The Midlifekicker Microarchitecture Evaluation Metric

Stamatis Vassiliadis
Computer Engineering
TU Delft
the Netherlands
stamatis@dutepp0.et.TUDelft.nl
http://ce.et.tudelft.nl

Leonel Sousa
Electrical and Computer Engineering
IST/INESC-ID
Lisboa, Portugal
las@inesc-id.pt
http://sips.inesc-id.pt

Georgi N. Gaydadjiev
Computer Engineering
TU Delft
the Netherlands
georgi@dutepp0.et.TUDelft.nl
http://ce.et.tudelft.nl

Abstract

*We introduce the midlifekicker metric for evaluating microarchitectures mostly during the design process. We assume a microarchitecture designed at a time **T-1** and estimate if a new microarchitecture projected for time **T** has advantages over the microarchitecture designed at **T-1** and remapped on the same technology at time **T**. We consider that microarchitects minimize the product cycles per instruction (CPI) x cycle time and estimate performance based on CPI with a "soft-threshold" to include cycle time product effects. Some measurements are also reported.*

Keywords: ILP, microarchitecture, pipeline.

1 Introduction

For a given architecture [1] performance improvements of processor products are derived mainly from technology, microarchitectures, and logic design. In this paper we consider primary microarchitectures (also called a "design point") and propose a metric as a means to evaluate them. It is well understood that architectures, technologies and logic design techniques change (advance) on time making microarchitectures to have an "expiration" date. The time period spawned between the product release based on a specific microarchitecture and the product released of the same architecture based on a completely new microarchitecture it is called the life of a microarchitecture. In this paper, based on real world design experience, we address microarchitecture related questions including the following:

- Is the microarchitecture to be developed better than the previous design point?
- Can we determine if the performance due to the microarchitecture alone is improving through the years?

The approach we propose was used by one of the author's team to estimate microarchitecture performance and it is termed the *midlifekicker* metric. The name, midlifekicker, is the name given by designers to a microarchitecture remapped to a technology other than the one originally planned and before its life expiration.

This paper is organized as follows. Section 2 introduces the metric and the justification for the approach. In section 3 the considered processors and the experimental setup are presented. The CPI results experimentally obtained with the SPEC2000 benchmark programs for the various processors are also reported. Finally, section 4 concludes the paper.

2 The Midlifekicker Metric

To take advantage of rapid technology improvements it is rather common in industrial practice to remap a design before a new design is made publicly available as a product. In the context of our discussion this means that a design is moved to a different (newer) technology (platform) than the one it was originally developed for. This common practice is possible because of the following conjunctures:

- It is well known that the number of gates per chip is increasing, consequently the design will fit to at least

[1] We assume the original definition described for example in [3] here and in the rest of the presentation.

the same number of chips it was originally designed for.

- The number of I/O's per chip is increasing implying that the communication between chips from the old to the new technology remapping can be easily achieved. The bottom line is that with minor changes (e.g. if two chips design has been merged to one then buffers may be added to emulate the chip crossings) a microarchitecture can be remapped to a newer technology.

- Remapping a design is also made possible because the primitive functional blocks irrelevant from technology has remained constant. The fan-in (FI), fan-out (FO) and functions performed by digital gates in a single stage is constant. This conjuncture may not appear evident to novice designers as it is counter intuitive when it is evident that the number of gates and I/O have been improving through the years. As an evidence of the conjuncture we suggest to investigate the logic gate depth of all common building blocks such as multipliers, adders, shifters, etc. It has not changed much the reason being the following. Irrespective of technology (including different families e.g. bipolar, CMOS etc.) the FI/FO of basic blocks (single logic stage blocks) has not be changed over the years. To clarify, consider gate arrays, further consider bipolar technologies used NAND and dot-AND (also known as wired-AND) logic as basic blocks. Seldom such technologies has NANDs FI exceeding 4 (mostly actually 3) and dot-AND of 8 (mostly 4-6). This in turn can be easily translated to basic CMOS blocks, where FI of a basic block in a library seldom exceed 3x8 AND-OR-INVERT (AOI) blocks (actually mostly 3x4 AOI blocks).

Improvements can be added to the midlifekicker as long as the additions make the new scheduled deadlines. A notable example is the addition of bigger caches. The basic microarchitecture remains however unchanged. Also mistakes (fixed usually with microcode or traps to software) and some special paths are also revisited and an attempt is made to improve them.

In this paper we assume that a design is not improved and that there is no improvements fixing old critical paths. This assumption is due to two factors. First of all this information usually does not became publicly available. Second, it is assumed that new designs have to be better than the old designs thus new designs have to be at least as good as unchanged old designs. In essence we assume that the necessary condition for a new design to be better than the old one is that the new design outperforms at least the old one (without improvements) remapped in the same technology.

A final conjuncture is that the design team follows what we term as the "consumer choice premise". The premiss is described by the following:

1. Designers face constraints or limits on their choices when designing;

2. Team preferences and knowledge determine what the designers will chose;

3. Designers will maximize the benefit for their microarchitecture designs to the constrains they face.

The consumer choice model is based on the premisses that designers are competent and have preferences. They understand the prices they have to pay due to their choices and they will act as consumers in the market place (they will maximize their benefit which is assumed to be the "best of bread" microarchitecture).

What has been argued thus far is that an microarchitecture of time **T-1** can be remapped to technologies available at time **T** with some (little) effort. It was also postulated (because of the consumer choice premise) that a design team will do the best it can to develop a microarchitecture. The implication being that design teams (given the skills and personnel) will always do the best possible design they can. It can be postulated then that microarchitecture designs at time **T-1** can be projected to time **T** and compared to designs at time **T** under the conjuncture of having designed the best possible microarchitecture given the constraints. It should be noted that designers will optimize "time per instruction" that is the product: cycles per instruction (CPI) and clock cycle time (among other things) and not only the cycle time or CPI. In evaluating this product in pipelined designs (a primary concern of this paper) we note the following: pipeline delays can be divided into useful computation delay and constant overhead delay. Constant overhead delays depend on technology and designer skills (e.g. clock skew). Such constants can have an impact on pipeline delays when the pipeline structure is changed. To exemplify, consider a pipeline structure where an ALU executes in a single cycle. For a given technology let the useful computation time be **U** and the constant overhead be **C** with the ALU delay being **U+C**. By assuming a two pipeline stage ALU we do not have **(U+C)/2** cycle time but at best around **U/2 +C**. At best because seldom we can cut a design in half and by doing so have the critical path also sliced by half. Computing the constant overhead delay precisely may not be possible in advance but it can be estimated in the order of 15-20% of an single ALU cycle time. Examples in the literature exist that justify the above statement. Some examples follow.

Numerous processor designs rely on a static CMOS logic and edge triggered flip-flops. This reduces the available time for useful work within each clock cycle. Usually this is referred as flip-flop or *latch overhead*. In addition, real systems have imperfect clocks due to several (mostly analog in nature) phenomena. In this paper we will lump those effects (mainly clock skew and jitter) together and term them *clock overhead*. The latch and the clock overheads are assumed to scale with the technology, hence the same technol-

ogy independent measure as for measuring delays is used - the fanout-of-four (FO4). In [7] the sum of the latch, clock skew and jitter overhead (referred as *pipeline overhead* from now on) is found as 1.8 FO4. In absolute time this is approximately 129 ps for $.18\mu m$ process using the methodology as presented in [5]. If we relate this number to the SIA roadmap of clock frequencies assuming $.18\mu m$ CMOS technology, the total overhead is 16 % of the cycle time (1.8 out of 11.1 FO4). This is close to our expectation for the pipeline overhead (the constant overhead delay). To confirm our assumption, we refer to the numbers presented in [11]. First, the authors confirm the 125 ps pipeline overhead using a standard design flow and $.18\mu m$ technology. Next, 90 ps overhead for a 2 GHz Pentium 4 processor is reported using Intel custom design flow. In relation to the 500 ps (2GHz) clock cycle time, this would lead to 18 % overhead. Kurd et. al [8] describe a Multigzahertz clocking scheme that decreases the total skew and jitter of a 2GHz NetBurst implementation from 129ps to 51ps. Assuming such optimized design, the pipeline overhead becomes 10%.

Briefly stated, the performance metric proposed here assumes the projection of a design at time **T-1** at a technology at time **T** where time **T** is the time of a new microarchitecture design delivery. It further assumes the minimization of the product *CPI x cycle time* (time per instruction) and the factoring in this product of an estimation of constants that will effect both old and new designs for the same technology.

We measure the CPI ration between microarchitectures and we vary it with a tolerance to provide some guidelines for evaluation. We assume the number of executed instructions to be about the same for all microarchitectures.

3 Experimental setup and evaluation

A set of experiments is performed on desktop computers based on IA-32 processors using the midlifekicker described previously. Three different microarchitectures and pipeline depths (see table 2) were used in these experiments:

- Intel Pentium 4 processor with the NetBurst microarchitecture [6];
- Intel Pentium II and Celeron processors with the P6 microarchitecture [9, 4] ;
- AMD Athlon processor with the K7 microarchitecture [2].

As stated above, two different processors with the P6 microarchitecture were considered for our experiments. However, the CPI for the Pentium Celeron is not directly comparable with the other processors since it was specifically designed as a low cost processor. It does not fulfill the assumption of a well balanced design, since the internal data cache capacity has been intentionally reduced to lower the cost. Our experiments confirmed the importance of the multilevel cache system for the processor's efficiency [10]. This was done by sampling events related to memory accesses on Celeron and Pentium II processors. The Celeron results, however, are not considered as part of the discussion to follow and will not be reported in this paper.

	(a) NetBurst	(b) P6	(c) K7
Processor	*Pentium 4*	*Pentium II*	*Athlon*
Frequency	2400 MHz	350 MHz	700 MHz
Physical memory	256 MB	256 MB	128 MB
gcc version	2.96 & 3.2.2	2.96	3.2
g++ version	2.96 & 3.2.2	2.96	3.2
g77 version	3.2.2	2.96	3.2
Red Hat release	9.0	–	8.0
Linux Kernel ver.	2.4.18-27.8.0 patched with perfctr 2.4.5		

Table 1. Microarchitectures and compilers

To compare the performance of the considered microarchitectures, the CPI was experimentally measured by using the SPEC2000 integer and floating point benchmarks. Each of the analyzed microarchitecture implementations have dedicated on-chip hardware for performance monitoring, which can be used to obtain statistics about a variety of performance events [12]. In addition to the number of cycles and executed instructions, more data is collected by counting several events, such as: cache misses, branch miss-prediction and floating point operations. The *Performance Programming Interface* (PAPI) tool (*version 2.3.4.3*) [1] was used to configure these event detectors and counters and to collect the data.

The work stations that were used in the experiments ran Red Hat (RH) Linux operating system. The benchmarks were compiled with the *gcc*, *g++* and *g77* compilers from GNU, using the "-O3 " optimization level. The main characteristics of computers and compilers can be found in table 1. The benchmarks were applied with the standard "test" input sets from the SPEC distribution. Several independent runs were made for each benchmark in each machine. The complete SPECint suite was used and all the SPECfp benchmarks, except for the Fortran 90, were also applied.

Table 2 presents the number of pipeline stages required to perform some relevant operations on the considered microarchitectures. The frequency ratio of about 1.5 referred in [6] for NetBurst and P6 can be used as a starting point for relating the critical paths. This is exactly the achieved value if the integer ALU is considered to be the critical path and the clock overhead, due to latch delay, clock skew and jitter, is ignored. Since the critical path is unknown, similar approach can be applied in respect to the floating point functional units. The following frequency ratios can be assumed for the NetBurst based processor by using the P6 and K7 microarchitectures as reference:

$$1.4(FPM) < 1.5(ALU) < 1.67(FPA) \quad \text{for P6}$$
$$1.25(FPA) < 1.5(ALU) < 1.75(FPM) \quad \text{for K7}$$

where, FPA and FPM are floating point adder and floating point multiplier pipeline stage ratios (see table 2). For simplicity, we base our considerations around the 1.5 frequency ratio. It is interesting to notice that this central ratio does not only directly correspond to the ALU but it is also the average value for the FPA and FPM. In addition, we assume that K7 and P6 clock architectures and ALU designs perform similarly. We understand that such assumption is not in favor of the aggressive AMD design, e.g. K6 uses Partovi pulsed latches, however it is considered as a good approximation for our study.

Based on the clock overhead percentages as introduced earlier, and in case it is assumed that all microarchitectures are implemented on the same silicon technology, e.g. $.18\mu m$, the frequency of the NetBurst will be somewhere between $f_l = 1.275$ and $f_h = 1.725$ times greater than P6 and K7. This corresponds to a tolerance of approximately $\pm 15\%$ around the average value of 1.5 times. In case one of the clock system architectures is highly optimized this difference may become $\pm 20\%$, e.g. $f_l = 1.2$ and $f_h = 1.8$ respectively. Those four frequency ratios are used to setup the upper and lower bounds for the relative CPI from which it can be clearly stated the performance of one microarchitecture is better than the other:

$$\begin{cases} \frac{CPI_{NetBurst}}{CPI_{P6,K7}} \leq 1.2 & \text{NetBurst performs better} \\ 1.2 < \frac{CPI_{NetBurst}}{CPI_{P6,K7}} \leq 1.275 & \text{NetBurst probably better} \\ 1.275 < \frac{CPI_{NetBurst}}{CPI_{P6,K7}} < 1.725 & \text{none is better} \\ 1.725 \leq \frac{CPI_{NetBurst}}{CPI_{P6,K7}} < 1.8 & \text{P6, K7 probably better} \\ \frac{CPI_{NetBurst}}{CPI_{P6,K7}} \geq 1.8 & \text{P6, K7 perform better} \end{cases} \quad (1)$$

Microarchitecture	NetBurst	P6	K7
integer ALU	1.5	1	1
FP adder (FPA)	5	3	4
FP multiplier (FPM)	7	5	4
Fetch → Retirement	20	13 [a]/14 [b]	11

[a] the reservation station write and dispatch can usually be performed in only 2 (vs 3) cycles.

[b] often it is said that P6 has 10 pipeline stages, since after 10 cycles the results are available for other instructions to use (before the instruction that generated them retires).

Table 2. Number of pipeline stages.

The experimentally obtained CPI for the integer benchmarks are presented in tables 3 and 4 and illustrated in figures 1 to 3. These tables provide different results for the NetBurst due to the two different compiler versions used on the P6 and K7 based machines. In this way, similar compiler versions are used for each comparison.

For the SPECint benchmarks, the NetBurst/P6 CPI ratio is equal or bellow 1.2 for five of the benchmarks. For the remaining seven benchmarks (more than half of the total number), the relative CPI is between 1.275 and 1.725. These results show that by using SPECint benchmarks it can not be clearly stated which of the microarchitectures exhibits a better performance. If all the uncertainty results are considered on the side of the older P6 microarchitecture, the latter exhibits better performance than the NetBurst for approximately 60% of the benchmarks.

	NetBurst	P6	NetBurst/P6
Processor	*Pentium 4*	*Pentium II*	–
Bzip2	2.37	1.4	1.69
Crafty	1.52	1.35	1.13
Eon	1.85	1.26	1.46
Gap	1.82	1.32	1.38
Gcc	1.75	1.74	1
Gzip	1.3	1.2	1.09
Mcf	3.8	2.3	1.65
Parser	1.47	1.31	1.12
Perlbmk	2.27	1.55	1.47
Twolf	2.31	1.63	1.41
Vortex	1.35	1.23	1.1
Vpr	1.93	1.4	1.38

Table 3. SPECint CPI results (gcc/g++ 2.96)

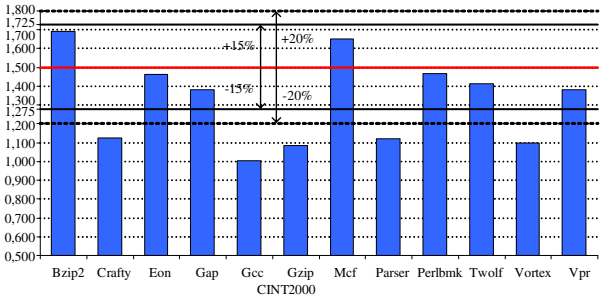

Figure 1. NetBurst/P6 SPECint CPI ratio

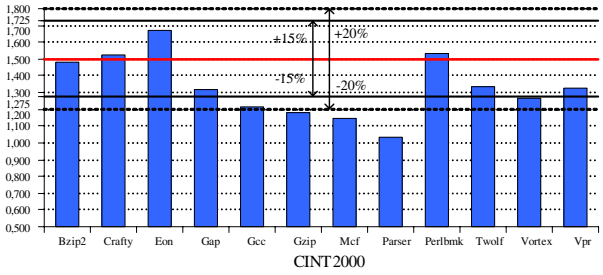

Figure 2. NetBurst/K7 SPECint CPI ratio

The CPI results for the NetBurst and K7 microarchitectures are presented in table 4 and illustrated in fig. 2. In this case, the CPI of the NetBurst is equal or bellow 1.2 for only three benchmarks. Two other benchmarks are in the uncertain region between 1.2 and 1.275. For the remaining seven benchmarks, the relative CPI is in the "no improvement" region (1.275 and 1.725). Moreover it should be noticed that two out of the three CPI results bellow 1.275 are close to the lower limit for the CPI relations defined in expression 1.

	NetBurst	K7	NetBurst/K7
Processor	*Pentium 4*	*Athlon*	–
Bzip2	1.78	1.2	1.49
Crafty	1.48	0.97	1.52
Eon	1.9	1.14	1.67
Gap	1.82	1.38	1.31
Gcc	1.75	1.44	1.22
Gzip	1.33	1.13	1.18
Mcf	3.76	3.28	1.15
Parser	1.38	1.33	1.04
Perlbmk	2.2	1.43	1.53
Twolf	2.13	1.6	1.33
Vortex	1.41	1.12	1.26
Vpr	1.73	1.3	1.33

Table 4. SPECint CPI results (gcc/g++ 3.2.X)

These results where obtained using a slightly older compiler version on the K7 based computer (see table 1).

In fig. 3, the CPI of the P6 and K7 microarchitectures are compared. The CPI values were normalized by using the results for the two compiler versions for the NetBurst. It can be seen that for exactly half of the SPECint programs the CPI of the K7 is smaller than the CPI of the P6. On average, the CPI of the K7 is slightly lower than the CPI of the P6 (please mind the K7 cycle time assumption in this paper).

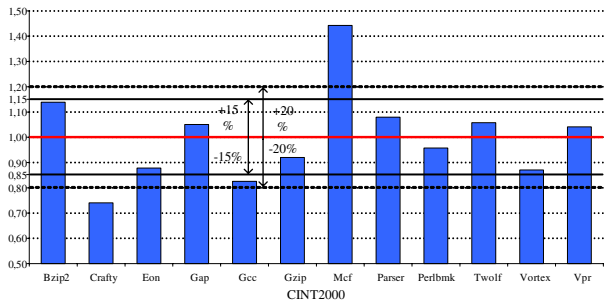

Figure 3. K7/P6 SPECint CPI ratios

The experimentally obtained CPI for the 3 microarchitectures with the floating point benchmarks (SPECfp) are presented in tables 5 and 6 and illustrated in figures 4 and 6. As for the integer benchmarks, the two versions of the C compiler were used on the NetBurst powered computer. Unfortunately, one unique Fortran77 compiler version is not available for both the NetBurst and P6. This puts the P6 microarchitecture in a disadvantageous position. For the *sixtrack* benchmark compiled with the g77 3.2.X versions, the output data does not conform with the expected output, and therefore the CPI could not be considered in the comparison.

The ratio between the CPI obtained with the NetBurst microarchitecture and the CPI obtained with the P6 in two of the nine considered benchmarks is much above 1.8 times.

	NetBurst	P6	NetBurst/P6
Processor	*Pentium 4*	*Pentium II*	–
wupwise [a]	0.99	0.97	1.02
swim [a]	4.5	2.84	1.59
mgrid [a]	1.65	1.99	0.83
applu [a]	1.79	1.54	1.16
apsi [a]	1.84	1.5	1.22
art	9.77	4.36	2.24
equake	2.1	1.8	1.17
ammp	5.39	2.44	2.21
mesa	1.86	1.35	1.37

[a]Fortran- g77 2.96 on Pentium II and g77 3.2.2 on Pentium 4

Table 5. SPECfp CPI results (NetBurst and P6)

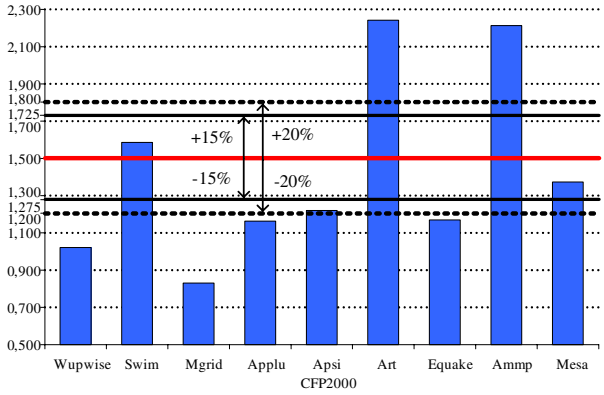

Figure 4. NetBurst/P6 SPECfp CPI ratios

For two other benchmark programs, this relationship is between 1.275 and 1.725 (none is better), while *apsi* is in the "NetBurst can be made better" region (1.2 - 1.275). The remaining four benchmark programs lead to values bellow 1.2 and for those it can be clearly stated that NetBurst performs better than P6. These results are more favorable to the NetBurst microarchitecture than those obtained with SPECint programs. However, it should be mentioned that four values from the set of results below 1.2 have been obtained with Fortran programs and a more recent version of the Fortran compiler was used on the NetBurst powered computer (see table 5). In case only the *C* benchmark programs are considered, just one is bellow 1.2, one other is in the non-defined area and the remaining two benchmark programs are above 1.8.

In what concerns the K7 microarchitecture, the NetBurst to K7 CPI ratio is greater than 1.725 (K7 may be better)for one of the nine considered benchmarks. The majority of the benchmarks (five) produce values in the uncertainty area. The CPI ratio for the remaining 3 benchmarks is bellow the 1.2 threshold value. It can also be observed in figures 4 and 5 that the CPI ratio is much more balanced over the different benchmarks for the K7 than for the P6. By com-

	NetBurst	K7	NetBurst/K7
Processor	*Pentium 4*	*Athlon*	–
wupwise [a]	0.99	0.92	1.07
swim [a]	4.5	3.06	1.47
mgrid [a]	1.65	2.06	0.8
applu [a]	1.79	1.32	1.36
apsi [a]	1.84	1.3	1.41
art	11.3	6.98	1.62
equake	2.19	2.22	0.99
ammp	4.9	2.77	1.77
mesa	1.54	1.13	1.37

[a]Fortran- g77 3.2.2 on Pentium 4 and g77 3.2 on Athlon

Table 6. SPECfp CPI results (NetBurst and K7)

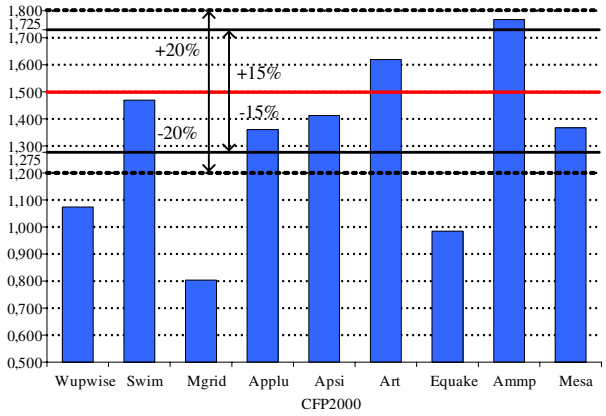

Figure 5. NetBurst/K7 SPECfp CPI ratios

puting the average and the standard deviation values (tables 5 and 6) it can be concluded that the average CPI ratio is slightly smaller for the K7 (1.42 against 1.32) and the standard deviation is considerably lower for this microarchitecture (0.1) compared to (0.17) for P6. This last difference is probably due to the different number of cycles required for computing the two basic floating point operations in P6 (see table 2).

These CPI results show that different but balanced design points were achieved for all analyzed processors. None of the processors shows a clear superiority in performance against the others. Therefore, the claimed improvement on performance for the most recent processors does not arise from a real step forward in the pipeline structure/microarchitecture but mainly due to advances in the technology.

4 Conclusions

The paper has introduced the midlifekicker metric to estimate the performance of microarchitectures. It has been argued that designs at time period **T-1** can be projected to

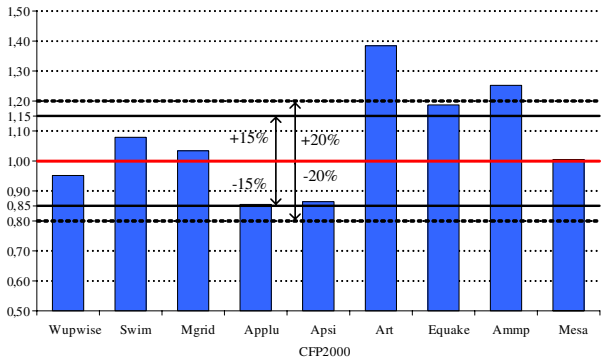

Figure 6. P6/K7 SPECfp CPI ratios

time **T** and that the estimation of the product *CPI x cycle time* can be established as a CPI soft threshold taking into account constant and useful computation delays. Numerous experiments have also been presented. We note that the midlifekicker is an estimation methodology thus it provides approximations (as most estimators) not certainties.

References

[1] Papi. http://icl.cs.utk.edu/projects/papi.

[2] AMD. *Athlon Processor x86 Code Optimization: Processor Microarchitecture (App A) and Pipeline and Execution Unit Resource Overview (App. B)*, September 2000.

[3] G. Blaauw and F. Brooks Jr. *Computer Architecture*. Addison-Wesley, One Jacob Way, 1997.

[4] R. Colwell and R. Steck. A 0.6 micron bicmos processor with dynamic execution. In *Proc. IEEE Int. Solid-state Circuits Conf.*, 1994.

[5] D. Haris. *Skew-Tolerant Circuit Design*. Morgan Kaufman, 340 Pine Street, SF, 2001.

[6] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyler, and P. Roussel. The microarchitecture of the pentium 4 processor. *Intel Technology Journal*, Q1 2001.

[7] M. S. Hrishikesh, N. P. Jouppi, K. I. Farkas, D. Burger, S. W. Keckler, and P. Shivakumar. The optimum logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *Proc. of the 29th Annual International Symposium on Computer Architecture*, pages 14–24, 2002.

[8] N. Kurd, J. Barkatullah, R. Dizon, T. Fletcher, and P. Madland. Multi-ghz clocking scheme for intel(r) pentium(r) 4 microprocessor. In *Proc. of the International Solid-state Circuits Conference*, pages 404–405, 2001.

[9] J. Shen and M. Lipasti. *Modern Processor Design: Fundamentals of Superscalar Processors*. McGraw-Hill, 1st edition, 2003.

[10] L. Sousa, S. Vassiliadis, and G. Gaydadjiev. Experimental Characterization of IA-32 Microarchitectures. Technical report, Computer Engineering, TU Delft, 2004.

[11] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *Proc. of the 29th Annual International Symposium on Computer Architecture*, pages 25–34, 2002.

[12] B. Sprunt. The basis of performance-monitoring hardware. *IEEE Micro*, pages 64–71, July 2002.