SD1 ParaDIME Stack Target Success Criteria Document

Target Success Criteria (and theoretical limits) for the ParaDIME Stack including: Devices (D2.1), Hardware Architecture (D3.1), Runtime (D4.1) and Applications (D5.1)

See SD2 for the Initial Requirements (D2.1, D3.1, D4.1) for the ParaDIME Stack. See SD3 for information regarding Applications Selection (D5.1).

Version 1.0

Document Information

Contract Number	318693
Project Website	www.paradime-project.eu
Contractual Deadline	Month 6 (30 Mar 2013)
Dissemination Level	Public
Nature	Report
Author	Oscar Palomar (BSC)
Contributors	Anita Sobe (UNINE), Thomas Knauth (TUD), Arindam Mallik (IMEC), Adrián Cristal / Osman Unsal / Georgios Passas, David Marquéz (BSC)
Reviewer	Christof Fetzer (TUD)
Keywords	model, baseline, energy savings, target success criteria

Notices:
The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the ParaDIME Project (www.paradime-project.eu), grant agreement n° 318693.

© 2012 ParaDIME Consortium Partners. All rights reserved.

Change Log

Version	Description of Change
v1.0	Initial Draft released to the European Commission (based on internal v20)

Table of Contents

1 Introduction and executive summary

The SD1 ParaDIME Stack Target Success Criteria Document includes a discussion of the theoretical limits and target energy savings as originally proposed in the following documents:

- D2.1 Theoretical limits and target energy savings metrics success criteria document (Device)
- D3.1Theoretical limits and target energy savings metrics success criteria and initial hardware architecture requirements document
- D4.1 Theoretical limits and target energy savings metrics success criteria definition and initial energy-efficient runtime requirements document
- D5.1 Theoretical limits and target energy savings metrics success criteria and applications selection document

The contents of these deliverables have been consolidated in order to present the target success criteria for the entire stack and thus, the entire project, in a more coherent and concise manner.

However, it should be noted that the document does not include the Requirements for each of the stack layers, nor does it include information on applications selection. Instead, these requirements have been consolidated in separate deliverable, SD2 ParaDIME Initial Requirements and SD3 ParaDIME Application Selection Document.

1.1 The ParaDIME Stack at a Glance

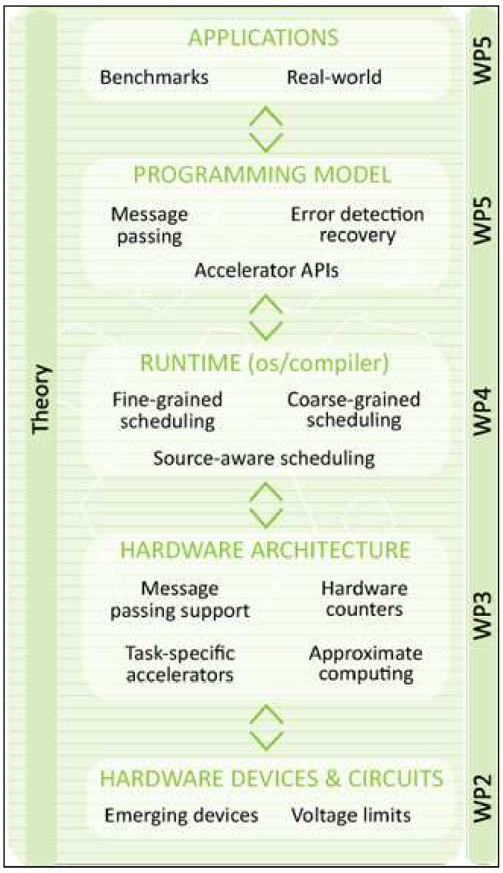


Figure 1: The ParaDIME Stack

The final outcome of the ParaDIME Project will be a complete system stack that includes novel architecture components based on "beyond the state-of-the-art" technology, a programming environment and an execution framework for energy-efficient execution of concurrent applications. This document defines the ***target energy savings*** for this stack (including the applications, the programming model and respective APIs, the runtime, the hardware architecture, the device level) by which the success of the project will be measured during the evaluation phase of the project. In order to define these targets, we first introduce analytical models for the different methodologies studied in the ParaDIME project: energy-efficient message passing, circuit and architecture operation below safe voltage limits for drastic energy savings, approximate computing, specialized energy-aware computing accelerators, device scaling, energy-efficient and proportional data centers, carbon aware scheduling and energy-efficient storage in order to propose a “best outcome” for the project.

The analytical models described in this document are used as a basis for calculating the theoretical limits of the researched ParaDIME methodologies. These limits depend on many parameters and characteristics that may not be defined or known yet. Moreover, it is likely that the models will need to be refined in order to bind the theoretical limits in a more fine-grained way.

This document begins with an overview of the key concepts and models used throughout the remainder of the document. Each section that follows focuses on one specific methodology that we plan to employ in our researching ParaDIME in order to minimize energy consumption. First, we describe the methodology or technique. Then, we introduce the relevant analytical models, and finally, we define our target success criteria and the baseline against which we plan to measure our success.

1.2 Background and Key Concepts

This section introduces several concepts used throughout the document. First, metrics related with energy consumption are discussed, followed by a discussion on the use of cost-related metrics with respect to energy minimization.

1.2.1 Energy-related metrics

We will consider two main metrics for energy: energy (E) and energy-delay product (EDP).

Energy (E) needed to execute an application is defined by the execution time (T) and the power of the system (P).

$$E = P * T$$

Thus, we can aim to reduce energy by reducing execution time, power or both. Many well-known techniques to reduce execution time typically incorporate more resources to the system that imply an increase of power dissipation. For example, adding more processors can speed up the application; however if the speed-up is not sufficiently high the additional power dissipation of the added processors may increase the system energy consumption.

Energy-delay product (EDP) weights more execution time than power. This metric is valuable even for energy-constrained systems because performance is still a concern.

$$EDP = E * T$$

Energy savings typically incur costs on other parts of the system. For example, if we lower the voltage, we can gain energy, with the cost of a higher failure probability. To handle the higher number of failures, we have to spend energy on reliability measures. In the following we identify factors that influence the energy consumption of an application. Basically the energy consumption with a new methodology ($E_{methodology}$) is the difference of energy gains (E_{gain}) and costs (E_{cost}) in comparison to the energy consumption of the baseline ($E_{baseline}$). There can be several different sources of gains and costs.

$$E_{methodology} = E_{baseline} - \sum E_{gain} + \sum E_{cost}$$

Depending on the requirements of the resources, the application and the developer; one could define weighting factors for the different costs and/or gains. This would allow us to define individual energy profiles.

We start by considering the processor power dissipation. CPU power can be modeled as follows.

First, power is divided into static power (P_{static}) and dynamic power (P_{dyn}), power dissipation due to capacitance charging or discharging the transitions from 0->1 and 1->0.

$$P = P_{dyn} + P_{static}$$

$C_{transistor}$: Parasitic capacitance of a transistor and the net the transistor drives, V_{dd}: supply voltage, f: frequency, F_{toggle} is the probability of toggling a bit.

$$P_{dyn} = \sum_{all\ transitors} \frac{1}{2} * C_{transistor} * {V_{dd}}^2 * f * F_{toggle}$$

E_{dyn} is the dynamic energy.
T_i: execution time of task i.

$$E_{dyn}^{i} = P_{dyn} * T_i$$

1.2.2 Cost-related metrics

We consider that energy minimization can be achieved by minimizing the cost of computation. With reduced energy consumption or reduced energy-delay product, the cost in electricity is reduced. With higher machine utilization, overall cost is reduced because fewer machines need to be powered on to run the same applications.

If E_A is the energy consumed to execute application A and $Cost_A$ is the cost of running the same application,

$$E_A \sim Cost_A$$

From the point of view of the application developer, it may not be clear how to optimize an application to minimize energy. It can be easier to measure cost than energy at higher levels of abstraction. Moreover, it is not obvious that application developers are concerned about energy consumption, while they have a motivation to reduce costs.

For this reason, we include cost in addition to energy and energy-delay product as a target success criterion of the ParaDIME methodologies. Energy-related metrics are useful when examining the methodologies at the device, circuit or hardware architecture level, where the impact in energy consumption is clear. Cost-related metrics will be generally used at higher layers of the ParaDIME stack (Run-time, programming model or applications).

We can calculate the total cost of running an application from the instantaneous cost during execution and the total execution time. The application runs on a number of virtual machines. We assume that machines that are not used are switched off, thus the cost varies during execution according to the utilization of the system.

The total cost of an application A is $Cost_A$.
T_A is the stop time of application *A*.
$Cost_V(t)$ is the cost of virtual machine at time *t*.

$$Cost_A = \sum_{t=0}^{T_A} Cost_V(t)$$

P_{whole} is the power of the system when all machines are turned on, $CostPerVmH$ is the cost of utilizing one machine during one hour.

$numVM$ is the number of virtual machines used.

The energy consumed by A is

$$P_{whole} * \frac{Cost_A}{numVM * CostPerVmH}$$

From the previous formula, we can estimate the energy savings of two implementations of *A*, *A1* and A2 that have different *Costs*, i.e. utilization.

Assuming

$$Cost_{A1} > Cost_{A2}$$

The saved energy is

$$P_{whole} * \left(\frac{Cost_{A1} - Cost_{A2}}{numVM * CostPerVmH}\right)$$

Assuming each implementation uses different hardware that will have different maximum power consumptions and as a consequence different $CostPerVmH$ and uses a different number of virtual machines, the formula changes to

$$P_{whole_{A1}} * \frac{Cost_{A1}}{numVM_{A1} * CostPerVmH_{A1}} - P_{whole_{A2}} * \frac{Cost_{A2}}{numVM_{A2} * CostPerVmH_{A2}}$$

A.Stack level methodologies

2 Efficient message passing

2.1 Methodology overview

Message passing programming models are known for their energy efficiency [VS10] [MMS+07]. For this reason, in ParaDIME we will adopt a message passing model that will be more efficient and scalable than shared-memory-based alternatives.

Another advantage of message passing is that the different machines involved in the computation are decoupled. This can be used to increase the utilization of the machines, which reduces the cost of computation and leads to higher efficiency at the data-center level.

In addition to the inherent efficiency of message passing, we will provide hardware support to further improve energy efficiency of message passing.

2.2 Model

In the following, we provide an extension to a power model for parallel tasks [RaRu11]. We take this model as a basis for the following extensions and add incrementally measures to save energy such as parallelization.

Parallelization (assumption: parallel tasks). According to Amdahl's law, it is assumed that a task consists of a perfectly parallelizable part and a sequential part. The sequential part remains constant.

(q: processors, i: task, $T_i(q)$: execution time of task i with q processors, σ: sequential execution time over total time)

$$0 \leq \sigma \leq 1$$
$$T_i(q) = T_i(1) * \left(\frac{(1-\sigma)}{q} + \sigma\right)$$

From $T_i(q)$ we can model the energy consumption of the parallel version.

$$E_{dyn}^i(q) = q * P_{dyn}^i(1) * T_i(q)$$
$$= E_{dyn}^i(1) * q * \left(\frac{1-\sigma}{q} + \sigma\right) = E_{dyn}^i(1) * (1 + (q-1) * \sigma)$$

Compared with the energy of the sequential version $E_{dyn}^i(1)$, we can see that the energy overhead of parallelization is$(q-1) * \sigma$

Communication costs: If we assume distributed memory machines, we have to consider communication costs. Execution time includes communication time ($T_i^{comm}(q)$) and is modeled as

$$T_i(q) = T_i(1)\left(\frac{1-\sigma}{q} + \sigma\right) + T_i^{comm}(q)$$

The communication time can be modeled based on the communication type as described by MPI. m is the size of the message and p the number of processors. τ1, τ2 and t_c are machine specific constants.

single transfer	$T(m) = \tau1 + t_c \cdot m$
single-broadcast	$T(q, m) = \tau2 \log(q) + t_c \cdot \log(q) \cdot m$
single-accumulation	$T(p, m) = \tau2 \log(q) + t_c \cdot \log(q) \cdot m$
multi-broadcast	$T(q, m) = \tau1 + \tau2 \cdot q + t_c \cdot q \cdot m$
gather and scatter	$T(q, m) = \tau1 + \tau2 \cdot q + t_c \cdot q \cdot m$

The energy spent in sending message (E_{MP}) can be modeled as follows. N_m is the number of messages sent and E_m is the energy spent sending a single message. This can be split into two parts: E_{data} is the energy spent transferring data and $E_{overhead}$ is the energy spent anywhere else (for example, in the network stack or transferring message headers).

$$E_{MP} = \sum_{j=1}^{N_m} E_{m_j} = \sum_{j=1}^{N_m} \left(E_{overhead_j} + E_{data_j}\right)$$

In ParaDIME, we will reduce $E_{overhead}$ by avoiding the network stack and providing very efficient and fast means to send data (for example using registers).

E_{data} will be reduced by approximate computing (see Section 4) and by providing accelerators for intelligent data movement, which can handle passing of complex data patterns, transferring only required data (see Section 5). This will also help to reduce N_m, as well as exploiting locality (performing computation where data is instead of moving data). The interconnection network can also play an important role to reduce the energy spent to send messages.

Processor usage: One way to save energy is to shut down unused cores. In the following, we model efficient core utilization.

P_{whole} All processors/cores are switched on and all cores are fully utilized.

P_{idle} Power when a processor is powered on but not used.

$0 \leq U \leq 1$ is the utilization of cores, where the non-used cores are switched off or in low power mode.

$$P = P_{idle} + U * (P_{whole} - P_{idle})$$

Utilization: Our hypothesis is that the decoupling allowed by message passing will lead to higher utilization (U_M) and throughput (Thr_M) when compared to the utilization (U_S) and throughput (Thr_S) of a shared-memory implementation.
That is,

$$U_M(t) > U_S(t)$$

and

$$Thr_M(t) > Thr_S(t)$$

We consider an application to be formed by a set of tasks. N_T is the number of tasks. The throughput is the number of tasks processed per second.

$$Thr = \frac{Tasks}{s}$$

If $P(t)$ is the power consumed at time t by the application, the energy required to execute the application is

$$E = \int_{t=0}^{N_T/_{Thr}} P(t)$$

The improvement in throughput will reduce the execution time but higher utilization implies higher power. However, by increasing utilization the importance of P_{idle} is reduced, which reduces wasted energy, thus improving energy efficiency.

2.3 Baseline

The proposed methodology will be compared against a) shared-memory model and b) conventional message passing implementation that doesn't include our improvements for energy minimization.

2.4 Target success criteria

For certain methodologies, we cannot provide quantitative criteria because the amount of savings is too dependent on the characteristics of the applications that are used. For example, the kind of messages sent, the frequency of message sending in addition to task dependences, etc. can significantly affect results. Nevertheless, we expect to:

- Reduce the cost of sending one message over conventional message passing.
- Reduce the number of messages sent with respect to conventional message passing.
- Reduce the energy-delay product of an application with respect to shared-memory or conventional message passing.
- Increase the utilization of the system with respect to shared-memory.
- Increase the throughput of the system with respect to shared-memory.
- Reduce the cost of running one application with respect to shared-memory or conventional message passing.

3 Operation below safe Vdd

3.1 Methodology overview

We want to reduce power consumption and increase the energy-efficiency at the CPU level by decreasing the CPU's supply voltage. Modern CPUs already incorporate energy-efficiency measures. The processor supports several power states. The ACPI standard defines exactly four different power states: C0 - C3. Besides power states, the processor may also support different performance states. The number of performance states differs between processors. They are numbered P0, P1, … , Pn.

Each successively higher state reduces the processors performance, because the voltage and/or frequency is reduced.

Usually the operating system exclusively controls power management features. However, by overriding the operating system it is possible to scale voltage and/or frequency even more aggressively, enabling even higher savings. More aggressive scaling is only possible by operating the CPU outside of the manufacturer's specified safety margin.

Due to the recent issues impacting device scaling as we approach the end-of-the-CMOS-roadmap, safe operation margins have been increasing. In particular, there is a substantial "tax" in the case of guard-bands for supply voltage. Due to systematic and random variability, increased thermal stresses and noise margins; this guard-band is increasing. If we go below the safe limit and the associated guard-band, one might encounter sporadic errors while simultaneously saving power dramatically. This will require support from several levels, for example using selective duplex replication at the architectural or programming model levels. For this approach to be worthwhile, the energy savings must be higher than the cost of correcting the resulting errors.

Voltage reduction comes with the cost of increasing delay substantially. Thus, it is not the most appropriate approach for time-critical tasks or applications. However, it is still very attractive in many cases. For example, it has been proposed for data centers servicing web pages, where it has been observed that 75% of overall energy is spent in requests to tier 1. The workload is extremely parallel with independent requests to render web pages. Many cores operating with reduced voltages can provide high throughput in a very energy-efficient way [DWBS10].

3.2 Model

s is the scaling factor applied to the nominal voltage, When s is 1, the processor operates at the nominal Vdd.

$$s \geq 1$$

Energy includes energy spent in computation ($E_{computation}$) but also the overhead energy for error detection ($E_{detection}$) and recovery ($E_{recovery}$); all three components are a function of s.

$$E(s) = E_{detection}(s) + E_{recovery}\ (s) + E_{computation}(s) < E(1)$$

In order to save energy under scaling voltage, $E(s)$ must be smaller than $E(1)$, which limits the energy that can be spent in error detection and recovery. Also, the error detection capability of the fault tolerance scheme should keep the vulnerability of the system lower than the user's restrictions.

Energy Saving in Scaling Voltage

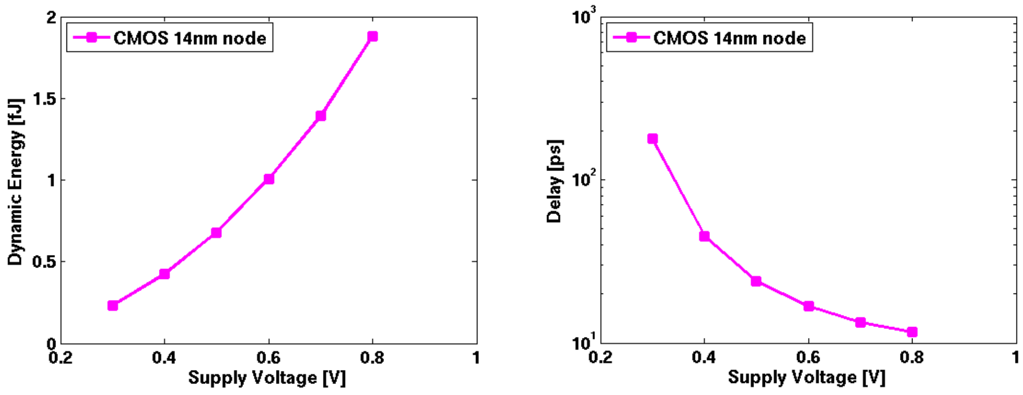


Figure 2: (a) Circuit energy; (b) Delay characteristic as a function of supply voltage for 14nm CMOS technology

The effect of scaling down the Vdd for a given technology node is illustrated in Figure 2. This figure shows the simulated circuit energy and delay as a function of supply voltage for 14nm CMOS FinFET technology. The circuit consumes less energy when operating at a lower supply voltage, but at the same time the computation delay increases. By lowering the supply voltage from nominal 0.8V to 0.6V, the circuit energy consumption reduces by 47%, with a 44% increase in delay. If the supply voltage is lowered to 0.4V, the energy can be further reduced by 77% when compared to 0.8V operating supply. This would result in dramatic increases in computation error event and error rate. Moreover, as can be seen in Figure 2, the circuit delay increases significantly by 289%.

From these results we can expect substantial energy gains when Vdd is lowered as well as a reduction of power. Regarding EDP, it will remain approximately the same when reducing supply voltage, since energy and delay change roughly by the same amount, as can be seen from the results for 0.6V. However, when the supply voltage is reduced to voltages near the threshold, higher increases in delay will offset the energy reduction, which subsequently increases EDP. For this reason, EDP is not a target success criterion of this methodology.

The power reductions achievable with resilient computing are bound by the increased error rate. Reducing the supply voltage will inevitably lead to more errors. The goal is thus to find a sweet spot where the trade-off is still beneficial. Figure 3 illustrates the relationship between supply voltage, error rate, and computation speed. As the supply voltage decreases, the error rate increases linearly. The sweet spot for the circuit investigated in this example is between 1.16 and 1.17 Volts. At this point, the error rate increased by about 0.04% and the computation speed was reduced by 0.2%. However, the reduced voltage led to a net gain in terms of power consumed.

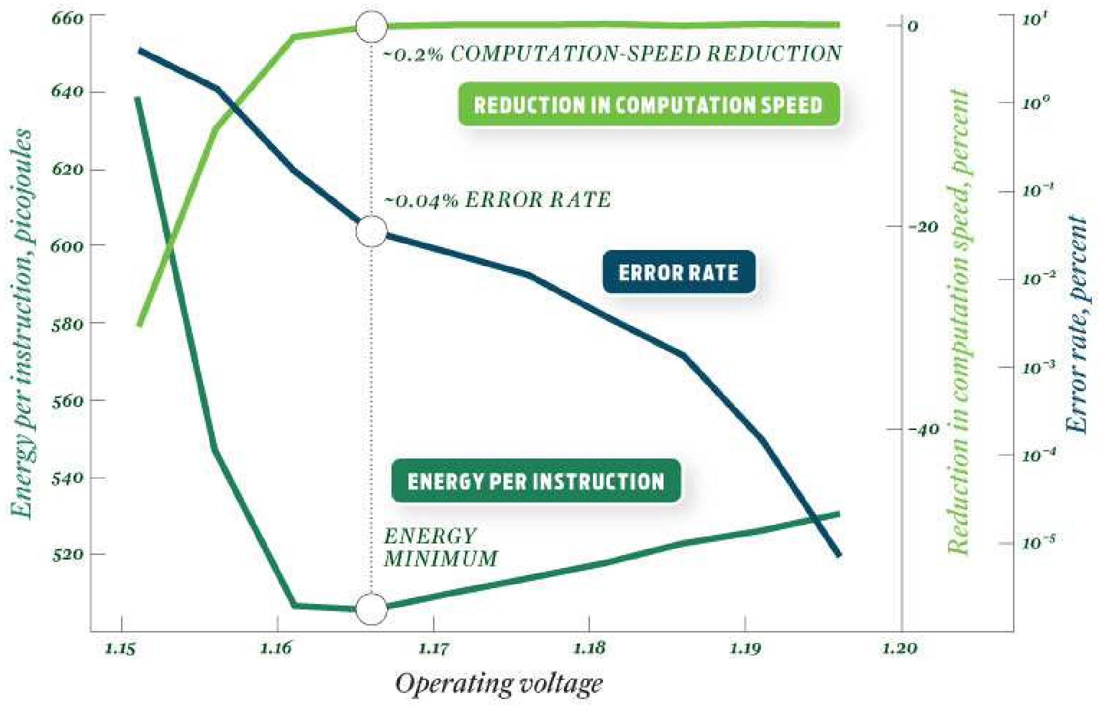


Figure 3: Impact of reducing supply voltage [“CPU, Heal Thyself”, IEEE Spectrum, August 2009]

From the formula for energy,

$$E = P * T$$

if we assume that the increase in delay will be reflected in the execution time in the same proportion through a lower frequency, we can see that power reduction will be proportional to the product of the energy reduction and the delay increase.

$$\frac{P(s)}{P(1)} \sim \frac{E(s)}{E(1)} * \frac{T(1)}{T(s)}$$

With a reduction of E of 45% and an increase in delay of 45%, we can expect around 38% reduction of power.

$$\frac{P(s)}{P(1)} \sim \frac{E(s)}{E(1)} * \frac{T(1)}{T(s)} = 0.55 * \frac{1}{1.45} = 0.379$$

Nevertheless, the energy spent in error detection and correction mechanisms will limit the reduction in power.

Error Detection [CUYF13]
Although fault tolerance is essential in order to tolerate the drastically increasing fault rate in the scaling voltage, it also comes with an overhead in energy.

Example: Replication (i.e., execute twice and compare results at the end) needs double the power the execution time remains the same, because the tasks are executed in parallel. We will apply a selective replication to replicate only when it is necessary.

rep is the number of replicas. q is the number of processors, σ is the sequential execution time over total time

$$P_{dyn}^{i}(q,s,rep) = rep * P_{dyn}^{i}(q,s,1)$$

$$E_{dyn}^{i}(q,s,rep) = rep * P_{dyn}^{i}(q,s,1) * T_i(q) = E_{dyn}^{i}(q,s,1) * rep$$

Error Recovery (Transactions) [CUYF13]
A rollback means that after the error detection the whole task has to be repeated, and might fail again. For recovering from errors, we can consider reliability-purposed transactions (TX) as the basic recovery block [YUCH11]. A TX has failure atomicity, i.e. in the event of a failure it aborts the TX, undoes all the TX side effects, and restarts the TX.

Prob$_{\text{core}}$(f): Probability of having a faulty bit in the core at a time under given frequency

Prob$_{\text{TX}}$(f): Probability that a TX has a fault under given frequency

Prob$_{\text{TX}}$ is calculated as the probability of having a failure in TX is subtracted from 1. It depends of the size of the transaction size$_{\text{TX}}$

$$Prob_{TX} = 1 - (1 - Prob_{core}(f))^{size_{TX}}$$

A recovered transaction also may fail and require a recovery again.

$$T_i(t_x) = T_i(q) * Prob_{TX} * \left(\frac{1}{(1 - Prob_{TX})}\right)$$
$$E_{dyn}^{i}(q,s,rep,t_x) = T_i(q) * Prob_{TX} * \left(\frac{1}{1 - Prob_{TX}}\right) * P_{dyn}^{i}(q,s,rep,1)$$
$$= E_{dyn}^{i}(q,s,rep,1) * \left(1 + Prob_{TX} * \left(\frac{1}{1 - Prob_{TX}}\right)\right)$$

3.3 Baseline

The proposed methodology is compared against operation with nominal Vdd values. Baseline is a processor with similar resources operating at nominal Vdd.

3.4 Target success criteria

- Reduce P dissipation by approximately 30% for Vdd below the safe limit, including the overhead of error detection and correction mechanisms.
- Reduce E by approximately 40% for Vdd below the safe limit, including the overhead of error detection and correction mechanisms.
- Limit performance decrease (to less than 5%) when operating at nominal Vdd.
- Enable lower values of Vdd or higher performance at nominal Vdd with existing error detection and correction mechanisms.

4 Approximate computing

4.1 Methodology overview

Several studies have shown [EUVG09] that between 50% and 70% of the values in the processor are narrow, depending on the architecture, application and the number of bits needed to consider a value narrow. This means that the processor doesn't need to load, store or compute the upper bits of an integer value if it has been identified as a narrow value. Errors occurring in these bits are also not relevant.

A similar approach can be used for floating point values when maximum precision is not required. In this case, the least significant bits of the mantissa are not relevant.

4.2 Model

From the number of instructions, the average cycles per instruction (CPI) and the frequency ($freq$), we can calculate the execution time.

$$T = I * CPI * \frac{1}{freq}$$

Similarly, the number of instructions and the average energy per instruction (EPI) provide the total energy.

$$E = I * EPI$$

Approximate computing will reduce CPI and EPI of the instructions that operate on narrow data and reduced precision.

We can classify all instructions as floating point (FP), integer (INT) or memory (MEM). Approximate computing can be used to reduce the energy require to execute instructions of all three types.

$$I_{FP} + I_{INT} + I_{MEM} = 1$$

R_{MEM}, R_{INT}, $R_{FP}$are the fraction of each instruction type that operates on narrow values or reduced precision. R_{MEM}, R_{INT}, R_{FP}depend on the number of bits that are used for narrow values or reduced precision. Smaller widths decrease R_{MEM}, R_{INT}, R_{FP} because less computation can be performed within an acceptable accuracy.

INT instructions that operate with known narrow values and FP instructions that operate with reduced precision can save energy in the ALUs and the register files. Also associated logic such as the bypass logic is smaller and consumes less energy. Existing techniques to exploit narrow values typically detect which data is narrow dynamically, which requires including additional fallback mechanisms. We won't incur in the overhead of detection and fallback mechanisms because narrow values will be annotated.

MEM instructions save energy by reducing the amount of data that has to transfer. If narrow integer data and reduced precision FP data is packed in memory, also performance can increase. Another possible source of energy reduction is the use of narrow values for address calculation This can also help to reduce the number of TLB accesses.

From Amdahl's law

$$Speedup = \frac{1}{(1-P) + \frac{red}{S}}$$
$$CPI_{new} = (1 - red) * CPI_{old} + \frac{red * CPI_{old}}{S}$$
$$EPI_{new} = (1 - red) * EPI_{old} + \frac{red * EPI_{old}}{S_{energy}}$$

red is the portion of instruction that will benefit from the reduced precision.

$$red_{FP} = I_{FP} * R_{FP}$$
$$red_{INT} = I_{INT} * R_{INT}$$
$$red_{MEM} = I_{MEM} * R_{MEM}$$

Energy and latency of FP multiplication decreases linearly with mantissa bit-width [ToNR00] so S and S_{energy}will be linear to the difference between the original data width and the reduced data width.

INT instructions also have a linear S_{energy}but typically S will be 1, because latency is already small in cycles. In order to benefit from the latency reduction frequency should be increased, which increases power. Another possibility is to exploit narrow integer values by computing several operations in parallel on packed values.

For MEM instructions, there are fewer bits to transfer from data cache. The reduction in $S_{energy}$is linear but only to the part that applies to data transfer. Address calculation and TLB accesses can also benefit from narrow values.

4.3 Baseline

The proposed methodology is compared against full-width IEEE754-compliant data and arithmetic.

4.4 Target success criteria

For certain methodologies, we cannot provide quantitative criteria because the amount of savings is too dependent on the characteristics of the applications that are used. For example, the portion of computation that can be approximated (that can use narrow values), the minimum width that can be used to produce acceptable results, etc. can significantly affect final results. Nevertheless, we expect to:

- Reduce E for execution of applications that use reduced precision and narrow values.
- Reduce EDP for execution of applications that use reduced precision and narrow values.

When using reduced precision, the difference in output should be acceptable (what is acceptable depends on the application).

5 Heterogeneous computing

5.1 Methodology overview

CPUs are general purpose microprocessors. But even they have a multitude of special purpose circuitry to help with, e.g., floating point operations and streaming data manipulation (SSE1/2/3/4). Besides the CPU, there are other components, which take over specialized tasks. The most prominent example is the graphics processing unit (GPU). The GPU is an *accelerator* for graphics processing. Rendering 3D scenes is a complex task which can, however, be speed up significantly with special-purpose hardware. The idea of accelerators is to implement certain functionality in hardware instead of executing it in software on a general purpose processing unit. By offloading tasks to the accelerator, the general purpose unit is free to do alternative work, or sleep if there is nothing else to do. The accelerator, because it is specialized, will perform the same task more efficiently.

The challenge with accelerators is to identify small tasks which are executed frequently. With each accelerator there is an associated development cost which makes it prohibitively expensive to blindly cast everything in hardware. The accelerator’s cost must be amortized by executing it frequently. Otherwise, the development, circuitry, and energy costs do not amortize.

A recent study [WL08] has shown that for a given power chip budget, one can improve both energy efficiency and performance by combining few state-of-the-art superscalar processors with many small but energy-efficient cores (see Figure 4). However, this project will venture beyond this issue as the effectiveness of such “classical” heterogeneous cores will be limited and will not deliver the expected performance with increasing core count towards the end-of-the-CMOS-roadmap. In fact, due to power envelope issues not all of the chip will be able to be powered on at a given time giving rise to the dark silicon phenomenon [EB11].

One way to deal with the dark silicon problem is through the use of specialized accelerators [BC11]. Hence, we foresee the emergence and eventual dominance of special-purpose energy and power efficient accelerators that are tailored for a certain group of applications such as security, speech recognition, image processing, and artificial intelligence. Recent research [CG+11] shows that specialized accelerators can improve energy efficiency and performance by 32x and 9x respectively. The performance benefits of accelerators could be exploited to utilize them for error detection and as special-purpose units to increase the efficiency of message passing. Less specialized accelerators such as vector units can also improve both energy efficiency and performance of many tasks. These observations will force us to rethink the way we develop concurrent software and approach it in a way that achieves high scalability while leading to a radical reduction of energy consumption.

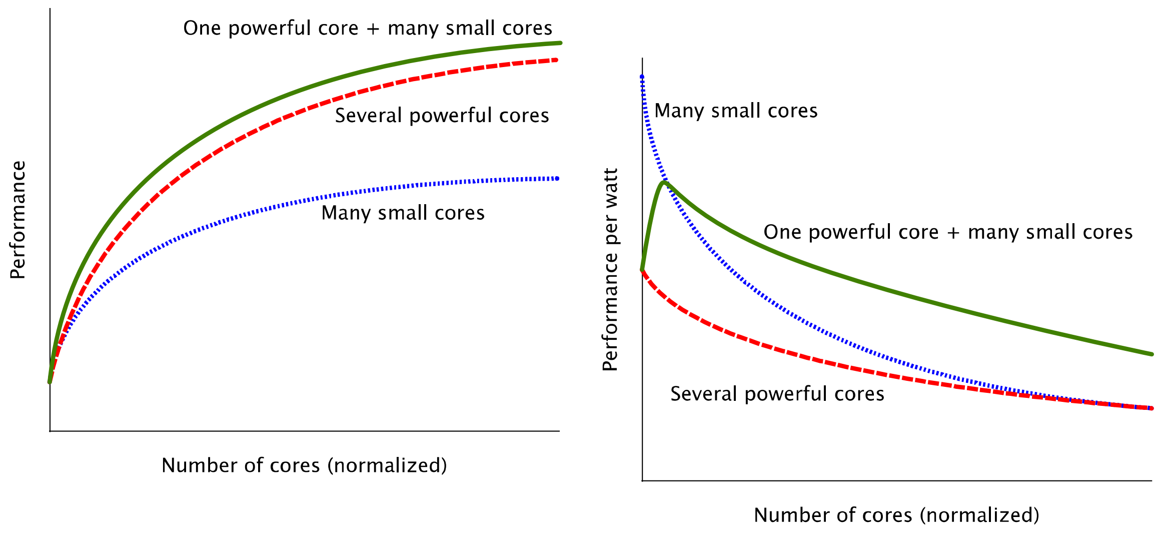


Figure 4: By exploiting heterogeneous core, one can improve performance and power efficiency. In this example, combining one state-of-the-art superscalar processor combined with many smaller more energy-efficient cores is better than using only one type of processors for a given chip power budget [WoLe08]

5.2 Model

The following model is inspired by the work from Chung et al. [CMHM10]. The model includes three types of cores: 1) basic computing engine (BCE), which is a very simple in-order processor with limited or non-existing speculation, 2) sequential core (SEQ), which represents aggressive out-of-order processors and 3) accelerators (ACC). The parameters and characteristics of SEQ and ACC are normalized to BCE.

We characterize cores with Area, performance ($Perf$) and power (P), all normalized to BCE. Energy is derived from performance and power.

$$E = \frac{P}{Perf}$$

5.2.1 Core level

A BCE core has the following characteristics.

$$Area_{BCE} = 1$$
$$Perf_{BCE} = 1$$
$$P_{BCE} = 1$$
$$E_{BCE} = \frac{P_{BCE}}{Perf_{BCE}} = 1$$

SEQ utilizes more area than BCE.

$$Area_{SEQ} = r$$

We use Pollack's law [Poll99] to estimate the performance of the sequential core as a function of the area it uses.

$$Perf_{SEQ}(r) = \sqrt{r}$$

For this kind of processor, there is a super-linear relationship between power and performance [CMHM10].

$$P_{SEQ}(r) = Perf_{SEQ}(r)^{\alpha} = r^{\alpha/2}$$

The value of alpha has been estimated to be 1.75. [GrAn06]

In order to model the ACC cores, we first assume a BCE-sized accelerator, which is characterized by its power and performance relative to BCE.

$$Area_{ACC} = 1$$
$$Perf_{ACC}(1) = \mu$$
$$P_{ACC}(1) = \varphi$$

For larger accelerators, we assume that both power and parallel performance scale linearly with the resources used to execute parallel sections of code. [CMHM10]

$$Area_{ACC} = k$$
$$Perf_{ACC}(k) = \mu(k) = \mu * k$$
$$P_{ACC}(k) = \varphi(k) = \varphi * k$$

5.2.2 System level

Now we consider a heterogeneous system HET with 1 SEQ core and 1 ACC. The system has total area *n*.

$$Area_{HET} = Area_{SEQ} + Area_{ACC} = n$$

We assume that the SEQ core has area *r*.

Using Amdahl’s law we can model performance of the HET system, being f the portion of time when the accelerator is used , f is the fraction of time where an accelerator is used.

$$Perf_{HET}(n) = \frac{1}{\frac{1-f}{Perf_{SEQ}(r)} + \frac{f}{\mu(n-r)}}$$

Assuming perfect power gating, we can also model the power of the HET system.

$$P_{HET}(n) = (1-f) * P_{SEQ}(n) + f * P_{ACC}(n)$$

$$
\begin{aligned}
E_{HET}(n) = \frac{P_{HET}(n)}{Perf_{HET}(n)} &= \frac{(1-f)P_{SEQ}(r) + f * P_{ACC}(n-r)}{\dfrac{1}{\dfrac{1-f}{Perf_{SEQ}(r)} + \dfrac{f}{Perf_{ACC}(n-r)}}} \\
&= \Big((1-f) * P_{SEQ}(r) + f * P_{ACC}(n-r)\Big) \\
&\quad * \left(\frac{1-f}{Perf_{SEQ}(r)} + \frac{f}{Perf_{ACC}(n-r)}\right) \\
&= \Big((1-f) * r^{\alpha/2} + f * \varphi * (n-r)\Big) * \left(\frac{1-f}{r^{\alpha}} + \frac{f}{\mu * (n-r)}\right)
\end{aligned}
$$

We are not considering parallel execution and message passing but accelerators fit in there as well. The HET system is simply a node of the complete system.

Accelerators for message passing may work in parallel with the main system. In this case, SEQ wouldn't be powered off.

$$P_{SEQ}(r) + f * P_{ACC}(n-r)$$

In this case, the improvement in performance will apply to the communication costs. From Section 2:

$$T_i(q) = T_i(1)\left(\frac{1-\sigma}{q} + \sigma\right) + \frac{T_i^{comm}(q)}{Perf_{ACC}}$$

5.3 Baseline

The baseline is a system without accelerators. The accelerator is compared against an aggressive out-of-order superscalar core (SEQ) and a simpler less power-hungry core (BCE).

5.4 Target success criteria

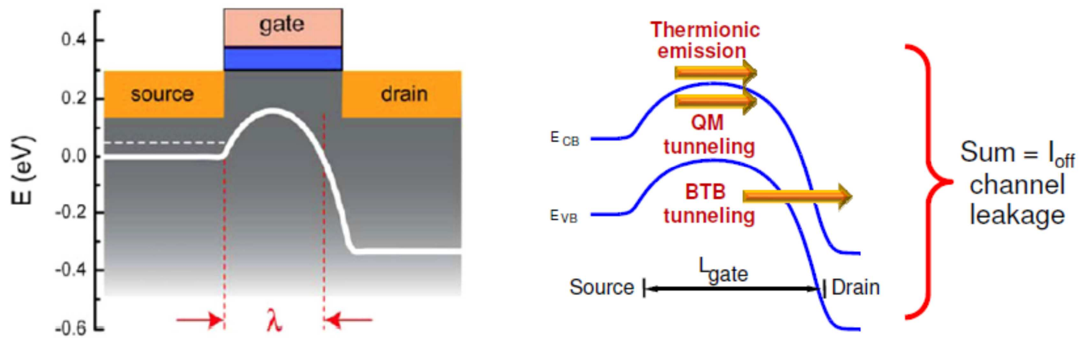
- Accelerators that have low φ and high μ, thus improving EDP above SEQ.
- Accelerators that have lower φ and/or higher μ over GPUs which today represent the state-of-the-art in available accelerators. Our reference values are empirically measured φ and μ for GPUs [CMHM10]: φ: 0.26-1.27, μ: 0.75 – 17.0. φ can be higher than 1 (higher power than BCE) as long as it is compensated by higher values of μ, therefore improving power efficiency.

B.Device level specific methodologies

6 Device scaling

6.1 Methodology overview

The scaling of conventional planar MOSFETs has been facing problems such as sub-threshold swing degradation, significant DIBL, fluctuation of device characteristics, and leakage. The MOSFET can be thought of as consisting of two wells (source and drain) separated by a barrier (channel).



- Planar gate: limited electrostatic control of the channel $\lambda = \sqrt{\frac{\varepsilon_{ch}}{\varepsilon_{ox}} d_{ox} d_{channel}}$

Figure 5: Short channel effect in MOSFETs

When the channel length reduces, no effective barrier is formed between the source and drain and the transistor "OFF" current increases. Devices with an improved electrostatic control over the channel are needed. Tri-gate transistors (FinFETs, see Figure 6) have better control of short-channel effects which enables further gate length scaling than planar *Si* devices (see Figure 7). We have observed improved subtreshold slope and Drain Induced Barrier Lowering (DIBL) for scaled. FinFETs as compared to planar devices. FinFETs are enabling devices for the 14nm technology (and beyond) technology generation.

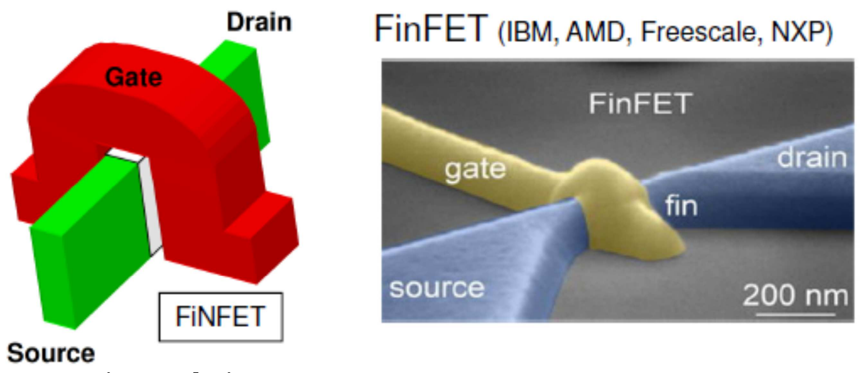


Figure 6: FinFET devices

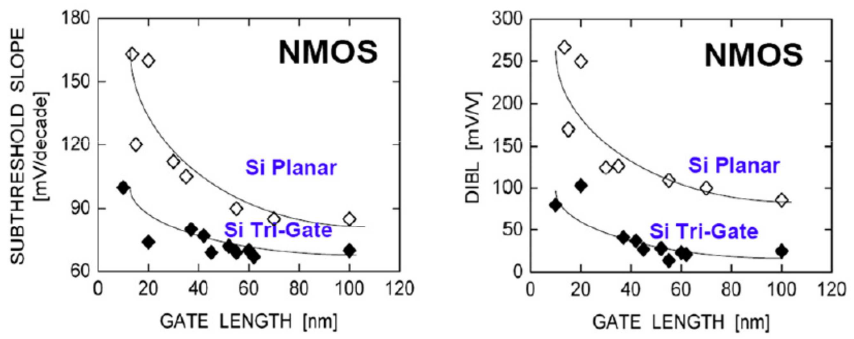


Figure 7: FinFET devices- better scalability over planar devices

As we scale down even more, the need for high mobility channel materials become more significant for improvement in performance. In the context of carrier transport in short channel devices, as it is dominated by ballistic transport, mobility loses its meaning. However, low effective mass is still important to obtain high source injection velocity and therefore mobility is still a good indicator for high drive current. *Ge* and *III/V* materials have a high carrier mobility, but do not have a stable natural oxide with good interface properties (such as the *Si/SiO2* system). Electrical passivation of the high-µ / high-k interface is a major challenge. *Ge* pMOS devices with *Si* passivation have better performance than *Si*. Excellent performance of short channel (70nm) is observed in *Ge* pFET devices. Drive current scales with EOT as expected (40% increase in Ion when EOT scales from 1.25nm to 0.85nm).

6.2 Model

6.2.1 Effect of scaling on Device performance and power

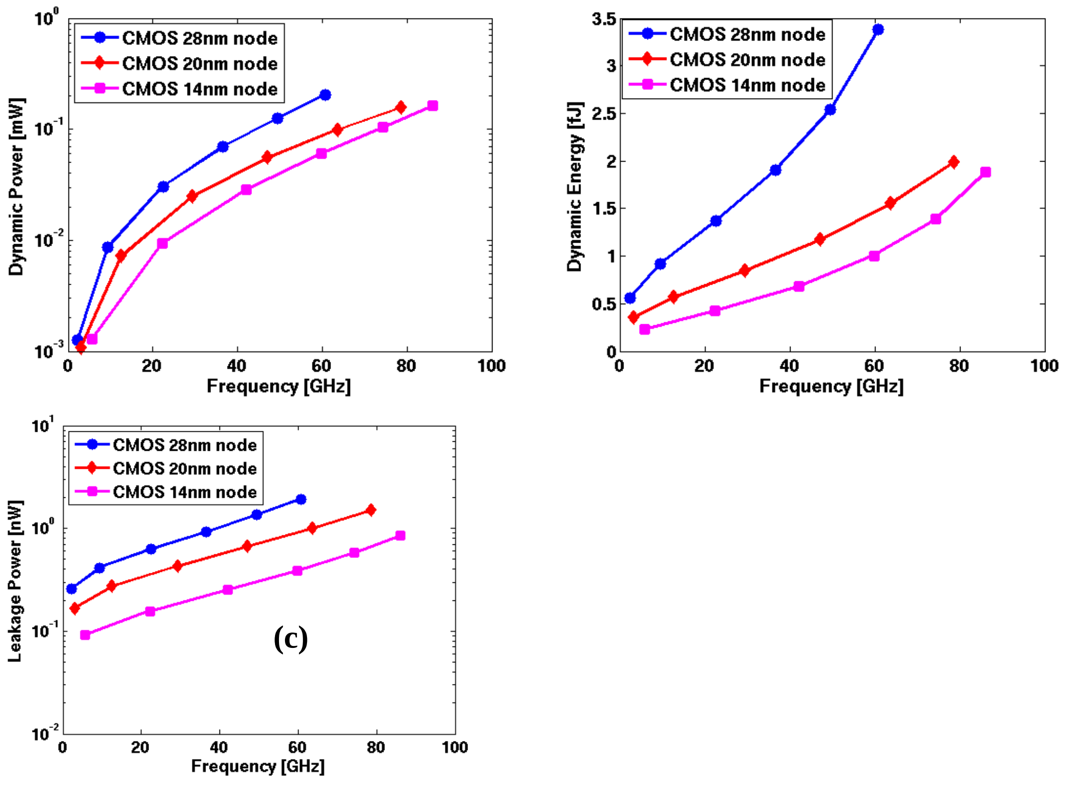
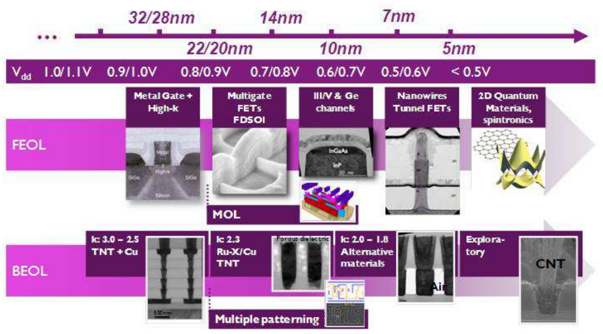


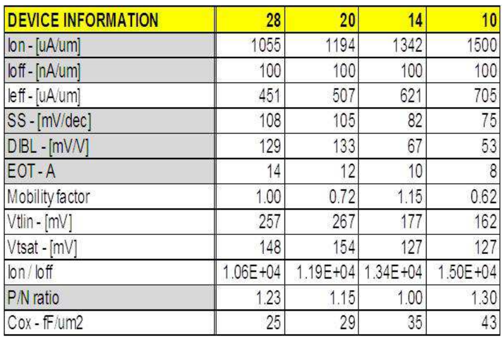
Figure 8: (a) Performance and power ; (b) Performance and energy; (c) Performance and leakage comparisons with different CMOS technologies

This section evaluates the performance improvement and reduction in power consumption due to technology scaling. Due to limited availability of compact models beyond 14nm, an early comparison between 28nm technology node and 14nm technology node is performed. Please note that, all results reported in this section is based on early assumptions in the compact model. During the course of the project, compact models will be developed to evaluate the power and performancegains for advanced technology nodes and different devices. Figure 8 shows the simulated power, energy, leakage and speed performance of a FO4 inverter circuit based on representative CMOS 28nm, 20nm and 14nm (FinFET) technology nodes. We can see by continuously scaling down the device dimension together with novel device structure, the 14nm process offers 79% reduction in leakage, 69% reduction in power and energy while maintaining identical speed performance, when compared to the 28nm technology node. If we average the amount of improvement over two generations of technology node (28nm and 20nm), we can foresee we can expect the energy consumption of at a device level is reduced by over 25% per technology node transition.

6.2.2 System level evaluation

It is important to evaluate the effect of technology innovation at an application level at the initial phase of the device architecture and process assumptions. Process assumptions are chosen to enable pitch scaling under the limitations of tools and materials with some early learning from process development of individual modules. The applications Key Performance Indicators considered cover High Performance Computing (HPC) and High Performance Mobile (HPM). Packaging and thermal requirements define the specifications of HPC while battery is the major bottleneck for HPM and autonomous sensors (AS). Targeting framework takes into account several elements of system-on-chip (SoC) that range from PPAC (power, performance, area, cost), power and clock integrity to applicability of traditional low-power techniques such as multi-Vt and dynamic voltage and frequency scaling (DVFS).





DEVICE INFORMATION	28	20	14	10
Ion - [uA/um]	1055	1194	1342	1500
Ioff - [nA/um]	100	100	100	100
Ieff - [uA/um]	451	507	621	705
SS - [mV/dec]	108	105	82	75
DIBL - [mV/V]	129	133	67	53
EOT - A	14	12	10	8
Mobility factor	1.00	0.72	1.15	0.62
Vtlin - [mV]	257	267	177	162
Vtsat - [mV]	148	154	127	127
Ion / Ioff	1.06E+04	1.19E+04	1.34E+04	1.50E+04
P/N ratio	1.23	1.15	1.00	1.30
Cox - fF/um2	25	29	35	43

Figure 9: Estimated device parameters for advanced technology nodes

Traditionally, target pitches for device and interconnect are selected to enable 50% area downscaling node-to-node. Ion-Ioff targets are given in Figure 9 as well as short-channel behavior targets such as subthreshold slope and DIBL. Considered devices for targeting are bulk planar for 28 and 20nm, Si-based bulk finFET for 14nm, and III-V or SiGe based finFET for 10nm. For finFET advantages of subthreshold slope and low Ioff are reflected in the selection of device parameters as well as careful selection of fin geometries to ensure good electrostatic control. We analyzed these targets at application level in order to if PPAC targets are met or not.

6.3 Baseline

The characteristics for one technology node are compared with the previous technology node.

6.4 Target success criteria

Averaging the amount of improvement over two generations of technology node, we will:

- Reduce the energy consumption at the device level by more than 25% per technology node transition.

C. Data center level specific methodologies

7 Energy-efficient and energy-proportional data centers

7.1 Methodology overview

Data centers typically show a low average server utilization of 50% or less. The reasons are manifold: first, systems start to behave unpredictably approaching 100% utilization. Second, to cope with load surges there must be enough "headroom", i.e., systems are traditionally overprovisioned.

We acknowledge that energy efficiency is conflicting with these goals. Low system utilization directly translates into low efficiency. A server's energy efficiency is highest at 100% utilization. A fully utilized server may draw, for example, 300 W. The same server at 0% utilization will still draw about 40% of its peak power, i.e., 0.4*300 W = 120 W. Even though some server components, e.g., most prominently the CPU, support different power states, this is unfortunately not the case for all server components. The base power consumption of an idle server is far from the ideal 0 W.

To increase the energy efficiency at the data center the goal is to increase average utilization levels. By pushing utilization levels up, the comparatively high baseline power consumption is compensated for. We plan to combat the previously mentioned drawbacks of high utilization levels by executing a mix of compute tasks on each server. We distinguish between two types of tasks: interactive and batch. Interactive tasks have stringent performance requirements expressed as service level agreements (SLAs). They are latency-sensitive because they are end-user driven. A typical example is a web server. The end user requests a web page from the server which should answer as quickly as possible, e.g., within one second.

Batch tasks, on the other hand, have turnaround times that are 2 - 3 magnitudes larger than interactive tasks, i.e., hours or days. This flexibility allows us to achieve utilization values of 90% and higher. Each server executes a mix of interactive and batch tasks. We have to ensure that interactive jobs never account for more than, say, 50% of the load. Additional capacity is consumed by batch tasks. Whenever there is a spike in interactive load, batch tasks will yield their resources to the interactive tasks. As soon as the surge in interactive load subsides, the batch tasks will continue executing, occupying all available spare resources.

Energy-proportional computing follows naturally from our goal to increase server utilization. Operating the server as close to 100% utilization as possible will ensure energy-proportionality. The worst operating point for a server is at 0% where it consumes up to 50% of its peak power while doing no useful work.

7.2 Model

We model the projected energy savings by defining an upper bound of the utilization due to interactive jobs called U_{inter_max}. Whenever the current interactive utilization, U_{inter}, exceeds the maximum more servers must be brought online. The utilization due to batch jobs is referred to as U_{batch}. Together, U_{inter} and U_{batch} form the overall utilization U

$$U = U_{inter} + U_{batch}$$

The current interactive and batch utilization are measured. If either of the three utilization values exceeds its limit additional servers must be brought online. The target number of new servers is calculated according to the following formula

$$N_{new} = U_{current} * N_{current}/U_{target}$$

That is, the new number of required machines can be calculated based on the current utilization and server count combined with the target utilization. Consider the following example: 100 servers are online, and the utilization mix is 60% interactive jobs (U_{inter}) and 30% batch jobs (U_{batch}), for a total utilization of 90% (U). The threshold for interactive jobs is 50% (U_{inter_max}), i.e., U_{inter} must be reduced by 10 percentage points. The overall target utilization drops from 90% to 80%, i.e., 90% = 60% + 30% - 10%. 100 servers ran at 90% utilization.

7.3 Baseline

Our baseline is a data center where servers are powered on all the time. Energy consumption is calculated based on the utilization level of each server over time. The total consumption is approximated by summing the consumption of each individual server. The utilization of server n is denoted by U_n and P(U_n) is the consumption for a given utilization. The total number of servers is N.

$$\sum_{n=0}^{N} P(U_n)$$

Aggregating the total consumption for a time period gives the total consumption for the period

$$\sum_{t=0}^{T}\sum_{n=0}^{N} P(U_n^t)$$

7.4 Target success criteria

- Increase the average server utilization from current utilization by 20-50%. The higher we are able to push the utilization the more energy-efficient the data center will become. Ideally, we aim to reach 90%, but this depends on the requirements outlined earlier. We increase the energy-proportionality by switching off idle machines.
- Achieve savings of 50% of power consumption compared to a static resource provisioning. The current modus operandi is to leave all servers powered on all the time. The possible savings are restricted by the load variations. Because the pessimistic approach to capacity planning is to

provision for twice the peak load, we believe it possible to achieve savings of 50% compared to a static resource provisioning.

8 Carbon aware scheduling

8.1 Methodology overview

Energy efficiency is less important if sufficient cheap and emission-free energy sources are available. Because energy is a growing cost factor for data center operators, reducing the overall consumption in turn reduces the overall operating expenditures. Coupled with penalties for carbon emissions the urge to cut energy consumption is even stronger. If, however, a cheap and "green" energy source is available, the overall consumption may suddenly be secondary.
When data centers have access to alternative energy sources, say solar and coal, the question of where to process a task is then also dependent on where energy is cheap, plentiful, and green.

8.2 Model

To model the effect of carbon-aware scheduling we introduce a brown energy tax Tax into the cost calculation. The cost per job, $Cost$, is calculated according to the following formula:

$$Cost = PUE * (B * (P_b + Tax) + G * P_g) * L$$

PUE	power usage effectiveness of a data center
B	percentage of brown energy for a data center
P_b	price of brown energy in Euro per kilowatt hour (EUR/kWh)
Tax	brown energy tax in Euro per kilowatt hour
G	percentage of green energy for a data center
P_g	price of green energy in Euro per kilowatt hour (EUR/kWh)
L	length of compute job in hours (h)

This allows us to calculate a projected cost for each job. The job cost varies for each data center as the mix and cost of available energy sources is different for each data center. By incorporating a brown energy tax into the cost we include a mechanism to favor CO2-friendly data centers. The job cost is calculated for each data center and the job is executed in the data center with the least projected costs.

8.3 Baseline

The baseline is a system which is oblivious to the carbon emissions. Jobs are scheduled on availability and cheap computation resources alone. The energy mix used for the computation may incidentally incorporate renewable energy sources, but no conscious effort is made to increase the percentage of renewable energy sources.

8.4 Target success criteria

- Increase the proportion of “green” energy (to “brown” energy used in computing whenever it makes economic sense to compute with “green” energy.

9 Energy efficient storage system

9.1 Methodology overview

The energy-efficient storage system is an object store with a simple interface to get, put, update, and delete objects. Objects are binary data blobs as far as the storage system is concerned. Each object is replicated rep times, where rep is the replication factor. The replication factor is tunable. It allows different trade-offs for data availability and storage overhead. Besides the minimum replication factor rep, there exists additional copies of popular objects. These exist solely to cope with increased read requests. Whenever the aggregated client read throughput exceeds the available bandwidth of live replicas, additional copies are brought online. This ensures that the storage system only consumes energy in proportion to the client demands.

9.2 Model

The minimum number of disks N_{min} required to store all data items is defined by the total data size Sz, the replication factor rep and the disk capacity Cap. The parameters are connected by the following relationship

$$Sz \leq Cap * N_{min}/rep$$

That is the total data size times the replication factor must fit not exceed the drive’s capacity. However, total storage size is only one concern. Disk read and write bandwidth is a second factor. The aggregated client throughput Thr_{client} must not exceed the available total disk throughput Thr_{total} where the total throughput is defined by the individual disk throughput T_{disk} and the total number of disks

$$Thr_{total} = N_{min} * Thr_{disk}$$

Saving energy in the disk subsystem is only possible because of load variations. When the load is low and consequently the client throughput is also low, disks can be spun down. The power consumed by an active disk is denoted by P_{active}. The total power consumption P_{total} of the disk subsystem is defined by the total number of disks

$$P_{total} = N_{min} * P_{active}$$

We save power by deactivating disks. While a disk is deactivated it consumes no power at all. The goal is to deactivate as many disks as possible while still guaranteeing sufficient storage space and read/write bandwidth.

9.3 Baseline

We compare our disk management scheme against a baseline where all disks are constantly powered on, regardless of their utilization.

9.4 Target success criteria

- Reduce energy consumed by 25% if the workload shows sufficient load variability via the ability to turn off individual disks without compromising availability and performance: Because disks only consume about 10% of the total server energy, we expect the absolute savings to be less than what is achievable with our other energy saving mechanisms. Considering only the energy consumed by the disk subsystem we conservatively aim at a 25% reduction if the workload shows sufficient load variability.

10 References

[BB09]	A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, A. Singhania, *The Multikernel: A new OS architecture for scalable multicore systems*, 22nd ACM Symposium on OS Principles, Big Sky, MT, USA, October 2009.
[CMHM10]	E.S. Chung, P.A. Milder, J.C. Hoe, K. Mai,*Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?*, Microarchitecture (MICRO), 43rd Annual IEEE/ACM International Symposium on , vol., no., pp.225-236, 2010
[CUYF13]	A. Cristal, O. Unsal, G. Yalcin, C. Fetzer, J-T. Wamhoff, P. Felber, D. Harmanci, A. Sobe, *Leveraging Transactional Memory for Energy-Efficient Computing Below Safe Operation Margins*, TRANSACT 2013 - 8th ACM SIGPLAN Workshop on Transactional Computing. Texas, 2013.
[DWBS10]	R.G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, T. Mudge, *Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits*, Proceedings of the IEEE , vol.98, no.2, pp.253,266, Feb. 2010
[EUVG09]	O. Ergin, O. Unsal, X. Vera, A. Gonzalez, *Reducing Soft Errors through Operand Width Aware Policies*, IEEE Trans. Dependable Secur. Comput. 6, 3 (July 2009), 217-230.
[GrAn06]	E. Grochowski, M. Annavaram, *Energy per Instruction Trends in Intel Microprocessors*, Technology@Intel Magazine, March 2006.
[MM07]	M. Michael, J.E. Moreira, D. Shiloach, R.W. Wisniewski, *Scale-up x Scale-out: A Case Study using Nutch/Lucene*, Proceedings of IPDPS, 2007.
[Poll99]	F. J. Pollack, *New microarchitecture challenges in the coming generations of CMOS process technologies (keynote address)*. In Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture (MICRO 32), 1999.
[RaRu11]	T. Rauber, G. Rünger, *Modeling the energy consumption for concurrent executions of parallel tasks*. In Proceedings of the 14th Communications and Networking Symposium (CNS '11). Society for Computer Simulation International, San Diego, CA, USA, 11-18, 2011.
[ToNR00]	J. Y. F. Tong, D. Nagle, R. A. Rutenbar, *Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic*, IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 3, June 2000.
[VS10]	A. Vishnu, S. Song et al., *Designing Energy Efficient Communication Runtime Systems for Data Centric Programming Models*, IEEE/ACM International Conference on Green Computing and Communications (GreenCom). Dec. 18- 20, 2010
[WoLe08]	D.H. Woo, H.S. Lee, *Extending Amdahl’s Law for Energy-Efficient Computing in the Many-Core Era*, IEEE Computer, 41(12), 2008.
[YoEr09]	D.H. Yoon, M. Erez, *Memory mapped ECC: low-cost error protection for last level caches*, in Proceedings of the 36th annual International Symposium on Computer architecture, 2009).

[YUCH11] G. Yalcin, O. Unsal, A. Cristal, I. Hur, M. Valero, *SymptomTM: Symptom Based Error Detection and Recovery Using Hardware Transactional Memory*, In 20th International Conference on Parallel Architectures and Compilation Techniques, October 2011.

[ZiKW99] T. Zidenberg, I. Keslassy, U. Weiser, *MultiAmdahl: How Should I Divide My Heterogeneous Chip?*," IEEE Computer Architecture Letters, vol. 11, no. 2, pp. 65-68, July-Dec, 2012.