POLITECNICO DI TORINO
Repository ISTITUZIONALE

A methodology for the design of dynamic accuracy operators by runtime back bias

Original
A methodology for the design of dynamic accuracy operators by runtime back bias / JAHIER PAGLIARI, Daniele; Durand, Yves; Coriat, David; Molnos, Anca; Beigne, Edith; Macii, Enrico; Poncino, Massimo. - ELETTRONICO. - -:(2017), pp. 1165-1170. (Intervento presentato al convegno 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE) tenutosi a Lausanne (Switzerland) nel 27-31 March 2017) [10.23919/DATE.2017.7927165].

Availability:
This version is available at: 11583/2674328 since: 2020-11-01T15:00:39Z

Publisher:
IEEE

Published
DOI:10.23919/DATE.2017.7927165

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

12 October 2024

A Methodology for the Design of Dynamic Accuracy Operators by Runtime Back Bias

Daniele Jahier Pagliari*, Yves Durand†, David Coriat†, Anca Molnos†
Edith Beigne†, Enrico Macii* and Massimo Poncino*
*Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, ITALY
Email: {daniele.jahier,enrico.macii,massimo.poncino}@polito.it
†CEA-LETI, Minatec Campus, Grenoble, FRANCE
Email: {yves.durand,david.coriat,anca.molnos,edith.beigne}@cea.fr

***Abstract*—Mobile and IoT applications must balance increasing processing demands with limited power and cost budgets. Approximate computing achieves this goal leveraging the error tolerance features common in many emerging applications to reduce power consumption. In particular, *adequate* (i.e., energy/quality-configurable) hardware operators are key components in an error tolerant system. Existing implementations of these operators require significant architectural modifications, hence they are often design-specific and tend to have large overheads compared to accurate units.**

In this paper, we propose a methodology to design adequate datapath operators in an automatic way, which uses threshold voltage scaling as a knob to dynamically control the power/accuracy tradeoff. The method overcomes the limitations of previous solutions based on supply voltage scaling, in that it introduces lower overheads and it allows fine-grain regulation of this tradeoff. We demonstrate our approach on a state-of-the-art 28nm FDSOI technology, exploiting the strong effect of back biasing on threshold voltage. Results show a power consumption reduction of as much as 39% compared to solutions based only on supply voltage scaling, at iso-accuracy.

I. Introduction

Energy efficiency has become one of the primary objectives in the design of digital systems, due to the widespread diffusion of energy autonomous devices for mobile and IoT applications, with increasing data processing demands and limited budget. Many emerging applications in these fields exhibit common *error tolerance* (or *resilience*) features, which can be leveraged to improve energy-efficiency [1]. Indeed, these applications do not require the exact calculation of a numerical result, but rather aim at providing a "good-enough" output. Error tolerance may stem from different factors, such as the presence of noisy inputs in the application, or the limited perceptive capabilities of human sense organs [1].

This trend has recently brought to the rise of a design paradigm generically referred to as Approximate Computing [2]. The idea of approximating a result is not new per se, but the goal of this research branch is to investigate general design patterns and platforms to support inexactness in computing [2]. Techniques have been devised at various levels of abstraction, from device/circuit to software. Although some solutions leverage approximations to obtain a performance improvement, most of them do it to improve energy efficiency [3]–[7].

One of the most popular embodiments of Approximate Computing is the design of *approximate hardware units*, in particular the main datapath operators. These works aim at simplifying the structure of an operator by relaxing the equivalence between specification and implementation. By doing so, they reduce area, power and delay, at the expense of controlled errors [3]–[6]. Other works focus on automating these simplifications by means of synthesis algorithms [7].

The main limitation of these approaches is that the amount of approximation is *fixed at design time*, making them unusable in systems with time-varying tolerance to errors.

Researchers have thus gradually shifted their interest from *approximate hardware* to operators whose accuracy is reconfigurable at runtime, which we will refer to as *adequate hardware* (and more in general as Adequate Computing). As for approximate operators, one approach consists in modifying the architecture of a unit in order to support multiple accuracy "modes" [8]–[13]. However, these solutions are mostly based on manual redesign of a single architecture, and cannot be automated. Moreover, they often have large overheads in terms of latency, area, and power at maximum accuracy [14].

An alternative that relies mostly on technological knobs rather than on architectural modifications is Dynamic Voltage and Accuracy Scaling (DVAS) [14]. In DVAS, power savings are obtained by scaling the supply voltage of an operator, and reducing the input dynamic range (i.e., bitwidth) to cope with the corresponding delay increase. Although this simple idea has proven very effective, its usability is limited by the fact that, in realistic implementations of hardware operators, a high percentage of timing paths have a delay close to the critical one (the so-called *wall of slack* phenomenon) [15].

In this paper, we propose a new technique for designing adequate operators, that combines DVAS with dynamic threshold voltage (V_{th}) scaling. By acting on the V_{th} of the cells, we speed up selected timing-critical areas of the operator, in order to increase the usable dynamic under scaled voltage. Our approach can be applied to any form of dynamic V_{th} tuning, although we demonstrate it on FDSOI technology and dynamic Back Biasing [16], [17]. To the best of our knowledge, this is the first time that Back Biasing and dynamic V_{th} assignment are used for adequate computing.

Thanks to the use of V_{th} tuning, our method overcomes the main limitations of DVAS: (i) it circumvents the wall-of-slack allowing larger bitwidth operation at iso-voltage, (ii) it permits to independently configure the bitwidth of different units in the same die without the need of inserting level shifters. The

proposed technique is fully automated and integrated with state-of-the-art tools, and yields a power reduction of more than 39% with respect to DVAS alone.

II. Background and Motivation

A. Architectural Methods for Adequate Hardware Operators

When designing hardware operators using a traditional *approximate* paradigm, the tolerated error is fixed at design time. This limitation can be overcome by designing *adequate* operators, which can support time-varying accuracy constraints. In this section, we briefly review the state of the art in architectural solutions for the design of adequate hardware. Details on approximate units can be found in [3]–[7].

Adequate operators are designed to support multiple runtime-selectable *accuracy modes*, each with a corresponding power consumption. At architectural level, this is achieved reducing the complexity of the active part of the operator in low-accuracy modes, by disabling or replacing some of its internal logic. This translates in a reduction of dynamic power and delay. The reduced delay is then exploited for further power reduction by means of supply voltage scaling.

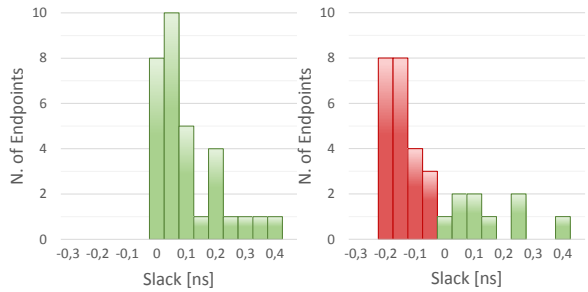
Most literature on these operators focuses on adders and multipliers. Several adequate adders are obtained enhancing an approximate unit with *error recovery* circuitry, which is selectively activated (in an additional clock cycle) when accurate results are needed [8], [9]. For these designs, maximum accuracy operation has a significant overhead in terms of latency, as well as area and power. An alternative solution is reconfigurable carry-chain segmentation [11], in which an adder is split in sub-adders, and the carry signal from each sub-adder to the next can be selectively replaced with the output of a simpler *carry prediction* circuit. For what concerns multipliers, the architecture of [10] is based on disabling columns of the partial product matrix by means of signal gating. In [13], instead, partial product accumulation is implemented by means of special adders, able to produce an approximate result and a *correction output*, which depending on the selected accuracy mode, is used to reduce the accumulation error. A general approach for the design of adequate operators is proposed in [12], where some of the previously described techniques are applied to more complex operators such as Multiply And Accumulate units, L1 Norm, etc.

Building adequate operators by means of architectural modifications has several drawbacks. Most techniques work for a single type of operator (e.g. [10] and [13] are specific to array multipliers), and often the resulting units have very large overheads in terms of area, latency, and power at maximum accuracy [8], [9]. Moreover, most approaches support a limited number of accuracy "modes" (typically one approximate mode and one accurate mode).

B. Dynamic Voltage and Accuracy Scaling

The authors of [14] have proposed Dynamic Voltage and Accuracy Scaling (DVAS). Like previous solutions, DVAS leverages voltage scaling to reduce power consumption in the operators. However, rather than changing the architecture, the authors propose to cope with the increased delay due to voltage scaling by reducing the *input dynamics*, i.e., gating some of the LSBs of each input operand. The comparison between DVAS and some of the previous architectural techniques, shows that despite its simplicity this solution obtains better error/power saving tradeoffs. Furthermore, for a single unit, DVAS has almost zero overheads in terms of area and power/delay at maximum accuracy, and can be applied, in principle, to *any* operator in an automatic way.

However, DVAS has a major drawback when combined with a standard ASIC implementation flow. Indeed, synthesis tools optimize timing-critical paths of an operator (i.e., paths with smallest *slack*) for performance, whereas less critical (largest slack) paths are exploited for area and power optimization. As a result, the delay of non-critical paths increases, to the point where their slack is comparable to that of critical ones, causing the so-called *wall-of-slack* [15]. An example is shown in Figure 1a, which shows the slack histogram of the endpoints of a 16x16-bit multiplier. The histogram is obtained after P&R, at the nominal implementation supply voltage ($1V$).



(a) $V_{DD} = 1.0V$ (b) $V_{DD} = 0.8V$

Fig. 1: Endpoint slack histogram for a 16x16-bit multiplier.

The effect of the wall-of-slack on DVAS is that the input dynamic usable without incurring in timing violations decreases rapidly when V_{DD} is scaled. As an example, Fig. 1b shows the multiplier endpoint slack histogram, after the supply voltage has been downscaled to $0.8V$. Red bars correspond to endpoints violating timing constraints. The large percentage of violating paths indicates that the input dynamic must be reduced significantly in order to restore timing compliance. Conversely, to obtain high-accuracy configurations, V_{DD} must be kept to a value very close to the nominal one.

Moreover, when multiple DVAS operators in the same design must work in independent accuracy modes, every unit must be placed in a separate *voltage domain*. In MOS technology, voltage domains must be separated inserting level shifters, which introduce significant power overheads.

C. UTBB FDSOI and Back-Biasing

The proposed solution to overcome the limitations of DVAS is demonstrated on 28nm *Ultra-Thin Body and Box* (UTBB) FDSOI technology [16]. For the purposes of this paper, the most important characteristic of this technology is the wide applicable Back Biasing (BB) voltage range, spanning more than $2V$, thanks to the presence of the Buried Oxide (BOX) layer, which acts as back-gate and removes body-source P/N junctions. Compared to the conventional range for bulk technology's Body Biasing ($\pm 300mV$), FDSOI Back Biasing has a stronger impact on V_{th}, and in turn on the performance/power

tradeoff. Forward Back Biasing (FBB) allows to lower V_{th} and reduce the switching delay of a transistor, at the expense of an increase in sub-threshold leakage currents. Reverse Back Biasing (RBB) has the opposite effect. In both cases, the *body factor* (i.e., the sensitivity of V_{th} to BB) is as high as $85mV/V$ [17]. One issue when using BB is that deep N-Well areas of the circuit with independent BB voltage must be separated by means of *guardbands*, which introduce area overheads. This makes independent control of Back Biasing to single devices impractical. Conversely, devices with common Back Bias must be grouped into geometrically localized BB *domains* (or *islands*). In our technology node, the minimum BB domain guardband width is about $3.5\mu m$, a value comparable to the height of the standard cells ($1.2\mu m$).

III. Back Biasing for Adequate Computing

The key limitation of DVAS with respect to the wall-of-slack is that when V_{DD} is lowered, the *entire operator* timing is slowed-down. This is clearly a problem when different parts of the circuit require different speeds (and hence a different power consumption), and in particular, as shown later, when the accuracy "modes" of an operator are implemented through reduction of input bitwidth (as in DVAS).

One possible solution to *selectively* tune the delay of different parts of the circuit would be to partition it in multiple independent *supply voltage islands*. However, due to the large overheads, this solution is only feasible at the SoC-level, and not at the fine granularity required for the relatively small hardware operators considered in this work; in particular, the insertion of *level shifters* between domains would have a relevant impact on power consumption [18].

Thus, in this paper we propose a methodology to achieve such selective tuning of the power/accuracy tradeoff using *dynamic threshold voltage (V_{th}) scaling* by means of Back Biasing (BB). As described in Section II-C, BB domains require a small separation guardband but no level shifters, and thus have a much smaller power overhead compaed to supply voltage domains. Nevertheless, independent assignment of bias voltages to individual cells is obviously unfeasible; instead, to admortize the area overheads, groups of topologically close cells must share a common BB. Given the similarity between the size of separators and the height of a cell, the minimum size of these regions should be in the order of a few tens of rows of the placed design. Therefore, in our methodology, the circuits are partitioned in V_{th} *domains* or *islands*. Each domain has an independent BB control, whereas the entire operator shares a single V_{DD}. The additional V_{th} knob is leveraged to speedup only timing critical parts of the circuit.

In this work we consider only *two* possible distinct V_{th} assignments to domains: Standard V_{th} (SVT) as the nominal condition, and Low V_{th} (LVT) as the "boosting" condition. In 28nm UTBB FDSOI technology, we map SVT to No Back-Bias (NoBB) and LVT to Forward Back-Bias (FBB). This simplifies both the search for the optimal V_{th} assignment to domains, as well as the generation of Back Bias voltages in the final circuit. In particular, two DC-DC converters (e.g., charge pumps) can be used to generate FBB voltages (for N-Well and P-Well) and some power switches to selectively connect the

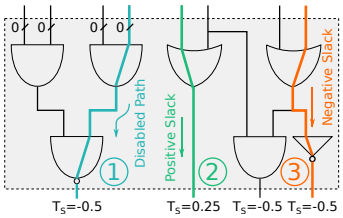


Fig. 2: Timing paths in an operator working with reduced bitwidth. Zeroed inputs are marked with a 0. Each output is associated a slack T_S computed assuming a clock period of 1 "unit", and that all gates have a delay of 0.75 units.

Well pins of each domain either to the converters output or to ground. Our methodology can however be applied to more than two V_{th} values, as well as to other forms of V_{th} tuning (e.g., Body Biasing bulk CMOS). Moreover, it is worth emphasizing that our approach is not alternative but rather complementary to DVAS; it is possible to set a given V_{DD} for the entire operator and then use BB to fine-tune the consumption and speed of different parts of the circuit.

While runtime BB assignment to affect circuit speed has been already exploited in literature for other reasons [17], we apply this principle to a completely new objective, i.e., optimizing the power consumption of a hardware operator based on runtime-varying *accuracy* requirements. We will later show the relevance of this objective on the methodology.

A. Runtime Accuracy Scaling

In our methodology, runtime adaptation of the operators accuracy is obtained using the same principle of DVAS [14]: depending on the required accuracy, some LSBs of the operator inputs are simply *set to zero*. In this paper we focus on the design of individual operators, and we assume that the number of zeroed LSBs is specified by means of external control signals. The selection of the optimal accuracy is determined at application level, and is outside the scope of this work.

Clamping some input LSBs at zero has an obvious effect on the timing paths that have those inputs as source points, which become deactivated (labeled with ① in Figure 2). The remaining active paths in the circuits include as usual paths with positive slacks (②) and with negative slacks (③).

These sets depend on the selected input width as well as on the global V_{DD}. Ideally, dynamic V_{th} assignment by means of BB should be used to speed-up *only critical paths* (③), leaving the other two sets in low-speed/low-leakage state.

B. Partitioning of the operator into V_{th}/BB Domains

As already mentioned, BB cannot be applied at an arbitrarily fine granularity; the maximum number of V_{th} domains N_{MAX} must be determined based on the acceptable area overheads, and traded off with the benefits in terms of power saving.

Once N_{MAX} is defined, however, there are still many degrees of freedom available regarding the possible shapes of the V_{th} domains. The definition of these regions is a fundamental issue in the proposed methodology. The basic problem of partitioning a circuit into multiple regions for selective application of a knob (V_{th} tuning, in our case) to control some design tradeoffs (power/accuracy, in our case) has been already studied the literature for various knobs and tradeoffs [20].

An important issue in the partitioning process is its impact on the other figures of merit of the circuit (timing, dynamic power consumption, etc.), due to the fact that V_{th} domains are physically isolated regions in the die. Indeed, cells should be kept as close as possible to their optimal position (i.e., the one determined by a standard placement algorithm) in order to minimize the impact on timing and power.

In addition to these traditional constraints, our problem introduces a new relevant element of complexity in the partitioning. In fact, **adequate operators must work in multiple accuracy configurations (i.e., input bitwidths)**. The ideal goal would be to optimize power consumption in *all* these configurations. However, each of them imposes different constraints on the optimization, due to the impact of zeroed LSBs on timing paths described in Section III-A. Determining a *single* physical partitioning that is optimal for different bidwidths is not trivial; since each accuracy mode implies a different set of critical paths, a solution that is optimal, (i.e., that allows to speed up *only* the critical paths) for a given input bitwidth, might not be optimal for another bitwidth.

In this work, our objective is to show a proof-of-concept of the usability of dynamic Back Biasing for adequate computing. Therefore, we do not investigate complex partitioning algorithms; we rather consider the simplest possible partitioning of a die into N_{MAX} sub-blocks, that is, a *regular grid* of V_{th} domains. Each of the N_{MAX} domains is assigned an identical area of rectangular shape, equal to a fraction $1/N_{MAX}$ of the total; Figure 3 shows an example for $N_{MAX} = 9$. This solutions has the desirable property of a regular structure which eases the physical implementation [19]. Moreover, it minimizes the overheads in terms of timing and power at maximum accuracy (i.e., full bidwidth), since none of the cells are displaced in order to be assigned to a V_{th} domain. The drawback of a regular tiling is that it might fail to isolate gates belonging to the paths that require speedup for a given bitwidth (i.e., set ③ in Figure 2). Thus, cells that do not need speed-up might end up in the same domain of cells that must be boosted. In this case, in order to meet timing constraints, the *entire* domain has to be assigned to LVT/FBB, resulting in a power overhead. However, due to the small granularity of V_{th} domains, this overhead will be much smaller with respect to that of raising V_{DD} in DVAS.

C. Implementation Flow

Figure 4 shows the proposed fully automated flow for the implementation of adequate operators based on V_{th} scaling. The flow consists of two main parts. The first *implementation phase* (green background in the figure), consists in the physical design of the operator. The result of implementation then undergoes a second *optimization phase* (blue background), in which the goal is determine the best assignment of the technological knobs for each bitwidth of interest.

The implementation phase takes as input a gate-level netlist of the original (accurate) circuit. The first step consists in performing a standard placement of the operator at nominal voltage, *without* V_{th} domains. In this step, the cells of the operator are placed according to "standard" constraints (i.e., timing, area and power) as determined by the P&R tool.

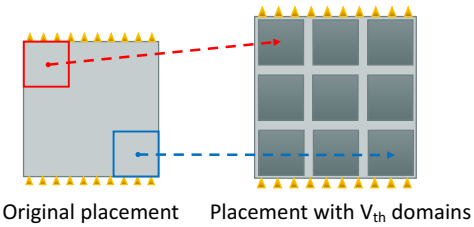


Fig. 3: Regular Grid Partition that identifies the V_{th} domains.

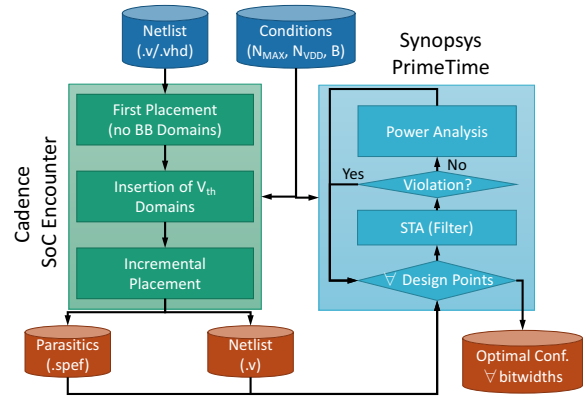
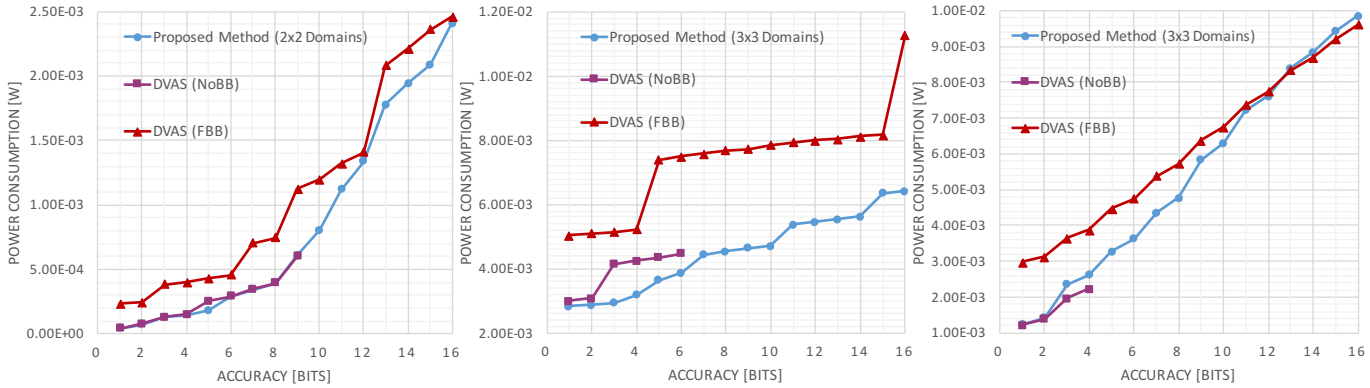


Fig. 4: Implementation Flow.

Then, the silicon area is enlarged in order to accommodate the separation guardbands, using the regular grid partitioning described in Section III-B, and depending on the desired number N_{MAX} of V_{th} domains. Finally, an incremental placement step is executed. In this step, the tool takes into account the newly inserted V_{th} domains and their possible operating modes (NoBB or FBB), and consequently modifies gate sizing, position, etc., in order to meet all timing (setup and hold) and DRC constraints. The tool is also instructed to insert *Well Taps* for the connection of Back Bias voltage rails to the different domains. Finally, the implementation is completed with routing. The outputs of the implementation phase are the placed and routed netlist of the operator, including V_{th} domains, and the corresponding parasitics file.

These results are then fed to the optimization phase, in which an exhaustive exploration of all possible operating conditions is performed. Specifically, the circuit is analyzed for all possible combinations of (i) V_{BB} assignments (NoBB or FBB) to the N_{MAX} domains and (ii) input bitwidths. Moreover, when our approach is combined with DVAS, the analysis of each point is repeated at multiple values of the global V_{DD}. The overall complexity of the exploration is $O(2^{N_{MAX}} \cdot B \cdot N_{V_{DD}})$, where B is the range of bitwidths of interest, and $N_{V_{DD}}$ is the number of supply voltages to be explored. These values are all relatively small, making exhaustive exploration feasible. N_{MAX} is in the order of a few tens; B in the worst case coincides with to the full bitwidth (16 or 32) but is typically smaller; $N_{V_{DD}}$ depends on the resolution of the supply voltage generator and the allowed range of variation of V_{DD}: assuming a 100mV step and a range between 0.6V and 1.0V, $N_{V_{DD}} = 5$. With these values, the number of points to consider is in the order of some thousands. Furthermore, points that are unfeasible can be easily filtered out without the need of a complete analysis. Unfeasibility is checked by running



(a) Booth Multiplier (b) FFT Butterfly (c) FIR Filter

Fig. 5: Comparison with DVAS in terms of bitwidth versus power tradeoff.

Design	A $[mm^2]$	f_{clk}[GHz]	Groups	A_{ovr}[%]
Booth	$2.59 \cdot 10^{-3}$	1.25	$2x2$	15
Butterfly	$7.71 \cdot 10^{-3}$	1.00	$3x3$	17
FIR	$9.10 \cdot 10^{-3}$	0.75	$3x3$	16

TABLE I: Post P&R design characteristics and groups configurations for DVAS comparison.

Static Timing Analysis (STA) on the netlist; if it incurs in a timing violation, that point is discarded. For instance, the all-NoBB configuration with $V_{DD} = 0.6V$ will very likely be not timing compliant at maximum accuracy. STA is quite fast (in the order of 0.1s for the scale of our operators), and even faster for lower accuracy netlists, where some paths become inactive after shrinking the bitwdith. On average, we have observed that about 75% of the configurations are filtered by STA.

Feasible configurations are instead analyzed for power, taking into account both leakage and dynamic components. This phase can optionally use realistic inputs for switching activity annotation. For each accuracy (i.e., bitwidth) of interest, the operating condition that produces the *minimum* power consumption is stored. The corresponding knob configurations (V_{BB} assignment to the different domains, and possibly global V_{DD} value) are output for each bitwidth of interest. Power estimation including importing of VCD traces takes about 1s on our server (Intel Xeon E5-2630@2.4GHz, 128GB DDR3).

IV. Experimental Results

A. Experimental Setup

To evaluate the effectiveness of our method, we applied it to three different hardware designs common in DSP applications, which are typically error tolerant but often have time-varying accuracy requirements: a Booth multiplier with Wallace tree, a 30-tap FIR filter, and a *butterfly* unit, i.e., the main datapath component of a FFT accelerator. We considered 16-bit fixed-point implementations of the operators (larger bit-widths are seldom used in this type of applications).

We started from behavioral specifications of the operators in VHDL or Verilog. For synthesis and implementation we targeted a 28nm FDSOI standard-cell library from STMicroelectronics. We used Synopsys DC-I-2013.12-SP4 for synthesis and Cadence Innovus v15.13 for P&R. Post implementation netlists and parasitics have been loaded in Synopsys PT-I-2013.12-SP1 for timing and power analysis. A summary of the post-P&R characteristics of the three designs is reported in Table I, where A represents the silicon area and f_{clk} is the nominal clock frequency used for synthesis. For all operators, we used a nominal supply voltage $V_{DD} = 1.0V$.

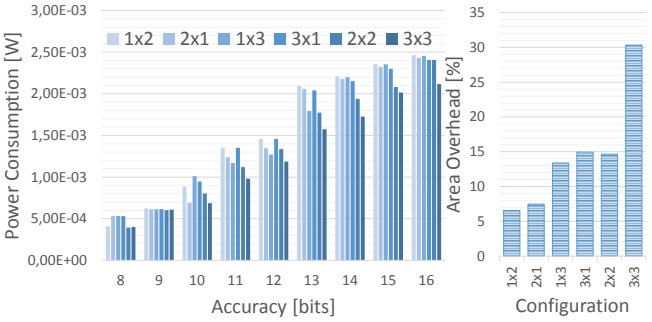
Notice that, in order to fully exploit the flexibility of our method, it is preferable that the maximum accuracy mode at nominal V_{DD} corresponds to a fully-boosted configuration of the domains. Thus, during this first P&R of the three operators without V_{th} domains, we considered a standard-cell library characterization in which *all* cells are in FBB state.

B. Comparison against DVAS

DVAS was already shown to provide better results than traditional architectural solutions for adequate hardware operators design ([10], [13], etc.). Therefore, we limit the comparison of our method against DVAS. We implemented both methods on the three target designs. The selected configuration of V_{th} domains is shown in Table I, together with the area overhead A_{ovh} introduced by separation guardbands. For both DVAS and our method we fixed the clock frequency at the nominal value of Table I, and we considered 5 V_{DD} conditions, from $1.0V$ (nominal) to $0.6V$, in steps of $0.1V$. We used a BB voltage of $\pm 1.1V$ (N-Well/P-Well) as FBB condition.

Figure 5 shows the results of the comparison on an bitwidth versus power consumption plane. Each curve is a Pareto frontier, i.e., it shows the *minimum power* configuration for each bitwidth, considering all possible combinations of V_{DD} and (for our method) V_{th} assignment. The graphs contain only configurations that *fully* meet timing constraints, taking into account disabled paths due to reduced dynamic. Power includes both leakage and dynamic components. As explained in Sec. IV-A, the original operators were implemented using FBB as nominal condition. Thus, for a fair comparison, Figure 5 reports both the result of DVAS with NoBB, and that of DVAS with FBB (i.e., boosting *all cells* of the operator).

A first important observation is that DVAS with NoBB, i.e., the standard implementation reported in [14], cannot reach maximum accuracy: the curves are limited to very small bitwidths in all three cases, indicating there is no solution for higher bitwidths at that frequency. Under the same timing



(a) Power consumption at different accuracies (b) Area overhead

Fig. 6: Power saving and area overhead for different numbers and configurations of V_{th} domains in the Booth Multiplier.

constraints, our method can instead reach maximum accuracy for all designs. DVAS with FBB, conversely, *can* work at maximum accuracy for all three designs; however, due to the wall-of-slack, the power vs. bitwidth tradeoff is significantly worse with respect to our method. For multiplier and FIR, the overheads of DVAS are particularly evident looking at the "step-wise" shape of the Pareto frontier, in which each step corresponds to a change of V_{DD}. At 10-bit, the power saving of our method with respect to DVAS is 32.67% for the Booth and 39.92% for the FIR. For what concerns the butterfly, the more linear curves of DVAS show that this unit is less affected by the wall-of-slack, and consequently the improvement thanks to Back Biasing is less marked. Nevertheless, it is still possible to obtain a 16.5% power saving with respect to DVAS at 8-bit.

The only case in which DVAS yields (marginally) better results than our approach is for the butterfly, at very small or very large (close to the maximum) bitwidths. These are due to the incremental placement after insertion of V_{th} domains. The same overheads are not visible for Booth and FIR, mainly because the impact of guardbands is smaller (the Booth has a smaller number of domains, and the FIR has a larger area).

C. Impact of the number of V_{th} domains

The number of V_{th} domains has a strong impact on all figures of merit in our method. Figure 6 depicts this effect using the Booth multiplier as an example. Figure 6a shows the power savings in different accuracy modes for different groups configurations. For visualization purposes, the graph only reports accuracy modes between 8 and 16 bits (those < 8 bits are seldom needed in realistic applications). Increasing the number of groups produces a general reduction in power consumption, especially at high accuracy modes. This is expected, as more groups allow a finer-grain selection of the parts of the operator to be sped up. However, this trend is not always respected; for particular accuracy conditions, increasing the number of groups has a negative effect on power (e.g. 2x1 versus 3x1 at 10-bits accuracy). This is due to the addition of guardbands between groups, which may cause operator cells that were previously closer to move away from each other, extending the corresponding routes. Figure 6b reports the area overheads due to guardbands for the groups configurations analyzed. Expectedly, area increases with the number of groups, while the dependence on the structure has a less evident effect.

As a general observation, it is evident that the selection of the number of V_{th} domains strongly depends on the metrics of interest. In general, the power reduction in accuracy modes of interest must be balanced with the area budget. However, since our method is automated, the design space can be explored exhaustively, at least for a small number of groups (≤ 10).

V. Conclusions

In this paper, we have presented a novel method for the implementation of adequate hardware operators. Thanks to the flexibility provided by the additional V_{th} knob, this method achieves significantly larger power savings at iso-accuracy compared to the previous DVAS, at the price of a small area overhead. Moreover, it removes the need for level shifters in complex systems including multiple operators. Future developments include the study of alternative V_{th} domains construction methods, and an investigation of the optimal number and configuration of domains.

References

[1] V. Chippa et al, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. DAC* 2013, pp. 1–9.

[2] Q. Xu, N. S. Kim, and T. Mytkowicz, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.

[3] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proc. GLSVLSI* 2015, pp. 343–348.

[4] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. EDSSC*, 2010, pp. 1–4.

[5] J. Huang, J. Lach, and G. Robins, "A methodology for energy-quality tradeoff using imprecise hardware," in *Proc. DAC* 2012, pp. 504–509.

[6] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, "New approximate multiplier for low power digital signal processing," in *Proc. CADS* 2013, pp. 25–30.

[7] A. Lingamneni et al, "Synthesizing parsimonious inexact circuits through probabilistic design techniques," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 93:1–93:26, May 2013.

[8] A. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. DATE* 2008, pp. 1250–1255.

[9] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. DAC* 2012, pp. 820–825.

[10] M. de la Guia Solaz, W. Han, and R. Conway, "A flexible low power dsp with a programmable truncated multiplier," *IEEE Trans. on CAS I*, vol. 59, no. 11, pp. 2555–2568, Nov 2012.

[11] R. Ye et al, "On reconfiguration-oriented approximate adder design and its application," in *Proc. ICCAD* 2013, pp. 48–54.

[12] D. Mohapatra et al, "Design of voltage-scalable meta-functions for approximate computing," in *Proc. DATE*, 2011, pp. 1–6.

[13] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. DATE* 2014, pp. 95:1–95:4.

[14] B. Moons and M. Verhelst, "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *Proc. ISLPED* 2015, pp. 237–242.

[15] A. Kahng et al. "Slack redistribution for graceful degradation under voltage overscaling," in *Proc. ASP-DAC* 2010, pp. 825–831.

[16] P.-E. Gaillardon et al, "A survey on low-power techniques with emerging technologies: From devices to systems," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 2, pp. 12:1–12:26, Sep. 2015.

[17] E. Beigné et al, "A 460 mhz at 397 mv, 2.6 ghz at 1.3 v, 32 bits vliw dsp embedding f max tracking," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 125–136, Jan 2015.

[18] J. Hu et al, "Architecting Voltage Islands in Core-based System-on-a-chip Designs," *Proc. ISLPED* 2004, pp. 180–185.

[19] T. Kutzschebauch and L. Stok, "Regularity Driven Logic Synthesis", *Proc. ICCAD* 2000, pp. 439–446.

[20] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-power Design", *Proc. ISLPED* 1995, pp. 3–8.