Energy Efficient Near-threshold Chip Multi-processing

Bo Zhai, Ronald G. Dreslinski, David Blaauw, Trevor Mudge, Dennis Sylvester
{bzhai, rdreslin, blaauw, tnm, dennis}@eecs.umich.edu, University of Michigan, Ann Arbor, MI

ABSTRACT

Subthreshold circuit design has become a popular approach for building energy efficient digital circuits. One drawback is performance degradation due to the exponentially reduced driving current. This had limited subthreshold circuits to relatively low performance applications such as sensor networks. To retain the excellent energy efficiency while reducing performance loss, we propose to apply subthreshold and near-threshold techniques to chip multi-processors. We show that an architecture where several slower cores are clustered together with a shared faster L1 cache is optimal for energy efficiency, because processor cores and memory operate best at different supply and threshold voltages. In particular, SPLASH2 benchmarks show about a 53% energy improvement over the traditional CMP approach (about 70% over a single core machine).

Categories and Subject Descriptors

C.1.4 [Parallel Architectures]: Mobile Processors

General Terms Design, Reliability, Performance

Keywords energy efficient, CMP, near-threshold, subthreshold

1 Introduction

Due to its high energy efficiency subthreshold design has been recently proposed for use in low performance applications. For example, Zhai et al. [1] proposed a low-end sensor network processor and Wang and Chandrakasan [2] proposed subthreshold use for an FFT engine. However, the drawback of subthreshold design is that the increased energy efficiency comes at the cost of performance loss. Thus previous papers have only targeted low end applications. Our intention is to use the energy efficiency of subthreshold designs to target parallelizable embedded applications requiring higher performance, but where battery life is important.

Previous work by Zhai et al. [3] and Calhoun [4] has shown that for a CMOS digital circuit there exists an energy optimal supply voltage (V_{min}) below which energy consumption increases because of exponentially increased propagation delay and leakage energy. V_{min} usually occurs in the subthreshold region. However, these papers did not address how to choose the threshold voltage (V_{th}) to further improve energy savings. In this work we show that varying V_{th} changes the energy optimal voltage and that greater savings can be achieved by controlling the V_{th} properly.

Zhai et al. [1] shows that, due to the different activity factors and leakage rates for memory cells and logic, the V_{min} of the processor and memory are usually different. As a result, operating the entire system under a single V_{dd} leads to sub-optimal energy efficiency. Therefore we propose using a chip design that has two separate V_{dd}/ V_{th} domains for the core and memory.

However only changing the V_{th} does not solve the issue of performance loss from aggressive voltage scaling. So we propose to employ multiple cores in the subthreshold or the near-threshold regime. We explore separate control of V_{dd} and V_{th} for the core and the memory, where memory is allowed to operate at speeds both slower and faster than the core it is attached to. In particular, we create a cluster that has several slower cores connected to the same faster cache.

Clusters have advantages compared to the traditional CMP approach. Applications that have high communication to computation ratios can share data with other cores in the cluster without the coherence overhead of communicating over the bus that connects the different L1's. However, the cores are now contending for the same cache space, which may result in effectively smaller cache sizes due to conflict and capacity misses generated by the other cores within the cluster. L1 sharing also requires a bus between the cores and the L1. More cores within a cluster increases the size and capacitance of this bus. We investigated all the major factors that could affect the system energy efficiency, such as L1 cache size, the cluster size, total number of clusters and the selection of V_{dd} and V_{th} within a cluster. Architectural simulation together with circuit-level modeling shows that for most of the SPLASH2 [5] benchmarks the energy optimal point is 2 cores in a cluster. This configuration provides about a 53% increase in energy efficiency over the traditional CMP design.

The paper is organized as follows: Section 2 explains the advantage of the subthreshold design and the impact of threshold voltage selection on energy efficiency. We then introduce the proposed architecture in Section 3 and provide a theoretical analysis. Section 4 and Section 5 detail the energy modeling of a complete system and the architectural simulation results of the SPLASH2 benchmarks. Finally, Section 6 presents concluding remarks. All technology numbers are from an industrial 0.13um CMOS technology.

2 Energy Efficient Subthreshold Design

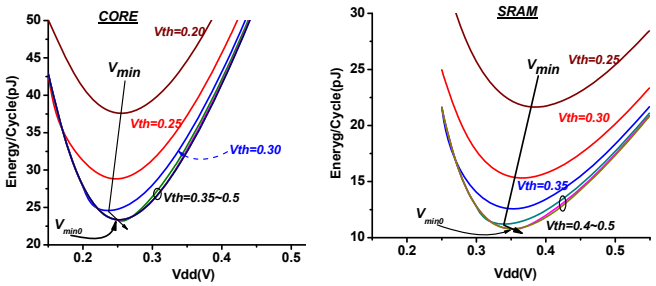
Zhai et al. [3] identified the supply voltage at which the minimum energy is achieved, V_{min}. They also showed that minimum energy consumption at that voltage, E_{min}, is independent of the threshold voltage, $V_{th,}$ if V_{min} is below V_{th}. In other words, we can increase the speed of a subthreshold circuit by reducing V_{th} while maintaining the same energy consumption. However, V_{min} varies with the activity rate of the circuit. Less active components such as an SRAM/cache usually have a higher V_{min} than a core. When the V_{min} moves close to V_{th}, E_{min} and V_{min} shift higher.

To analyze this interaction, we write out the energy per switching for a circuit block as (short-circuit power is lumped into switching power as described in [6]):

$$E_{total} = E_{act} + E_{leak} = CV_{dd}^2 + I_{leak} \cdot V_{dd} \cdot \frac{CV_{dd}}{I_{on}} \qquad \text{(EQ 1)}$$

where E_{total}, E_{act}, and E_{leak} are the total, active, and leakage energy consumption respectively; C is the load capacitance; and I_{on} is the on-current of the driving gate. Among the two components of energy consumption, E_{act} always scales quadratically with V_{dd} and is independent of V_{th}. What may change with V_{th} is the leakage energy component E_{leak}. If V_{th} is high enough, E_{leak} holds constant regardless of V_{th} because I_{on} has the same scaling characteristics with V_{th} as I_{leak}, and both V_{th} terms cancel out:

ISLPED'07, August 27-29, 2007, Portland, Oregon, USA.



(a) (b)

Figure 1. Energy-V_{dd} for core and SRAM with different V_{th}s

$$E_{total,sub} = CV_{dd}^{2} + I_{s0} \cdot e^{-\frac{Vth}{mV_T}} \cdot V_{dd} \cdot \frac{CV_{dd}}{I_{s0} \cdot e^{-\frac{V_{dd}-Vth}{mV_T}}} \quad \text{(EQ 2)}$$

Therefore V_{min} stays unchanged. However, as V_{th} moves closer to V_{min}, the leakage energy component E_{leak} increases because the circuit moves out of the subthreshold regime and into the near V_{th} transient region. In this region I_{leak} increases more than the delay reduces. In the energy equation EQ2, this implies that the V_{th} terms can no longer be cancelled out. If we assume an α-power law [7] for near-threshold on-current, E_{total} can be written as:

$$E_{total,super} = CV_{dd}^{2} + I_{s0} \cdot e^{-\frac{Vth}{mV_T}} \cdot V_{dd} \cdot \frac{CV_{dd}}{I_{on0} \cdot (V_{dd}-Vth)^{\alpha}} \quad \text{(EQ 3)}$$

We simulated an inverter chain by applying a pulse input using SPICE and fitted models for leakage current and delay with V_{dd} and V_{th}. V_{th} is varied by changing the V_{th0} parameter in the SPICE model. Then we applied the fitted model for a commercial processor core [8] and plotted the energy consumption per clock cycle with V_{dd} for different V_{th}s in Figure 1(a). The total energy shows a minimum as expected. More importantly, the energy consumption is insensitive to V_{th} values that are larger than 0.35V. In this region, V_{min} stays constant, which we refer to as V_{min0}, the *inherent optimal energy voltage*. In this case V_{min0} is ~0.25V. When V_{th} drops below 0.35V, the overall energy consumption increases considerably. As a result, V_{min} shifts with the change in V_{th}. Interestingly V_{min} first decreases a bit and then it rises. SRAM/cache usually has a higher V_{min} than the processor core due to the lower activity rate and higher relative leakage power. For comparison we have plotted the energy consumption with V_{dd} under different V_{th}s for a SRAM in Figure 1(b). It is evident that SRAM has a higher V_{min}/V_{min0}. It also exhibits dependency on V_{th} like the processor core.

It follows that operating the processor core and the memory at their individual optimal V_{dd} and V_{th} will result in the best overall energy efficiency. As noted above, the speed depends exponentially on V_{th} and we could gain "free" performance without additional energy consumption if V_{min} is lower than V_{th}. Assuming a nominal speed of 233MHz at a supply voltage of 1.2V and a threshold voltage of 0.4V, we solved for the optimal V_{min} and V_{th} at different performance targets and plotted the results in Figure 2. Memory, the leakier component, runs at a higher V_{th} and V_{dd} than the core in order to balance the active and leakage energy. For both components, the optimal V_{th} increases almost linearly with the log of the decreasing target frequency. However the optimal V_{dd} reduces dramatically at higher performance targets and then stabilizes near 10MHz. For the best

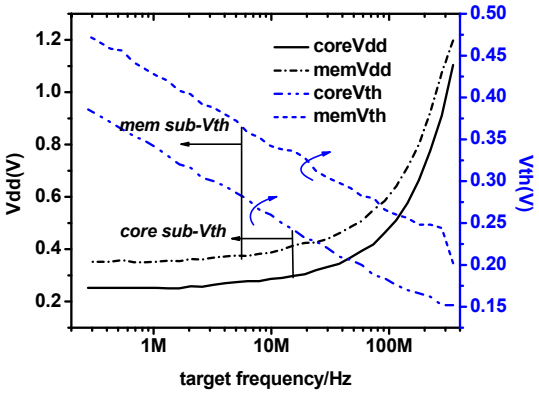


Figure 2. V_{dd}/V_{th} assignment for different frequency targets

performance-energy trade-off, we would like to operate them slightly above threshold voltage to avoid the performance degradation of scaling deeper into the subthreshold region.

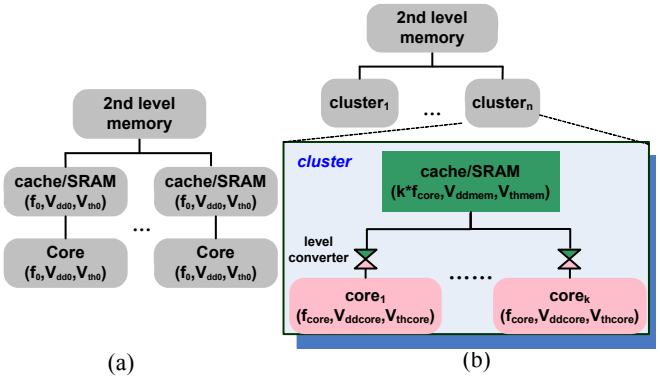
Although it is straightforward to build a system based on the optimization results in Figure 2, the flexibility of tuning V_{th} and V_{dd} separately for the core and memory provides even more savings when combined with a novel multi-processing (MP) structure.

3 Proposed Near-threshold Architecture

3.1 New Memory Architecture

Traditional computer design is usually limited by the SRAM/cache speed leading to traditional designs in which the caches run at the same speed or slower speeds than the core. However, the picture changes in the subthreshold domain because we can now take advantage of the V_{dd} and V_{th} knobs, as discussed in Section 2, and have the memory module operate at a speed faster than the core. In the subthreshold regime, a 100mV boost in V_{dd} from 300mV to 400mV can bring almost 10X performance speed-up according to silicon measurements in [1]. We can also tune the V_{th} similarly to adjust speed by applying body bias or choosing from the foundry preset V_{th}s. However, this is not feasible in the superthreshold regime because driving current depends on V_{dd}/V_{th} following the α-power law [7]. One issue with subthreshold design is the increased sensitivity to variation and previous work [10] has shown that it can be successfully mitigated using proper circuit design techniques.

We further investigated the use of multi-processing techniques in conjunction with voltage scaling techniques to reduce energy consumption. The traditional way that multi-processing is implemented is illustrated in Figure 3 (a) where there is one cache per core. In this paper we propose a new micro-architecture shown in Figure 3 (b) where there are several cores sharing one local cache forming a



(a) (b)

Figure 3. Traditional and proposed MP micro-architecture

"cluster". The local cache serves k cores by running k times faster than each individual core. We achieve this by assigning proper V_{dd} and V_{th} to the cores and the SRAM/caches. To simplify the problem, we assume that all the cores are running at the same speed and same V_{dd} and V_{th}. But the V_{dd}/V_{th} between the cores and the SRAM/cache could be different. The clusters are connected to the same next level memory, which could be an on-chip cache or an off-chip DRAM.

Because the cores and the memories could potentially operate at different supply voltages, level converters will be needed in between the cores and memories. Subthreshold level converters have been shown feasible in [1]. Considering the V_{dd} space we are exploring, the delay of the level converter is minimal compared the critical delay of the core and the memory. In addition, since we are now connecting multiple cores to a single L1, the tightly coupled nature of the L1 will be removed and a bus will be needed to connect the cores and the cache. This bus will become larger and expend more energy as the number of cores within the cluster is increased.

3.2 Theoretical Analysis and Projection

In order to obtain some quick insights on how traditional and proposed micro-architectures scale differently, we first apply theoretical models before turning to detailed architectural simulation. When the system scales from single core to multi-core processing, the total number of clock cycles across all the cores increases due to the necessary application synchronization between the cores (explicit software synchronization instructions and barriers) and cache coherence traffic (implicit hardware execution overhead). [9] has shown a scaling model for multi-processing for high end processors. Although in this work we are not targeting high end applications, the basic scaling theories remain the same. [9]'s model is shown below:

$$Speedup = \frac{n_p^{x+y}}{(n_p^{x+y}-1)\cdot P_s + 1} \quad \text{(EQ 4)}$$

where n_p is the number of the processors, P_s is the percentage of the serial portion of code, and x and y are application dependent parameters describing non-linear speed-up due to various overheads. The maximum speed-up is bounded by the non-parallelizable portion P_s of the application. This scaling model applies to the traditional CMP as shown in Figure 3 (a). Note that in this work we assume a snoop-type/shared-bus cache coherence protocol.

However, we cannot directly use EQ4 for the proposed new cluster mode configuration in Figure 3 (b) because having multiple cores in a cluster reduces the coherence traffic. Therefore, we have proposed the following model:

$$Speedup = \frac{n_{mem}^{x+y}}{(n_{mem}^{x+y}-1)\cdot P_s + 1} \cdot \frac{k}{k\cdot P_s + 1} \quad \text{(EQ 5)}$$

where k is the cluster size, or the number of cores present in a cluster, and n_{mem} is the number of caches or clusters. The first term is similar to what is used in EQ4 to capture the cache coherence overhead and speed up. This speed up is related to the computation to communication ratio of the application. The second term describes the speed-up that occurs from sharing cache locally among several cores. This sharing reduces the coherence traffic between first level and second level memory because modifications made by cores within a cluster are visible to the other cores within that cluster. If some other core in the cluster needs the data, there is a higher chance that the block resides in the local cluster memory. However both terms are still limited by the serial part, P_s, of the application.

3.2.1 Memory Area Scaling Model

Previous work [10] has shown that current sensitivity to subthreshold variation increases dramatically with reduced V_{dd}. Previous work [2][11][12][13][14] has illustrated successful SRAM designs that operate in subthreshold regime. However, all these works result in an area overhead. Part of the reason for the area overhead is that larger channel area helps suppress random dopant fluctuations (RDF), the dominating factor in the subthreshold regime [15].

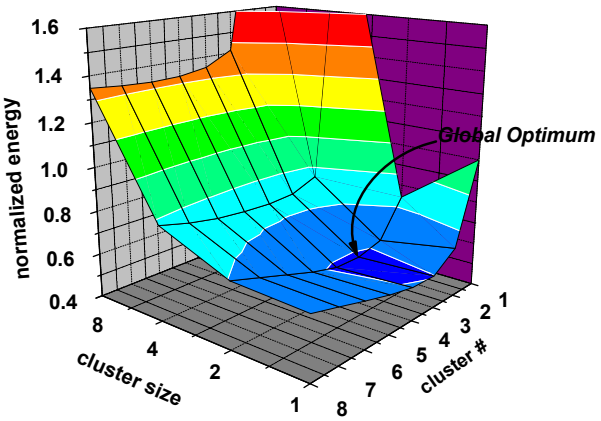


Figure 4. Theoretical scaling results

In order to factor in the design area overhead of voltage scalability for the SRAM, we carried out Monte Carlo simulations and determined the amount of up-sizing needed for the memory cells under a certain yield constraint. An exponential function is then fitted to the results and used in the rest of the paper.

3.2.2 Problem Formulation and Theoretical Results

Given a certain task deadline or a execution time for a certain application, we want to find the optimal combination of n_{core}, $V_{dd,core}$, $V_{th,core}$, n_{mem}, $V_{dd,mem}$ and $V_{th,mem}$ to achieve the minimum total energy consumption. The optimization is formulated as follows:

$$\begin{aligned} &Minimize\ energy\ consumption \\ &E_{core}(n_{core}, V_{dd,core}, V_{th,core}) + E_{mem}(n_{mem}, V_{dd,mem}, V_{th,mem}) \\ &s.t. \\ &speedup \cdot f_{core} = freq_{tar} \end{aligned} \quad \text{(EQ 6)}$$

where n_{core} is equal to n_{mem}*k, and $freq_{tar}$ is the target frequency. We assume a 16kB unified L1 cache for each core in the system similar to what is used in a commercial processor [8].

The energy consumption is modeled as the sum of switching energy and leakage energy from both the memories and the cores. After applying EQ5 and the fitted circuit-level power models from SPICE simulation, we can numerically solve the optimization problem in EQ6. The results are presented in Figure 4 where we swept the cluster size (k) and the number of clusters (n_{mem}), both from 1 to 8. The energy consumption is normalized to the single-core single-memory case. The energy increases considerably as the cluster size increases beyond 4 because the memory has to run at a much higher voltage and/or lower V_{th} and significantly raises the energy per access to the memory. In addition, energy savings are improved with more than one cluster but saturates with large cluster numbers due to the synchronization overhead and cache size. Combining these two factors, there exists a global optimal at 3 clusters with 2 cores per cluster. The parameters used are $(x+y)$=0.90 and P_s=0.05.

Although these results are based on hypothetical application and the theoretical speed-up model as in EQ5, it does provide us with the insights into cluster mode organization. We will further verify these observations with detailed power modeling and micro-architectural simulation in Section 5 on actual applications.

4 Detailed Power Model for Simulation

In order to properly attain energy numbers from our architectural simulation, we need to determine a power model for all the compo-

Table 1. Baseline architecture

Component	Value
CPU	86mW@(233MHz,1.2V) in-order functional model
Unified D/I L1	64kB, 2-way, block size=64B
L2	2MB 8-way, latency=10

nents in the system. The core energy estimations are based upon the processor core frequency and power consumption numbers from ARM946 in [8]. The same fitted model as in Section 2 is used to capture the V_{dd} and V_{th} dependency of delay and leakage. The cache power/energy numbers of different sizes were extrapolated from a memory compiler in 0.13um technology. The baseline machine detail is listed in Table 1. Off-chip access to DRAM is power-hungry because of the high capacitance in the chip package and off-chip wires. Therefore, a energy-oriented design needs to have a big enough L2 cache so that it can shield off the majority of the conflicting misses from the L1 cache from accessing the off-chip memory. For our analysis we have chosen a 2MB on-chip L2 cache operating at nominal voltage (1.2V). As aforementioned, designing a large L2 for voltage scalability with high yield implies significant area and energy overhead. Also since the L2 has a much lower activity rate than the L1, we assume L2 cache utilizes low standby leakage techniques such as drowsy cache [17]. In order to include bus switching energy, we have extracted the physical parameters from a 0.13um technology. Worst case coupling capacitance and repeater insertion have been taken into account. The length of the CPU-L1 buses scales linearly with cluster size and the length of the L1-L2 buses linearly with cluster number.

5 Architectural Simulation and Results

In order to analyze the ramifications of the proposed architectures in energy efficient processor design, we must quantify the system performance energy trade-off using architectural simulations. We performed the simulation using the *M5 Simulator* [16] to determine the energy consumption for different configurations. The analysis was done using the SPLASH2 parallel benchmark suite [5] and all the benchmarks were run to completion. Although SPLASH2 benchmarks are considered high performance scientific applications it was the only parallel benchmarks we had running in the simulator. The trends that we observed in the SPLASH benchmarks will still hold for other parallel applications, but may vary based on the amount of parallelism present and the communication to computation ratio of the application. In order to support the proposed clustered configuration, we have modified M5 to include the capability of simulating clustering. During the cluster mode, we assume that the L1's are k times faster than the core, where k is the number of cores per cluster and that each of the cores is clocked on a different phase of the caches clock to provide a round-robin access pattern and avoid an arbiter on the bus between the cores and the L1.

The relative latency of the components need to be scaled so as to capture the voltage scaling effect. For instance, when we have a multi-core system, each core and L1 inside the system can be accordingly slowed down compared to the single-core baseline machine, since our constraint is to keep the runtime constant. In architectural simulation, this is equivalent to having a faster L2 and DRAM.

5.1 Optimal L1 Size

It is known that the size of the L1 affects the system performance in terms of average access latency, but more importantly we found that it also affects the energy efficiency of the system. In order to understand how the choice of L1 size impacted the energy performance of a system, we first performed an analysis on a supply voltage scaled uni-processor system. We varied L1 sizes and optimized the energy consumption for each size by tuning the V_{dd} and V_{th} of each component. Figure 5 shows the energy consumption for *Cholesky*. We found similar trends for all the SPLASH2 benchmarks, although the optimal cache size varied across the applications. The energy consumption is broken down into processor core, L1, and L2. As L1 size increases from 4kB to 128kB, the overall accesses to the L1 remain constant because of the same instruction stream and data pattern from the processor core. However, L1 energy consumption increases with larger L1 sizes because the energy per access of the L1 requires more power due to larger array size and larger capacitance. The other implication of various L1 sizes is the L1 performance. The L1 miss rate reduces significantly with larger L1s, which also results in a lower number of access to the L2 due to less L1 misses and write-backs. Therefore the L2 energy reduces with larger L1 sizes. The core's energy consumption also reduces with larger L1 sizes because of reduced cycles per instruction (CPI) from less total number of clock cycles that results from a higher L1 hit rate (lower average access latency to the memory system).

Figure 6 shows the optimal V_{dd}s and V_{th}s of the core and the L1 with different L1 sizes for *Cholesky*. The total number of CPU cycles reduces with larger L1 sizes, implying relaxed frequency requirements under the same execution time constraint for the core. Therefore the core V_{dd} reduces. However this improvement is not large enough to change the core V_{th}. As a comparison, the L1 V_{dd} increases and the amount of change is much higher than that of the core because larger caches inherently run slower. In order for the L1 to keep up with the core the V_{dd} has to be raised. At the same time a larger cache is leakier, resulting in a higher V_{th} in order to maintain a good switching energy and leakage energy balance.

5.2 Optimal Cluster Size

To study the impact of multi-processing and clustering on the energy consumption, we set up our analysis as follows: the number of clusters is varied from 1 to 64, the number cores per cluster is varied from 1 to 8 and L1 size per cluster is varied from 4kB to 128kB. We did a complete analysis of the 240 different configurations across the complete design space for 7 SPLASH benchmarks. The other SPLASH benchmarks were also simulated but did not provide sufficient configuration combinations (due to the nature of the benchmarks).

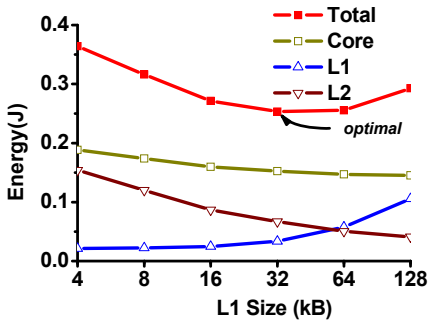
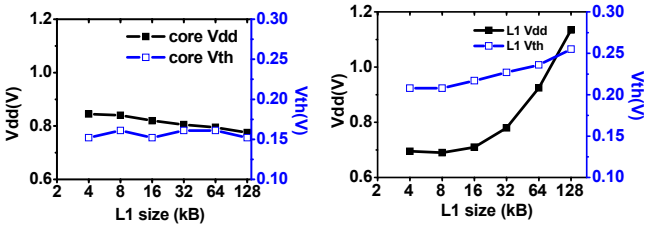
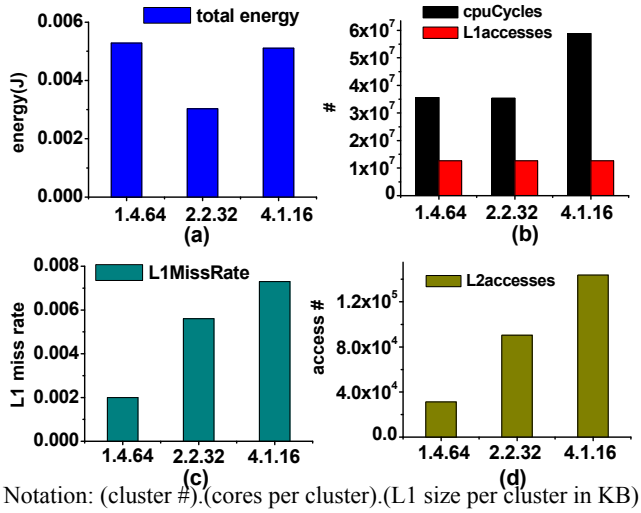


Figure 5. Optimal energy with one core and one L1 (*Cholesky*)



(a) (b)

Figure 6. Optimal V_{dd} and V_{th} settings of the core (a) and L1 (b) with different L1 sizes for *Cholesky*



Notation: (cluster #).(cores per cluster).(L1 size per cluster in KB)

Figure 7. Three configurations comparison for *LU*

In order to fully understand the benefit of running in cluster mode Figure 3(b) vs. traditional connection Figure 3(a) we specifically compare three cases in which the overall die size is held constant and present the results in Figure 7. Figure 7(a) shows the energy consumption of the three candidates for *LU* in the SPLASH2 benchmark suite. The best configuration for energy efficiency is 2 cores per cluster, with 2 clusters. This can be explained with Figure 7(b-d). At 4 clusters with 1 core per cluster, a traditional CMP, the number of CPU cycles is larger than both of the clustering cases. This occurs because the average memory access latency is long because the L1 miss rate is high do to the fact that shared memory accesses are forced to miss in the local L1 cache and snoop neighboring L1's to get the data. As the number of cores in a cluster is increased to 2 some of the shared memory of other cores is visible within the cluster's L1 to both cores without having to access another L1. This significantly reduces the average memory access time and results in a reduced number of CPU cycles. Meanwhile the larger shared cache within the cluster results in a higher hit rate and reduced accesses to the L2. Reducing the number of access to the L2 reduces the energy consumed by the L2 as was shown in Section 5.1.

As we scale to 4 cores per cluster, the L1 miss rate is reduced even further, but the energy per access of the L1 and the energy to operate the large bus to connect the cores within the cluster begin to outweigh the gains we see in reduced L2 traffic. Also there is not much reduction in CPU cycles. This happens because the first set of clustering encapsulated most of the memory sharing and synchronization that took place in this application.

For this study it is important to note that we chose three points at which the die size was the same, in the later studies we will present

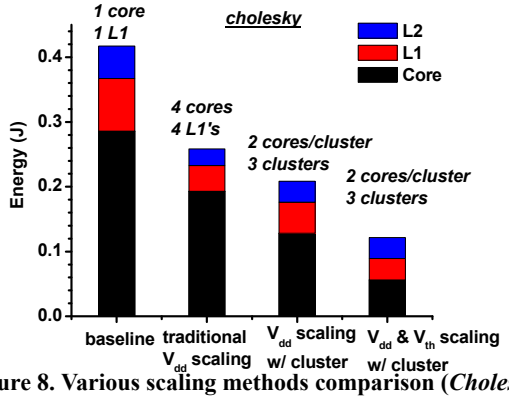


Figure 8. Various scaling methods comparison (*Cholesky*)

Table 2. Optimal V_{dd}/V_{th} for various scaling methods (*Cholesky*)

	core		L1	
	V_{dd}/V	V_{th}/V	V_{dd}/V	V_{th}/V
baseline	1.2	0.4	1.2	0.4
traditional CMP	0.782	0.4	0.782	0.4
clustering w/ V_{dd}	0.665	0.4	0.845	0.4
clustering w/ V_{dd} & V_{th}	0.37	0.189	0.64	0.264

the global optimal configuration without regard to die size. We wanted to show in this study that given a fixed die size, clustering is the optimal choice for this benchmark.

5.3 Various Energy Saving Modes

In this section, we explore the entire design space and analyze the impact of using the V_{th} to further improve performance. We again use the same 240 configurations per benchmark as in Section 5.2. Three different scaling approaches are evaluated: 1) traditional V_{dd} scaling using MP while maintaining one core per L1 cache (Figure 3 (a)), 2) V_{dd} scaling using MP but with cluster mode configuration and 3) V_{dd} and V_{th} scaling with cluster mode configuration. The runtime of all three systems is set to match that of the baseline machine. The results for *Cholesky* are presented in Figure 8. Traditional MP V_{dd} scaling brings us 38.1% savings over the baseline machine, but clustering cores with V_{dd} scaling results in a 18.9% improvement over traditional scaling techniques and 49.9% improvement over the baseline. Finally we show that clustered cores with both scaled V_{th} as well as V_{dd} yields a global optimal energy, and about a 53% percent improvement over using traditional scaling techniques.

The optimal V_{dd}/V_{th} values for the different techniques are presented in Table 2. With nominal V_{th} at 0.4V, V_{dd} scaling with traditional MP finds an optimum with 4 cores and 4 L1 caches at a V_{dd} of 0.8V. With clustering, the system finds an optimal with 2 cores per cluster at 3 clusters. In order to maintain the speed difference between the L1 and the core, the L1 supply voltage needs to be raised higher and in turn the core voltage can be dropped. The best clustered machine using V_{th}/V_{dd} scaling finds its optimum at a lower V_{dd} and V_{th} than V_{dd}-only clustered scaling. Also the L1 cache operates at a higher V_{th} than the core due to the higher leakage, confirming our observation in Figure 2. The optimal L1 size for all 3 scaling approaches is 64kB for this benchmark. The energy optimal point was chosen without regard to die size. If given a die size constraint the energy optimal point can be found within the configurations, out of the 240 explored, that meet that die constraint. Section 5.2 provides an example of such an analysis where we present three configurations with the same die size.

Table 3. Optimal configurations for different benchmarks

	n_c	k	L1 size /kB*	core		L1		*energy savings*
				V_{dd}/V	V_{th}/V	V_{dd}/V	V_{th}/V	
cho	3	2	64	0.370	0.189	0.640	0.264	70.8%
fft	2	2	32	0.415	0.180	0.620	0.236	72.6%
fmm	8	2	128	0.310	0.236	0.610	0.377	79.7%
luc	3	2	32	0.380	0.198	0.610	0.292	77.8%
lun	2	2	64	0.425	0.180	0.765	0.255	68.4%
rad	16	1	128	0.285	0.236	0.610	0.480	84.2%
ray	3	2	128	0.405	0.208	0.780	0.283	65.1%

k: # of cores per cluster n_c: # of cluster *L1 size is per cluster
energy savings is relative to baseline uni-processor machine

We ran the same analysis for additional SPLASH2 benchmarks: *Cholesky (cho), FFT, FMM, LU non-contiguous blocks (lun), LU-contiguous blocks (luc), Radix (rad), and Ray*trace *(ray)*. The results are presented in Table 3. Clustering is optimal for 6 applications with the optimal cluster size of 2 cores. The key reason is that cluster mode can enhance the memory efficiency but too many cores sharing one L1 forces the L1 to be much larger and to run a lot faster and eventually the energy per access to L1 takes over the savings from traffic to the L2 and from the CPU. Additionally the bus connecting the cores within the cluster becomes larger and requires more energy as the cluster size increases.

5.4 Optimality under Different Performance Constraints

In this section we investigated the optimal energy consumption for *Cholesky* under different performance requirements using clustering and optimal V_{dd}/V_{th} selection. The results are presented in Figure 9(b). Performance on the x-axis refers to the frequency of the baseline single-core single-L1 machine. With reduced target performance, the energy savings increases first because of relaxed frequency constraints on each core and L1 cache, both of which operate in the near-threshold regime. Then energy consumption increases because the cores and L1s begin to scale into subthreshold regime causing considerable performance loss in comparison to the power gain. Also the lengthened execution time prolongs the L2 cache leakage energy.

The V_{dd} and V_{th} settings with different performance targets are shown in Figure 9(a). The V_{dd} for the L1 slightly reduces from 233MHz to 100MHz due to relaxed frequency. However it holds around 600mV because lower V_{dd} implies significant up sizing as described in Section 3. And this area increase from SRAM redesigning nullifies the savings from reduced V_{dd}. The V_{th} increases with lowered performance requirements to balance switching energy and leakage energy. The cores operate at a significantly lower V_{dd} and V_{th} than the L1s for all of the swept performance range because it is running at half the speed of the L1. In addition logic gates are more robust than SRAM and voltage scale without area overhead.

In Figure 9(a) the optimal number of clusters increases from 2 to 3 for targets above 150MHz. This is attributed to the fact that in order to meet the higher performance constraint we need more computational power. The increase in the number of clusters, and thus total cores, requires less energy than voltage scaling the smaller number of cores to meet the same constraint. Figure 9(a) also shows that the optimal L1 size changes from 64kB to 128kB for targets below 76MHz because the relative contribution of L2 energy consumption starts to increase. A larger L1 helps to suppress L2 accesses. Although the larger L1 incurs more energy per access, the amount of energy saved from reducing the L2 accesses outweighs any increase in the L1. For more explanation on this refer to Section 5.1.

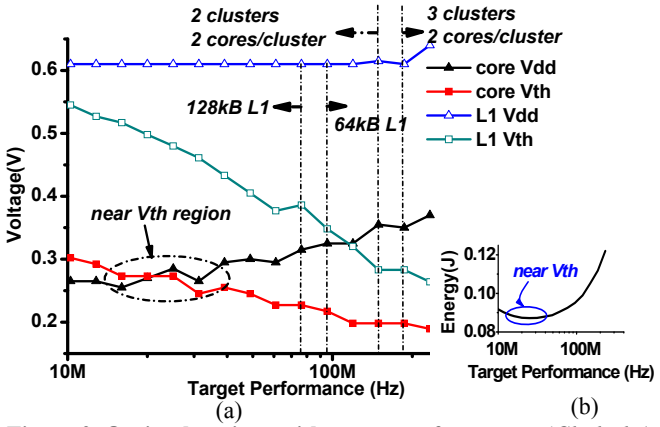


Figure 9. Optimal settings with target performances (*Cholesky*)

Finally, we have highlighted the near V_{th} region on both plots in Figure 9. Optimal energy consumption is achieved at this voltage regime and at a frequency of tens of megahertz (~15MHz-50MHz).

6 Conclusions

In this paper we have investigated the optimal threshold voltage selection for energy efficient subthreshold design. By separately controlling the supply voltage and the threshold voltage of the core and the memory, we can achieve better energy efficiency. We have combined these techniques with a novel multi-processor architecture where multiple cores share one faster L1 cache in a cluster to further improve energy savings. We found that for a typical SPLASH2 application the proposed architecture can provide about 70% energy savings over a uni-processor system and about 53% over conventional multi-processor scaling. The optimal cluster size is 2 cores for most of the SPLASH2 benchmarks that we investigated, and the system achieves best energy efficiency when operating in the near-threshold voltage regime.

Acknowledgements

The authors acknowledge the support of NSF, SRC, Intel and the Gigascale Systems Research Focus Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.

References

[1] B. Zhai, L. Nazhandali, *et al.*, "A 2.60pJ/Inst Subthreshold Sensor Processor for Optimal Energy Efficiency", *IEEE VLSI Technology and Circuits*, 2006

[2] A. Wang, A. Chandrakasan, "A 180mV FFT processor using subthreshold circuits techniques", *IEEE ISSCC* 2004

[3] B. Zhai, D. Blaauw, *et al.,* "Theoretical and practical limits of dynamic voltage scaling", *DAC* 2004

[4] B. Calhoun, A. Chandrakasan, "Characterizing and modeling minimum energy operation for subthreshold circuits," *ISLPED 2004*

[5] S. C. Woo, M. Ohara, *et. al.* "The SPLASH-2 Programs: Characterization and Methodological Considerations", *ACM ISCA,* 1995.

[6] B. Zhai, D. Blaauw, *et al.,* "The Limit of Dynamic Voltage Scaling and Insomniac Dynamic Voltage Scaling", *IEEE TVLSI*, Nov 2004

[7] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE JSSC*, vol. 25, no. 2, pp. 584-594, Apr. 1990.

[8] http://www.arm.com/products/CPUs

[9] R. A. Hankins, T. A. Diep, *et al.*, "Scaling and Characterizing Database Workloads: Bridging the Gap between Research and Practice", *IEEE/ACM MICRO* 2003.

[10] B. Zhai, S. Hanson, *et al.*, "Analysis and Mitigation of Variability in Subthreshold Design", *IEEE ISLPED*, 2005

[11] B. Calhoun and A. Chandrakasan, "A 256kb Sub-threshold SRAM in 65nm CMOS", *IEEE ISSCC*, 2006

[12] N. Verma, A. Chandrakasan, "A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy", *IEEE ISSCC*, 2007

[13] T-H. Kim, J. Liu, *et al.*, "A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtual-Ground Replica Scheme", *IEEE ISSCC*, 2007

[14] B. Zhai, D. Blaauw, *et al.*, "A Sub-200mV 6T SRAM in 0.13um CMOS", *IEEE ISSCC*, 2007

[15] M. J. M. Pelgrom, *et al.*, "Matching properties of MOS transistors," *IEEE JSSC*, vol. 24, no. 5, pp. 1433-1440, 1989.

[16] N. L. Binkert, R. G. Dreslinski, *et al.*, "The M5 Simulator: Modeling Networked Systems.", *IEEE Micro*, pp. 52-60, 2006

[17] N. S. Kim, K. Flautner, *et al.* "Single-Vdd and Single-Vt Super-Drowsy Techniques for Low-Leakage High-Performance Instruction Caches", *IEEE/ACM ISLPED*, 2004.