Exploration of Low Power Adders for a SIMD Data Path

G. Paci
IMEC and DEIS, U. Bologna
gpaci@deis.unibo.it

P. Marchal
IMEC
marchal@imec.be

L. Benini
DEIS, U. Bologna
lbenini@deis.unibo.it

Abstract – Hardware for Ambient Intelligence needs to achieve extremely high computational efficiency (up to 40GOPS/W). An important way for reaching this is exploiting parallelism, and more specifically data-level parallelism enabled by SIMD. Whereas a large body of research exists on the benefits *of*, the architectural design *of* and compilation *onto* SIMD, the design of energy-optimal functional units for SIMD has received limited attention. It appears that existing SIMD functional units are designed in an area optimal, but not energy optimal way. By exploiting the difference in critical path length for the types of operations (e.g., 4x8/2x16/1x32), SIMD adders can be developed that save up to 40% of energy. In this paper, we will present these adders, the issues of building them and quantify their benefits for different usage scenarios and operating frequencies.

I Introduction

Ambient intelligence is driven by low power, real-time, digital systems. Economics demand that these systems are made at a low cost (in terms of NRE and manufacturing) and that, even more importantly, they are introduced in due time onto the market. Given both constraints, flexibility and programmability is key. Unfortunately, these features come at an important energy penalty.

It is well known that parallelism, and in particular SIMD can partially recover the loss in energy efficiency due to flexibility. Firstly, more parallelism increases performance, which can be exchanged for energy savings in many ways (e.g., by voltage scaling). Secondly, SIMD reduces the cost associated with instruction decoding.

A SIMD unit is usually derived from an existing data path. The data path is basically subdivided in units operating on smaller words. This explains the origin of its name: a subword parallel data path.

Whereas many related work exists on the benefits of and compilation techniques for subword parallelism, the implementation of subword parallel data paths for energy has received limited attention.

Subword parallel data paths require dedicated adder/shuffler/multiplier/logic units that process heterogeneous operations. E.g., an adder may have to operate on 4x8- or 2x16- or 1x32-bit words. For this purpose, 4x8-bit adders are usually cascaded with multiplexers into a 32-bit unit, which can be programmed to perform all three types of operations. During synthesis the gates of the design are sized such that the longest critical path of the design, in case the critical path through all units composing a 32-bit operation, meets the delay constraint. As a result, the adder is not energy-efficient for performing 8-bit operations, which have a shorter critical path. Indeed, for 8-bit operations smaller gates suffices and energy can be saved: 4x8-bit operations on an 8-bit ripple adder consume 1.8 times less compared 1x32-bit operation on a 32-bit adder.[1]

The contribution of this paper is to increase the energy efficiency of SIMD adders by exploiting these differences in critical path length. The benefits of combining dedicated adders for the different operand lengths into a low power subword parallel adder are explored. The optimized adder is composed of adders of different types (ripple, carry-look-ahead, brent-kung, etc.) and/or having different gate sizing. The most energy-efficient SIMD adder is identified for different operating frequencies and several operating scenarios (how many times each operator is used). Experimental results obtained with Physical Compiler are used for quantifying the energy benefits of this approach and to precisely characterize its limitations. The enhanced SIMD adders can save up to 40% energy.

This paper is organized as follows: first, the related work will be described (see section II. Related Work); thereafter, the explored adders will be described in detail (see section III. Exploration Space), experimental results will be provided for quantifying the energy benefits (see section IV Experimental Results) and finally, the main conclusions of this paper are summarized.

II. Related Work

Historically, SIMD finds its roots in vector processing where it was investigated for increasing performance [3]. The same ideas have been exploited for improving energy efficiency on embedded and (VLIW) cores (e.g., [2], Trimedia, ASPROCORE, ARM11 SIMD, the cell's SPE cores, …). The key idea behind these extensions is the exploitation of subword parallelism in a SIMD fashion.

Two factors determine the computational efficiency of subword parallel units:

1. the capability of mapping code on the subword parallel units. This depends (1) on the instruction set and (2) to the extent that the designer can detect parallelism inside the application's code and map that

[1] Assuming a ripple adder operating at 100Mhz.

1-4244-0630-7/07/$20.00 ©2007 IEEE.

on the available units. Today, SIMD is mainly been adopted through the use of assembly libraries and compiler intrinsics for media-rich applications (e.g., [1]). To reduce the programming effort, research is ongoing for developing optimizing compilers that can exploit (better) subword parallelism [4] [6]. An important part of this work focuses on managing the data stream into the SIMD unit [9] [8]. Besides optimizing the data layout of the software, hardware extensions for alleviating the memory access bottleneck have been proposed too (e.g., [7]), but they come at the cost of extra hardware, thereby often consuming more power too.

2. the efficiency of the hardware of the subword parallel unit. For VLIW architectures, subword parallelism is mostly implemented using the same resources as the ILP units. $m \times n$-bit functional units are cascaded together with multiplexers into a SIMD unit [5]. This approach is not optimal for energy. It's well known that depending on the operating frequencies and operand lengths, different adder sizes and adder type are more energy optimal.

The contribution of this paper is to reduce the energy cost of a sub-word parallel unit, by exploiting the difference in critical path for several operands-lengths. In the following section, we describe the explored adder types for SIMD more in detail.

III. Exploration Space

Two templates for SIMD adders are explored throughout this paper: (1) we use as a reference a typical SIMD adder, in which a k-bit functional unit is split in $n \times m = k$ operands and (2) we propose our energy-optimized design, a SIMD template where multiple adders are combined into a single unit. Both templates are discussed in the following subsections.

SIMD Adder By Cascading Smaller Adders

Typically, SIMD adders are built by cascading subunits for the smallest operand length into a larger structure. E.g., k-bit adder is composed of $k=n \times m$-units, where the carry output of each unit is conditionally fed into the next unit. In case that the single units can perform the programmed operation, a zero(one)-signal is applied to the carry input of each unit.[2] In case that more units are required for computing the sum, multiplexers combine two or more adders, feeding the carry signal from one unit into the next one. The template of a cascaded SIMD adder is shown in Figure 1. It consists of three parts: the input control unit, the output control unit and the adders themselves. The Input Control Unit (ICU) has two functions: (1) It provides support for subtractions (A-B). It negates B for this purpose. Moreover, it contains the necessary logic for providing the correct carry inputs to each adder unit. The user should set the *Sub*-signal high for activating subtractions. (2) It joins the adders together based on the selected operand length. The user should use the *U* signal for this purpose.

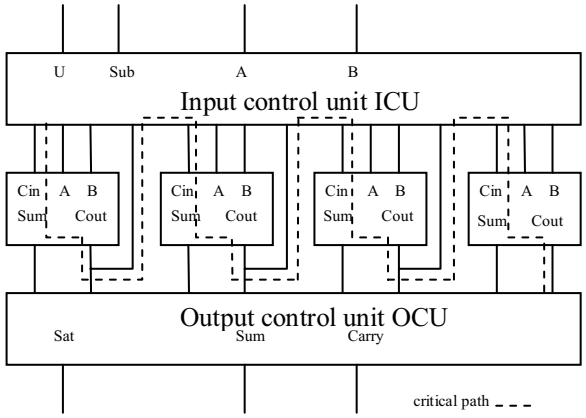

Figure 1 Classical SIMD Adder: mxn-bit adder units are combined into a programmable SIMD unit.

The Output Control Unit (OCU) sets the right carry output vector. E.g., if two 16-bit operations are performed and both have carry, the 32-bit carry vector will be set as [0x0101]. Besides, the OCU supports saturation of the sum vector in case of over-flow. This is a useful option for signal processing applications. The saturation fixes the result to maximum/minimal value in case of over-flow/under-flow.

Finally, our design contains the adders units themselves. We have explored 4x8-bit cascaded SIMD adders based on Brent Kung (bk), carry look ahead (cla), carry look ahead select (clsa), ripple carry (rpl) and ripple carry select (rpcs) adders. All these adders were realized with the DesignWare Synopsys library and were optimized for a range of target frequencies. The synthesis was steered to reach the most energy-efficient design, satisfying the timing constraints.

Compared to a simple adder design, there is considerable area overhead for providing the SIMD functionality. The area breakdown is as follows for a SIMD adder operating at 300Mhz and based on Brent Kung adders: the ICU, OCU and adders occupy respectively 14.7%, 26.7% and 58.4 % of the total area. 42.4% of the critical delay is spent in the ICU and OCU units. The adders burn 60% of the total power.

In the next section, we will discuss a more energy-efficient SIMD adder.

SIMD Adder by Combining of Adder Units For Different Operand Lengths

In this section, we first indicate the potential benefits of combining several adders optimized for different operand lengths. Thereafter, we explain an architectural template for exploiting this idea.

The potential energy benefits

To analyze the potential energy benefits, we compare the power consumption of three usage scenarios of a SIMD adder. In the first case, we assume that the SIMD adder

[2] The carry input is set zero (one) for additions (subtractions).

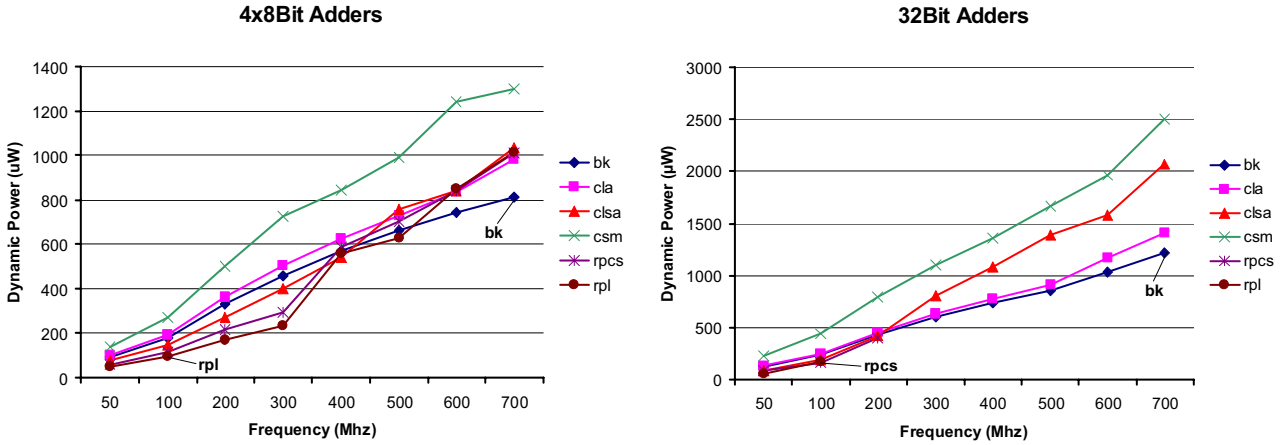

Figure 2 The power consumption of adder types for varying operating frequencies: (left) 4x8-bit additions and (right) one 32-bit addition. Note that the energy consumption of 4x8bit additions is always cheaper than equivalent 1x32-bit addition operated on the same type of adder. Also note that for different operating frequencies, different adder types are optimal. For instance, at 100Mhz ripple carry adders are most energy efficient whereas at 700 Mhz bk are the best ones.

performs only 32-bit operations, and thus actually corresponds to a normal functional unit. In Figure 2 (right), we present the energy consumption of this scenario at different operating frequencies. At 100Mhz, a 32-bit operation on a ripple carry select adder is the most energy efficient It consumes a 160µW per operation.

In the second case, we assume that the SIMD adder performs four 8-bit operations. Hence, the SIMD adder may be composed of four, small 8-bit adders, each optimized for energy. No support is needed for other types of operations such as one 32-bit operation or two 16-bit operations. The most energy efficient SIMD adder is now composed of four 8-bit ripple adders, which together burn 94µW. Hence, compared to a single 32-bit operation, this case consumes 42% less. This difference can be explained by the fact that the logic depth of the 8-bit operators is lower. Smaller and thus more energy-efficient adders suffice for achieving the delay budget.

As a last scenario, assume that respectively 50% and 50% of 8-bit/32-bit operations are performed. Usually, this case is supported with a configurable SIMD adder (as discussed in the previous subsection). A 32-bit adder (see cascade one) is partionned in four 8-bit units and sized such that the path delay on the long 32-bit carry path meets the delay budget. As a result, the 8-bit operations consume as much power as the 32-bit ones, whereas it's clear from scenario 2 that these operations could be performed with 42% less power (when operating at 100Mhz).

To alleviate this power cost, we propose below to combine adders optimized for different operand lengths into a single SIMD adder.

Note that at higher operating frequencies, different adders are energy-optimal, indicating that different combinations of adders are needed for building an energy-efficient SIMD adder. E.g., whereas ripple carry adders are most efficient at 100Mhz, Brent Kung adders for both 8-bit and 32-bit operations burn the least power at 700Mhz. However, both the 8-bit and 32-bit adders are not the same: they have a different gate sizing. The four 8-bit Brent Kung adders occupy 27% less area than a 32-bit one. As a result, they consume up to 44% less energy at 700Mhz (compare the Brent Kung power consumption in both the left and right Figure 2 for the 700Mhz operating point).

An energy-efficient SIMD adder template

In the next paragraphs, we explain the architecture template of the proposed SIMD adders, indicating the overhead of combining adders and discussing the exploration space in detail.

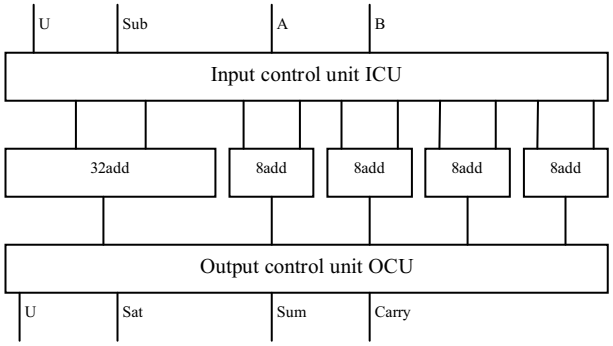

Fig. 3. Instance of an energy-efficient SIMD adder template. In this case, an energy-optimal 32-bit adder is combined with four separate 8-bit adder units. The ICU prevents activities on operand inputs A&B from propagating to the non-selected adder. The OCU selects the result from the correct adder units.

The template for the combined SIMD adders is composed of the Input control unit (ICU), one Output control unit (OCU),

and adders dedicated to each operand length. For instance, one 32-bit adder and four 8-bit adders are combined in Figure 3.

The ICU controls again the operation mode of each adder unit: (1) it selects the set of adders performing the operations. This mainly consists of isolating the operands of the non-selected units. As the adders are duplicated, the wires of the operand signals A&B are forked to respectively 4x8-bit units and the 1x32-bit unit. If the operating mode selects one set of adders, the activities on the input wires should not propagate to the other units. E.g., as 32-bit operations only use the 32-bit adder, the inputs to the four 8-bit adders should not be active at the same time. Therefore, the ICU conditionally latches the inputs to each of the adders, isolating their operands. Obviously, a power/area penalty needs to be paid for these latches, which we quantify in the next section; (2) the ICU differentiates between addition/subtraction operations by setting the input carry bit and negating the B-signal (if necessary).

The OCU selects the correct output from the different adders and builds up the carry output vector. It also provides support for output saturation (see subsection SIMD Adder By Cascading Smaller Adders).

As explained before, depending on the usage conditions different adders have to be combined for building a more energy efficient SIMD adder. The most important combinations that we have tested are indicated in Table 1.

For the 32-bit adders, the following adders were selected Brent Kung (bk), the carry look ahead (cla) and the carry look ahead select (clsa). They are the most energy-efficient adders over a large frequency range (see also Figure 2). Each of these adders is combined with either an 8-bit adder of the same type, a 8-bit ripple carry (rpl) and ripple carry select (rpcs) adder.

All the SIMD adders are implemented with the Synopsys DesignWare library and synthesized for different frequency targets while optimizing for power.

n°	32bit adder	8bit adder
1	Bk	Bk
2		Rpl
3		Rpcs
4	Cla	Cla
5		Rpl
6		Rpcs
7	clsa	Clsa
8		Rpl
9		Rpcs

Table 1 Explored Combinations for SIMD adders.

Compared to a cascaded SIMD adder, the area overhead of our novel SIMD adder thus consists of: (1) extra logic required for operand isolation; (2) extra logic required for output selection and (3) multiple adders rather than a single one. The area breakdown for the SIMD adder of Figure 3, implemented with Brent Kung adders and running at 300Mhz, is as follows: the ICU, the OCU, 4 8-bit and 1 32bit adder occupy respectively 22.9%, 13.0%, 28.2% and 35.9% of silicon estate. The ICU and OCU burn up to 48,2% of the total power. They contribute up to 56.6% of the critical path delay of the adder. A more detailed study of the overhead is provided in the next section.

IV Experimental Results

Experimental Setup

All generated components are built using a UMC High Performance 0.13um CMOS standard cell library. The designs were synthesized with a maximum fan-out of four, an operating voltage of 1.2V, and a temperature of 125 °C. Area and delay for various designs are estimated with Synopsys Physical Compiler. Every design was synthesized, while optimizing for power and constrained to a range of target frequencies. Note that Physical Compiler builds a provisional placement and uses wire-energy estimates for determining the optimal gate-sizes. It reduces the power consumption, exploiting information on the typical input pattern of the design. We provide the following input pattern: both operand inputs *A&B* toggle at the same rate as the frequency (i.e. operating frequency equals activity). Each bit of the input pattern has the same probability of being zero or one. Similarly, we change the subtraction and saturation controls signal every other cycle.

The inputs controlling the operating mode are set to mimic three different operating scenarios: only 8-bit operations (*U* signal set zero), only 32-bit operations (*U* signal set one) and scenario in which 8-bit/32-bit have an equal probability of occurring. In the latter case, the *U* signals toggle every 100 cycles and have the same probability of being one or zero.

Experiments

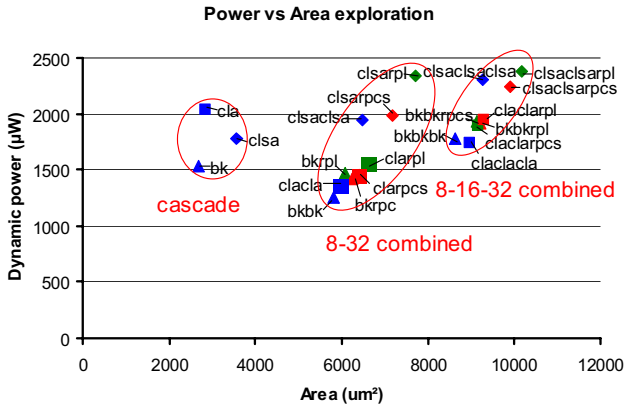

Figure 4 A power and area trade-off exists between the presented SIMD adder templates: the cascaded adders occupy less silicon, but consume more energy than the most energy efficient combined ones. Notice that not all combinations are energy-optimal. The results shown assume an operating frequency of 300Mhz and 50%/50% occurrence of 8/32-bit operations.

In the experimental results, we will first present the trade-off between area/power for the explored adders. Thereafter, we indicate which adders are best combined for different operating frequencies. Then, we explain which are the best combinations in terms of the usage conditions (scenarios). As expected, a trade-off exists between area and power: the cascaded adders are the most area-efficient ones, but consume more power than the best ones obtained by combining several adders (see Figure 4). Assuming a usage scenario of respectively 50%/50% 8/32-bit operations and an operating frequency of 300Mhz, combining adders is 18% more power efficient, but costs 2.2 more area. Furthermore, note that making a combined adder for 8/16/32-bit operations is in this case nor energy nor area optimal, due to the overhead for combining the different adders. This becomes more clear with Table 2.

	cascade bk		combined bkbk		combined claclalca	
	area	power	area	power	area	power
adders	2028	914	3724 (1.84)	697 (0.76)	5895 (2.91)	733 (0.80)
ICU	508	207	1327 (2.61)	259 (1.23)	2071 (4.07)	634 (3.06)
OCU	926	410	755 (0.82)	296 (0.72)	1009 (1.09)	376 (0.92)
Total	3462	1531	5806 (1.68)	1252 (0.82)	8975 (2.59)	1743 (1.14)

Table 2 Area is in µm² and power is in µW, while assuming a 50%/0%/50% between the 8/16/32-bit operations and assuming a 300Mhz operating frequency. The values between brackets are the relative values compared to the data of the cascaded adder.

The SIMD adder (*combined bkbk*), combining 8-bit and 32-bit operations and based on Brent Kung adders is most power efficient, despite the fact that there is a significant power penalty for the ICU unit (again see Table 2). This power penalty becomes a bottleneck for the SIMD adder, combining adders for three operand lengths (8-bit, 16-bit and 32-bit). The combined adder burns more power than the cascaded one, due to the extra power in the ICU unit. The extra area required *for* and the larger logic depth *of* the ICU unit delays the design. Consequently, the synthesis tool uses larger gates for closing the timing, and the design becomes more power-hungry. From this experiment, we conclude that both the area and power overhead of combining adders, or more in general, the cost of adding redundant logic is significant. When introducing redundant logic, one should carefully analyze whether its benefits outweigh the area/power overhead. As we will show in the remainder of this paper, the outcome of this analysis strongly depends on the usage conditions (such as e.g., operating frequency).[3]

[3] The experimental results indicate that a 8/16/32-bit combined SIMD adder is not energy-efficient due to the extra power overhead of the ICU. Throughout the remainder of the paper we will therefore only combine 4x8-bit adders with a 32-bit adder. This combined adder thus provides no support for 16-bit operations. If 16-bit operations are required too, the 32-bit adder should be replaced with a 2x16-bit cascaded one.

A first important parameter that determines the benefits of building a combined adder is the operating frequency.
Figure 5 represents the evolution of the power in function of the operating frequency for different SIMD structures and for a 50%/50%distribution between 8/32-bit operations. The power numbers are expressed relative to the power of the most cascaded SIMD adders, which is based on carry look ahead select adders as this is the only cascaded SIMD adder that can operate up to 500Mhz. The other cascaded adders are also shown, named *bk* and *cla* in the figure.

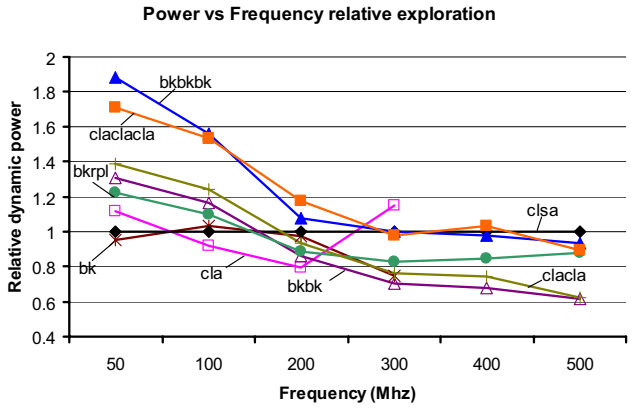

Figure 5 The power of SIMD adders normalized to the one of a clsa cascaded SIMD adder are presented for a 50%/50% 8/32-bit scenario. The cascaded SIMD adders are most energy efficient below 300Mhz (cla). Higher than 300Mhz, the SIMD adder combining 4x8-bit operations with 1x32-bit using Brent Kung units becomes most power efficient.

The cascaded adders are the most energy-efficient for performance targets below 300Mhz. Indeed, the experiments show that in the operating range all combined adders (*bkrpl, claclacla, etc.*) have a consumption larger than 1. The cascaded ones are best because the *cla* adders can achieve even with minimally sized standard cells the target frequency for both the 8-bit and 32-bit operations. More complex circuits do not provide any benefit, but rather increase the power dissipation.
For frequencies above 300Mhz, the delay target for the 32-bit operand lengths can no longer be achieved with minimally sized cells. Having multiple operand lengths then starts paying off. E.g., at 500 Mhz the SIMD adder that consists of four small Brent Kung adders combined with a large 32-bit Brent Kung one, is the most energy-efficient solution. It reduces the power up to 40%.
Surprisingly, four simple 8-bit ripple carry adders combined with a 32-bit Brent Kung adder is never an energy efficient solutions for operating frequencies between 50-500Mhz. Even though that an 8-bit ripple carry adder is the most energy-efficient design for low operating frequencies (see Figure 2), its gates are always up sized when integrated in a combined SIMD adder where 50% of the critical path delay

is spent in ICU and OCU unit.[4]

A second important parameter that determines the benefits of having a combined adder is the activity distribution. This is indicated in Figure 6, where the power dissipation of the SIMDs for different activity distributions between 8/32-bit scenarios is presented. For instance, the figure shows that the combined 8/32-bit *bk* SIMD (called in the graphs *bkbk*) is the best solution as long as less than 75% 32-bit operations occur. Thereafter, the 8/32-bit *cla* SIMD is the most energy efficient one.

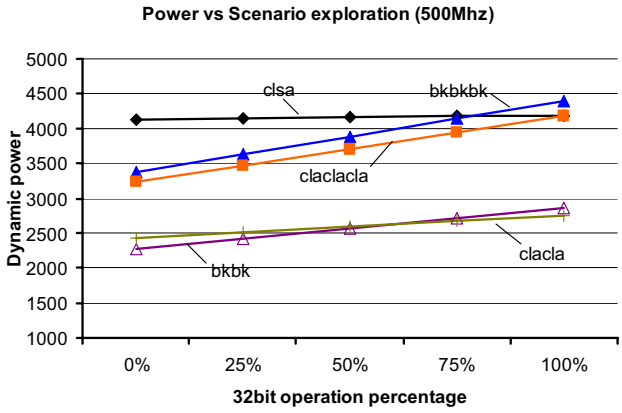

Figure 6 Power consumption of SIMD adders at 500Mhz is presented for multiple usage scenarios. The optimal combination depends on the operating scenario.

Finally, note that for high frequencies even combining adders for three operand lengths becomes more energy efficient than building a cascaded one. E.g., *bkbkbk* and *claclacla* outperform the cascaded *clsa* option. At these high frequencies and for long carry lengths, large and thus power-hungry gates are required for achieving the delay target. Adding the much smaller 4x8-bit adders to save energy for byte operations then really pays off. In this case, combining *bkbk* can result in the energy savings up to 45% compared to the *clsa* cascaded one.

V. Summary and Conclusions

SIMD is becoming more popular for embedded systems/DSPs as it is a flexible solution for performing data crunching in an energy-efficient way. Despite the fact that SIMD architectures have been extensively researched, limited work exists on the design of energy efficient functional units for SIMD. The common practice of partitioning a normal adder in sub-units is not the most energy efficient one. A better solution is combining adders optimized for each operand length into a single SIMD unit. Significant energy savings can be achieved in this way, particularly for high operating frequencies. The proposed technique should be carefully applied as there is a considerable penalty for operand isolation and an extra area cost has to be paid compared to existing SIMD adders. In future work, we want to apply the same concepts to an entire SIMD data path.

[4] Note that we assume 50Mhz as the minimal delay target.

Acknowledgements

Intra-European Marie Curie Actions funds

References

[1] H. Nguyen and L. John, "Exploiting SIMD Parallelism in DSP and Multimedia Algorithms Using the AltiVec Technologies", *Proc. ACM Int. Conf. Supercomputing,* pp.11-20, June 1999

[2] H. Hunter and J. Moreno, "A New Look at Exploiting Data Parallelism in Embedded Systems", *Proc. ACM CASES,* pp.159-168, Oct. 2003

[3] C. Kozyrakis and D. Patterson, "Overcoming the limitations of conventional vector processors", *Proc. ACM ISCA,* pp.399-409, Nov. 2003

[4] S. Larsen and S. Amarasinghe, "Exploiting superword lebel parallelism with multimedia instruction sets", *Proc. ACM PLDI,* pp.145-156, 2000

[5] R. Lee, "Efficiency of microSIMD architectures and index-mapped data for media processors", *Proc. IS&T/SPIE Symposium on Electric Imaging,* pp 34-36, 1999

[6] R. Lee, "Subword Parallelism with MAX-2", *Proc. IEEE MICRO,* Aug., pp 51-58, 1996

[7] D. Talla et al., "Bottlenecks in Multimedia Processing with SIMD Style Extensions and Architectural Enhancements", *Proc. IEEE Trans. Computer,* Vol. 52, N. 8, pp 34-36, 1999

[8] J. Fridman., "Data alignment for sub-word parallelism in DSP", *Proc. IEEE Workshop on Signal Processing Systems,* , pp 251-, 1999

[9] K. Masselos, F. Catthoor, C. Goutis and H. De Man, "Combined Application of Data Transfer and Storage Optimizing Transformations and Subword Parallelism for Power Consumption and Execution Time Reduction in VLIW Multimedia Processors", *J. VLSI Signal Processing Systems, Vol 32, Nr.1*, pp 53-73, 2004