorgenson et al. In that work the authors look at the productivity of U.S. industries, producing the data shown in Figure 1. It categorizes U.S. industries into following groups: (1) I.T. producing industries, (2) I.T. using industries, (3) non-I.T. using industries and two accounting terms (not shown). It then reports the aggregate contribution to productivity from each.

Figure 1

I.T. using industries go from being the largest contributor to productivity growth in 1995-2000 and 2000-2005, to a negative contributor in 2005-2010. It is not the absolute level of these effects that is interesting, but rather the comparison to I.T. producing industries which fall much less. This drop suggests a change for I.T. using industries. Some of this

[9] This paper draws heavily on Thompson (Working Paper) for the underlying computer science. Various parts of that work are repeated here for exposition.

[10] IDC (2002).

is likely to be the 2008 financial crisis, but Jorgenson, Ho and Samuels (2010) do an industry-level disaggregation that shows that the financial sector is not the main driver of their results.

This paper will propose a potential mechanism that might underlie some of this change, but will not (as yet) be able to fully connect to the macro-level results. That work is on-going.

Microeconomists have also tackled the question of the impact of I.T. on firm productivity at the firm level, perhaps most notably in the 2003 paper by Brynjolfsson and Hitt, which looks at firm I.T. expenditure and correlates it with improvements in multi-factor productivity. They conclude that there is a sizable return to investments in I.T., but that the benefits are delayed ~5 years as management learns to take advantage of the new capabilities.[11] Despite the plausibility of good management and I.T. systems being complements in producing productivity increases, Brynjolfsson and Hitt's findings don't rule out the possibility that these results are driven by reverse causality (more productive firms buy more I.T.) or other missing variables (e.g. good managers buy more I.T. and make firms more productive).

A recent paper by Aral, Brynjolfsson and Wu has attempted to address these causality issues using data on the purchase and implementation of large enterprise software systems (e.g. enterprise resource planning systems - ERP). They observe that the purchase of ERP systems is not correlated with performance improvements, but that the 'go-live' events are.[12] But, if the 'go-live' decisions are also timed to firm needs (for example expanding sales), then this might also be endogenous.

Computer scientists have also tried to quantify the benefits of computing in a number of studies, for example in the airline, automobile and semiconductor industries. These find substantial benefits from the usage of computers, and heavy costs from instances where computers are down.[13] There are also other industries, such as finance, logistics, and oil exploration, where the benefits of computation are understood to be large, but where firms are reluctant to share detailed information for fear of undermining their competitive advantage.

[11] Brynjolfsson and Hitt (2003).

[12] Aral, Brynjolfsson and Wu (2006).

[13] For example, see discussions in: National Research Council (2005), Council on Competitiveness (2005), Hennessy and Patterson (2007).

Collectively, these studies suggest that I.T. impacts productivity. Unfortunately, most of these estimates are either indirect, for example by measuring the *cost* of I.T., or analyzing aggregated, rather than firm-specific, numbers. These choices make endogeneity and reverse causality confounds harder to separate out, and thus make causality harder to establish. This paper takes a different approach. It looks at a change in the technology at the heart of computers, microprocessors, and argues that it will impact firm performance. To make this case, and to substantiate why it should have a causal interpretation, it is important to understand how computing power has increased over the years.

3 Moore's Law and the rise of the microprocessor

At the heart of each modern computer is a microprocessor, usually in the form of a central processing unit (CPU). This is the part of the computer that performs virtually all computations; it takes data from memory, manipulates it, and then returns the results back to memory, perhaps to be acted upon again later. Microprocessors are comprised of an enormous number of switches, which are used both to store data and to process instructions. In early computers these switches were vacuum tubes, but in modern machines they are comprised of transistors. Roughly speaking, computers that have more transistors have more computing power.

Gordon Moore observed in 1965 that the number of components per computer chip were doubling. This was later re-articulated as the number of transistors doubling every (roughly) two years. This exponential growth, known as "Moore's Law," became a hallmark of computing power increases and a roadmap used by the semiconductor manufacturing industry to coordinate improvements.

Miniaturization has played a pivotal role in fulfilling Moore's Law. As transistors become smaller, more can be fit on the chip and the power that each consumes drops. Together, these allow computers to run faster than they had previously – the clock-speed[14] of the computers increases. This is important because the clock-speed determines how fast the microprocessor can perform a computation, so, roughly speaking, if you double the clock speed, you double the speed of a computation.[15] This correlation has led to the more-

[14] Roughly: the speed at which the computer can perform a simple operation.

[15] There are many factors that can alter this relationship, for example memory access.

popular conception of Moore's Law, that the performance of computers doubles (roughly) every two years.[16]

Over the past 40 years, the speed of computers has increased a thousand-fold. Figure 1, based on data compiled by various computer scientists including the Stanford VLSI group,[17] documents the increase in the number of transistors per chip and the clock-speed of those chips.

[Figure 1]

Figure 1 also highlights the 2004 change in the relationship between the numbers of transistors and clock-speed, and the subsequent rise in the number of multicore chips. This changeover is at the heart of this paper, and so is described in detail.

In November 2000, Intel launched the Pentium 4 chip. Like previous Intel processors, the Pentium 4 had a single processor (i.e., had a single core). It operated at a clock speed of 1.4 GHz, meaning that it performed 1.4 billion computations per second.[18] Intel also announced that, as with previous models, additional versions of the Pentium 4 would be released that would be faster. They sharpened this prediction in 2002 (as shown in the quotation in the introduction), claiming that future versions would scale up to 10GHz.[19] That prediction never come to pass, and instead chip speeds stalled in the 3-4 GHz range.

The reason for the failure to deliver faster clock speeds is heat.[20] Along with more and faster transistors came increased heat buildup in the chips – so much that ultimately the chips began to melt. Thus, on October 14, 2004 Intel announced that it would not release any faster versions of the Pentium 4, but "instead will dedicate resources to pushing dual-core processors to market."[21] It released its first dual-core offering, the Pentium D, in May 2005.[22] AMD faced similar challenges, and released their first dual core offering, the Turion 64 X2, in August 2005. Because Intel controlled 86% of the processor market at this time,[23] this paper focuses on its behavior. Since Intel's 2004 announcement, the

[16] At a technical level this is more correctly identified as a result of "Dennard Scaling".

[17] Data collected by Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, Krste Asanoviç and Angelina Lee. Data available at http://cpudb.stanford.edu/.

[18] Intel (2000). Here 'a computation' is used loosely, to avoid unnecessary technical detail.

[19] PC & Tech Authority (2009).

[20] Appendix A discusses this in more detail.

[21] Regan (2004).

[22] Olukotun et al. (2012). As Figure 1 shows, an IBM dual core chip was released in 2004, but it was a reduced instruction set computing (RISC) chip and so not usable for most code written for Intel hardware.

[23] Frost and Sullivan (2005).

number of cores on a chip has continued to rise and clock-speeds have continued to languish.

While the change to multicore computing may sound merely technical, it is fundamental to how hardware works and to how software performs, as the next section will show. Patrick Gelsinger, a Senior VP at Intel and chief architect for the 486 microprocessor describes the shift to multiple cores as the single greatest shift to date in microprocessor architecture.[24] The National Research Council, part of the National Academy of Sciences, went further, describing their fears about the impact of the switch to multicore: "One might expect that future IT advances will occur as a natural continuation of the stunning advances that IT has enabled over the last half-century, but reality is more sobering."[25]

To understand why this changeover from clock speed improvements to multicore could have such an important effect on the performance of I.T., the connection between hardware and software performance needs to be made clear.

4 The connection between hardware and software performance

Broadly speaking, all software programs benefit from an increase in clock speed since the calculations they perform run faster. In his famous 2005 article, Herb Sutter, Chair of the C++ programming language's ISO committee, discusses how increases in clock-speed improve software performance:[26]

> *"Most classes of applications have enjoyed free and regular performance gains for several decades, even without releasing new versions or doing anything special, because the CPU manufacturers (primarily) and memory and disk manufacturers (secondarily) have reliably enabled ever-newer and ever-faster mainstream systems"*

This highlights a critically important element of these clock-speed improvements: they produced improvements in software performance *even when no change was made to the software itself.*[27] Another way to think about this is that AMD and Intel were selling a

[24] Brock (2006).

[25] National Research Council (2011).

[26] Sutter (2005).

[27] National Research Council (2011).

bundle to consumers; they were selling both a physical microprocessor as well as an implicit improvement in the performance of software that users already owned.

But even this understates the value of clock-speed improvements because it only highlights changes on the intensive margin – firms using software to do their existing computations more quickly. Given the long history of innovative new software applications, it is plausible that at least as large a could arise from changes on the extensive margin – firms being able to use new software, or being able to use old software, now faster, for new purposes. Computer scientists John Hennessy (Stanford) and David Patterson (Berkeley) emphasize the extent of this change, arguing that the increase in computing power "has significantly enhanced the capability available to computer users. For many applications, the highest-performance microprocessors of today outperform the supercomputer of less than 10 years ago."[28]

So, to summarize, clock-speed improvements in hardware provide automatic speed-ups in software, improving both existing performance and creating new capabilities. The end of clock-speed increases stops this free improvement for most software. As Herb Sutter said "the free lunch is over."[29] The rise of multicore processors does not compensate for the end of clock-speed improvements because, in order to take advantage of multiple processors, a software program must be able to split up the work amongst them. That is, it must be *parallelized*. If not, it will only run on a single processor, and the other processors will sit idle.[30] As a consequence, the incremental improvement for an unparallelized program from multiple cores will be small.[31] In contrast, a parallelized program would be able to take advantage of these cores, and thus its benefits would be higher. Historically, few programs were parallelized, with the most-common exception being high-end simulations on supercomputers (e.g. for simulating nuclear explosions). Later, some parallelism was introduced in the form of *multi-threading*, an attempt to use single processors more effectively by sequencing their work differently. Results from these attempts were mixed, and interviews and software assessments carried out for this paper suggest that, even today, it is the exception, rather than the rule that most programs are able to make effective use of parallelism.

[28] Hennessy and Patterson (2007).

[29] Sutter (2005).

[30] Although, if multiple programs are being run at the same time, the operating system may be able to use the other processors for other purposes.

[31] There will still be some benefits, for example from larger on-chip memory (cache).

The impact of being able to use parallelism is reflected in the following graph, which charts the performance of the multiprocessor (i.e. parallelized) version of Stata.[32] It illustrates several important points about software parallelism.

[Figure 2]

The black lines represent the two extremes of performance, perfect scaling, where the increase in the speed of the calculation grows proportionally to the number of cores, and no improvement. For Stata, the median improvement for all commands of running on an 8-core machine is only 2.7X. Thus, as a software program, Stata's performance is neither completely unparallelized (1X), nor perfectly parallelized (8X) despite the fact that it has been re-engineered for multicore machines. This likely reflects two issues: (i) parallel programming is hard, (ii) there is a theoretical limit to how parallel a particular task can be made.[33] This second point is reflected in the differing level of parallelism for various commands. Logistic regression scales quite well, getting 6.5X performance gain from 8 cores, whereas Arima regression scales poorly, benefiting only 1.2X. The difference between these is that Logistic regression can been broken into smaller calculations that are relatively independent (and thus can be run on different processors). In contrast, the auto-regressive and moving-average parts of Arima regression link calculations together, making such a split difficult.[34]

The much-reduced benefit of multicore on unparallelized code can been seen not just in specific examples like Stata, but in general computing performance benchmarks. The SPEC benchmark suite is "designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems."[35] It contains tests for a variety of tasks, including many business-relevant ones. Figure 3, from computer scientists Hennessy (Stanford) and Patterson (Berkeley), summarizes progress on this benchmark suite since 1978 (footer added):[36,37]

---

[32] Stata is a statistical analysis tool used by economists and other researchers. See Stata (2012b) for more details.

[33] See, for example, the Wikipedia article on "Amdahl's Law" for a discussion of one of the reasons for this.

[34] For further discussion of the parallelism of individual Stata commands, see Stata (2012). For a technical discussion of algorithms and their parallelism, see Asanovic et al. (2006).

[35] SPEC (2012).

[36] Hennessy and Patterson (2007). The benchmarks shown are the SPECint benchmarks. The SPEC floating point operations show a similar trend – see NRC (2011).

[37] In order to create this time series, various SPEC benchmarks have been strung together.

[Figure 3]

This graph shows the transition in the mid-1980s from mini-computers, which used distinct components, to integrated chips that benefited from Moore's Law-driven improvements in computing speed. This transition was accompanied by an increase in the pace of improvement on the benchmark from 25% per year to 52% per year. This remarkable growth in performance of integrated chips then fell back to 22% per year with the changeover to multicore microprocessors.[38, 39] In the words of Hennessy and Patterson: "this 16-year renaissance is over. Since 2002, processor performance improvement has dropped to about 20% per year due to the triple hurdles of maximum power dissipation of air-cooled chips, little instruction-level parallelism left to exploit efficiently, and almost unchanged memory latency."[40]

It is clear that the changeover to multicore chips is having a profound impact on improvements in software performance, as measured by computer science metrics. The question then becomes, are the impacts of this change visible in the performance of firms?

5 Connecting software parallelism to firm performance

If software performance is growing more slowly for firms using unparallelized software after the multicore transition then the natural question would be: are they also growing their productivity more slowly? Figure 4 shows this hypothesis schematically:

[Figure 4]

Here a (hypothetical) firm with perfectly parallelized software continues growing productivity as rapidly as during processor speed-up times because it can take full advantage of the exponential increase in the number of cores. Conversely, firms with unparallelized software get a much-reduced benefit.[41]

[38] In keeping with the original authors, the changeover is marked in 2003, rather than the 2004 date used elsewhere. Both dates are reasonable.

[39] Benchmarks of completely parallelizable algorithms (for example LINPACK) do not show a change with the introduction of multicore.

[40] They date this change to the end of clock speed improvements rather than the introduction of multicore chips.

[41] Increases in performance improvement are unlikely to fall to zero because of other chip improvements (larger caches, improved chip layout, etc.) continue.

Investigating this hypothesis requires three kinds of data: measures of firm performance, knowledge of the software that firms are using, and a measure of the parallelism in each type of software. The data used for each of these purposes is outlined below.

More than 30 qualitative interviews were conducted to supplement this quantitative data. Interviewees were from academia as well as semiconductor and software firms.

5.1 Firm performance data

The firm performance data used in this paper is from the Bureau Van Dijk Orbis firm database.[42] It contains typical accounting information, including line items needed to calculate Total Factor Productivity (TFP), which is described in Section 6.2. These line items are relatively more complete for the Swedish data, which was the decisive factor in choosing that country as the setting for this study.

Despite the greater completeness of data for Sweden, key data are missing for many firms. As a result, only ~35% of the sample can be used in the full analysis.[43] A more complete discussion of the Bureau Van Dijk coverage of Swedish firms can be found in Bloom, Sadun, and Van Reenan (forthcoming).

5.2 Firm software data

Harte Hanks, a market research firm, gathers data on software and I.T. usage.[44] They do this using phone surveys that ask establishments[45] to report the details of their:

- Computer System / Servers
- Local Area Network
- Network Connection Hardware
- Network Connection Services
- Operating Systems
- Personal Computers
- Server Operating Systems
- Software

Harte Hanks collects this data to sell it to software and hardware vendors. This creates an incentive for Harte Hanks to be accurate in their assessments, as data purchasers may be in a position to verify these details if the data is used for sales leads.

[42] Bureau Van Dijk (2007).

[43] Even amongst this reduced sample, some smaller line items must still be imputed to get reasonable sample sizes. This is done using a simple regression framework (R^2 for the regression is 84%).

[44] More details on this research is available in Harte Hanks (2002), and Harte Hanks (2006).

[45] An establishment is one location of a business.

Of the data collected by Harte Hanks, this analysis focuses on firm software, as captured by the Operating Systems, Server Operating Systems, and Software categories.[46] The following list summarizes the most-widely used types of software from those categories:

[Table 1]

The Harte Hanks data used in this paper covers 6,600 Swedish establishments from 2001-2007.[47] Figure 5 shows the percentage of these establishments using each type of software.

[Figure 5]

The following two figures decompose these averages. The first shows the share of establishments using software based on whether or not the firm (of which they are a part) has below- or above-median sales. The second figure repeats this analysis, but for below- or above-median I.T. intensity (as measured by PCs per employee).

[Figure 6]

[Figure 7]

Sales seem to have little effect on the percentage of establishments using different types of software. However, firms that have high I.T. intensity, as measured by PCs per employee, have higher levels of penetration of virtually every type of software. The ~20% increased penetration of databases is of particular interest.

5.3 Software Parallelism Data

The parallelism present in various software types is determined using the Berkeley Software Parallelism Survey,[48] one part of Thompson (Working Paper). This online survey was sent to ~300 computer scientists at the University of California, Berkeley; Stanford University; the University of Illinois, Urbana-Champaign; and in Industry. The survey had a ~20% response rate, but not every question was answered by every

[46] Unfortunately these categories do not provide any details on the intensity of the usage of software by the firm. This is an important limitation to this paper.
[47] Harte Hanks (2007).
[48] A list of the survey questions can be seen at http://faculty.haas.berkeley.edu/neil_thompson/Berkeley_Software_Parallelism_Survey/Berkeley_Software_Parallelism_Survey.pdf. [needs updating once the new site is live]

respondent. As a result, the sample sizes for individual questions vary, and in some cases are small.[49]

The level of software parallelism was determined by asking:[50],[51]

- "In practice, how parallelized are current implementations of this type of software…"

This was asked for each of the software types listed earlier (e.g. databases).[52] Because more respondents were able to rank the parallelism in categories of software (e.g. ERP vs. compilers) than between types of software (e.g. Microsoft vs. Oracle's implementation of databases), only variation in the *category of software* is used to construct measures of parallelism. This has the disadvantage of not using all of the differences in parallelism between firms. Conversely it has the advantage that it is less plausible that 'astute' firms could bias the result by strategically switching between types of software, since ERP programs are not a substitute for compilers.

The survey results indicate that software, such as Finance/Accounting or Antivirus programs, have more parallelism in practice, and others, such as Web Browsers or Compilers, have less.[53] Importantly for this paper, software that is associated with technically sophisticated firms, for example databases, compilers, and enterprise resource systems, have a broad range of parallelism scores; they are neither predominantly parallelized nor unparallelized.

6 Methodology and Identification Strategy

This section describes five aspects of the methodology and identification strategy of this paper. First it discusses the treatment variable, establishment-level software parallelism,

[49] A second phase of survey is planned to augment these values. See Thompson (Working Paper) for a more detailed discussion of this.

[50] All questions were asked using a 7-point Likert scale, but then converted using even spacing to a 0-100% scale for ease of interpretability. The Likert value for 0% was "not at all" parallelized and the value for 100% was "completely" parallelized. More details on the Likert scale can be found at http://en.wikipedia.org/wiki/Likert_scale, for example.

[51] Pilot testing of the survey asked this question retrospectively to 2004, but respondents were unable to answer in that form. This is discussed further in the robustness checks section.

[52] Some program categories thought to be important for productivity, such as CAD/CAM, are not included in the survey because they are used by so sparsely (for CAD/CAM < 1% of establishments)

[53] Once subsequent phases of the survey are completed, these results will be reported in greater detail.

and discusses how this is obtained from the software-level parallelism data. Second, it explains the outcome variable: total factor productivity, a metric of how efficiently firms turn their inputs into outputs. Third, it presents the regression specifications used to analyze the data. Fourth, it outlines the robustness checks for those specifications. And fifth, it argues that the coefficient estimate from these specifications should have a causal interpretation.

6.1 Firm-level parallelism

Creating a single variable that fully captures an establishment's software parallelism is hard in theory and harder in practice. In this sense, it may be akin to capturing national output in a single GDP figure (what about unpaid work?). But, like this example, the inability to construct a perfect measure does not imply that an imperfect one is not valuable.

In theory, a measure of software parallelism should indicate the benefit that the establishment will get from using multiple processors for its software. Since some software is used more than others, such a measure should reflect how intensively software is used. Similarly, it should reflect how important software is in the production process. Unfortunately, constructing such a measure requires detailed knowledge of individual workers, their software, and how these fit into the firm's production process. Such detailed knowledge, even about a single firm, is entirely beyond the scope of this paper.

Instead, this paper adopts a simple and admittedly imperfect, metric for establishment software parallelism: the unweighted average of the parallelism in the establishment's software. While this measure fails to account for differences in software usage intensity as well as its importance in the production process, it is the best starting point available. Figure 8 shows the distribution of this measure of establishment parallelism for the Swedish establishment data. For this graph, the Likert scale from the survey has been mapped to the 0-1 interval for clarity.

[Figure 8]

These results indicate that the average software for these establishments corresponds to "Somewhat" or "Moderately" parallelized in the language of the Berkeley Software Parallelism Survey.

6.2 Firm Productivity

Firm productivity is measured using total factor productivity. The following description of this metric draws heavily on Syverson's survey of the topic (2011).

Total factor productivity (hereafter just "productivity") is a ratio of the output of the firm to its inputs. The intuition is that a firm producing more output from the same inputs is more productive. The formula for calculating productivity for firm *i* is:

$$Total\,Factor\,Productivity_{Firm\,i} = \frac{Sales_i}{Labor_i^{\alpha_{i,L}} * Capital_i^{\alpha_{i,K}} * Materials_i^{\alpha_{i,M}}}.$$

In this equation, α_L, α_K, α_M are the cost shares associated with labor, capital and materials respectively. The denominator of the equation is a Cobb-Douglas production function. According to Syverson, a Cobb-Douglas production function is known to be "a first-order approximation of any production function."[54]

The conditions under which total factor productivity perfectly measures productivity are that firms are cost minimizing, that their factor markets (e.g. labor) have perfect competition, and that their production technology has constant returns to scale. These are strong assumptions, and this paper defers to the productivity literature to defend them. Nevertheless, this paper can address some concerns from the productivity literature *better* than other papers because the data on Swedish firms includes material costs. For many other countries, data on material costs are not publicly available and thus 'revenue' total factor productivity, a less good proxy, is used instead.

Figure 9 plots the productivity for the sample, with establishments with below-average software parallelism in red and above-average parallelism in blue. The data is shown as of 2003, which pre-dates both the announcement of the end of clock speed improvements (2004) and the launch of multicore chips (2005).

[Figure 9]

This graph suggests that, in terms of the pre-treatment productivity, the treatment and control groups are very similar.[55] Figure 10 shows that pre-treatment productivity *growth* is also similar.

[Figure 10]

[54] Syverson (2011).

[55] Statistical tests of these differences are presented in Table 3.

Together Figure 9 and Figure 10 suggest that, prior to exposure to the shock, those with more or less software parallelism looked 'similar' in terms of productivity, making them good for comparing the effect of the switch to multicore. Moreover, the similar pre-treatment growth suggests validates the usage of a difference-in-differences approach to measure this effect.

6.3 Regression specifications

The goal of the following analyses is to quantify how software parallelism impacts firm productivity after the switchover to multicore chips.

6.3.1 Difference-in-differences specifications

The base specification is a differences-in-differences estimation that looks at whether higher pre-treatment parallelism leads to a differential increase in productivity.

The following expression describes the quantity of interest (shown here with parallelism dichotomous):[56]

$$\beta = E[\Delta avg\ TFP\ growth(par_{high})] - E[\Delta avg\ TFP\ growth(par_{low})]$$

Where $\Delta avg\ TFP\ growth$ is the change in average productivity growth from the 'before' period, 2001-2003, to the 'after' period, 2005-2007, and $par_{high/low}$ reflects the software parallelism level of the establishment in 2003 (pre-treatment).[57] Growth in 2004 is excluded from the pre-period, since it is unclear what effect the announcement of the switch to multicore designs would have on chip replacement behavior.

Because this is a difference-in-differences estimator, any establishment-level characteristics that are not time-varying are netted out. This produces the following specification, here with parallelism continuous:

[56] Although firm software parallelism is a continuous variable, it is presented here as a dichotomous variable for expositional purposes.

[57] This is calculated using the 'in-practice' parallelism software measures from the Berkeley Software Parallelism Survey and applying them to the software being used by the firm in 2003. The Robustness Section discusses the implications of the timing of these two measures.

I. Base differences-in-differences estimator[58]

$$avg\ TFP\ growth_{2005-2007,\,i} - avg\ TFP\ growth_{2002-2003,i} = \alpha + \beta * Parallelism_{2003,e} + \gamma_1 * PCs\ per\ employee_{2003,e} + \gamma_2 * \log(\#\ I.T.Developers_{2003,e}) + \varphi * \log(\#\ Employees_{2003,e})$$

II. With ordinal parallelism rankings

$$avg\ TFP\ growth_{2005-2007,\,i} - avg\ TFP\ growth_{2002-2003,i} = \alpha + \beta * Parallelism\ Rank_{2003,e} + \gamma_1 * PCs\ per\ employee_{2003,e} + \gamma_2 * \log(\#\ I.T.Developers_{2003,e}) + \varphi * \log(\#\ Employees_{2003,e})$$

Here i indexes the firm, and e the establishment. Because productivity is at the firm level, whereas parallelism is at the establishment level, this will induce correlation in the error terms between establishments at the same firm. These are accounted for by clustering the standard errors at the firm level.

The coefficient of interest is β, the impact of establishment-level parallelism. This paper's hypothesis is that establishments with more parallelism in their software should have higher TFP growth in 2005-2007 than those with little parallelism in their software. Thus the prediction is that the sign of β will be positive.

The additional covariates, PCs per employee and # I.T. Developers, correspond to the number of PCs per employee and the total number of I.T. developers at the establishment. The number of employees the total number of employees at the establishment.

An important potential confound in this data would be industry productivity trends that pre-date the introduction of multicore chips. Since establishments in an industry are likely to do similar tasks, they are likely to use similar software types, and thus have similar levels of parallelism. If certain industries had particular growth paths, this could lead to incorrectly assigning industry trends to parallelism.[59]

[58] Strictly speaking, specifications I, and II have employees on both the right- and left-hand side of the estimating equation, since it is an input to productivity growth. However the estimated coefficients are so small that there is no substantive effect. The panel specifications show this directly.

[59] Here "particular growth paths" would be ones where the *acceleration* of productivity growth is correlated with parallelism. It is not enough for productivity growth itself to be correlated, since this would appear in both the before- and after-productivity growth measures and thus would cancel out in the difference-in-differences.

Figure 9 and Figure 10 address this issue, showing that both the levels of productivity and the growth of productivity are very similar between the high- and low-parallelism groups. This makes the pre-trend argument less plausible. Nevertheless, the hypothesis about industry trends can be tested directly in both the differences-in-differences specification and the panel specifications (following). In the difference-in-differences framework this can be done by testing on a placebo, pre-treatment, year. If it is industry trends that are driving the productivity changes, these should appear even prior to the multicore change. If not, it suggests that industry trends emerge at the same time as the multicore change, and that the extent is correlated with the parallelism of firms in that industry, which is more consistent with a story of firms using similar software being affected similarly.[60]

III. Difference-in-differences placebo specification (Placebo year: 2002)

$$avg\ TFP\ growth_{2002-2003,\,i} - avg\ TFP\ growth_{2001-2002,i} \\ = \alpha + \beta * Parallelism_{2003,e}$$

The importance of specifications II and III as robustness checks are discussed in detail in section 6.4.

The difference-in-differences analyses are for a balanced panel and present only a single, overall-impact value. The following panel specifications allow us to relax these assumptions at the cost of some (software) limitations on the standard error clustering.

6.3.2 Panel specifications

The second set of specifications for examining this question are panel-level regressions. These take the following forms:

IV. Base panel specification

$$Productivity\ (TFP)_{i,t} = \beta_t * Parallelism_{e,t=2001} + \alpha_t + \psi_e$$

[60] Notice that industry fixed effects (here measuring growth trends) cannot be used to disentangle this question, since the difference-in-differences collapses the panel into a cross-section, and thus industry fixed effects would simultaneously estimate both pre-existing industry trends and post-multicore industry trends together. In contrast, the panel specification, which follows, does allow for industry trends effects.

V. With linear industry trends pre- and post-multicore[61]

$$\begin{aligned}Productivity\ (TFP)_{i,t} \\ &= \beta_t * Parallelism_{e,t=2001} + \alpha_t + \psi_e + \tau_{SIC3} * Industry_{SIC3} * Year \\ &+ \omega_{SIC3} * Industry_{SIC3} * Post * Year\end{aligned}$$

VI. With industry-year fixed-effects

$$\begin{aligned}Productivity\ (TFP)_{i,t} \\ &= \beta_t * Parallelism_{e,t=2001} + \alpha_t + \psi_e + \kappa_{SIC3,t} * Industry_{SIC3} * Year_t\end{aligned}$$

In all cases, *i* indexes the firm, *e* the establishment, *t* the year, and *SIC3* the industry. The coefficients of interest are β_t, the productivity benefits of establishment-level software parallelism. Software parallelism is measured in 2001, pre-multicore, to ensure no reverse causality.

In specification V, linear industry trends are added. These are flexible, allowing both a pre-multicore trend and a post-multicore change in the trend. In specification VI this is replaced with an even more flexible industry-year fixed effects model, which can adjust for any industry time trend (including non-linear ones). In all cases industry values are calculated at the 3-digit SIC level.

6.4 Robustness checks

To ensure that the results presented are robust, three sets of tests are performed.

The first robustness check is the placebo test, checking whether the 'parallelism' impact emerges before multicore chips are released. This is done by considering 2001-2002 as the 'before' period, and 2002-2003 as the 'after' period. The difference-in-differences model is then estimated from that sample of data. This is presented in specification III.

The second robustness test checks the robustness of the results to the specific distribution of the software parallelism rankings from the Berkeley Software Parallelism Survey. This is important because the highly-specialized nature of the survey means that some questions have few answerers, so differences in interpretation of ranking scores are

[61] Errors here should ideally be clustered at the firm level, as in the difference-in-differences regressions, however current implementations of the panel regression software (plm) in R cannot handle the memory load required for this calculation for specifications V, VI. As a consequence, these results are presented with errors unclustered. Nonetheless, running specification IV with clustered standard errors at the establishment level produces only a small change in the p-values.

plausible and could impact the results. This robustness check is done by converting the software parallelism scores into rankings. The following equations summarize this change (j indexes the type of software):

[Table 2]

The third robustness check is a permutation test, also known as a Fisher Exact Test (Fisher, 1935), which non-parametrically relaxes the normality assumption in the output variable, productivity. This is accomplished by dividing the sample into those with above- and below-median parallelism. Under the null hypothesis that there is no relationship between parallelism and productivity, these groups should have equal productivity growth changes (i.e. equal values for the dependent variable from Equation 2). The test is run by randomly permuting the labels for these observations and determining the share of permutations that produce differences in productivity as large as observed in reality.

6.5 Causal interpretation

This paper argues that the estimate for the effect of parallelism on firm productivity should be interpreted causally: having parallelized software improves firm productivity in this period. This is based on several lines of argument: (i) the similarity between those with higher parallelism and lower parallelism prior to the changeover to multicore, (ii) the changeover to multicore being a surprise, which made it difficult for firms to anticipate the change, and (iii) the effect on productivity appearing before firms could firm adapt.

Table 3 shows the covariate balance between firms in 2003, based on whether they had parallelism above or below the average. It also shows where these differences are statistically significant, both as a difference in means (t-test) and as a difference in distributions (Kolmogorov-Smirnov test).

[Table 3]

These indicate that firms with greater parallelism have lower sales, fewer employees, and fewer I.T. staff (not shown). Despite some differences in the average values of some of these variables, there is still common support across them. Figure 11 illustrates this for the Sales variable.

[Figure 11]

Table 3 also tests the distributions presented in Figure 9 and Figure 10. It confirms that despite some differences in some pre-treatment covariates, both productivity levels and growth are not statistically significantly different between the groups. This suggests that the other differences in covariates are not creating differential productivity outcomes.

The argument that the switch to multicore computing came as a surprise is strongly supported by Intel's own statements. As already mentioned, Intel announced in 2002 that they would continue the decades-long trend of increasing microprocessor clock speed, but then in 2004 they publicly reversed course and began focusing on multicore processors.

Interviews with Intel support this view, suggesting that they adopted this change only hesitantly, saying that Microsoft "didn't want us to do this [switch to multicore], and we didn't either, but we were forced to by the physics."[62] Moreover, action to help important software partners adapt also didn't start until after the change. It was not until 2005 that Intel and Microsoft began to collaborate on multicore, and not until 2007 – 2009 that Intel released the first set of tools to help programmers write parallel code.

Statements by outside computer scientists also support the argument that the community was taken by surprise. To the author's knowledge, the first public announcement of the importance of the multicore changeover was by Herb Sutter, Chair of the C++ committee, who wrote in 2005 that "the free lunch has already been over for a year or two, only we're just now noticing."[63] Similarly, the 'Berkeley View' report, one of the first academic documents to highlight the importance of this change was published in December 2006, and stated that the group had been meeting over the past "nearly two years," suggesting that they also became engaged with this in early 2005.[64]

Finally, there is a question about whether I.T.-using firms could have adapted to the change within the sample period. To do so, firms would need to (i) substitute between the types of software that they are using, or (ii) re-implement their existing software to get more parallelism from it. The first of these probably happened to some extent, but is sharply limited by different software doing different tasks (by definition) and hence firms cannot replace their customer management software with, say, Powerpoint. The second of these could be done either by the company themselves in-house or by third-party software providers. There is evidence that both of these took time, even for firms well positioned to notice and adapt to this change. For example, STATA MP was not released until mid-2006. Similarly, a request by Wall Street traders for more parallelization in Excel for use in their Monte Carlo simulations was not met until the release of Office 2007.[65] In-house

[62] Conducted for this study.

[63] Sutter (2005).

[64] Asanovic et al. (2006).

[65] As related in interviews with developers working on the project.

parallelization of Dreamworks animation software *began* in 2008 (discussed in greater detail in section 8).

These examples show that, even for important applications by large, technically-savvy software companies, there were long delays in redeveloping software to take advantage of multicore chips. But, as will be shown later, firms get the benefit from multicore chips almost immediately *despite not getting a benefit beforehand.* This suggests that the level of parallelism that these firms have is a 'frozen accident,' with firms choosing their software before 2004 based on their business needs, and then finding after the switch to multicore that parallelism that they happen to have yields new productivity benefits.

Together, the similarity of firms before the changeover, the 'surprise' of the switch to multicore, and the emergence of a software parallelism effect before firms could adapt, suggest a causal interpretation to the estimated effects.

7 Results

This section estimates the effect of software parallelism on productivity. Unless otherwise noted, all results are initially calculated as a level, or change, in total factor productivity arising from increased software parallelism. However, because the interpretation of this quantity is unintuitive, the initial coefficient estimates are re-scaled to have the more-intuitive interpretation of *the percentage point increase in productivity level (or growth rate) for an average firm if it had one standard deviation more software parallelism.* That said, it is worth noting that a difference of one standard deviation in parallelism is a large change, and that firms within an industry (and thus having similar types of software) might have a difference in parallelism of less than a standard deviation.

7.1 Overall trends

Prior to adding any covariate controls, Figure 12 simply shows the average productivity of firms over time for two groups: those whose level of parallelism in 2001 was in the top-third of all firms, and those whose level of parallelism was in the bottom-third. This directly tests the hypothesis from Figure 4.

[Figure 12]

It shows the predicted pattern – no difference in the productivity of high- versus low-parallelism firms prior to the multicore switch and a large, statistically significant, difference afterwards. The subsequent analyses test to see whether this can be explained by other causes.

7.2 Difference-in-Differences analysis

This analysis estimates specifications I, II, and III for a balanced panel of firms. Enforcing balance results in the sample being approximately one-third as large as for the panel analysis.

The difference-in-differences analysis estimates the *per year* increase in *productivity growth* in 2005 – 2007 from having one standard deviation more firm software parallelism in 2003.

To get a more robust measure of central tendency the results below are on trimmed samples. These exclude a small number of firms with productivity changes more than 2 standard deviations from the mean.

[Table 4]

The coefficient estimates in the Parallelism row indicate that firms with greater software parallelism grow productivity more rapidly over this period. Column I shows that the magnitude of this effect is 0.75pp** per year. Column II shows the result when the parallelism measure is ordinalized. If parallelism was previously specified correctly, this would just add measurement error, biasing the result towards zero, which is what is shown with the 0.62pp* result. However, that the significance survives the variable transformation suggests that this result is robust to the exact formulation of the measure.

Column III shows the first robustness check: a placebo test. It checks whether firm software parallelism has an effect prior to the introduction of multicore chips. Such a result would call into question the mechanism proposed by this paper.

The placebo test recreates the specification described in Equation 2, but with the 'before' period as 2001-2002, and the 'after' period as 2002-2003. The placebo test shows no economically or statistically significant effect on the growth of productivity in the period prior to the introduction of multicore, consistent with the argument that this effect comes from the introduction of multicore and is not just representative of a longer secular trend.

7.3 Panel regressions

Figure 12 presents coefficient estimates from specification IV. It shows the extent to which a firm with 1 standard deviation more parallelism in 2001 has a higher *level* of productivity in that year. Consistent with the multicore story, it shows no statistically significant differences in productivity before 2005, and large statistically-significant ones afterwards, rising to 1.5% by 2007 (1% significance).

[Figure 12]

Despite the estimate being on a larger, unbalanced sample, this result agrees well with the cumulative impact that the increased *growth rate* of productivity, estimated in the difference-in-differences analysis, would have.

Table 5 presents these results in regression form and presents increasingly flexible industry-level effects.

[Table 5]

Columns V and VI show the additional effect of adding in industry-level controls. Even with Industry-Year fixed-effects the magnitude and approximate significance of the results survive. These results do *not* indicate that inter-industry parallelism is unimportant, but do suggest that intra-industry variation in parallelism is important. Interestingly, once industry effects are added the ramp-up in magnitude of the effect from 2005-2007 becomes smaller, perhaps suggesting that at the end of the period 'laggard' firms in industries that could parallelize were starting to adjust.

7.4 Permutation Test

This robustness test compares the productivity growth of firms with above- and below-median parallelism. It does this by permuting the labels "above" and "below" for the data points and calculating the difference in productivity between the groups *as if those were those the true labels.* This is repeated 10,000 times to produce an assumption-free distribution of outcomes. The true productivity difference between these groups is then compared to the distribution.

Only 2.9% of the 10,000 draws are as extreme (one-tailed test) as the value under the true labels, thus the test rejects the null hypothesis that these labels do not impact productivity gains.

8 Discussion

An underlying principle to this paper is the importance of performance of software for firms. However achieving proper functionality and performance in a new (or revised) piece of software, either for users or 3rd-party software developers, is famously difficult – with development and implementation costs often far exceeding planned budgets.

In this context it is important to understand the role that Intel has played as a hardware provider. In their 2002 book, *Platform Leadership*, Annabelle Gawer and Michael Cusumano discuss how Intel creates innovation. They argue that Intel actively promotes innovation throughout hardware, based on the recognition by Intel that it benefits from – and indeed needs – innovation by other players to create a demand for their chips.

This paper takes this argument one step further, arguing that implicit in the sales of their microprocessors was ready-made improvement of users existing software, since that software run more quickly with little-to-no adjustments on the new hardware. This could enable firms to run software tasks that were previously too slow, or turn existing tasks into ones that could be run in real-time. It is easy to imagine how these could improve productivity. Thus, implicitly Intel was bundling improved software performance with their chips prior to the switch to multicore.

An Intel interviewee explained why firms rarely parallelized their code prior to the multicore switchover, even though they could have done so to run on supercomputers, mainframes, etc. Firm would have needed "to want performance and to want it enough to not just wait a few years for it to happen automatically". Given the uncertainty and costs of software redevelopment, it is perhaps not surprising that relatively few did.

The switch to multicore computing has overturned this relationship. Now firms must adapt their software to reflect their hardware. A number of different approaches to address this problem seem to be emerging.

As has always been the case, firms with sufficiently deep expertise can develop their software in house. Google is an example of a company that has done this, for example in developing its own file management system, BigTable, and its own adaptation of a database, MapReduce.[66]

Firms may also be helped by outside consultants in this process, for example by Intel. One instance of Intel's involvement was a successful collaboration with Dreamworks, who brought them in to help scale up the animation software used to produce the Shrek

[66] Chang et al. (2006), Dean and Ghemawat (2004). MapReduce is not a full database, although it can perform many of the same functions.

movies. Intel was able to achieve a 70% speed-up of Dreamworks' software, using a combination of parallelizing their code and other programming changes.[67]

New products focused on parallel computing are also being developed. One such example is Hadoop, an open-source implementation of Google's MapReduce.

A final possibility is that firms will do their parallel computing through Cloud Computing providers such as Amazon.com's EC2 Cloud. There are two rationales for this change. The first is that once firms are willing to invest to parallelize their software, either directly or through third-party providers, they may choose to scale not just to the 2, 4 or 8 processors on their desktop machines, or even the 16, 32 or 64 processors on their servers, but to thousands of processors on a Cloud. Secondly, since many Cloud providers pre-load and customize software for their machines, they may represent a new platform for computing, with hardward and software bundled as a service.

Although it is just a conjecture at this point, it is interesting to consider that firms may have chosen to move to Cloud Computing in part because their alternative of getting automatic speed-ups through new hardware has disappeared. As a point of contrast, had Moore's Law continued to speed-up computers, all computers would be at least 20 times as fast as they are today.

The effect of parallelism on productivity shown in this paper is a mechanism that gets at the importance of software performance improvements for firms. It will be interesting to see how such performance improvements are sought now that such a powerful source of these improvements (processor speed-ups) is no longer available.

9 Conclusion

This paper investigates how a change in the microprocessor technology that underlies Moore's Law has impacted the productivity of firms. It argues that clock-speed improvements in computer chips provided substantial productivity gains to firms, and that the engineering-driven switch to multicore computing in 2004/2005 has greatly diminished these benefits for some firms.

Analysis of the productivity changes in Swedish firms confirm that firms with greater software parallelism – the prerequisite to take advantage of multicore chips – had larger productivity gains. The magnitude of this effect is estimated to be 0.5pp - 0.7pp in total

[67] Based on interviews and personal communications with Intel staff.

factor productivity growth per year from 2005 – 2007 from one standard deviation greater software parallelism. Non-parametric tests support the robustness of this result qualitatively, as does a placebo test for 2002, which shows no effect. A rich set of controls for industry-level effects also reject the idea that this is simply different industries growing productivity at different rates.

This paper argues that this productivity impact should be interpreted causally – that parallelism allowed some firms to become more productive – because the switch to multicore came as a surprise, and because firms had little knowledge or ability to respond to it within the timeframe of the data.

To the author's knowledge, this is one of only a few papers to test the impact of I.T. on firm productivity in a way that addresses reverse-causality and omitted variable bias issues. It finds strong effects.

Bibliography

Asanovic et al. 2006. *The Landscape of Parallel Computing Research: A View from Berkeley.* UC Berkeley Electrical Engineering and Computer Sciences, Technical Report No. UCB/EECS-2006-183.

Aral, Sinan, Erik Brynjolfsson and D.J. Wu. 2006. *Which came first, IT or productivity? The virtuous cycle of investment and use in enterprise systems.* Twenty-Seventh International Conference on Information Systems, Milwaukee 2006.

Barroso, Luiz A., Kourosh Gharachorloo, Andreas Nowatzyk and Ben Verghese. 2000. *Impact of Chip-Level Integration on Performance of OLTP Workloads.* Sixth International Symposium on High-Performance Computer Architecture (HPCA), January 2000.

Bloom, Nicholas, Raffaella Sadun and John Van Reenan. Forthcoming. *The organization of firms across countries.* Quarterly Journal of Economics

Bloom, Nicholas, Raffaella Sadun and John Van Reenan. 2012. *Americans do I.T. Better: US Multinationals and the Productivity Miracle.* American Economic Review.

Borek, Chris, Laurits Christensen, Peter Hess, Josh Lerner and Greg Rafert. Working Paper. *Lost in the Clouds: The Impact of Copyright Scope on Investments in Cloud Computing Ventures.*

Brock, David. 2006. Understanding Moore's Law: Four Decades of Innovation. Chemical Heritage Foundation, Philadelphia.

Brynjolfsson, Erik and Lorin Hitt. 2003. *Computing productivity: Firm-level evidence.* MIT Sloan Working Paper 4210-01, http://ebusiness.mit.edu/research/papers.html.

Bureau Van Dijk. 2007. *Orbis.* Data extract of income statement and establishment data for Sweden for 2001-2007.

Cabezas, Victoria C., and Phillip Stanley-Marbell. 2011. *Parallelism and Data Movement Characterization of Contemporary Application Classes.* SPAA '11, June 4-6, 2011.

Cabezas, Victoria C., and Phillip Stanley-Marbell. 2011b. *Quantitative Analysis of Parallelism and Data Movement Properties Across the Berkeley Computational Motifs.* CF '11, May 3-5, 2011.

Chang, Fay et al. 2006. *Big Table: A Distributed Storage System for Structured Data.* OSDI: Seventh Symposium on Operating System Design and Implemenation.

Council on Competitiveness. 2005. *High Performance Computing and Competitiveness.*

Dean, Jeffrey and Sanjay Ghemawat. 2004. *MapReduce: Simplified Data Processing on Large Clusters.* OSDI 2004.

De Gelas, Johan. 2005. *The Quest for More Processing Power, Part One: "Is the single core CPU doomed?"* online at http://www.anandtech.com/show/1611.

ExcelTrader. 2011. *Excel Benchmark 2011: An Excel Speed Test (with trading functions).* ExcelTrader online at http://exceltrader.net/984/benchmark_et-xls-an-excel-benchmark-for-traders/.

Favero, Willie. 2008. *DB2 History 101: Version 4.* Toolbox database blogs, online at http://it.toolbox.com/blogs/db2zos/db2-history-101-version-4-23970.

Frost and Sullivan. 2005. *World Market for Dual Core Processors.* Market research report.

Gordon, Robert J. 2000. *Does the "New Economy" measure up to the great inventions of the past?* NBER working paper 7833.

Gower, Anabelle and Michael Cusumano. 2002. <u>Platform Leadership</u>. Harvard Business Review Press.

Harte Hanks. 2002. *CI Technology Database Methodology.* Harte Hanks marketing distribution document.

Harte Hanks. 2006. *Relational Database Format: A guide to using the CI Technology Database with relational database software.* Online at: http://www.europe.hartehanksmi.com/support/downloads/RDF_Manual.pdf.

Harte Hanks. 2007. *CI Technology Database.* Data extract from Sweden, 2001-2007, including establishment and equipment data.

Hennesy, John and David Patterson. 2007. Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publications, San Francisco.

IDC. 2002. *Worldwide IT Spending by Vertical Market Forecast, 2001-2006.* International Data Corporation.

Intel. 2000. *Intel Introduces the Pentium 4 Processor.* Intel News Release, online at: http://web.archive.org/web/20070403032914/http://www.intel.com/pressroom/archive/releases/dp112000.htm.

Jorgenson, Dale, Kevin Stiroh, Robert Gordon and Daniel Sichel. 2000. *Raising the Speed Limit: U.S. Economic Growth in the Information Age.* Brookings Papers on Economic Activity, Vol. 2000, No 1 (2000), pp. 125-235, Brookings Institution Press.

Jorgenson, Dale, Mun Ho, and Jon Samuels. 2010. *Information Technology and U.S. Productivity Growth: Evidence from a Prototype Industry Production Account.* Prepared for *Industrial Productivity in Europe: Growth and Crisis*, online at: http://www.economics.harvard.edu/faculty/jorgenson/files/02_jorgenson_ho_samuels%2B19nov20101_2.pdf.

Kaspersky. 2012. *Kaspersky Anti-virus for Novell NetWare.* Kaspersky Lab, online at http://www.kaspersky.com/anti-virus_novell_netware.

Malone, Michael. 1995. The Microprocessor: A Biography. Springer-Verlag, New York.

National Research Council. 2005. Getting up to Speed: The Future of Supercomputing. National Academies Press, Washington, D.C.

National Research Council. 2011. The Future of Computing Performance: Game Over or Next Level? National Academies Press, Washington, D.C.

Olukotun, Kunle, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, Krste Asanoviç and Angelina Lee. 2012. CPU DB. Data available at http://cpudb.stanford.edu/. Data received in 2012.

OPL Working Group. 2012. *A Pattern Language for Parallel Programming ver2.0.* ParLab Patterns Wikipage, online at:
http://parlab.eecs.berkeley.edu/wiki/patterns/patterns.

PC & Tech Authority. 2009. *The greatest tech U-turns of all time: Intel and Netburst.* Online at: http://www.pcauthority.com.au/News/163122,the-greatest-tech-u-turns-of-all-time-intel-and-netburst.aspx.

PostgreSQL wiki. 2012. *Parallel Query Execution.* PostGreSQL wiki online at:
http://wiki.postgresql.org/wiki/Parallel_Query_Execution.

Regan, Keith. 2004. *Intel Puts 4 GHz Processor on Back Burner.* E-Commerce Times, Oct 15, 2004. Online at http://www.ecommercetimes.com/story/37360.html.

SPEC. 2002. *SPEC's Benchmarks and Published Results.* Online at:
http://www.spec.org/benchmarks.html#cpu.

Stata. 2012. STATA MP homepage. Online at: http://www.stata.com/statamp/.

Stata. 2012b. *Why use Stata statistical software?.* Stata website, online at:
http://www.stata.com/why-use-stata/.

Sutter, Herb. 2005. *The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*, online at http://www.gotw.ca/publications/concurrency-ddj.htm.

Symantec. 2005. *Enabling Multithreaded Scans.* Symantec Knowledge Base Article. Online at
http://www.symantec.com/business/support/index?page=content&id=TECH101387.

Syverson, Chad. 2011. *What determines productivity?* Journal of Economic Literature, 49:2, pp. 326-365.

Televation. 2010. *The Fastest PC.* Online at
http://www.telovation.com/articles/fastest-pc.html

Thompson, Neil. Working Paper. U.C. Berkeley Computer Science Master's Thesis, draft as of December 2012.

Watt, Martin, Lawrence D. Cutler, Alex Powell, Brendan Duncan, Michael Hutchinson, and Kevin Ochs. 2012. *LibEE: A Multithreaded Dependency Graph for Character Animation.* DigiPro '12, the Proceedings of the Digital Production Symposium, pp 59-66.

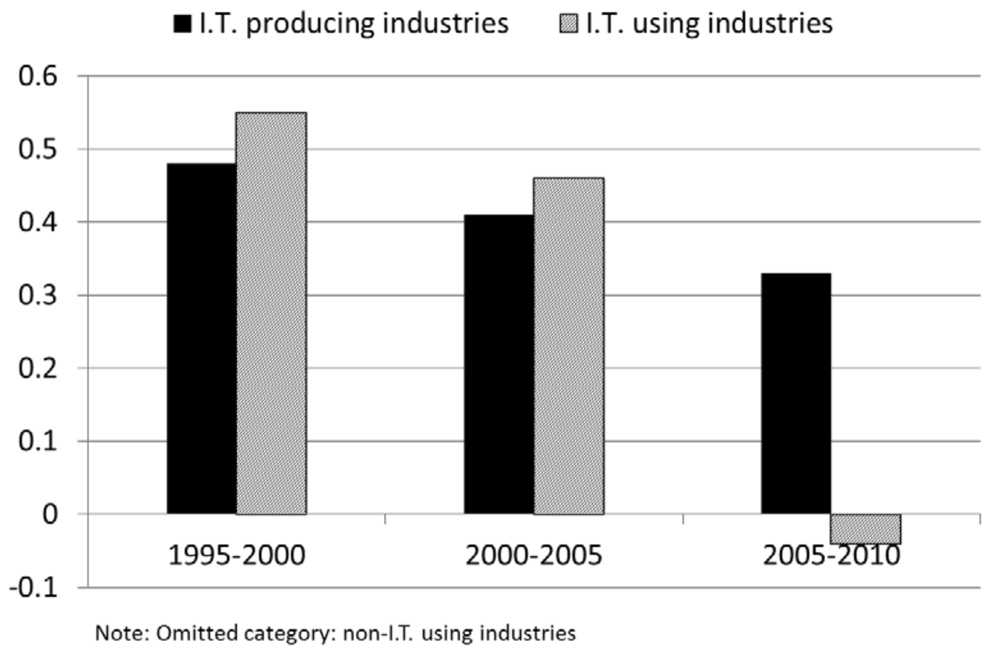


Figure 1: U.S. National Productivity Growth (Jorgenson)[68]

[68] Based on data from Dale Jorgenson, Harvard Economics, from a personal communication. The quantity on the y-axis is a measure of productivity, but is not straight-forward. A discussion of how the calculations are carried out can be found in Jorgenson, Ho & Samuels (2010).

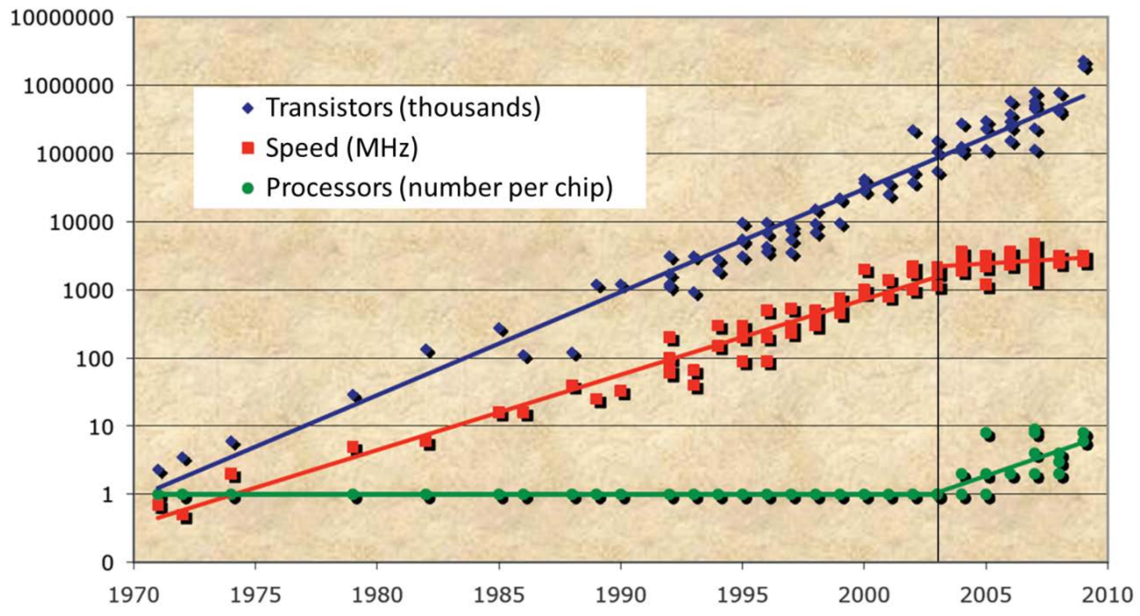


Figure 2: Moore's Law & the switch to multicore processors[69]

[69] UC Berkeley Parallel Computing Class.

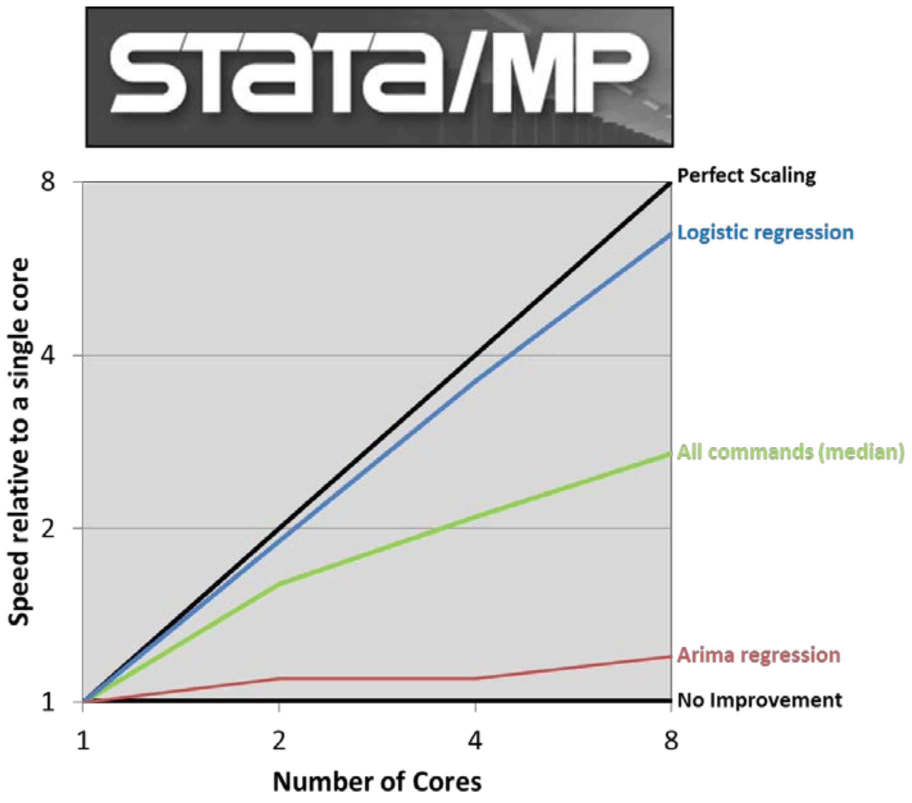


Figure 3: Performance of STATA MP[70]

[70] Stata (2012).

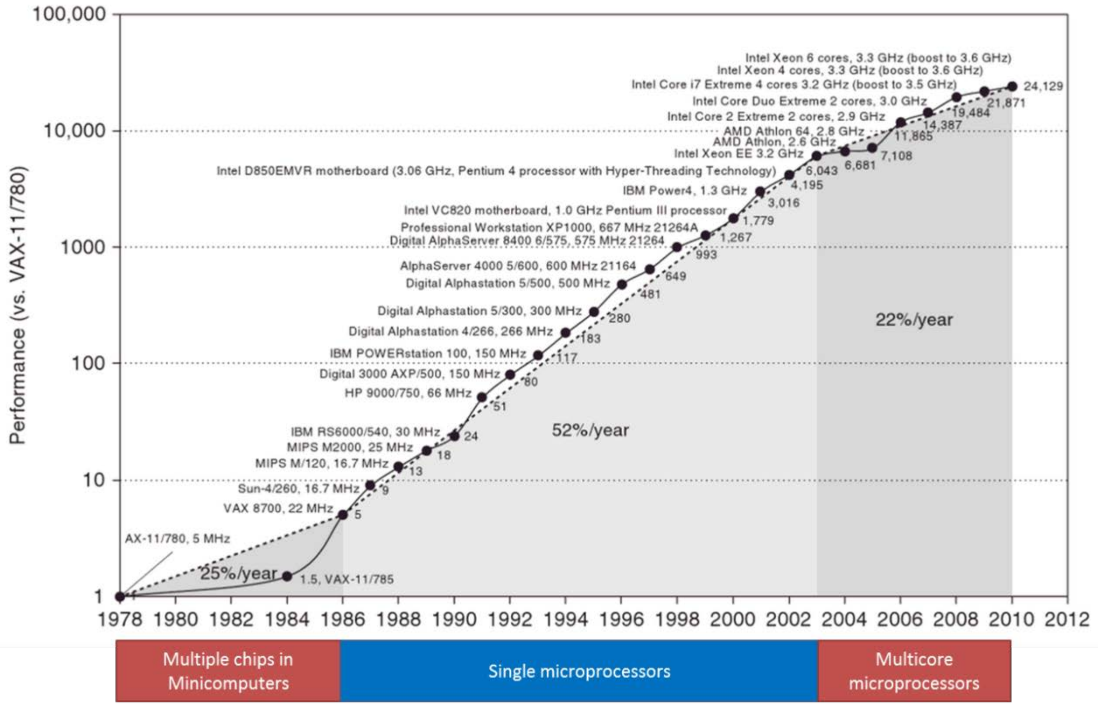


Figure 4: SPEC Benchmark

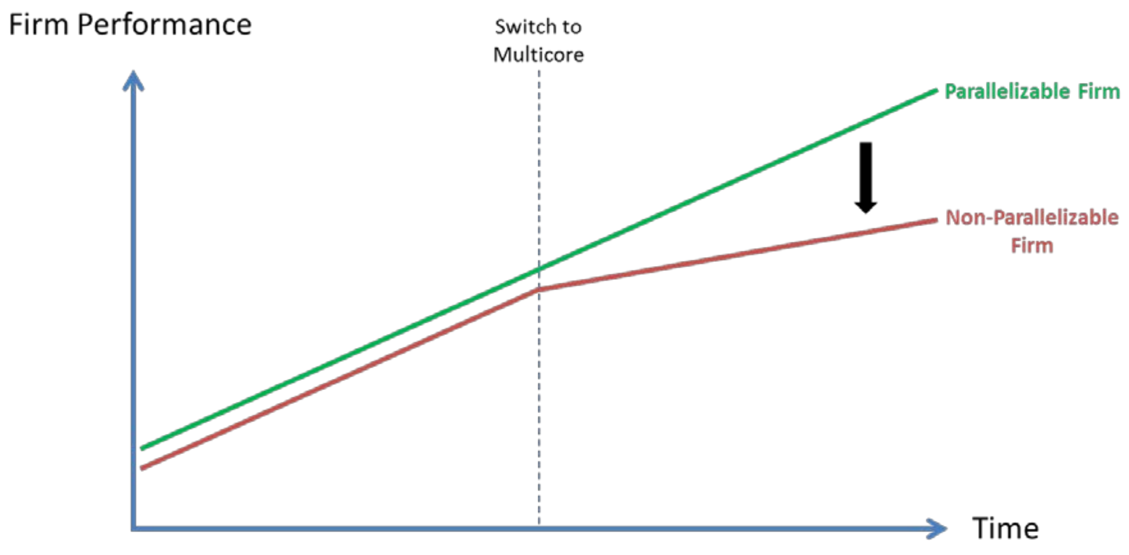


Figure 5: Hypothesis

Software Type	**Examples**
Antivirus	Norton Antivirus
Compiler	C++ compiler, Java compiler
Customer relationship mgt	SAP CRM
Database	SQL Server
Email	Microsoft Outlook
Finance & Accounting	SAP Financial Accounting
Human resources mgt	Oracle HRM
Network mgt	HP Openview
Network operating system (network os)	Windows Server
Operating system (os)	Microsoft Windows
Presentation	Microsoft Powerpoint
Resource planning system	SAP MRP
Spreadsheet	Microsoft Excel
Web browser	Internet Explorer, Firefox
Word processing	Microsoft Word

Table 1: Software Classes

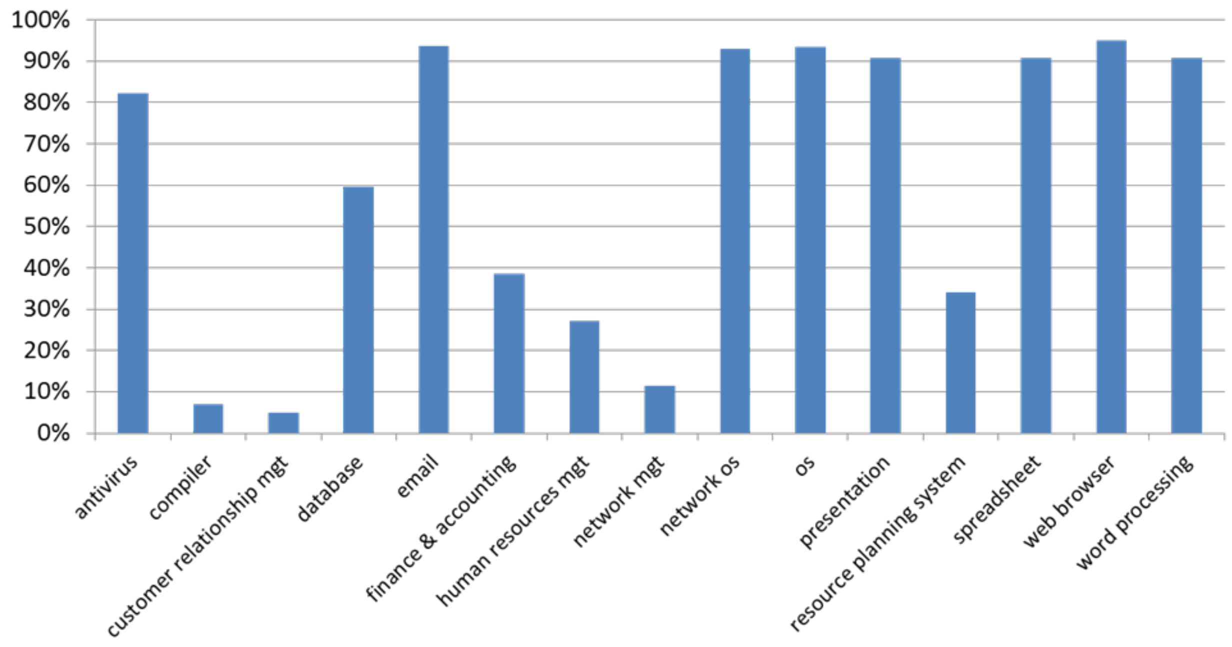


Figure 6: Share of Swedish establishments using various software types in 2003

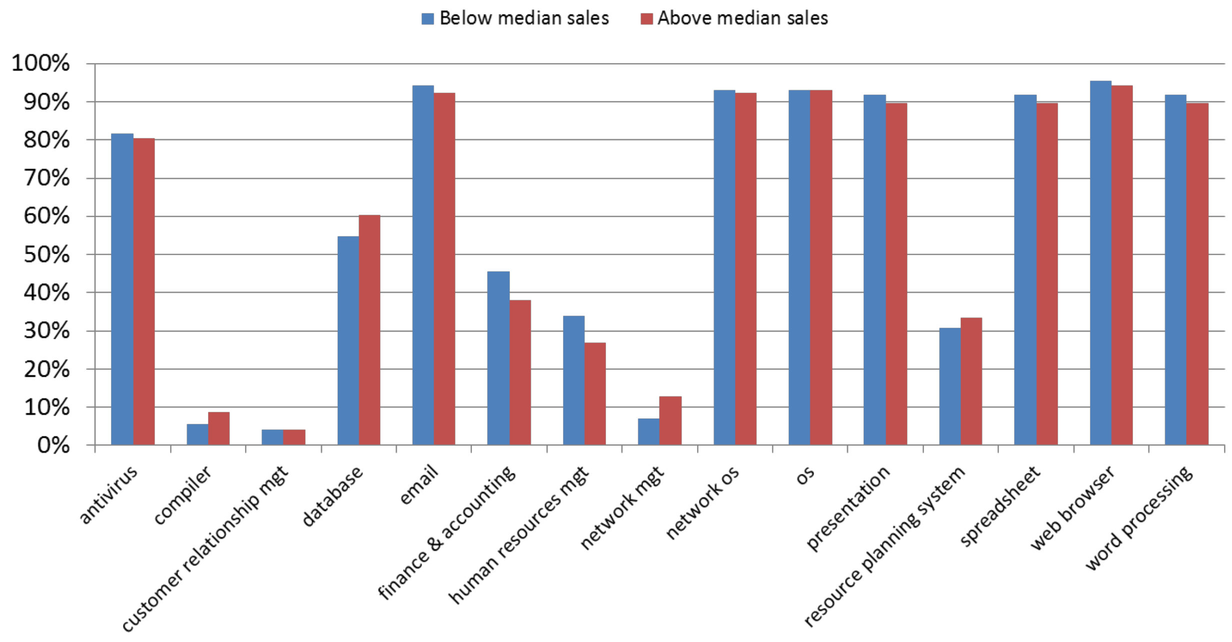


Figure 7: Share of Swedish establishments using various software types in 2003 (by sales level)

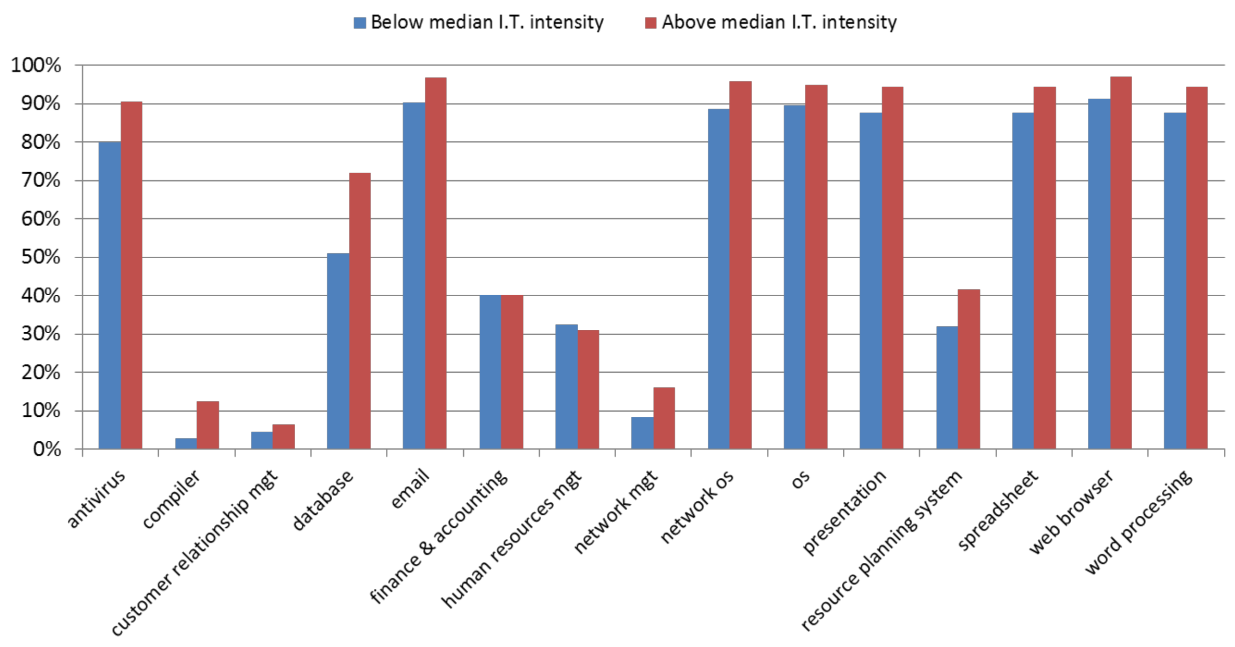


Figure 8: Share of Swedish establishments using various software types in 2003 by I.T. intensity level (PCs per employee)

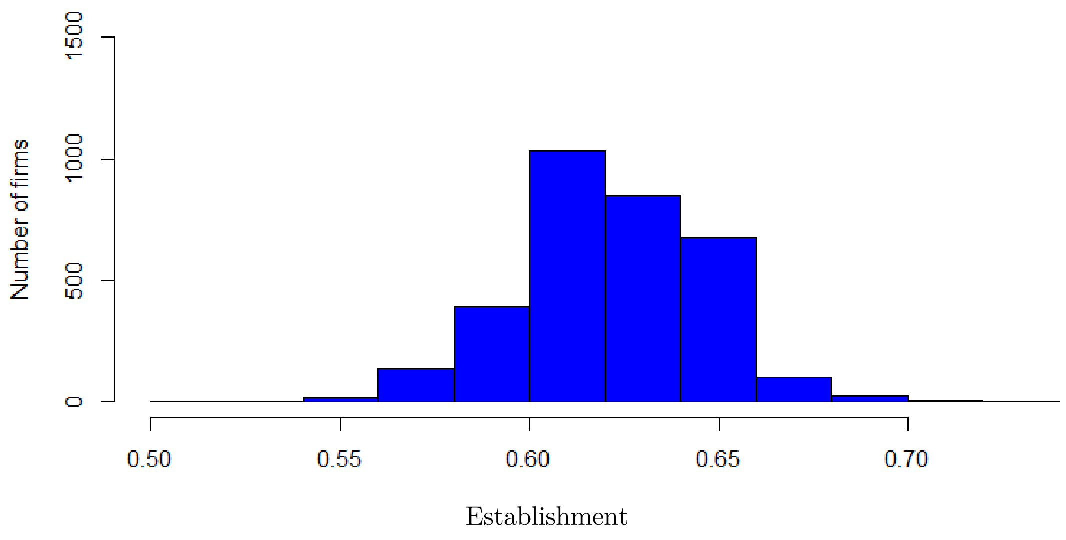


Figure 9: Establishment level parallelism

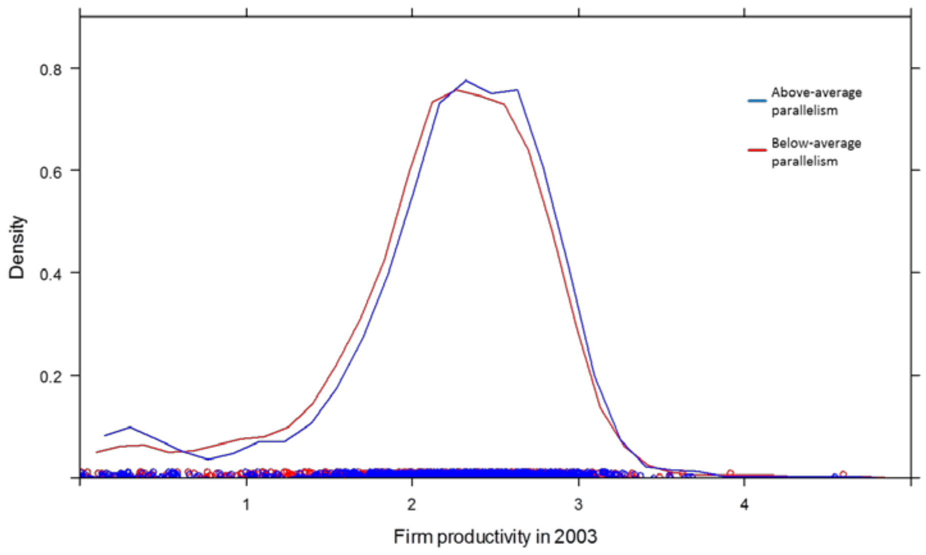


Figure 10: Distribution of Total Factor Productivity for Swedish Firms

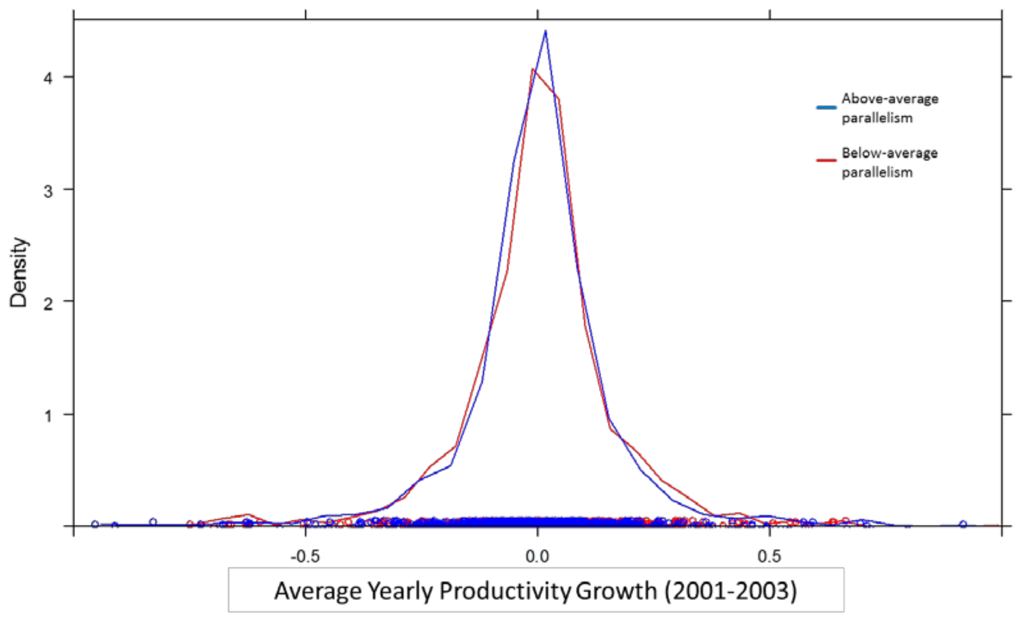
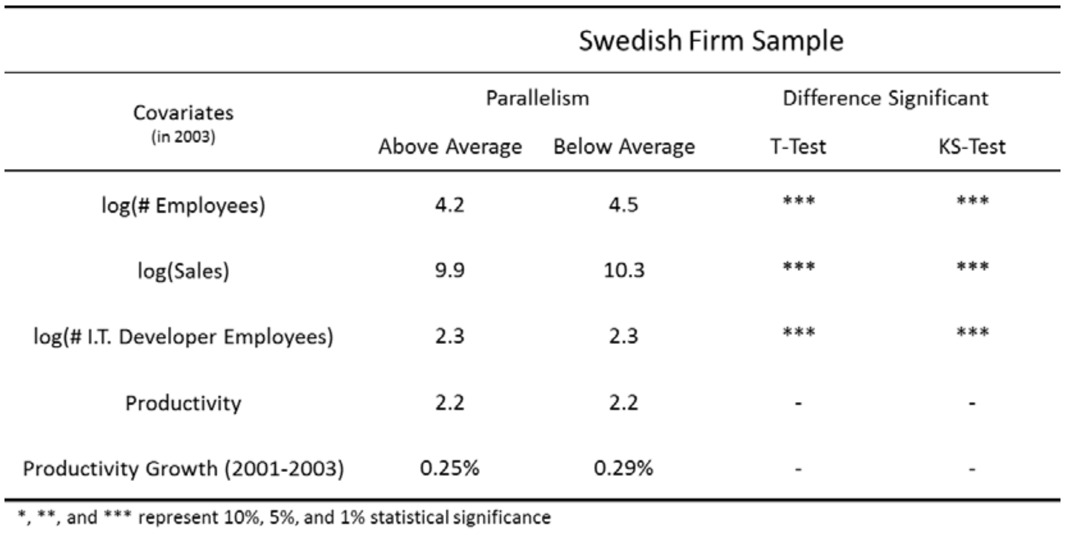


Figure 11: Pre-treatment trends in productivity

	Formula for Establishment Parallelism
Values Calculation (default)	$Parallelism_i = \frac{1}{m}\sum_{j=1}^{m} Software\ Parallelism_j$
Ranking Calculation	$Parallelism_i = \frac{1}{m}\sum_{j=1}^{m} Rank(Software\ Parallelism_j)$

Table 2: Definitions for software parallelism robustness check



	Swedish Firm Sample			
Covariates (in 2003)	Parallelism		Difference Significant	
	Above Average	Below Average	T-Test	KS-Test
log(# Employees)	4.2	4.5	***	***
log(Sales)	9.9	10.3	***	***
log(# I.T. Developer Employees)	2.3	2.3	***	***
Productivity	2.2	2.2	-	-
Productivity Growth (2001-2003)	0.25%	0.29%	-	-

*, **, and *** represent 10%, 5%, and 1% statistical significance

Table 3: Covariate Balance

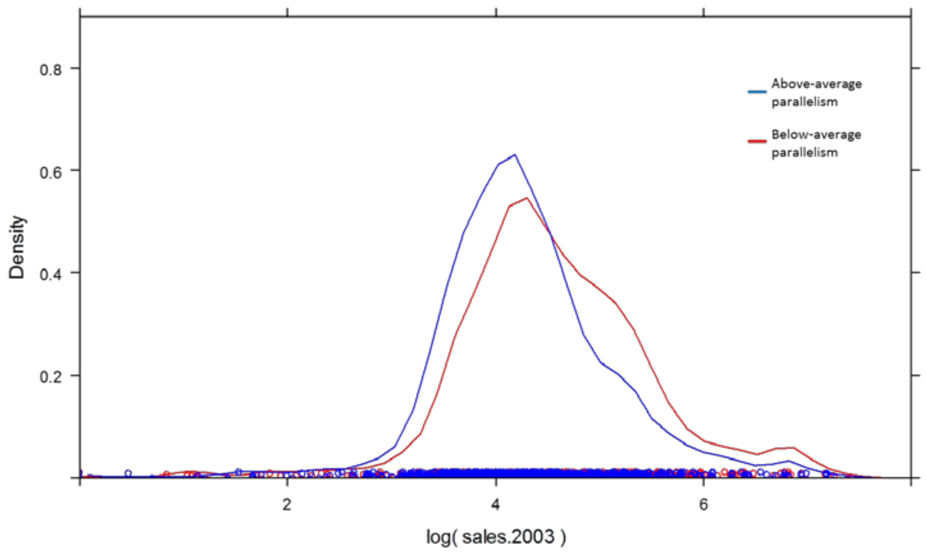


Figure 12: Distribution of Sales

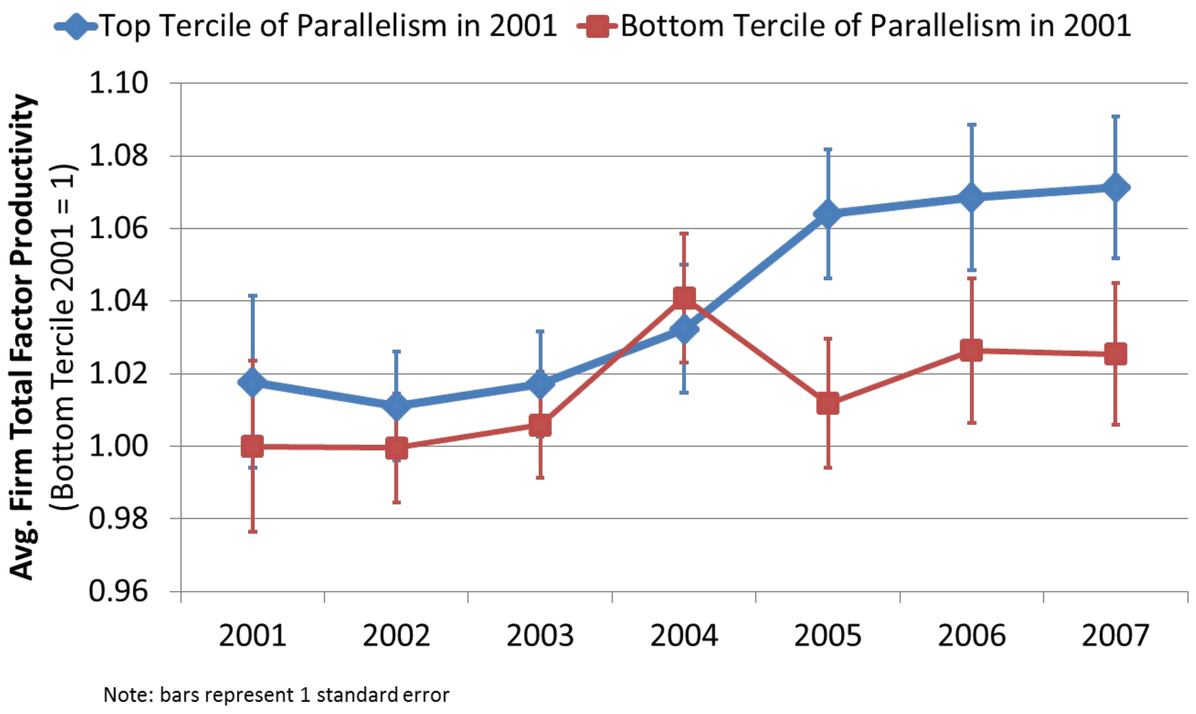


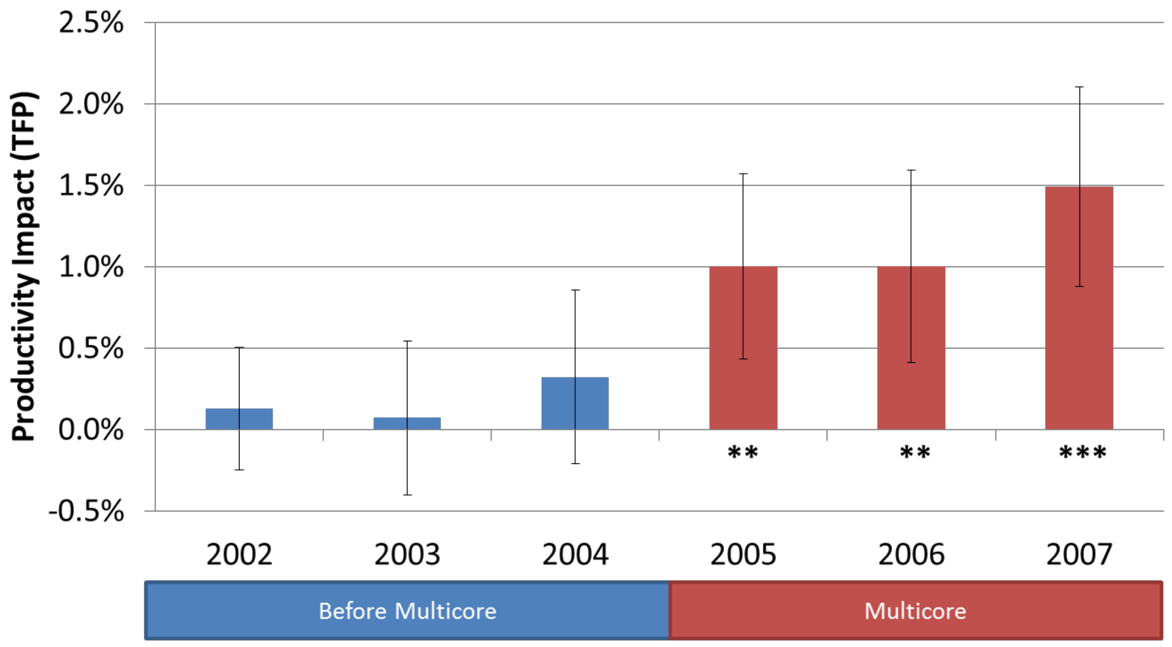
Figure 13: Overall Productivity Trends by Tercile

	Outcome: Change in Total Factor Productivity Growth (pp)		
	(I)	(II)	(III)
Parallelism Measure	REAL	ORDINAL	REAL (PLACEBO YEAR)
Parallelism	**0.75****	**0.62***	**-0.05**
(Standard Error)	**(0.35)**	**(0.35)**	**(0.26)**
[p-value]	**[0.03]**	**[0.08]**	**[0.85]**
Observations	967	967	967
R^2	0.02	0.02	-

Standard errors are clustered at the firm level; *, **, and *** represent 10%, 5%, and 1% statistical significance

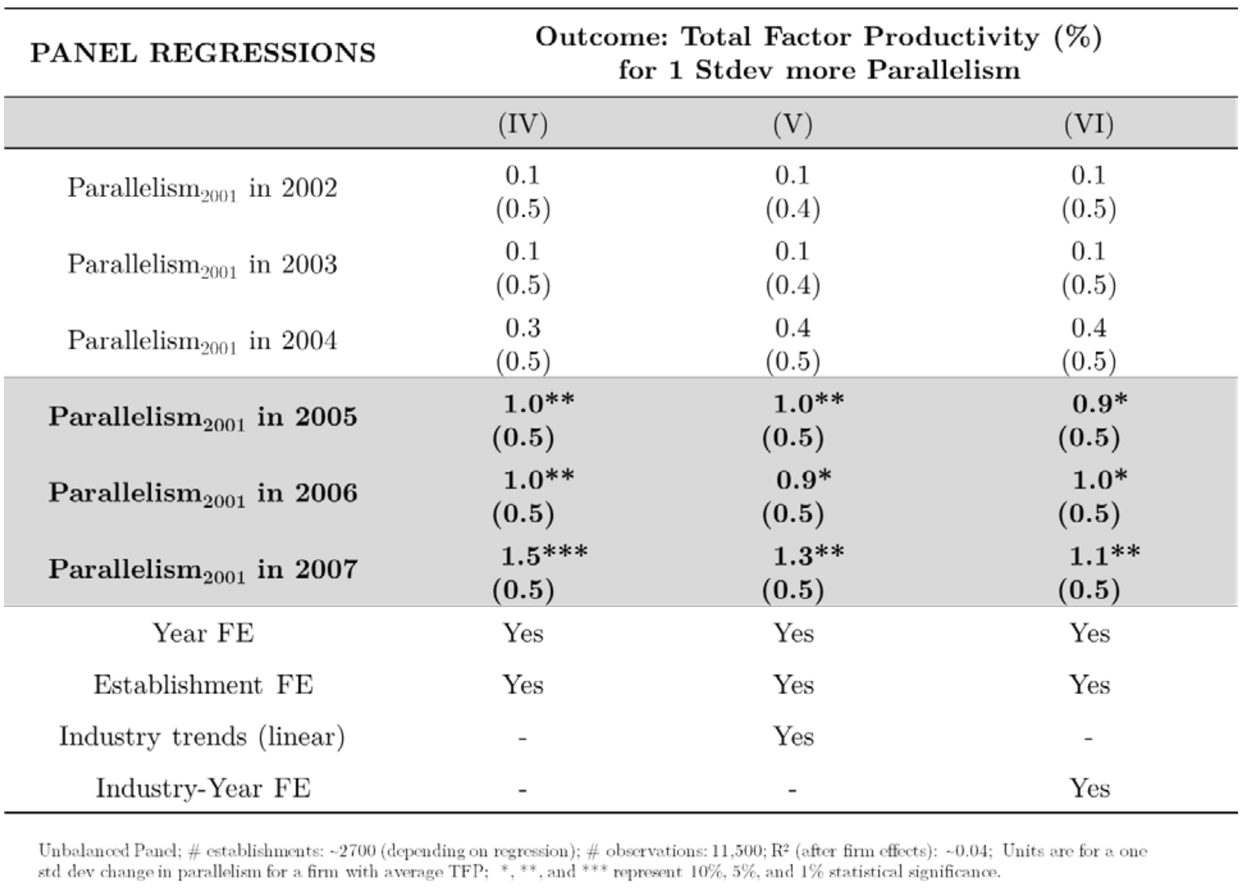
Table 4: Difference-in-differences Results[71]

[71] Coefficient estimates for the covariates are suppressed because the magnitudes are not economically significant (as might be expected from the diff-in-diff construction).



Note: Includes year and establishment fixed effects. N=2,971. Values are the coefficients from specification IV; panel is unbalanced; error bars are one standard error; *, **, and *** represent 10%, 5%, and 1% statistical significance

Figure 14: Productivity Impact of +1 standard deviation of software parallelism in 2001



PANEL REGRESSIONS	**Outcome: Total Factor Productivity (%) for 1 Stdev more Parallelism**		
	(IV)	(V)	(VI)
Parallelism$_{2001}$ in 2002	0.1 (0.5)	0.1 (0.4)	0.1 (0.5)
Parallelism$_{2001}$ in 2003	0.1 (0.5)	0.1 (0.4)	0.1 (0.5)
Parallelism$_{2001}$ in 2004	0.3 (0.5)	0.4 (0.5)	0.4 (0.5)
Parallelism$_{2001}$ in 2005	**1.0**** **(0.5)**	**1.0**** **(0.5)**	**0.9*** **(0.5)**
Parallelism$_{2001}$ in 2006	**1.0**** **(0.5)**	**0.9*** **(0.5)**	**1.0*** **(0.5)**
Parallelism$_{2001}$ in 2007	**1.5***** **(0.5)**	**1.3**** **(0.5)**	**1.1**** **(0.5)**
Year FE	Yes	Yes	Yes
Establishment FE	Yes	Yes	Yes
Industry trends (linear)	-	Yes	-
Industry-Year FE	-	-	Yes

Unbalanced Panel; # establishments: ~2700 (depending on regression); # observations: 11,500; R² (after firm effects): ~0.04; Units are for a one std dev change in parallelism for a firm with average TFP; *, **, and *** represent 10%, 5%, and 1% statistical significance.

Table 5: **Panel Regression Results**[72]

[72] In future versions of this paper these results will be clustered at the firm level. The author is in the process of reimplementing these in the R statistical language to make this possible. This is already complete for the diff-in-diffs specification, which is clustered at the firm level.

Appendix A: Heat dissipation from chips

This paper contends that the switch-over to multicore was a surprise to industry. And while the increase in heat build-up was foreseeable, it may have been foreseeable in the way that countless other technical challenges are in semiconductor manufacturing (e.g. lithography). In the face of a 30+ year history of speed-ups and an announcement from Intel that the Pentium 4 would scale to 10GHz, it seems reasonable that software providers would bet on the existing trajectory of single processor speed-ups.

The following chart, from Intel, summarizes the evolution of the power density. It also shows that a previous increase in power density (for the 8086) was overcome in the 1980s.

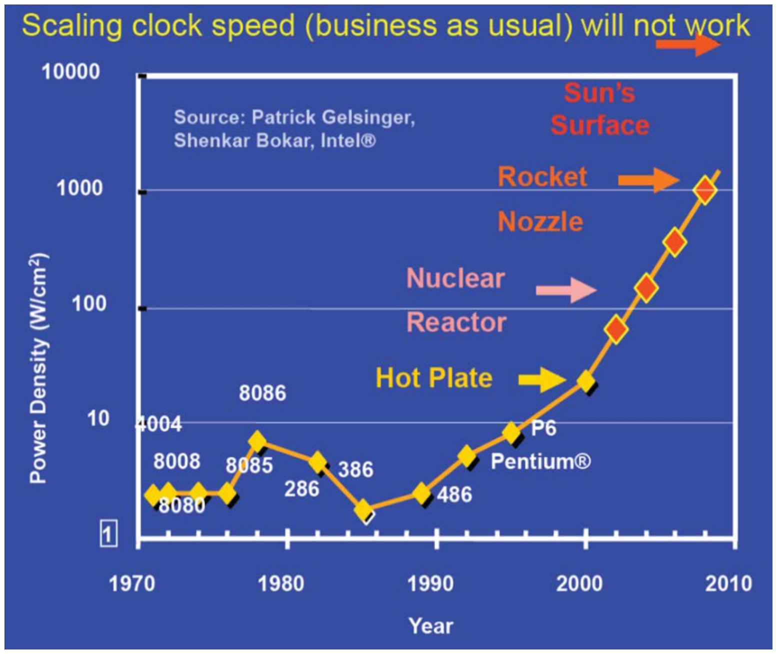


Figure 15: Heat dissipation challenge

Notice, this graph also makes clear the scale of the challenge of heat dissipation if another type of engineering solution could not be found. It is telling that even today the most ardent seekers of computer speed (mostly gamers) have only achieved speed-ups to 8.2 GHz, and only then by using liquid nitrogen to cool the processor.[73]

[73] Telovation (2010).