On the Selection of Adder Unit in Energy Efficient Vector Processing

Ivan Ratković, Oscar Palomar, Milan Stanić, Osman S. Ünsal, Adrian Cristal and Mateo Valero
Barcelona Supercomputing Center
Email: {first}.{last}@bsc.es

***Abstract*—Vector processors are a very promising solution for mobile devices and servers due to their inherently energy-efficient way of exploiting data-level parallelism. Previous research on vector architectures predominantly focused on performance, so vector processors require a new design space exploration to achieve low power. In this paper, we present a design space exploration of adder unit for vector processors (VA), as it is one of the crucial components in the core design with a non-negligible impact in overall performance and power. For this interrelated circuit-architecture exploration, we developed a novel framework with both architectural- and circuit-level tools. Our framework includes both design- (e.g. adder's family type) and vector architecture-related parameters (e.g. vector length). Finally, we present guidelines on the selection of the most appropriate VA for different types of vector processors according to different sets of metrics of interest. For example, we found that *2-lane* configurations are more *EDP (Energy×Delay)*-efficient than single lane configurations for low-end mobile processors.**

I. Introduction

Power dissipation has become the primary design constraint for almost all computer systems. Let us consider two fast-growing markets such as mobile devices and data centers. The main design goal of mobile, battery constrained, devices is to reduce power dissipation and achieve good performance (energy efficiency) while computer systems like data centers suffer from both power density and energy consumption issues. It is still expected that each new generation of microprocessors is better than the previous one. Since frequency does not scale with technology in an energy-efficient way anymore, we need changes at the architectural level that allow faster execution without a power increase [1].

Vector processors are an inherently energy-efficient architecture [2]–[4] for applications that exhibit data-level parallelism (DLP), i.e. that operate on vectors of independent elements. They express DLP in a very compact form, thus removing a lot of redundant work (e.g. instruction fetch, decode and issue). Lee et al. [4] observe that the best vector-based machines are generally faster and/or more energy-efficient than scalar multicore processors. In some cases scalar multicores perform better though at a greater energy cost.

There are server and mobile workloads with a lot of DLP. For example, Facebook's face recognition feature, running on data centers; or Google's offline voice typing system introduced in Android 4.1 (Jelly Bean). Taking all this into account, we consider that low power vector processors are quite interesting designs for mobile devices and servers. These computer system have fairly different design goals than old vector processors that were used almost exclusively for supercomputing. Their designers prioritized performance without caring much about power, and this is not desirable in today's energy-conscious climate. Vector processors require new components that are energy and power efficient.

We perform a design space exploration of the adder unit for vector processors (Vector Adder Unit-VA) as it has a significant impact in the overall performance and power. Addition is one of the most used operations in both general-purpose and application-specific processors and this also applies to vector processors. The adder is a fundamental block of functional units and it is often in the processor's critical path [5]. Therefore, selecting an appropriate adder unit is of crucial importance. To the best of our knowledge, such a study for vector processors has never been addressed in the open literature so far.

There are architecture-related characteristics, specific only for vector processors, that affect adder unit usage. Thus conclusions found for other architectures might not be valid for vector ones. For example, the activity factor of a VA is affected by architectural parameters (introduced in Section II) such as the maximum vector length or number of lanes. Generally, the data of a vector are correlated, so the activity factor of the VA is less than that of an adder unit operating with scalar data. Moreover, the VA is used in a burst fashion (with idle periods), since a single addition instruction implies adding all vector elements in consecutive cycles. This makes clock-gating more efficient as the overhead of its buffers is minimized.

We perform a comprehensive (power, delay, energy, area) design space exploration of VA. The exploration includes design parameters like number of pipeline stages and clock-gating presence, as well as architectural parameters such as the number of lanes. The results are obtained for a modern CMOS low power standard cell library (TSMC40LP [6]) rather than for a high-performance one as we target low power. We propose a complete framework that consist of several simulators at different levels, and we do this exploration mostly with real microbenchmarks (consisting of application data) obtained from vectorized SPEC applications. This is necessary for this exploration as we need to observe how architectural-level parameters (e.g. vector length) affect the circuit-level metrics (e.g. adder's power dissipation) and how circuit-level parameters (e.g. adder's clock cycle) impact the execution time of a microbenchmark. We discover that these observations are not possible with a framework based on random value input.

So far we have not found in the open literature this kind of coupling architectural and circuit simulators intended for adder unit exploration.

The final goal of this design space exploration is to provide guidelines for designers of low power and energy efficient vector processors. We take into account four types of vector processors and propose suitable VA configurations for each one.

In the context of adders there is considerable research on their comparison in the Energy-Delay space [7]–[9]. However, our analysis differs as we include architecture-related parameters and additional ones like clock-gating. Apart from that, our analysis targets vector architectures and the adders for these architectures lead to distinct conclusions. While there is very interesting research on the selection of adder units for particular architectures such as DSPs [10] and crypto-processors [11], their exploration demands are different so they do not utilize the same kind of architectural parameters as we do. Moreover, they do not test the adders with architecture-driven benchmarks and assume adders are always busy.

This paper presents the following contributions:

- Development of a novel framework for exploration of the VA design space taking into account both architectural- and design-level parameters (Section III-A).
- A discussion of adders' power characteristics considering clock-gating and timing stage analysis for various kinds of testbenchs and a modern technology (Section IV-A).
- Study of vector multi-lane usefulness. We found it useful for achieving low *Energy-Delay* products. (Section IV-B).
- Discussion of advantages of using application-based testbenchs over traditional random-data ones (Section IV-C).
- Guidelines on VA configuration selection for different low power vector processors. For example, we discover that vector multi-lane with *1-stage* Brent-Kung adders is an effective approach for mobile devices (Section V).

II. Target Vector Architecture

Vector processors [2] exploit DLP by expressing a lot of operations in a single instruction. For example, a vector add instruction indicates that all the elements stored in the two source vector registers must be added in an independent fashion and written in the destination vector register. The maximum vector length indicates the number of elements stored in a vector register.

Vector ALUs are fully pipelined and the elements of the vector register are streamlined to the unit, one per cycle. This is the main difference between vector processors and Single Instruction Multiple Data (SIMD) multimedia extensions such as AVX [12], which are a common way to exploit DLP. SIMD extensions are typically implemented with multiple ALUs that operate on all independent elements in parallel. Operating on n elements in parallel with n ALUs would be inefficient for vector processors because they typically operate on much longer vectors.

The execution time of a vector instruction is the vector length plus the start-up latency. This time can be reduced with the introduction of multiple vector lanes, i.e. replicated

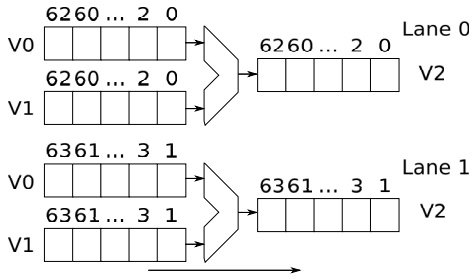

Fig. 1. A 2-lane VA executing `VADD V2<-V0,V1`.

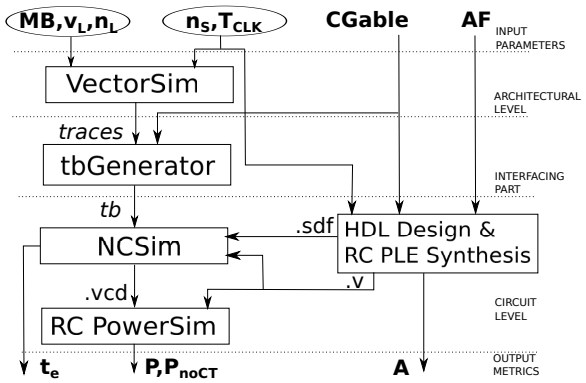

Fig. 2. Block diagram of the framework's steps, parameters, and metrics.

vector ALUs, where each vector lane is connected to a slice of the vector register file. Such slicing reduces the number of ports needed to feed the multiple ALUs. An n-lane VA has n identical adders, and all lanes operate in parallel in a lockstep fashion, thus simplifying control logic. Fig. 1 shows a 2-lane VA of a vector processor with a maximum vector length of 64.

In this paper we assume that our VA is attached to a low power vector processor configuration. Most modern low power processors are in-order due to its simplicity. For this reason, we assume an in-order vector processor. The processor has one ALU per lane and implements chaining (vector equivalent of data forwarding) and dead time elimination (allowing to reuse the ALU immediately after the current instruction). It has separated vector and scalar execution units. The vector unit accesses the L1 cache (shared with the scalar unit).

III. Methodology

This section describes the framework used to perform the design space exploration of VA as well as the framework parameters and the test benchmarks that we use.

A. Framework

The framework is depicted on Fig. 2. It includes architectural- (*VectorSim*) and circuit-level (*RC, NCsim*) simulators and tools, as well as an interfacing tool (*tbGenerator*). For various circuit- and architectural-level parameters (explained in detail in section III-B) we obtain the metrics of interest of our VA (described in Table I).

Since we perform an exploration of the VA using vectorized microbenchmarks (*MBs*), we need to use an architectural simulator of the targeted vector processor. The first step in the framework is to feed our in-house *VectorSim* with the vector

parameters (microbenchmark (*MB*), vector length (V_L), number of lanes (n_L)) as well as some design parameters (number of stages (n_S), clock period (T_{CLK})). This stage generates data and timing traces. The traces include information about vector additions only, excluding scalar additions.

The next step is transforming this architectural-level information into verilog test benchmarks (*tbs*). In order to do this transformation we develop a tool named *tbGenerator*. Apart from the traces, we include here one more parameter (*CGable*) which indicates if the design has clock-gating support. We generate independent *tb* files for each lane of the VA. Section III-C describes the different *tbs* that we generate.

We incorporate design parameters (adder family (*AF*), T_{CLK}, n_S, *CGable*) into HDL codes of adders which are supplied to Cadence RTL-Compiler (*RC PLE Synthesis*) [13] to produce different adder's synthesized mapped netlists. The tool tries to meet timing constraints (T_{CLK}) while prioritizing power over area optimization. In order to meet timing constraints, incremental synthesis optimizations are performed, primarily through gate sizing by selecting adequate gates and buffer insertion. However, we prevent optimizations that could change the initial adder hierarchy, to ensure a fair comparison between different families. For multiple lane VA configurations, adders in all lanes are identical. In this stage we obtain one metric of interest: area (*A*). We use physical layout estimation (PLE) [13] in order to estimate wire delays i.e. to include routing overhead. The most critical paths are verified with Spectre [13] in order to ensure there are no timing violations in the synthesized designs. This step is not shown in Fig. 2 for the sake of simplicity.

We use a 40nm low power standard cell library (TSMC40LP), and we obtain results for *typical* operating conditions. We chose effective current based model (ECSM) [13] as it delivers accuracy to within 2% of SPICE [14]. It is better than conventional models like non-linear delay model (NLDM) which differ as much as 20%. We obtain the results for static CMOS. However, as it is observed in [7], the topology that is the most energy efficient in one logic style, is also the most energy-efficient in the other styles.

The next step is to simulate each synthesized adder in *NCSim* [13] for each matching *tb* with back-annotated delays using standard delay format (sdf) files [15]. This is done in order to obtain the execution time t_e, verify the synthesized designs and extract resulting switching activity information using Value-Change-Dump (vcd) files [13]. The final step of the framework is precise computation of power metrics (P and P_{noCT}) using *RC Power Simulation*. The inputs are synthesized designs in verilog and vcd files.

B. Framework Parameters

We first present the vector processor specific parameters:

- ***MB*** is a vectorized microbenchmark (kernel) extracted from an application and it consists of integer data. It is a representative part of the application and small enough (between 100k and 150k test vectors) to keep circuit simulation time reasonable. We use three different *MBs* extracted from three vectorized SPEC applications (described in Table II) that are used in mobile devices and can also be found in server workloads. In the *MBs* there are long and short vector data, so our application set is comprehensive for our needs. They are addition intensive (in average 27.14% of total instructions executed), so adder's impact on the *MBs* t_e is significant.
- $\boldsymbol{v_L}$ is the maximum vector length of the vector processor. Possible values are 16 and 128 to represent both extremes of short and long maximum vector lengths.
- $\boldsymbol{n_L}$ is the number of lanes. Possible values are 1, 2, and 4. We do not examine more lanes as our initial analysis showed it would not satisfy well a low power core budget.

We now present the design parameters which are primarily needed for the circuit-level part of the framework.

- ***CGable*** indicates whether the design has clock-gating capabilities. Clock-gating is quite common today, and makes pipelining more effective. It prevents useless switching activity in circuits, but we still have almost equal power dissipation in the clock tree. Clock-gating is done at the stage level so we gate pipeline stages separately on demand. We implement clock-gating in the HDL code, and use clock-gating cells from the cell library.
- ***AF*** is the adder family (algorithm). We choose 5 adder families: Brent-Kung (*bk*), Kogge-Stone (*ks*), carry-lookahead (*cla*), ripple-carry adder (*rca*), and conditional sum adder (*cosa*) [16], all implemented in HDL code. We choose *rca* as it is a basic algorithm and it is the simplest to implement (as for each bit it practically requires only one full adder cell), *cla* as a basic model of fast adders, and prefix (*bk* and *ks*) and *cosa* adders as they are fast and suitable for pipelining.
- $\boldsymbol{T_{CLK}}$***(=1/f)*** is the clock period of adder and the whole processor, as double-clocked ALU is unusual, especially in low power design. We define (0.1-5.0)GHz as a practical frequency range for the processor types we look at (Section V), in order to explore a complete design space.
- $\boldsymbol{n_S}$ is the number of pipeline stages, i.e. pipeline depth. *bk*, *ks*, and *rca* adders are pipelined up to 8 stages, *cosa* up to 7 stages, and *cla* is pipelined up to 4 stages. Due to its feedback-like structure further pipelining of *cla* is not fruitful. Our adder designs have input registers, so they can be easily incorporated into a datapath.

TABLE I
METRICS OF INTEREST

Measured Metrics
P and $\boldsymbol{P_{noCT}}$. Average power of adder (or adders if we have more than one lane) including and excluding the clock tree respectively.
t_e. The execution time of a test benchmark *tb*, also referred as Delay (*D*).
A. The area of adder(s).
Computed Metrics
E=PDP. Power-Delay product is total energy spent in adder during a *tb*.
EDP= $\boldsymbol{P^2}$***DP and*** $\boldsymbol{E^2}$***DP=*** $\boldsymbol{P^3}$***DP*** are commonly used Power-Delay products.
P_d. Surface power density = *P/A*. We need this metric as it is related with the temperature of the given surface by Stefan-Boltzmann law.

TABLE II
VECTORIZED APPLICATIONS

Hmmer (SPEC2006) applies profile Hidden Markov Models (HMMs) and is useful in many areas such as speech synthesis, handwriting, gesture recognition, part-of-speech recognition, etc. Most vector data is long.
Facerec (SPEC2000) is an implementation of a face recognition system. Most vector data is short.
H264ref (SPEC2006) is the reference implementation of H.264/AVC standard for video compression. There are both short and long vector data.

C. Test Benchmarks

We generate two kinds of *tbs*: *app-tb* are obtained from real *MBs* and *synth-tb* are synthetic. *app-tb* are a function of all the parameters used in *VectorSim*. On the contrary, *synth-tbs* are not related with the vector simulator and are a function of T_{CLK} only. There are three types of *app-tbs* (*noCG*, *CG* and *100%*) and two types of *synth-tbs* (*rnd* and *0%*).

- ***noCG*** is used to evaluate designs without clock-gating. The input values of the VA are provided for each cycle.
- ***CG*** enables evaluating designs with clock-gating support. Clock-gating is enabled when we have idle cycles. Here we need additional clock-gating signals, one per stage.
- ***100%*** represents a case where the VA is always busy, assuming that memory is fast enough to provide data on time and that consecutive adds are independent. Therefore, t_e depends only on the characteristics of the VA.
- ***rnd*** is a testbench with random values that are supplied each cycle to the adder. This is the traditional methodology of testing digital designs.
- ***0%*** is used to evaluate the case when there is no vector additions in the MB, i.e. clock-gating is always active.

CG and *0%* are supplied to the adders with clock-gating, while others are supplied to adders without clock-gating logic integrated.

IV. DESIGN SPACE EXPLORATION

In this section we first analyze the adders' characteristics and then the effectiveness of multiple lanes, showing their behavior on the measured frequency range using averaged results of all mentioned *MBs*. The execution times (t_e) for all *MBs* are normalized before averaging in order to assure the correctness of averaged results. At the end of the section we present the major drawbacks of using a synthetic benchmark of random data for our research.

A. Adders Characteristics Discussion

We compare the power characteristics of the examined adder families by comparing their power dissipation for the measured frequency range. For a given frequency and adder family we plot the lowest power configuration (Figure 3). The goal here is to examine power dissipation of adders alone so we test configurations with one lane and do not include the clock tree power. Results for v_L=16 and 128 are almost the same, so we do not comment on them separately. The only exception is *CG* where for v_L=128 we have slightly higher power dissipation. It is due to better exploitation of VA, so the clock-gating mechanism is active less percentage of time than for v_L=16.

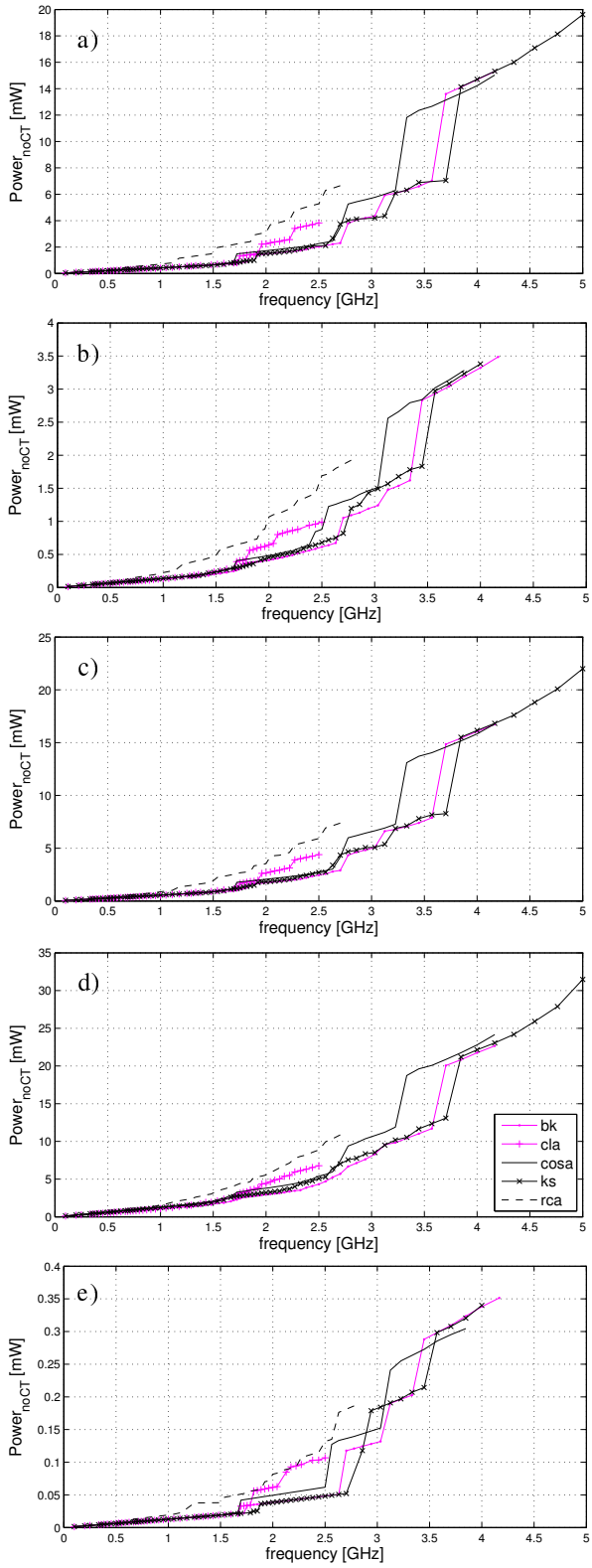

Fig. 3. Power dissipation of analyzed adder families for the following test benchmarks: a) *noCG*, b) *CG*, c) *100%*, d) *rnd*, and e) *0%* for v_L=128.

noCG: As we can see in Fig. 3a, power dissipation is highly dependent on the number of pipeline stages. For a given frequency, the lowest power configuration is always the one with the fewest stages. On the graph lines we can observe

steps, that occur when the number of stages increases for a given adder family. *bk* adder family is the most efficient one except for the lowest and highest frequency ranges. For the lowest frequencies, until 0.57GHz, *rca* is slightly better than others due to its simplicity. However, it requires more stages to achieve a given frequency than other adder families because of its intrinsically poor timing characteristics. Therefore, it is worse than the other designs for higher frequencies. For the highest frequency range, *ks* is the best as it is the only one that can satisfy the frequency range from 4.17GHz to 5GHz. There is a short range (3.85GHz-4.17GHz) where *cosa* is the lowest power adder.

CG: Clock-gating reduces power dissipation roughly 5x (Fig. 3b). Designs are slightly different in this case as clock-gating logic is incorporated in baseline designs used in the previous case. As a side effect, timings are a bit worse, especially for *ks* adder family, which is not the fastest adder as in the previous case. In fact, it can achieve "only" 4GHz, causing *bk* to perform the best in both mid- and high-range.

100%: The graph (Fig. 3c) is fairly similar to *noCG* one, but power consumption in general is around 10% higher than for *noCG* case as the activity factor here is higher.

rnd: As we can see in Fig. 3d, power dissipation is significantly higher than for any *app-tb* case, even for *100%*, which is the most similar to *rnd*, as in both cases we have data constantly supplied to the adder inputs. Results are up to 2x higher than ones obtained with *noCG*, and up to 10x higher that *CG* ones. The reason for this difference is that random data is not correlated, which makes the activity factor higher. This is particularly emphasized in *cosa* as its conditional logic suffer from excessive power dissipation when it is driven by uncorrelated values. A higher activity factor implies that logic uses relatively more power than in *app-tbs* (diminishing pipelining relative overhead) and the slope of the *(f, P_{noCT})* graph is higher. As a result, power gaps between stages are smoother than in previous cases.

0%: This case (Fig. 3e) actually shows the maximal possible power reduction using clock-gating. The greatest consumer in this case is clock-gating logic. The size of this logic is directly proportional to the number of stages - number of FFs to be gated. As a result, power dissipation is practically defined by the number of pipeline stages. Therefore, for a given frequency and number of stages two adders have quite similar power dissipation. Power is around 10x lower than for *CG*.

General Observations: Despite the glitch reduction achieved with pipelining [17], the overhead of adding registers is too high. Even with clock-gating presence, it is not possible to compensate it. Thus, for all 5 *tbs* we find that, for a given frequency, the adder with the lowest number of stages has the lowest power. *bk* is generally the most power efficient one. Although *ks* and *cosa* can provide high frequency, due to its high area, they suffer from long wires that cause excessive net power dissipation. *rca* is the most power and area efficient for low frequencies, and this makes it suitable for low power and low performance requirements where we can hide its long carry propagation delay.

TABLE III
POWER DISSIPATION RATIO OF 2- AND 4- OVER 1-LANE VA

f[GHz]	1.0				4.0			
n_L	2		4		2		4	
v_L	16	128	16	128	16	128	16	128
noCG	1.99	1.98	3.95	3.88	1.98	1.99	3.94	3.96
CG	1.79	1.74	3.1	2.9	1.68	1.65	2.69	2.62
100%	2.12	2.06	4.63	4.17	2.06	2.02	4.29	4.08

From an implementation point of view, the most relevant results are for *CG* (Fig 3b). Power dissipation of the adders with clock-gating ranges from 11.68μW for 0.1GHz and 3.49mW for 4.2GHz, which is adequate for the low power design demands.

B. Multi-lane Effectiveness

In order to examine the fruitfulness of multi-laning over the measured frequency range we look for *P*, *E*, *EDP*, and *ED^2P* characteristics of *1-*, *2-*, and *4-lane* VAs. For a given combination of v_L, *app-tb* and *f*, we pick the configuration with the lowest power (including clock tree) for each n_L.

Table III shows ratios of power dissipated by *2-* and *4-lane* configurations over single lane power for 1GHz and 4GHz. For *noCG* having twice n_L causes doubling power almost independently from v_L and *f*. On the contrary, for *CG*, doubling the n_L increases power by less than 2x. The reason is that addition is faster while the rest of the hardware is the same, so a higher percentage of time the VA is clock-gated. Due to the same reason the ratio decreases when frequency increases. For *CG* and *100%* multi-lane is more power efficient for longer v_L, especially for 4 lanes. There is a correlation between data inside a vector, and with multi-lane we actually slice a vector into n_L shorter vectors, which causes decrease of the correlation and increase of the activity factor. Due to this reason having n_L lanes causes power increase more than n_L times in *100%*. This trend decreases with frequency as for higher frequencies we need to use adders with more stages. By having more stages a design has less glitches, which means that a decrease of data correlation has less effect in terms of glitching increase for *8-* than for *1-stage* adder.

Table IV shows the speed-up of the multi-lane configurations over single lane both for 1GHz and 4GHz. As it consists of additions only, in *100%* doubling the number of lanes halves the t_e. For *noCG* and *CG* we have the same result, as clock-gating does not affect t_e for a given frequency. For these *app-tbs*, the speed-up linearly decreases with frequency, so adding more lanes is more productive for lower frequencies (low power region). This happens because the parallelism offered by multiple lanes has lower effect, as the *app-tb* do not have only arithmetic instructions (*Amdahl's law*), and cache and main memory latencies become more noticeable at higher frequencies. Cache latency is more difficult to be hidden with short v_L, so speed-up decreases with frequency more in this case.

Results for *PD^nP* are functions of *P* and t_e, thus they are directly explained with the discussion above. We just highlight the most interesting observations. While for *100%* all lane

TABLE IV
SPEED-UP OF 2- AND 4- OVER 1-LANE VA

f[GHz]	1.0				4.0			
n_L	2		4		2		4	
v_L	16	128	16	128	16	128	16	128
CG, noCG	1.41	1.52	1.74	1.97	1.29	1.42	1.48	1.73
100%	2	2	4	4	2	2	4	4

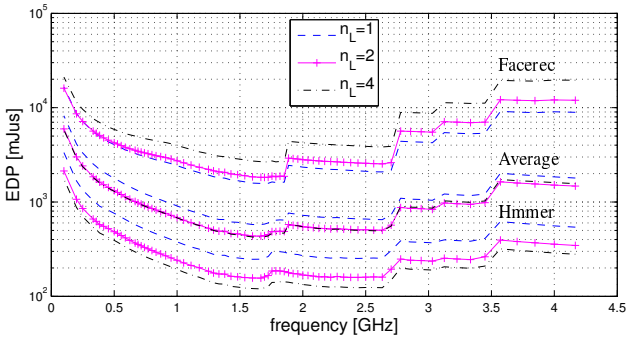

Fig. 4. *EDP* of *1-*,*2-*, and *4-lane* VA for *CG* and v_L=128.

configurations have the same energy independently of f and v_L, for other *app-tbs* the single lane configuration is the most energy efficient one. Adding more lanes becomes more fruitful if we look for performance, so *2-* and *4-lane* VAs are in most cases better solutions than *1-lane* for *EDP*, especially for *CG* and long v_L. While for other metrics averaged results lead to the same conclusions as individual *MBs'* results, we observe that adding lanes improves *EDP* for long-vector *MBs* (Hummer), while it is the opposite for short-vector ones (Facerec) (Fig. 4). Results for H264ref are similar to the averaged ones. For ED^2P, *4-lane* VA is better than *1-* or *2-lane* for all v_L and *app-tbs*. The *(f,* ED^nP*)*(n>1) graphs have the minimum around 1.67GHz and, as it is shown in the next section, this is actually the most energy-efficient operating region.

C. Application-based vs. Random Data Benchmarking

We found the following problems with the traditional random data-based benchmarking:

- The main problem is that we cannot measure t_e, so we do not know how some design and architectural parameters affect t_e of *MBs*. Therefore, we cannot speak about the energy spent (or *EDP and* ED^2P) during a MB execution.
- We cannot incorporate architectural-level parameters (v_L and n_L) into power measurement, so we cannot analyze their effects on P and how this changes with frequency.
- By supplying random data instead of real ones, we overestimate power dissipation as it is proved by comparing adders power of *rnd* and *100%* cases. This happens because the value range of real data is "narrow", especially inside a vector, so activity factor is lower.
- Additionally, there are not idle cycles, so we are not able to analyze power of tested designs with clock-gating.

V. VA SELECTION GUIDELINES

In this section we provide some guidelines, derived from our results, on the choice of the best VA for the following types of low power vector processor:

- **Embedded.** A vector processor of this type would resemble "classic" embedded processors from TI and ARM that could be used in embedded signal processing solutions, including portable devices in audio, voice, communications, medicine, etc. Assumed frequency range is (0.1-0.4)GHz. The requirements are long battery life (E), as we need autonomous work and battery often is not rechargeable, and small area (A). Therefore, the desired VA should have a low product of E and A. Here power is indirectly limited with energy and frequency requirements, so there is low risk of temperature violation.
- **Low-End Mobile.** The reference in this case is low-end ARM-like processors used in various handheld devices like GPS navigations and low-end smartphones. Targeted frequency range is (0.33-0.8)GHz. This kind of processors needs fast response (t_e) as it is a soft real time system, and long battery life (E). Therefore *EDP* is the right metric here as it usually is for this kind of low power devices [18]. Additionally, we need to maintain low P and A in order to satisfy processor's power/area budget.
- **High-End Mobile.** Here we assume a vector processor that resembles current processors like high-end ARM, Intel Atom, or AMD Bobcat which can be found in high-end smartphones, tablets, and notebooks. Assumed frequency range is (0.8-1.8)GHz. We have similar requirements like for low-end mobile processors with the difference that we stress performance more, so the desired metric is ED^2P [18].
- **Low Power - High Throughput.** Targeted vector processor in this case would have similar design goals as low power processors for servers like AMD Bulldozer, PowerPC A2 or Sun SparcT3. The frequency range of interest is (1.6-4.2)GHz. We took a pretty wide frequency range as from one side we target low frequency processors like the mentioned ones, but from the other side we also want to explore possibilities and trends of high frequency range. In general, when we consider server design we always need to care about performance (t_e). Additionally, we distinguish and consider two cases. In the first case we assume that the server is always busy so we need to care about power dissipation P as it is constantly on, so the targeted metric is E. However, if the server is supposed to have idle periods, during which it goes to low power states, then we usually want to finish the job as soon as possible and go to one of these states. Therefore, in this case we are interested in ED^2P.

We only consider designs with clock-gating support as today it is standard in any low power and energy efficient design. Results for each processor type are observed separately for short (16) and long (128) v_L.

Fig. 5 shows the best design points according to the metric of interest for three types of processor. For example, if *EDP* is the metric of interest, for each t_e observed, we select the configuration with the least E. We discard slower designs that

consume more energy than faster ones. For other metrics of interest configurations are selected in a similar manner. The figure actually shows the t_e vs. P, E, or EDP trade-offs and the black curves indicate the best observed result for the metric of interest. The embedded case is not shown as there is a single solution. There are graphs for both cases of server processors.

Table V presents detailed information of the most relevant design points according to the metrics of interest discussed above. The best design for a given metric is highlighted using a bold font in the metric column. The table also includes some alternative design points that trade off the desired metric with significant gainings in other metrics. This is indicated with italic fonts in the metrics involved. Effect of parameters on a particular metric are explained in Section IV.

As it was observed in Section IV-A, *1-stage bk* adders are usually the best design choice. However, when speed is not critical, like in embedded vector processors, *1-stage*, *1-lane*, *rca* VA is the best choice due to its simplicity. It confirms that where the performance requirements are pretty low, *rca* is the most area and energy efficient choice [19]. When we stress performance, *2-stage* adders are more desirable, as we cannot achieve high frequencies with single stage. For high speed designs, *ks* adders are also an interesting solution.

When we consider low-end mobile vector processors, we can confirm *1-stage bk* as the most efficient solution. The VA configurations shown in Fig 5a appear in series that correspond to different n_L values. In terms of *EDP*, *2-lane* configurations perform the best. For v_L=128, configurations with 4 lanes are more desirable as we can exploit them better than for v_L=16. When we deal with high-end mobile vector processors, *2-* and *4-lane* configurations provide lower ED^2P than single lane ones. With long v_L, *4-lane* configurations are the only ones to be considered.

For power-constrained servers, where demands are conservative in terms of t_e, single lane configurations are usually the most desirable due to their low power. This still remains true for long v_L. However, when we consider VA design for fast, low power servers, multi-lane is as effective as in high-end mobile case. In this case, *2-stage* adders running around 2.6GHz perform the best in terms of ED^2P for both long and short v_L.

In terms of the frequency we can identify two efficient operating ranges. The first one is around 1.67GHz while the second one lies around 2.6GHz. Until these points we can consider increasing frequency as a way to achieve energy efficient speed-up, but after that adding more lanes becomes a more fruitful solution.

VI. Conclusions

The presented design space exploration addresses the issue of optimal VA selection in vector processor design. A novel, multi-level (circuit and architectural) methodology, on identifying the best VA is presented. We showed how design (e.g. frequency) and architectural (e.g. maximum vector length or number of lanes) parameters can direct the selection of the VA design. We reevaluate various adder's families characteristic with this novel methodology and showed that multi-lane approach is efficient in terms of EDP and ED^2P, especially with clock-gating and longer vector lengths. Additionally, we showed advantages of our benchmarking method over the traditional random values-based one for this exploration. Finally, as the final goal of this research, we propose suitable VA configurations for four vector processor types according to several metrics of interests. We found that *2-* and *4-lane* configurations with single stage *bk* adders are optimal for mobile, while *1-lane*, *rca* configurations perform the best for embedded vector processors. However, *4-lane* configurations with *2-stage bk* adders running around 2.6GHz are optimal solutions for VA if we target high-performance, low power server vector processors.

Acknowledgments

The authors would like to thank Oscar Gustafsson for providing us with the base HDL code for some of the adders used in this paper. We would also like to thank Carlos Alvarez for his useful advice and fruitful discussions. This work was partially supported by the cooperation agreement between the Barcelona Supercomputing Center and Microsoft Research, by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contracts TIN2007-60625 and TIN2008-02055-E, and by the European Network of Excellence on High- Performance Embedded Architecture and Compilation (HiPEAC).

References

[1] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, pp. 67–77, May 2011.
[2] K. Asanović, "Vector microprocessor," 1998, *PhD Thesis*, UC Berkeley.
[3] C. Lemuet *et al.*, "The potential energy efficiency of vector acceleration," in *SC '06*, p. 1.
[4] Y. Lee *et al.*, "Exploring the tradeoffs between programmability and efficiency in data-parallel accelerators," in *ISCA '11*, pp. 129–140.
[5] C. Zhou *et al.*, "64-bit prefix adders: Power-efficient topologies and design solutions," in *CICC '09*, pp. 179 –182.
[6] TSMC. [Online]. Available: http://www.tsmc.com/
[7] D. Patil *et al.*, "Robust energy-efficient adder topologies," in *ARITH '07*, pp. 16–28.
[8] S. Sun and C. Sechen, "Post-layout comparison of high performance 64b static adders in energy-delay space," in *ICCD '07*, pp. 401 –408.
[9] V. Oklobdzija *et al.*, "Comparison of high-performance vlsi adders in the energy-delay space," *VLSI Trans.*, vol. 13, no. 6, pp. 754–758, 2005.
[10] Y. Liu and T. Zhang, "On the selection of arithmetic unit structure in voltage overscaled soft digital signal processing," in *ISLPED '07*, pp. 250–255.
[11] S.-W. Heo *et al.*, "Study of optimized adder selection," in *ASIC '03*, vol. 2, pp. 1265 – 1268.
[12] N. Firasta *et al.*, "Intel® AVX: New frontiers in performance improvements and energy efficiency," White Paper, Intel Corporation, May 2008.
[13] Cadence Design Systems. [Online]. Available: http://www.cadence.com/
[14] Cadence Design Systems, "Encounter library characterizer datasheet."
[15] Standard Delay Format. [Online]. Available: http://www.eda.org/sdf/
[16] M. Ercegovac and T. Lang, *Digital Arithmetic*. MKP, 2003.
[17] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," in *MICRO 35*, pp. 333–344.
[18] M. Monchiero *et al.*, "Design space exploration for multicore architectures: a power/performance/thermal view," in *ICS '06*, pp. 177–186.
[19] S. Ghosh and K. Roy, "Exploring high-speed low-power hybrid arithmetic units at scaled supply and adaptive clock-stretching," in *ASP-DAC '08*, pp. 635–640.

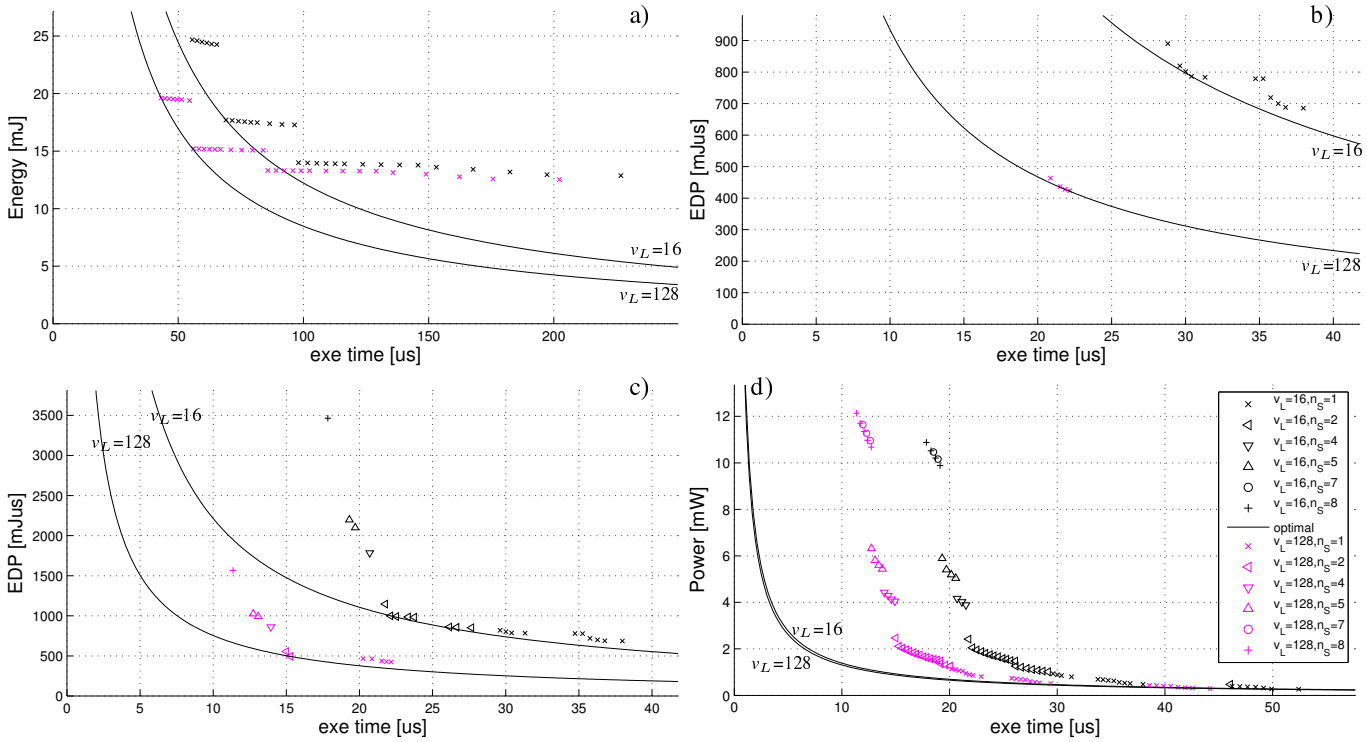

Fig. 5. Trade-off graphs of E (a), EDP (b), EDP (c), and P (d) vs. t_e for low-end mobile, high-end mobile, fast low power servers, and power-constrained servers respectively. Hyperbolic curves ("optimal" in the legend) represent EDP (a), ED^2P (b), ED^2P (c), and E (d) of the best design points for both 16 and 128 v_L.

TABLE V
OPTIMAL VA UNIT CONFIGURATIONS

Vector Processor Type	Input Parameters					Output Metrics						
	v_L	n_L	AF	n_S	f[GHz]	$t_e[\mu s]$	P[mW]	E[mJ]	EDP[mJμs]	ED^2P[mJμs^2]	$A[\mu m^2]$	P_d[mW/m^2]
Embedded (0.1-0.4)GHz	16	1	*rca*	1	0.1	749.1	16.79x10^{-3}	**12.58**	9422	7.06x10^6	**281.2**	36.0
	128	1	*rca*	1	0.1	668.6	18.35x10^{-3}	**12.27**	8204	5.48x10^6	**281.2**	41.53
Low-End Mobile (0.33-0.8)GHz	16	2	*bk*	1	0.8	69.08	0.2562	17.7	**1223**	84.47x10^3	792.6	187.5
		4	*bk*	1	0.8	**55.62**	0.4436	24.67	1372	76.33x10^3	1585.0	144.1
		1	*rca*	1	0.33	226.8	56.76x10^{-3}	**12.87**	2919	66.18x10^6	309.4	111.0
		1	*bk*	1	0.8	*98.05*	0.1427	***13.99***	1371	0.134x10^6	396.3	224.2
	128	4	*bk*	1	0.8	**43.21**	0.45369	19.61	**847.5**	36.62x10^3	1585.0	150.6
		2	*bk*	1	0.8	56.12	0.2711	15.21	***1143***	47.91x10^3	*792.6*	206.3
		1	*rca*	1	0.33	202.2	61.9x10^{-3}	**12.52**	2532	51.2x10^6	309.4	127.6
		1	*bk*	1	0.8	*85.8*	0.1552	***13.32***	1143	98.03x10^3	396.3	255.9
High-End Mobile (0.8-1.8)GHz	16	4	*bk*	1	1.67	30.4	0.851	25.87	786.5	**23.91x10^3**	2244.0	255.5
		4	*ks*	1	1.8	**28.8**	1.073	30.91	890.3	25.64x10^3	3847.0	205.8
		2	*bk*	1	1.6	37.97	0.4753	18.05	**685.3**	26.02x10^6	1064.0	317.5
	128	4	*bk*	1	1.69	21.85	0.8958	19.57	427.7	**9.34x10^3**	2477.0	249.2
		4	*ks*	1	1.8	**20.86**	1.065	22.22	463.5	9.67x10^3	3847.0	203.7
		4	*bk*	1	1.67	22.17	0.8623	19.12	**423.8**	9.40x10^6	2244.0	260.6
Power-Constrained Servers (1.6-4.2)GHz	16	1	*bk*	1	1.69	49.9	0.2779	**13.87**	692	34.53x10^3	619.2	373.8
		4	*bk*	8	4.2	**17.84**	10.88	194.1	3464	61.79x10^3	8327.0	747.9
		4	*ks*	5	3.45	***19.71***	*5.405*	*106.5*	2100	41.39x10^3	5195.0	621.3
		1	*bk*	1	1.6	52.42	**0.2686**	14.08	738	38.69x10^6	531.9	375.6
	128	1	*bk*	1	1.6	44.2	**0.2957**	**13.07**	577.6	25.53x10^3	531.9	426.6
		4	*bk*	8	4.2	**11.36**	12.14	137.9	1566	17.79x10^3	8327.0	898.5
Fast Low Power Servers (1.6-4.2)GHz	16	4	*bk*	2	2.63	22.12	2.045	45.24	1001	**22.14x10^3**	3877	372.9
		4	*bk*	1	1.6	37.97	0.4753	18.05	**685.3**	26.02x10^6	1064.0	317.5
	128	4	*bk*	2	2.63	15.29	2.11	132.26	493.3	**7.54x10^3**	3877.0	389.5
		4	*bk*	1	1.67	22.17	0.8623	19.12	**423.8**	9.40x10^6	2244.0	260.6