Cross-Layer Approximate Hardware Synthesis for Runtime Configurable Accuracy

Tanfer Alan, Andreas Gerstlauer, *Senior Member, IEEE*, Jörg Henkel, *Fellow, IEEE*

***Abstract*—Approximate computing trades off computation accuracy against energy efficiency. The extent of approximation tolerance, however, significantly varies with a change in input characteristics and applications. We propose a novel cross-layer approach for the synthesis of runtime accuracy configurable hardware that minimizes energy consumption at area expense. To that end, first we explore instantiating multiple hardware blocks in the architecture with different fixed approximation levels. These blocks can be selected dynamically and thus allow to configure the accuracy during runtime. They benefit from having fewer transistors and also synthesis relaxations in contrast to state-of-the-art gating mechanisms which only switch off a group of paths of the circuit. Our cross-layer approach combines instantiating such blocks in the architecture with area-efficient gating mechanisms that reduce toggling activity, creating a fine-grained design-time knob on energy vs. area. We present a systematic methodology to explore this joint design space and find energy-area optimal solutions as a function of required accuracies, their utilization in the workload, together with hardware parameters: dynamic power savings, area of the hardware block, and leakage of the technology. Examining total energy savings for a range of circuits under different workloads and accuracy tolerances show that our method finds Pareto-optimal solutions providing up to 32% and 60% energy savings compared to state-of-the-art accuracy-configurable gating mechanism and an exact hardware block, respectively, at 2x area cost.**

***Index Terms*—Approximate computing, low-power, design space exploration, variable precision, quality configurable.**

I. Introduction

Approximate computing leverages the application error resilience by relaxing exactness in computation towards a primary design goal: improving energy efficiency [2]. Several modern and prominent application domains such as machine learning, gaming, and computer vision are tolerant to varying degrees of approximations while still meeting their requirements. The impact of approximations on energy facilitates solving larger problems at both ends of the computing spectrum [3]. It is an enabling factor for workloads such as multimedia, gaming, and object recognition with limited energy budget as in AR/VR devices [4]. Also, it is a recent and major driver of machine learning at both embedded edge devices [5–7] and high-performance servers [8–10] through precision scaling, i.e., quantization in machine learning terms.

This work is an extension of [1]. Manuscript received December 1st, 2020; revised February 14, 2021; accepted March 14, 2021. This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre Invasive Computing (SFB/TR 89).

Tanfer Alan and Jörg Henkel are with the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: alan@kit.edu; henkel@kit.edu).

Andreas Gerstlauer is with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: gerstl@ece.utexas.edu).

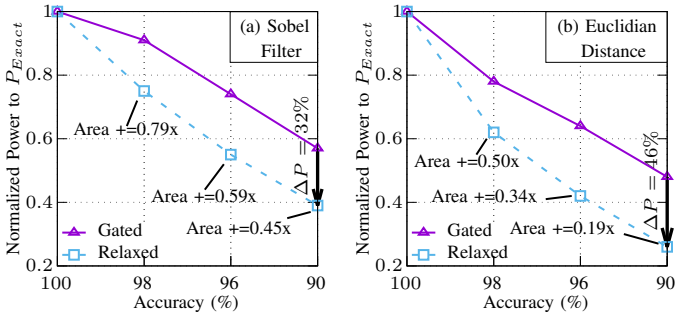

Fig. 1: Power vs. accuracy to compare gating an exact circuit against instantiating relaxed, approximate circuits. Area costs of instantiated circuits are noted. All values are relative to the exact version of the corresponding circuit. ΔP is dynamic power savings of instantiating approach over gating.

Traditionally, a large class of research explored approximate computing at the hardware level targeting a single accuracy in manual [11, 12] and automated design [13–16] of functionally approximate circuits. The hardware is designed to have fewer transistors and shorter critical paths, where boolean functionality deviates from an exact specification to a limited extent. Instantiating such approximate hardware has a two-fold effect on energy: fewer transistors cause less toggling activity and shorter paths allow for voltage scaling or *synthesis relaxations*, i.e., circuits can be composed of smaller transistors that require less power at the same performance. In this way, the slack in shorter paths can be exploited by the synthesis tool. The evident disadvantage is that the approximations on these circuits are fixed and hardwired. It is not possible to configure their accuracy at runtime.

Accuracy configurability is essential in practice for two main reasons: (i) Output quality of approximate hardware strongly depends on its inputs, and (ii) A workload may tolerate significantly different levels of approximation depending on its context and environment [17]. Runtime methods have shown that a fixed accuracy may be too conservative and accuracy configuration is necessary to maximally exploit the opportunities of approximate computing for energy efficiency improvement [17–20]. In particular, an offline profiler in [17] has shown that there is a significant variation in precision requirements between different applications and also between different phases of an application.

Existing accuracy configurable hardware proposals [21–25] and design methodologies [26–28] primarily utilize circuit-layer *gating* mechanisms: They disable a configurable portion of the paths by not propagating data or inserting control circuitry into the exact hardware with a small area overhead. Notwithstanding their potency, such approaches only benefit from reduced toggling activity. Because they do not structurally simplify

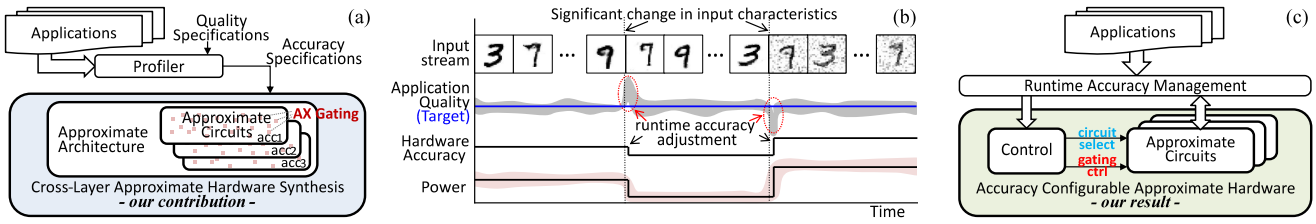

Fig. 2: Background of a runtime accuracy configurable system. (a) Synthesis of approximate hardware with multiple circuit instantiations and a gating mechanism. (b) An application example targeting a quality, while input characteristics change over time. (c) System-level abstraction of accuracy management, where the hardware can work in tandem with a runtime system.

the circuit, e.g. shorten the critical path, they cannot exploit the full extent of power savings that static approximate hardware can achieve with synthesis relaxations.

A potential solution can be instantiating multiple static accuracy circuits in the architecture and switching between them despite their area and leakage power costs [7, 29]. In Figure 1, we compare gating an exact hardware, similar to existing work [26–28], against static accuracy approximate hardware by means of precision scaling their inputs, i.e., discarding a number of LSBs for Sobel Filter and Euclidian Distance computation hardware blocks. Additional dynamic power savings reach up to 46% when leakage is neglected. Notably, the additional area cost of instantiating a circuit is reduced significantly as the accuracy is reduced. For instance in Figure 1b at 90% accuracy, the circuit requires only $0.19\times$ the area of the exact circuit. This example demonstrates that instantiating and connecting additional approximate circuits can be a lower-power and higher-area overhead alternative to gating. However, instantiating an additional circuit incurs leakage cost even when this circuit is not used. The dynamic power benefits of an instantiated logic is proportional to its utilization whereas its leakage power and area costs are fixed. At fine granularity, having many instantiations would reduce the utilization per circuit. Hence, instantiating may not always result in net energy benefits.

In this paper, we propose a novel, cross-layer approach for the synthesis of runtime accuracy-energy configurable hardware. Our approach combines circuit-level gating mechanisms and instantiating multiple approximate circuits in the architecture, to exploit both toggling activity and also synthesis relaxations. It enables fine-grain energy vs. area trade-offs in a design space that is a superset of two distinct approaches. Finding energy optimal solutions in this joint design space is a non-trivial function of required accuracies, their utilization in the workload, power savings that can be achieved at required accuracies, and leakage in the used technology. In [1], we introduced the basic hybrid design approach combining gating and instantiation applied in a manual fashion. This paper extends our previous work into a systematic and automated cross-layer methodology that explores the design space efficiently yet exhaustively and finds the minimum energy requirement solution given an RTL block, a workload with specified accuracies and a maximum area constraint. Additionally, we significantly expand our evaluations to showcase generality, impact of the integration overheads, accuracy reconfiguration costs, input dependency of energy savings, and a detailed analysis of the leakage impact with 2 different technology libraries. All combined, our work makes the following key contributions:

- We propose a novel and cross-layer runtime accuracy-configurable hardware design approach. Our approach is general, supports several accuracies, and utilizes both circuit and architectural techniques in significantly reducing dynamic power consumption.
- Our work demonstrates the existence of a larger design space of accuracy-configurable hardware, with non-obvious trade-offs linked to the workload, hardware architecture, and technology.
- We present a systematic methodology to ensure selecting Pareto-optimal solutions. Our methodology creates a design-time knob on energy vs. area while matching given dynamic accuracy requirements.
- Our experiments show significant energy savings can be achieved despite the increase in area and leakage energy. We present these results in a technology-independent manner.

In our evaluations, we examine a range of circuits under different workloads and accuracy requirements. Our experiments show at $2\times$ area cost and the same performance, up to 60% energy reduction compared to an exact hardware block and up to 32% energy reduction compared to state-of-the-art accuracy-configurable gated hardware while matching the accuracy.

II. Background

Figure 2 illustrates the background of a runtime accuracy configurable system and highlights the position of our work. At design time (Figure 2a), we aim at synthesizing an energy optimum hardware that consists of multiple instantiations of approximate circuits and a gating mechanism. Here, optimality depends on the accuracy specifications that can be profiled for the given applications and the quality specifications. In Figure 2b, we show a handwritten digit recognition example, in which, the input stream characteristics change in handwriting style and noise. A monitoring runtime system reacts to maintain the application quality at a target. It switches our hardware to a different accuracy at a different power requirement and minimizes the energy consumption when possible. Runtime accuracy management is orthogonal to and out of our scope. We distinguish the quality and accuracy terms as follows: quality is an application demand, for instance, a classification error rate of $\leq$ 1%. Accuracy is the correctness of the underlying hardware and software, with metrics such as 95% correctness in mean magnitude (5% mean relative error distance). Hence, quality changes with input characteristics and also with hardware accuracy. Accuracy can be changed by, e.g., changing the precision. In Figure 2c, hardware synthesized by our methodology offers accuracy controls to work in tandem with an external runtime system for accuracy management.

III. Related Work

Approximate computing has received significant interest with research efforts spanning from software to architecture and circuits. The majority of the hardware research efforts explored targeting a single accuracy in manual [11, 12] and automated design [13–16] of functionally approximate circuits. Runtime monitoring techniques, however, have shown that with temporal variations in input characteristics, the optimal accuracy to meet an application quality target also changes [18–20]; the single accuracy circuit delivers suboptimal benefits or violates the quality targets [17].

To utilize the temporal variations with approximations, accuracy configurable hardware [21–25], and generic design methodologies are proposed [26–28]. The *de facto* method to configure accuracy is data precision scaling, i.e., not propagating the LSBs of data. Energy-aware precision scaling of floating-point data is proposed in [24]. Data packing: using non-interfering paths of a circuit for simultaneous calculation is proposed in [34]. At the system level, precision scaling is applied in the memory controller [25]. A vector coprocessor with data precision reducing FIFO input buffers is proposed in [21]. It is extended with an internal PID controller [22] and later with accuracy aware ISA extensions [23]. All of these designs apply precision scaling on data, not on the circuit. Moreover, function-specific ways are proposed for configurable approximate units [30–33]. But they also only benefit from the reduced toggling activity, and not from synthesis relaxations.

Generic design methodologies can offer improving performance by synthesizing partially faster circuits [34–36] or energy efficiency by means of disabling low significance logic groups in hardware [26–28]. Voltage scaling may possibly reduce the energy in [34–36], but it comes at often ignored system-level costs of multiple additional voltage supplies, rails, and switches. Two gating mechanisms are utilized in [26]: (1) Masking logic groups by inserting control gates to their combinatorial path and (2) power gating the logic groups to partially switch off the hardware. To group the gated logic, a genetic programming based search is proposed in [27]. In [28] clock gating for approximations, *clock overgating*, is proposed. By disabling the clock signal of flip-flops, power savings are achieved in their fan-out cone. Similar to existing accuracy configurable hardware designs, gating mechanisms reduce the energy via reducing the toggling activity only.

Several approaches using the instantiation of distinct circuits that can benefit from synthesis relaxations have been proposed. In [37], two instantiations of adders and multipliers are used for reliability purposes, targeting circuit delay. The results are shown to even increase the energy, contrary to our primary goal. Multiple instantiations of floating-point units with different accuracies are used in [7]. Some of the exact processing elements of a CGRA are replaced with approximate ones in [29].These architectural solutions may not always be beneficial when leakage power is considered and they make a subset of our proposal.

Our work distinguishes from existing methodologies as it benefits from both gating and synthesis relaxations. It incorporates the existing design methodologies for accuracy configuration and existing single accuracy hardware designs towards creating a design space, that is a superset of these individual approaches. With a systematic methodology, it extends the design space of gating approaches for further energy savings at area expense.

IV. Accuracy Configurable Hardware Architecture

Dynamic accuracy configuration, as we interpret it in the scope of this paper, aims to maximally exploit energy efficiency benefits of approximate hardware while meeting a given quality target given at runtime. In this section, we first detail *gating* mechanisms and present *instantiating* approximate circuits, which is a distinct, architecture-layer design approach for accuracy configurable hardware synthesis. Next, we introduce a novel hybrid, *cross-layer* design approach that jointly considers gating and instantiating and enables a design space with fine-grain energy vs. area trade-offs. Finally, we explain the hardware execution of cycle-by-cycle accuracy configuration, facilitating dynamic runtime adjustments.

A. Gating Groups of Paths in Circuit

We utilize the gating mechanisms discussed in Section III as low area cost accuracy configuration methods [26–28]. When the clock gating, power gating, and masking compared; masking by inserting control gates into the combinatorial paths increases path delays of the circuit. Thus, either the circuit delay increases or circuit area and power increase to match the same delay with more aggressive synthesis optimizations. These effects are counter-intuitive to our energy optimization design goal. Power and area overheads are reported as up to 7.6% and 8.7% [26]. Power gating mechanism, used in [26, 27] require many cycles, prohibiting cycle-by-cycle dynamic adjustments. In comparison, clock gating for approximations can eliminate such delay overheads and allow dynamic adjustments. Clock gating is a mean for disabling configurable partitions of a circuit. In [28], *significance constrained overgating* strategy, together with clock gating candidates algorithm result in configurable degrees of precision scaling on the input registers.

While our design approach will allow for any gating mechanism to be employed, we use clock gating as a baseline in our comparisons to represent the gating approach in the remainder of the paper.

B. Instantiating Approximate Circuits with Different Accuracies

We employ adding and connecting multiple instantiations of a circuit for additional energy savings at area expense. As shown in Figure 1, the approximate instantiations have static accuracies and they exhibit lower power at the same delay as the exact instantiation. To design approximate instantiations of a circuit, we inherit the rich set of existing static approximation, i.e., single accuracy hardware design methods. By connecting separate instantiations of a circuit with different accuracies, the hardware is able to offer dynamic accuracy configuration to fulfill the varying requirements of the application at runtime.

Energy savings through functional simplification of the hardware architecture originate from having fewer gates and shorter paths compared to the exact circuit. When the exact and the approximate circuits are synthesized for the same clock delay, the shorter paths allow *synthesis relaxations*: Boolean remapping and undoing gate-level delay optimizations [38]. Boolean remapping converts a large number of parallel gates into a less number of serial gates while maintaining the boolean function (e.g., parallel-prefix adder to ripple carry adder). Undoing gate-level delay optimizations such as inserting

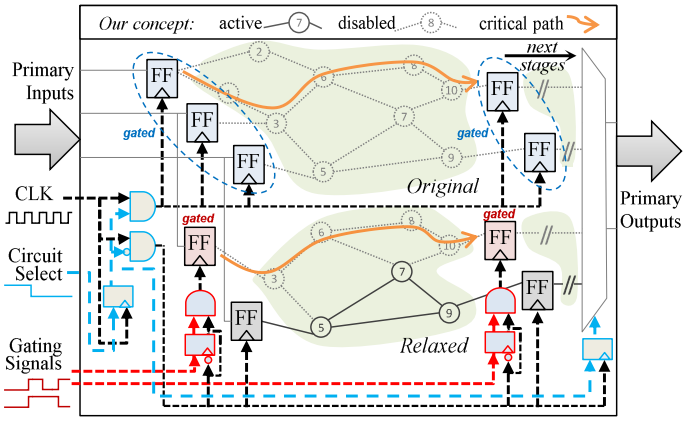

Fig. 3: Cross-layer accuracy configurable hardware architecture. The lower positioned circuit is instantiated and gated.

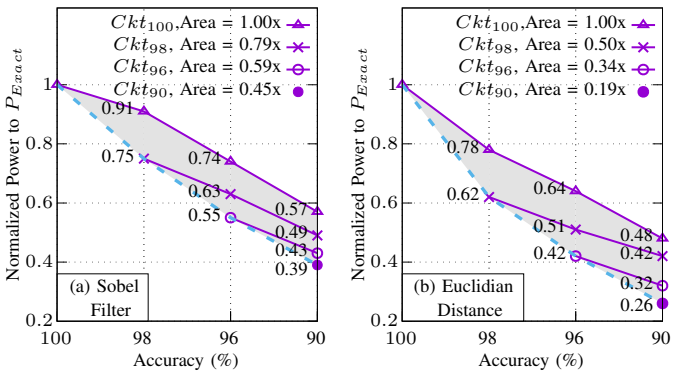

Fig. 4: Power vs. accuracy design space of the cross-layer approach. Power values are labeled. Area costs of instantiated circuits are given in the legend. All values are relative to the exact version of the corresponding circuit.

buffers (load isolation) and splitting the driving gates on the critical paths (load splitting) reduce the number of gates in the design. Gate resizing re-composes the circuit with smaller transistors that require less energy. Hence, synthesis relaxations generate inherently more energy-efficient circuits.

Our approach is agnostic to the design method of the approximate instantiations. Existing work includes tools that parameterize precision scaling [13, 14], other automated functional simplifications [15, 16], and selecting manually implemented approximate circuits from a library [29]. In this paper, we apply precision scaling to the primary inputs of the RTL and let the synthesis tool propagate this approximation using hardware compiler optimizations such as constant propagation and eliminating unused gates.

A challenge in instantiation is how to integrate the instances into the final design. Integration overheads can easily overweigh the benefits. If the accuracy-configurable hardware block is a component in a larger final design that includes a interconnection network such as a crossbar or a shared bus system, new instantiations can be connected as additional system components [29]. Larger hardware blocks such as systolic arrays [10, 21–23] share and reduce the integration overheads as they are connected through a single shared system interface. In case of multiplexer-based systems [39], this will require extending slave-to-master multiplexers for each instantiation. Such an approach also allows instantiating multiple functional units within the execution stage in a processor [7]. Alternatively, instantiations can be integrated as separate accelerators with distinct memory-mapped IO at no multiplexer cost [40]. As a guideline, in case neither memory-mapped IO nor extending a multiplexer is possible, adding a new multiplexer to the same hardware stage should be avoided for delay and power overhead reasons, as shown by previous work [26, 37].

In this paper, the instantiated circuits are at the level of at least one complete sequential stage. We call this granularity a *hardware block*. Potentially, instantiating at the combinational, sub-block level can exploit logic sharing opportunities for area savings across instantiated blocks. We utilize the final synthesis tool to find hardware common subexpressions between instantiated combinational blocks for sharing logic and thus exploit opportunities for sharing logic in an automated manner. To connect new instantiations, we consider two architectural integration decisions: memory-mapped IO at no multiplexer cost [40] and extending slave-to-master multiplexers in shared bus systems [39].

C. Cross-Layer Design Approach

While gating brings some energy benefits at low area overhead, instantiating can significantly improve the energy benefits with a higher area overhead. Our proposed *cross-layer* approach combines the two: It instantiates distinct approximated blocks at coarse accuracy levels and selectively gates them. Consequently, it enables fine-grained intermediate accuracies and additional energy savings on their computation, without the area and leakage cost of instantiating each intermediate accuracy circuit. We take an existing RTL design (*hardware block*) as input and apply instantiating at the complete block level only. Afterwards, we apply a chosen gating method which can potentially gate instantiations internally, according to its search algorithm. Figure 3 illustrates the proposed cross-layer approach. Here, an instance of the original design (top) is combined with an approximate instantiation of the original circuit (bottom). The instantiation has a shorter critical path and fewer gates. It maximizes the power savings of computation at a particular accuracy. The instantiation is also clock gated to enable further power savings for a further range of accuracies.

In Figure 4, we extend our motivational example from Figure 1 with the cross-layer design approach. Starting from exact versions of the Sobel filter and Euclidian distance circuits, we synthesize approximate circuits for each accuracy. Afterward, we gate each synthesized circuit for lower accuracies. The leftmost value of each line represents power values achieved with instantiations. The lines towards the right represent power values achieved with gating each instantiated circuit. Note that the design space of prior, gating-only approaches is limited to the Ckt_{100} line. By contrast, instantiations are limited to the left-most points on each curve (blue dashed line). The combined design space of our proposed cross-layer approach is shown by the shaded area. These examples also indicate a key insight to minimize dynamic power: We observe that for a single accuracy level, instantiating a circuit produces the most power-efficient solution, followed by gating the closest higher accuracy circuit available. In Section V, we use this insight to reduce the search space.

Given an area budget, we can instantiate a set of circuits to address varying accuracy requirements. The energy-optimal selection of such a circuit set is not a trivial task. It necessitates answering the following questions: (1) how many circuits to instantiate and at which accuracies (2) which instantiated circuits to gate and (3) how to associate different accuracy requirements of workloads with the hardware in an optimum manner. The energy optimal solution is a function of hardware and workload. Within an area budget, the additional area and associated leakage cost vs. dynamic power savings of instantiations over gating should be considered. From the leakage perspective, the impact of power savings through instantiation is thereby weighted by the utilization of the corresponding accuracy level in a given workload.

A multi-layer search, i.e., independent decisions in the architecture and the circuit would lead to first instantiating the highest utilized circuit (to make it most efficient) and then gating for other accuracies. For example, following Figure 4, if a low accuracy is utilized 70% of the time, the best strategy seems to be instantiating an efficient circuit for this accuracy. However, if gating a slightly higher accuracy instantiation results in a very close energy consumption, yet considerably increases utilization of this efficient circuit, let's say 90% of the time, instantiating a higher accuracy circuit and gating it 70% of the time can become the ideal solution. The inter-dependency between the layers, i.e., the impact of gating on instantiations influence the optimal decision. Such decisions can only be made with a cross-layer search in the joint design space. Note that the number of solutions increases quadratically with the number of accuracies as we can see in Figure 4. We need to consider their combination to address all required accuracies, which is a power set with the complexity $O(2^{n^2})$. Therefore, an automated and systematic exploration methodology is necessary. We address this challenge in Section V.

D. Runtime Accuracy Management

Our design approach supports dynamic accuracy configuration at cycle-by-cycle granularity. With this, the runtime accuracy changes can be set by a simple control unit. For each accuracy, there exists a single, static choice of circuit and gating configuration, determined at design time. I.e., no dynamic decisions are taken on circuit selection. Accuracy changes are orchestrated by an independent runtime system, such as [17–19], one cycle ahead of the operation. The frequency of accuracy changes is determined by the runtime system as a function of dynamic changes in the input stream or environment. The design of such a runtime system is out of the scope of this paper. Once an accuracy change is requested, the control unit sets *circuit select* and gating signals using a look-up table that holds the configuration for the selected accuracy. It propagates requested accuracies to the next stages in multi-stage hardware.

V. Exploration Methodology

In this section, we introduce our methodology to systematically and efficiently explore the joint design space. Here, energy optimality is a non-trivial function of required accuracies, their utilization in the workload, power savings that can be achieved at required accuracies, and leakage in the used technology.

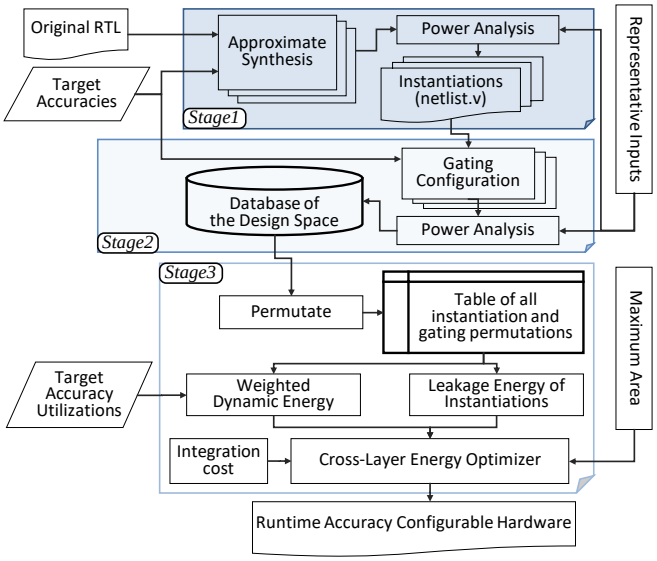

Fig. 5: Cross-layer synthesis flow.

Finding the highest energy saving combination that fits into an area constraint maps to the well studied Knapsack Problem, and it is known to be NP-complete.

In Figure 5, we present our cross-layer synthesis flow as a framework. We employ a divide and conquer alike algorithm, in which we search for the minimum energy solutions in 3 distinct hierarchical stages. We break down the cross-layer energy optimization problem to energy optimal instantiation and energy optimal gating problems. Both are further divided per accuracy. In stage 3, we combine the prior solutions to answer the original problem. Combinations of prior solutions, permutated over target accuracies gives us a complete table where each entry is a unique 3-tuple (target accuracy, instantiation, gating). Considering dynamic energy consumption weighted to circuit utilization, together with leakage and integration costs, we search for the lowest energy solution fitting into a given area constraint, by solving a 0/1 Knapsack problem with dynamic weight dependencies among elements. Hence, we decouple the cross-layer search and explore the joint meta design space on top of the separate design spaces of synthesis and gating. Our approach thereby enables: (1) use of existing, effective and well-studied tools for approximate synthesis and gating without modification, (2) modularity in choosing among approximate synthesis methods and gating mechanisms (such as clock gating, masking, or power gating). (3) significant design time savings by running synthesis and gating only one time to create a database, not recursively. Our search remains complete following the assumption that the ideal instantiations will also lead to ideal gated instantiations. This assumption held true in the extent of our experiments. Typically Knapsack Problem is solved using a form of greedy search (gradient descent/ascent, simulated annealing, etc.). With our flow, we can run an exhaustive search in a feasible time to find optimum cross-layer solutions. Next, we detail our methodology using pseudo-code and analyze its computational complexity.

Algorithm 1 describes a 3-stage exploration process, namely *approximate synthesis*, *gating*, and *optimizer*. The inputs to the algorithm are an RTL description, the maximum area constraint (A_{max}) per hardware block, representative inputs (In_{rep}), and

Algorithm 1 Accuracy configurable hardware synthesis

Input: Exact RTL block, area constraint: A_{max}, representative input: In_{rep}, accuracy and utilization list: $< ACC_{\text{list}}, U_{\text{list}} >$
Output: Accuracy configurable circuit: $Ckt_{dynamic}$

Stage 1 – Approximate Synthesis

1: **for** each $ACC_i \in ACC_{list}$ **do**
2: Ckt_acc_i = Synthesize_AX (*RTL*, ACC_i)
3: $<P_i, P_{leak\_i}>$ = get_power(Ckt_acc_i, In_{rep})
4: $Ckt_{all} = Ckt_{all} \cup Ckt\_acc_i$
5: **end for**

Stage 2 – Gating

6: **for** each $Ckt\_acc_i \in Ckt_{all}$ **do**
7: **for** each $ACC_j < ACC_i$, in ACC_{list} **do**
8: $Ckt\_acc_i\_g_j$ = apply_gating (Ckt_acc_i, ACC_j)
9: $P_{i,j}$ = get_power($Ckt\_acc_i\_g_j$, In_{rep})
10: **end for**
11: **end for**

Stage 3 – Cross-Layer Energy Optimizer

12: $CktSet_{all} = \wp^{Ckt_{all}} \setminus \emptyset$
13: $CktSet_{candidates} = \{CktSet \in CktSet_{\text{all}} | (get\_area(CktSet) \leq A_{\text{max}}) \wedge (Ckt\_acc_{max} \subseteq CktSet)\}$
14: **for** each $CktSet_k \in CktSet_{candidates}$ **do**
15: $CktSet_k$ = Synthesize($CktSet_k$)
16: **for** each ACC_i in ACC_{list} **do**
17: **if** $Ckt\_acc_i \in CktSet_k$ **then**
18: associate($CktSet_k.Ckt\_acc_i$, ACC_i)
19: $P_{CktSet_k} = P_{CktSet_k} + (P_{MUX}+P_i)*U(ACC_i) + P_{leak\_i} + P_{leak\_MUX}$
20: **else**
21: associate($CktSet_k.Ckt\_acc_{i+1}\_g_i$, ACC_i)
22: $P_{CktSet_k} = P_{CktSet_k} + P_{i+1,j} * U(ACC_i)$
23: **end if**
24: **end for**
25: **end for**
26: $CktSet_{minpower}$ = argmin($CktSet_{candidates}$, P_{CktSet})
27: $Ckt_{dynamic}$ = wrap($CktSet_{minpower}$)
28: **return** $Ckt_{dynamic}$

the list of required circuit accuracies with their utilization $(ACC_{\text{list}}, U_{\text{list}})$ in ascending order. The latter input pair can be obtained for a given workload using a profiler, e.g. from [3, 17].

Stage 1 synthesizes circuits at the same delay for all accuracies in ACC_{list}. Netlists can be generated by an automated tool that parameterizes precision scaling [13, 14] or functional simplification [15, 16], or synthesizing manually implemented approximate circuits from a library; our flow is agnostic to the synthesis method. We consider the synthesized approximate circuits optimal in the sense that they are the minimum power consuming circuits possible for that accuracy. We characterize the power of each circuit (Ckt_acc_i) using the given representative input in line 3. In line 4 we create a set of all synthesized circuits, Ckt_{all}.

Stage 2 applies a chosen gating method to each instantiation generated in stage 1 (Ckt_{all}), and for each target accuracy. In line 8, the *apply_gating* function returns the minimum power gated circuit that meets or exceeds the accuracy constraint, ACC_j. Note that gating can only reduce the accuracy of a netlist. So instantiated circuits are gated only to lower accuracies in the ACC_{list}. Possible gating strategies are given in [26] for masking and power gating, and in [28] for clock gating, where our flow is again agnostic to the chosen method. The output of this second stage is a database containing dynamic power, area values (as given in Figure 4), and also leakage power for all instantiation and gating combinations.

Stage 3 forms and evaluates circuit combinations, *CktSet*, to find the minimum power solution for a given area constraint. This decision depends on the utilization of each accuracy (U_i), the power consumption of each circuit (P_i), and also the integration cost (P_{MUX}). We show this relation in Equation (1) for a system that utilizes n different accuracies.

$$P_{CktSet_k} = \sum_{i=1}^{n} P_i U_i + P_{Leakage} + P_{\text{MUX}} \quad \text{where :} \quad U_{total} = \sum_{i=1}^{n} U_i, \;\; 0 \leq U_{total} \leq 1 \tag{1}$$

U_{total} is the portion of cycles in which at least one circuit instantiation is active, i.e., the hardware is not idle. P_{CktSet_k} denotes the total power consumption of a circuit set out of many.

The optimizer first generates all dynamic-accuracy solutions ($CktSet_{all}$) from the combinations of all prior, static-accuracy solutions (Ckt_{all}). This is the power set of Ckt_{all}, except the empty set. In line 13, we reduce the possible dynamic-accuracy solutions $CktSet_{all}$ to $CktSet_{candidates}$. These are the solutions that fit into a given area constraint and also include the highest accuracy circuit (Ckt_acc_{max}). If a *CktSet* does not include Ckt_acc_{max}, we invalidate it because a part of the workload cannot be computed at a required accuracy. In line 15, we instantiate each *CktSet* and synthesize it with the "-incremental" switch in Synopsys DC to find and exploit the opportunities for sharing logic, i.e., hardware common subexpressions, between the circuits of each set.

The second part of stage 3 is to evaluate each candidate *CktSet*. For each circuit set, we associate the accuracy requirements of the workload with circuits. When *CktSet* contains a circuit with matching accuracy ACC_i, we associate them in line 18. The *associate* function generates directives for the control system, binding configurations to accuracies. Our insight from Section IV-C has shown that when gated, the closest accuracy requires the lowest power. In case *CktSet* does not contain a matching accuracy circuit, we get the gating of the next higher accuracy circuit available in the set ($Ckt\_acc_{i+1}\_g_i$) in line 21-22, and use the configuration previously computed in line 8. This gives us a permutation of all dynamic-accuracy solutions (*CktSet*) over the target accuracies (ACC_{list}) as a complete table. With the workload associations in line 18 and 22, we know the utilization of each circuit (U_i) within each dynamic-accuracy solution (*CktSet*). We can fill our table with the power values. In lines 19 and 22, we accumulate the dynamic power consumption of each circuit weighted proportionally to its utilization. The total power consumption includes dynamic, leakage, and also integration costs. Our analysis on system integration has shown that multiplexer area, dynamic, and leakage power increase linearly with size. To consider the integration impacts, we synthesize *m-to-1* multiplexers where we changed m from 1 to 8, and used average increment in determining A_{MUX}, P_{MUX}, and P_{leak_MUX} at average primary output toggle rate. Finally, from all evaluated circuit sets, we pick the one with lowest power consumption, $CktSet_{minpower}$ in line 26. Since all circuits were synthesized to the same delay, the $CktSet_{minpower}$ is also the $CktSet_{minenergy}$. We instantiate all the circuits in $CktSet_{minpower}$ and connect them as explained in Section IV-B to generate a dynamic accuracy configurable circuit ($Ckt_{dynamic}$) in line 27.

TABLE I: Circuits used in our experiments

Name	Function	Bitwidth	I/O	$t_{ckt}[ns]$	Area [μm^2]	Input
FIR	4-Tap FIR Filter	8	8/16	0.30	1177	random uniform
Neuron	8 input ReLu Neuron	8	64/8	0.91	5865	random uniform
Sobel	3x3 Sobel Filter	8	64/8	0.60	1337	cameraman
Gaussian	3x3 Gaussian Filter	8	72/8	0.50	764	cameraman
Euclidian	Euclidian Distance (without square-root)	8	32/16	0.77	1380	random uniform

Thus, our proposed methodology systematically explores the design space of dynamic accuracy configurable hardware. It finds circuit sets, that require minimum energy, and match the required accuracies by their construction, in the joint design space of gating and instantiating approaches. With an adjustable area constraint, A_{max}, it creates a design-time knob on energy vs. area. It automates the generation of $Ckt_{dynamic}$, i.e. the energy-optimized dynamic accuracy configurable circuit.

The computations in Stage 1 and 2 are necessary only one time and can be parallelized. Afterward, our optimizer works on provided power and area values. Thus, it is completely decoupled from the previous stages. To ensure Pareto-optimal solutions, we run an exhaustive search with a complexity $\mathcal{O}(n2^n)$, where n is the number of elements in ACC_{list}. If we break it down, in line 14, the for loop operates on $CktSet_{candidates}$, which is a subset of the power set term $\wp^{Ckt_{all}}$ has the complexity of $\mathcal{O}(2^n)$. In line 16, another for loop operates on ACC_{list}, with complexity $\mathcal{O}(n)$. The insight we gained in Section IV-C allowed us to reduce the gating candidates in line 21 to one. As a result, this reduced the complexity by one order. Overall, an exhaustive search is possible within seconds for the practical range of n, where our carefully structured and computation efficient algorithm is capable of exploring a design space significantly beyond the previous work.

VI. Experiments and Results

We evaluate the effectiveness of the proposed cross-layer methodology on a variety of circuits as listed in Table I. For the synthesis, we use Synopsys Design Compiler with the ultra high effort option using an industrially proven technology library, TSMC 65nm generic plus (typical case, 1.0 Volt, 25°C). We synthesized a circuit for each given accuracy by precision scaling the primary inputs and letting the synthesis tool propagate optimizations to later stages with compiler optimizations (dead code/gate elimination, constant propagation, etc.). All instantiations of circuits are synthesized to match the same delay, i.e.,110% of the minimum delay of the corresponding exact circuit ($t_{ckt} = delay_{min} \times 110\%$). Therefore all circuits compared in our experiments have the same performance. The extra 10% delay budget was applied to give some headroom for optimizations; It is not a limitation. We instantiated each circuit directly in the HDL testbench. For gating, we reduced the number of primary inputs supplied to the netlists according to the algorithm given in the state of the art [28]. For the area and leakage cost of gating, we assumed a conservative 3% penalty per added accuracy, which is in line with [26]. We calculated energy by multiplying total power and time. To characterize the dynamic power consumption of our circuits, we generated toggling activity files (*.vcd* file) from gate-level simulations with ModelSim and provided them to Synopsys Primetime in *.saif* format. Similarly, we included the leakage power values that Primetime reported. We ignore the clock tree power of our post-synthesis netlists. In practice, there could be additional minor clock tree savings, up to 2.5% according to our experiments. The unit of length is μm in our library and we derive the area in μm^2. For a gate count comparisons, the smallest NOT and NAND gates are 1.08 and 1.44 μm^2, respectively. As inputs, in Sobel and Gaussian filter experiments we used a 512x512 pixel cameraman picture. These filters are used exclusively in image processing. For other circuits, we used random inputs with uniform distribution. Note that the 3x3 Sobel kernel inputs 8 pixels (64 bits), excluding the center pixel. The FIR filter module inputs an 8-bit value and internally propagates it through its stages.

TABLE II: Utilization distributions (U_i) of 4 different circuit accuracies for 3 synthetic workloads.

Utilization Distributions	Accuracy			
Workload	100%	98%	96%	90%
W_ex - *mostly exact*	0.5	0.2	0.2	0.1
W_eq - *even distribution*	0.25	0.25	0.25	0.25
W_ax - *mostly approximate*	0.1	0.15	0.05	0.7

Workload Profiling: The required circuit accuracies and their utilizations are application and input dependent. To abstract their effect on our methodology, we used 4 different accuracies and 3 different utilization distributions as shown in Table II. In our experiments on the Sobel filter, 98%, 96% and 90% accuracies corresponded to PSNR 45, 38 and 31dB, respectively.

Accuracy: We considered circuit accuracies in terms of $1 - MRED$ (Mean Relative Error Distance), as shown in Equation (2), where O_{approx_n} is the nth approximate and O_{exact_n} is the nth exact output value.

$$Accuracy = 1 - \frac{1}{N}\sum_{n=1}^{N}\frac{\left|(O_{approx\_n} - O_{exact\_n})\right|}{O_{exact\_n}} \qquad (2)$$

As we use the mean value in Equation (2), our error metric involves both the rate and magnitude.

Optimizer Runtime: We profiled the runtime of our Cross-Layer Energy Optimizer (*Algorithm 1 - Stage 3*) with MATLAB R2016a using integer variables on an Intel i5 6600 with 16GB RAM. In Table III, we show that an exhaustive search is feasible as it is in the order of seconds for the practical range

TABLE III: Optimizer runtime

n	$\leq$9	10	11	12
[s]	0.01	0.14	1.42	21.3

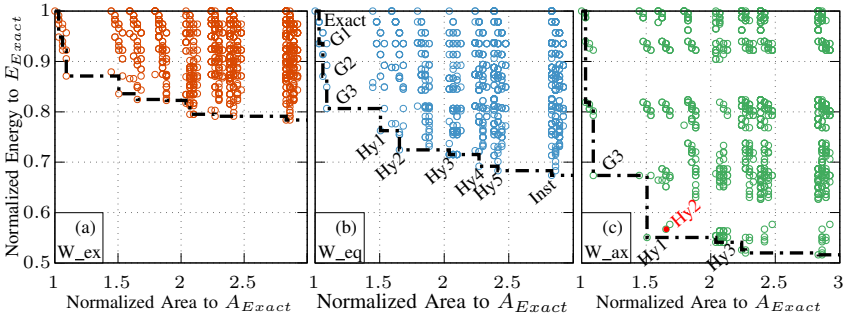

Fig. 6: Design space of an accuracy configurable Sobel filter under 3 different utilizations given in Table II. Pareto-front is obtained using the proposed methodology. Pareto solutions in Figure 6b are labeled to be used in Figures 7 and 12.

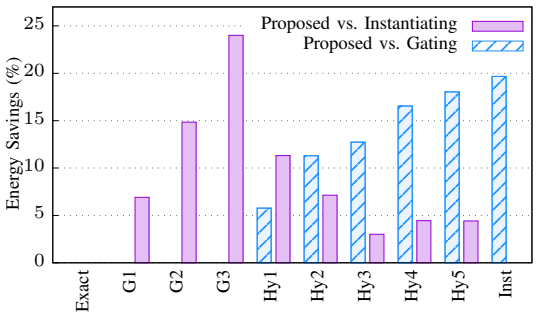

Fig. 7: Pareto front comparison of cross-layer against gating and instantiating solutions from Figure 6b.

of the number of required accuracies n (3-6 in [7, 17, 21, 23, 25–28, 34, 36, 41]). In fact, our computation-efficiently structured algorithm is capable of exploring a design space for double the n, that is quadratically larger than the previous work.

A. Design Space Exploration

We begin our evaluations by presenting the design space of a Sobel filter obtained using our proposed methodology in Figure 6 under 3 different workloads from Table II. Each solution on the design space corresponds to a particular combination of instantiated circuit, gating and workload accuracy association. We show the Pareto front obtained by our proposed methodology with a dashed line. The Pareto solutions we refer to in the text are labeled as follows: *G1-G3* are gating-only solutions, *Hy1-Hy5* are hybrid solutions of gating and instantiations in the cross-layer design space, and *Inst* is the instantiation-only solution. When examined at 2× maximum area constraint, the Pareto-optimum solution in Figure 6b is labeled *Hy2*. At the same area cost, under *W_ax* workload in Figure 6c, *Hy1* is the optimum solution, which dominates *Hy2*. *Thus, Pareto-optimal solutions are workload dependent.*

The dynamic power impact of instantiating an approximate circuit is proportional to the utilization of its accuracy. Our methodology first instantiates the highest power impact solution. At the excess area, only low impact circuits remain. As an example, under the mostly approximate workload in Figure 6c, 90% is the dominating accuracy. With solution *Hy1*: [$Ckt\_acc_{100}g_{98}g_{96}$, Ckt_acc_{90}] (i.e., instantiating exact and 90% accuracy circuits and gating the exact for 98% and 96%) we already achieve significant energy savings. Additionally instantiating a 96% accuracy circuit (Ckt_acc_{96}) only reduces the energy consumption by a mere 2.6% at $0.6\times A_{\text{Exact}}$ extra area cost. Similarly, under *W_eq*, where the accuracy utilization is even, an extra 0.65× area at first reduces energy by 28% w.r.t. *Exact* (10% w.r.t. *G3*), and at the end, only 1.3%, w.r.t. *Hy5*. *Thus, energy savings diminish at excess area.*

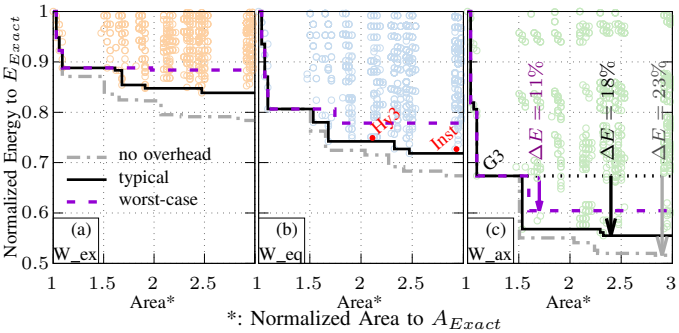

Fig. 8: Design space of an accuracy configurable Sobel Filter with *typical* integration costs (control+MUX+clk) under 3 different utilizations given in Table II. Pareto-fronts are obtained using the proposed methodology with varying integration costs.

B. Comparison of Pareto-Optimal Solutions

In Figure 7, we show the energy savings of cross-layer solutions over gating and instantiating alone for the Sobel Filter with *W_eq* workload. We obtain the gating Pareto front by providing only the exact circuit as the output of Stage 1 in Algorithm 1. Similarly, to obtain the Pareto front of the instantiating approach, we skip Stage 2 in Algorithm 1 and associate the missing accuracies with a higher accuracy circuit. The figure highlights that, as we increase the area budget, the Pareto solutions of *cross-layer* and *instantiating* approaches offer energy reduction over state-of-the-art gating approaches. *The joint design space offers solutions that are superior or equal to the two individual approaches.*

C. Analysis of Integration and Control Overhead

The integration overheads, as we have previously discussed in Section IV-B, are dependent on the connection to an enclaving hardware system. The experiments in Figures 6 and 7 represent energy savings when the system integration costs can be averted, such as in adding instantiations with memory-mapped IO [40]. For a shared bus architecture, we need to extend the slave-to-master multiplexers to connect new instantiations [39].

We included an m-to-1 multiplexer in our design space exploration in the experiments shown in Figure 8. The system-level integration overheads manifest themselves as additional energy and area as a function of m, i.e., the number of instantiations. In Figure 8, we show their impact on the Pareto-front for 3 scenarios: The worst-case line represents using the fastest multiplexer generated by the synthesis tool with the delay constraint set to 0 ns to maximize synthesis effort. The typical line represents a multiplexer with an excess delay budget. In our experiments, the multiplexer delay has been a fraction of the circuit delay ($< 41\%$). Hence, if necessary, multiplexers could always comfortably fit into a pipeline stage that is delay constrained by the experimented circuit. Finally, we included the 'no overhead' line from the previous experiments, as shown

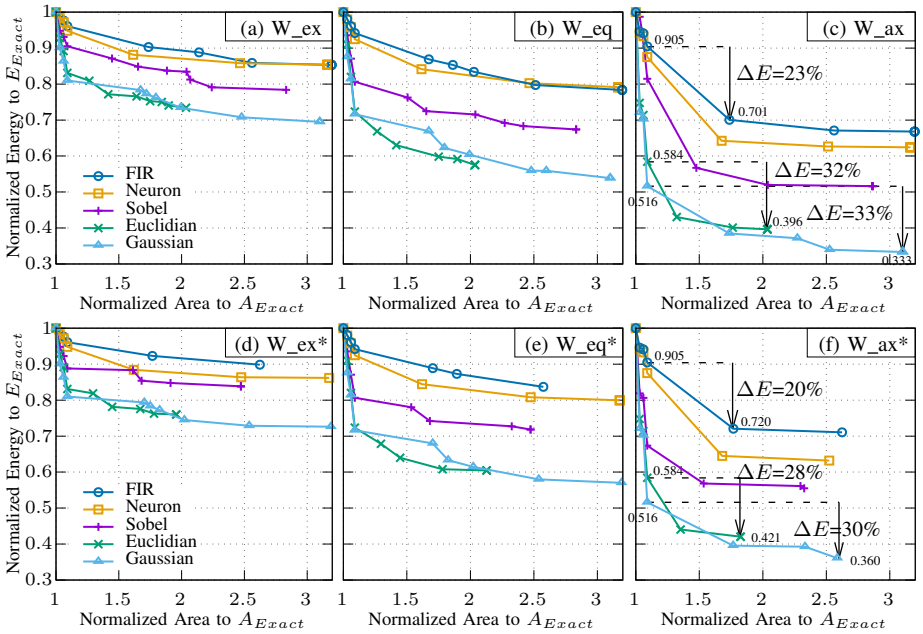

Fig. 9: Pareto curves for a range of circuits under workloads with accuracy utilizations given in Table II. The experiments denoted with a "*" in (d,e,f) include system integration overheads. Each point represents a particular dynamic accuracy configurable circuit, $Ckt_{dynamic}$, which is energy optimized for the given area and the workload.

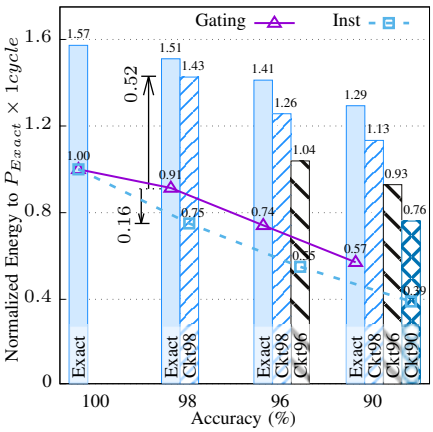

Fig. 10: Energy required for accuracy reconfiguration of the Sobel Filter: Each bar represents one-time energy to activate a circuit for the desired accuracy. The purple line represents clock gating the exact circuit for the desired accuracy. The blue line is the runtime energy requirement of instantiations at their designed accuracy, without switching. Normalized to the average of $P_{\text{exact}} \times 1cycle$, for Cameraman picture as input.

in Figure 6. Values are normalized to the area and the energy of the Sobel Filter. Depending on the scenario, Figure 8c shows 11% to 23% energy savings over *G3*, the most energy-efficient solution of the state-of-the-art gating method.

Note that in the typical case, the solution of instantiating all circuits (*inst*) is no longer on the Pareto front for any workload. We also observe that *Hy3* is dominated in Figure 8b. These are the first (lowest area) solutions that use 4 and 3 instantiations and their energy reduction do not compensate for the increased integration overheads. This finding supports the previous argument of *energy savings diminish at the excess area* for a different reason: increasing overheads.

We have also explored placing multiplexers inside the pipeline stages of the hardware block, similar to the previous work [26, 37]. As such, we can enable sharing some stages and configuring accuracy in others. This setup has not given energy savings more than gating as it significantly reduces the delay margin for both the instantiations and the multiplexer and thus more aggressive delay optimizations that increase energy.

D. Area vs. Energy Trade-offs

In Figure 9, we generalize our exploration by repeating it for a range of datapath circuits. We share the energy-area Pareto front obtained under 3 different workloads from Table II, with (d,e,f) and without integration overheads (a,b,c). For each circuit, after the exact, the first 3 data points represent the gating solutions and for Figure 9a,b,c the last data point represents the solution at which all circuits are instantiated. Figure 9 shows that *our cross-layer synthesis methodology is general and it applies to a wide range of circuits.*

We can observe that Pareto points change in number and x-axis position under different workloads. For instance, the Gaussian filter has 10 Pareto solutions under the workloads *W_ex* and *W_eq* whereas 8 under *W_ax*. This validates the workload dependency of Pareto solutions. Similarly, a maximum area constraint of $2 \times A_{Exact}$ results in an average 18%, 28%, and 48% energy savings under W_ex, W_eq and W_ax workloads, respectively. Afterward, an additional area budget of A_{Exact} only reduces energy by 4%, 5%, and 1%. These values support our previous finding on diminishing energy savings at excess area. In other terms, instantiating all accuracies solution (*inst*) has only a small, incremental energy saving over cross-layer solutions at relatively large area cost.

Previously, in Figure 1, we shared power savings with *instantiating* compared to an exact Euclidian distance calculator as up to 78%. Compared to gating an exact circuit, we reported additional power savings of up to 46% with instantiation at matching accuracy. These values set the maximum possible savings: under a workload utilizing constantly 90% accuracy (U_{90} = 1). Between our workloads, *W_ax* shows higher energy savings than *W_eq* and *W_ex*, as it utilizes the energy-efficient, approximate instantiations more. At 2x area cost, the power reduction under the mostly exact workload in Figure 9a is up to 26% whereas the reduction under the mostly approximate workload reaches 60% in Figure 9c. When compared to the state-of-the-art gating approach and at matching accuracy, in Figure 9c, *we achieve up to 32% additional energy savings.*

In the experiments shown in Figure 9(d,e,f), we considered the system-level integration overheads by including an *m-to-1* multiplexer in our design space exploration methodology, generalizing our previous overhead analysis in Section VI-C. We observe that some Pareto points such as solutions that instantiate all (*inst*) vanish; effectively, for the remaining Pareto-solutions, the energy overhead of integration is less than 4%. When Figure 9(d,e,f) are examined, apart from the initial low-cost gating solutions, *only the cross-layer solutions remain on the Pareto-front.*

E. Energy Cost of Runtime Accuracy Reconfiguration

Runtime systems decide and dictate an accuracy reconfiguration in relatively large periods [17, 18], and independently of our hardware synthesis methodology. Architecturally, as shown in Figure 3, an accuracy change is conducted by setting a small number of registers ($\sim log(n)$, for n accuracies). Although this register energy cost is negligible, activating a different instantiation can result in a context switch cost.

In the real-world, due to temporal and spatial correlations in input data, we can expect that not all input bits change and lead to toggles and energy consumption in every cycle. In other words, bit-level input correlation is a factor in energy consumption. For instance, in a picture, neighboring pixels are likely to have a similar value, as do successive frames in a video stream. Running a kernel on the equal value pixels of a graphic would not lead to any toggling activity. Small changes in the input are likely to toggle only LSB paths which are inherently short and more energy-efficient [35]. However, activating a different circuit instantiation by switching accuracy or changing the input data context leads to a high amount of toggle in the hardware. Upon a context switch, the correlation in inputs is lost and the energy consumption is expected to be higher. The impact can be generalized as energy consumption upon context change.

Each time we switch to a different instantiation, we can expect the current input values to not be correlated to what the inputs of the instantiation were the last time it was used: A context switch occurs to instantiations upon activation. To evaluate this energy impact of input data correlation, we have tested the Sobel Filter with uniform random input values that have no correlation from one cycle to the next, which emulates a setup in which accuracy re-configuration occurs in every cycle. In Figure 10, we compare the average power consumption per cycle with accuracy configuration (bars) against the runtime cost of using the same circuit with normal inputs benefiting from correlation (lines). In other words, bars represent the energy consumption of switching to and activating a specific instantiations when changing the hardware accuracy, while the lines represent the energy consumption at a fixed accuracy, where purple is a gated exact circuit and blue is instantiations without gating. The figure can be read as follows: Normalized to an energy unit of $P_{Exact} \times 1cycle$, 1.43x energy is required to switch to the 98% accuracy instantiation. In comparison, if there are no instantiations and only the exact circuit, the circuit could be gated to 98% at no context switch cost, resulting in a fixed energy cost of $0.91x$ as shown with the purple line. Hence, by switching to a different accuracy circuit, we incur an energy overhead. However, instantiated circuits are more energy-efficient than the gated ones at the same accuracy. Considering a net switching cost of $(1.43 - 0.91) = 0.52$x and following savings of $(0.91 - 0.75) = 0.16$x, our approach can amortize an accuracy reconfiguration within 4 cycles.

This analysis is particularly useful as it abstracts the hardware from the workload and hence retains generality. We see the impact of switching between instantiations on energy, independent of the workload characteristics such as the switch frequency and previous state (as in a finite state machine).

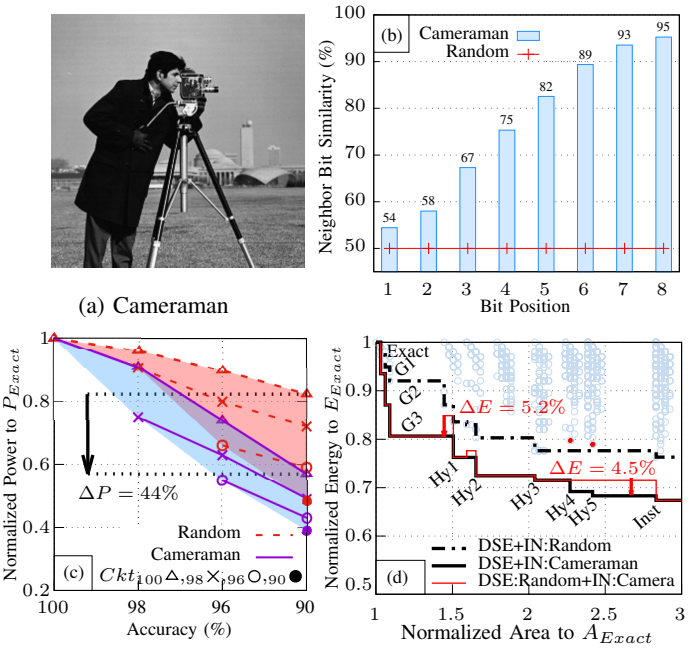

Fig. 11: Input dependency of cross-layer design space. (a) Cameraman picture used in our experiments. (b) Bitwise similarity between neighbor pixel values of the Cameraman picture vs. uniform random numbers. (c) Power consumption of a Sobel Filter with random inputs compared to the cameraman picture as input. Lines with different markers represent the dynamic power consumption of instantiations when gated, as in Figure 4. (d) Design Space of accuracy configurable Sobel filter under W_eq workload from Table II, with random inputs, highlighting the mismatch in solutions and its energy impact.

F. Input Dependency of Cross-Layer Design Space

The dynamic power consumption depends on the characteristics of the input stream. Figure 11a is the cameraman picture used in our experiments. We use it is a representative of real-world data with spatial correlations (similarly valued neighbor pixels). The image processing kernels we used, Sobel and Gaussian filters, work on neighboring pixels and shift over the image pixels like a sliding window for each output. The required computation is strongly affected by spatial correlations. In Figure 11b, we analyze the bitwise similarity between neighboring pixels of the cameraman picture using MATLAB. The figure can be read as follows: the MSB of the pixels have the same value for 95% of the neighbors; only 5% of the neighbor pixels have a different MSBs. The LSBs of the neighbor pixels, however, have the same value for only 54%. *Using real-world data, the kernel inputs have relatively high-frequency changes between LSBs compared to their MSBs.* In comparison, uniform random input has the same, 50% toggle rate for each bit position. As we lower accuracy through precision scaling, we omit the high frequency LSB toggles, which leads to bigger normalized savings in real-world data (up to 44%, in Figure 11c) than random. Since both experiments are on the same circuit, the area and leakage power remain the same. To the scope of our experiments, *precision scaling leads to higher normalized energy savings with inputs from real-world data than random inputs with uniform distribution.*

We run our cross-layer synthesis methodology using both random and representative input values for comparison by changing the inputs in line 3 and 9 of *Algorithm 1*. In Fig-

ure 11d, we present the design space exploration of the accuracy configurable Sobel filter with random input and the Pareto-fronts for random and representative inputs when evaluated running the same inputs as used during search. Comparing the Pareto-fronts, we notice a difference in energy consumption. The red line in Figure 11d shows the solutions for random input evaluated by running on the cameraman input. An in-depth analysis shows that most of the solutions on both Pareto-fronts are the same configuration. However, some Pareto-solutions are input dependent. Using an unrepresentative input during design space exploration can lead to suboptimal solutions, which cost up to 5.2% energy. By contrast, performing exploration using representative inputs reveals additional Pareto-solutions (*Hy4, Hy5*) for further energy savings of up to 4.5% in this example.

G. Leakage Energy Analysis and Technology Independence

Instantiating additional circuits increases area and hence the leakage of the hardware. The contribution of leakage energy over total energy is a function of the technology and hardware utilization, U_{total}: the portion of cycles, in which at least one circuit is active. In Figure 12, we present the total energy consumption of dynamic accuracy configurable Sobel filters under the workload *W_eq* for different technologies and varying utilization factors, achieved by introducing idle cycles. The y-axis shows the total energy (including the leakage), normalized to the total energy of the exact circuit. Note that the x-axes (utilization factor) are on a logarithmic scale. The graph can be read as follows: at a utilization factor of 1, there are no idle cycles. At 0.1 the hardware is used for 1 cycle out of 10 on average, i.e., 9 idle cycles. Here, the energy contribution of the dynamic part is also reduced to 0.1, while the leakage remains constant.

In Figure 12a, dynamic accuracy configurable circuits that we use are from the Pareto front of Figure 6b. We plot their energy under two different utilization scales: The main (bottom) x-axis represents our analysis with the TSMC65nm library at the typical corner. The second x-axis represents results for the worst-case corner instead. The TSMC65nm library characterized for worst-case conditions (0.9V, 125C) results in an increase in leakage to dynamic power ratio by 2x. This is equivalent to doubling the number of idle cycles, i.e., scaling utilization by a factor of 0.5 as shown in the figure. The highest energy savings are obtained with instantiation (inst) and cross-layer solutions (*Hy1-Hy5*) at high hardware utilization, such as 1 to 0.1 for the x-axis representing the typical corner. As the effect of leakage power becomes more prominent, the energy of *Hy1-Hy5, and inst* increase above gating-only solutions *G1-G3* between 0.1 and 0.05 (marked in Figure 12a). Until the range of 0.05 to 0.0125, the *Hy1-Hy5, and inst* remain superior to the exact circuit.

We applied the Sobel Filter to a camera system application with a 512×512 pixel video feed at 30 frames per second, running at a clock frequency of 100MHz. In this specification, the utilization factor is 0.08. Even though this can be considered as a low utilization to justify the accelerator, the best solution is *Hy2* (with typical x-axis). When we increase the resolution and frame rate to 1280×720 pixels and 60fps, the utilization

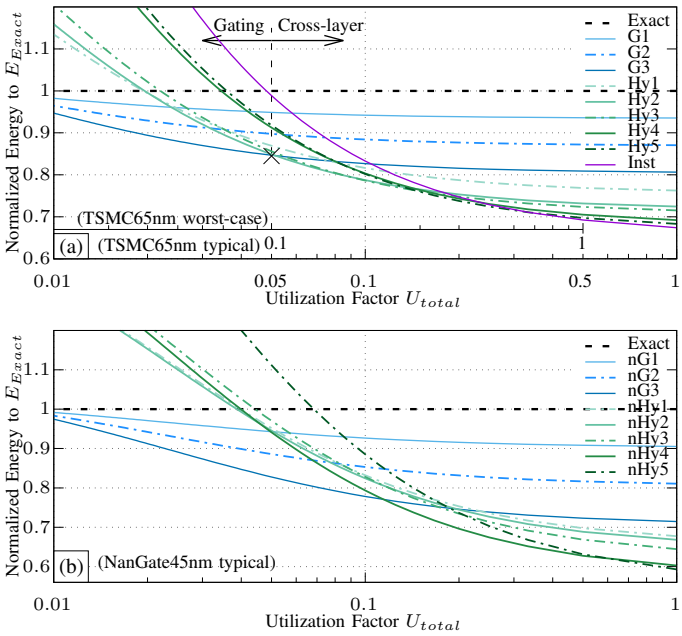

Fig. 12: Leakage impact on accuracy configurable Sobel filters under varying utilization factor U_{total} with different technology libraries. Each line is a Pareto solution under the workload *W_eq*.

factor becomes 0.54. At this value, leakage power has a much lower impact on total energy. The instantiation solution (*inst*) offers 0.4%, 26%, and 37% more energy savings than solutions *Hy5*, *G3* and *exact*, respectively.

The data can be read in a technology-independent manner, for different leakage impact on total energy, by scaling the utilization factor axis accordingly, as we show in Figure 12a. Furthermore, In Figure 12b, we redo the experiments (synthesis, gating, design space exploration, and leakage analysis) with a NanGate 45nm library. Note that the Pareto-points in Figure 12a and Figure 12b are not the same set. For instance, the NanGate *inst* solution is dominated by *nHy5* and not drawn in Figure 12b. There are differences in synthesis and gating results that change the design space and Pareto-points. The NanGate experiments validate our previous leakage analysis using a completely independent technology library. With these experiments, we can generalize that *for a significant range of utilization factor, the cross-layer approach produces superior solutions.*

VII. Conclusion

We addressed the necessity of accuracy-configurable hardware systems with the exploration of applying gating mechanisms to existing circuits together with instantiating more efficient circuits into the architecture. Jointly, they present a larger design space where non-trivial cross-layer decisions are necessary to find optimal solutions. We proposed a systematic methodology to ensure Pareto-optimal combinations towards minimizing energy consumption under given workload and area constraints. Our work has demonstrated that dynamic accuracy configurable hardware with significantly (up to 33%) reduced energy compared to existing gating solutions can be synthesized when more circuit area can be utilized.

REFERENCES

[1] T. Alan, A. Gerstlauer, and J. Henkel, "Runtime accuracy-configurable approximate hardware synthesis using logic gating and relaxation," in *DATE*, 2020.

[2] S. T. Chakradhar and A. Raghunathan, "Best-effort computing: re-thinking parallel software and hardware," in *DAC*, 2010.

[3] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC*, 2013.

[4] T. Lee, M. Hwangbo, T. Alan, O. Tickoo, and R. Iyer, "Low-complexity hog for efficient video saliency," in *ICIP*, 2015.

[5] S. Cass, "Taking ai to the edge: Google's tpu now comes in a maker-friendly package," *IEEE Spectrum*, 2019.

[6] J. Song, Y. Cho, J.-S. Park, J.-W. Jang, S. Lee, J.-H. Song, J.-G. Lee, and I. Kang, "7.1 an 11.5 tops/w 1024-mac butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile soc," in *ISSCC*, 2019.

[7] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *DATE*, 2018.

[8] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *ISSCC*, 2014.

[9] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *ICML*, 2015.

[10] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017.

[11] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *DATE*, 2008.

[12] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *ISIC*, 2009.

[13] J. Miao, A. Gerstlauer, and M. Orshansky, "Approximate logic synthesis under general error magnitude and frequency constraints," in *ICCAD*, 2013.

[14] S. Lee and A. Gerstlauer, "Fine grain precision scaling for datapath approximations in digital signal processing systems," in *VLSI-SoC*, 2013.

[15] J. Castro-Godínez, S. Esser, M. Shafique, S. Pagani, and J. Henkel, "Compiler-driven error analysis for designing approximate accelerators," in *DATE*, 2018.

[16] I. Scarabottolo, G. Ansaloni, and L. Pozzi, "Circuit carving: A methodology for the design of approximate hardware," in *DATE*, 2018.

[17] S. Yesil, I. Akturk, and U. R. Karpuzcu, "Toward dynamic precision scaling," *IEEE Micro*, 2018.

[18] W. Baek and T. Chilimbi, "Green: A system for supporting energy-conscious programming using principled approximation," in *PLDI*, 2010.

[19] M. A. Laurenzano, P. Hill, M. Samadi, S. Mahlke, J. Mars, and L. Tang, "Input responsiveness: using canary inputs to dynamically steer approximation," in *PLDI*, 2016.

[20] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: An online quality management system for approximate computing," in *ISCA*, 2015.

[21] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan, "Scalable effort hardware design," *IEEE TVLSI*, 2014.

[22] V. Chippa, A. Raghunathan, K. Roy, and S. Chakradhar, "Dynamic effort scaling: Managing the quality-efficiency tradeoff," in *DAC*, 2011.

[23] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Quality programmable vector processors for approximate computing," in *MICRO*, 2013.

[24] C.-C. Hsiao, S.-L. Chu, and C.-Y. Chen, "Energy-aware hybrid precision selection framework for mobile gpus," *Computers & Graphics*, 2013.

[25] A. Jain, P. Hill, S.-C. Lin, M. Khan, M. E. Haque, M. A. Laurenzano, S. Mahlke, L. Tang, and J. Mars, "Concise loads and stores: The case for an asymmetric compute-memory architecture for approximation," in *MICRO*, 2016.

[26] S. Jain, S. Venkataramani, and A. Raghunathan, "Approximation through logic isolation for the design of quality configurable circuits," in *DATE*, 2016.

[27] V. Mrazek, Z. Vasicek, and L. Sekanina, "Design of quality-configurable approximate multipliers suitable for dynamic environment," in *AHS*, 2018.

[28] Y. Kim, S. Venkataramani, K. Roy, and A. Raghunathan, "Designing approximate circuits using clock overgating," in *DAC*, 2016.

[29] M. Brandalero, A. C. S. Beck, L. Carro, and M. Shafique, "Approximate on-the-fly coarse-grained reconfigurable acceleration for general-purpose applications," in *DAC*, 2018.

[30] M. Imani, D. Peroni, and T. Rosing, "Cfpu: Configurable floating point multiplier for energy-efficient computing," in *DAC*, 2017.

[31] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Walking through the energy-error pareto frontier of approximate multipliers," *IEEE Micro*, 2018.

[32] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *DAC*, 2012.

[33] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *ICCAD*, 2013.

[34] B. Moons and M. Verhelst, "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *ISLPED*, 2015.

[35] T. Alan and J. Henkel, "Slackhammer: Logic synthesis for graceful errors under frequency scaling," *IEEE TCAD*, 2018.

[36] D. J. Pagliari and M. Poncino, "Application-driven synthesis of energy-efficient reconfigurable-precision operators," in *ISCAS*, 2018.

[37] B. Boroujerdian, H. Amrouch, J. Henkel, and A. Gerstlauer, "Trading off temperature guardbands via adaptive approximations," in *ICCD*, 2018.

[38] *Design Compiler® User Guide*, Synopsys, www,synopsys.com, 2010.

[39] *ARM AMBA 5 AHB Protocol*. [Online]. Available: https://static.docs.arm.com/ihi0033/bb/IHI0033B_B_amba_5_ahb_protocol_spec.pdf

[40] *MSP430 Hardware Multiplier*. [Online]. Available: http://www.ti.com/sc/docs/products/micro/msp430/userguid/ag_06.pdf

[41] D. J. Pagliari, E. Macii, and M. Poncino, "Automated synthesis of energy-efficient reconfigurable-precision circuits," *IEEE Access*, 2019.

Tanfer Alan is currently pursuing the Ph.D. degree at Chair for Embedded Systems (CES) at Karlsruhe Institute of Technology (KIT), Germany, under the supervision of Prof. Jörg Henkel. He received his bachelors degree from Hacettepe University, Turkey and his masters degree from TU Darmstadt, Germany, in 2011 and 2014, respectively. His current research interests include approximate computing, cross-layer design, design automation, and architectures for emerging workloads. He holds a US patent.

Andreas Gerstlauer (Senior Member, IEEE) received the Ph.D. degree in information and computer science from the University of California, Irvine (UCI), CA, USA, in 2004. He was an Assistant Researcher with the Center for Embedded Computer Systems, UCI, from 2004 to 2008. He is currently a Professor with the Electrical and Computer Engineering Department, The University of Texas at Austin, TX, USA. He has coauthored three books and over 100 refereed conference and journal publications. His research interests include system-level design automation, system modeling, design languages and methodologies, and embedded hardware and software synthesis. His work has received several best paper nominations from, among others, DAC, DATE, and HOST, and two best paper awards at DAC16 and SAMOS15. He was a recipient of a 2016-2017 Humboldt Research Fellowship. He has been a General and the Program Chair for conferences such as MEMOCODE and CODES+ISSS. He also serves as an Associate Editor for ACM TODAES and ACM TECS journals.

Jörg Henkel (Fellow, IEEE) received the Diploma and Ph.D. (summa cum laude) degrees from the Technical University of Braunschweig. He was a Research Staff Member with the NEC Laboratories in Princeton, NJ, USA. He is currently the Chair Professor for embedded systems with the Karlsruhe Institute of Technology. His research work is focused on co-design for embedded hardware/software systems with respect to power, thermal, and reliability aspects. Prof. Henkel received six best paper awards throughout his career from, among others, ICCAD, ESWeek, and DATE. He has led several conferences as the General Chair including ICCAD, ESWeek, and serves as a Steering Committee Chair/Member for leading conferences and journals for embedded and cyberphysical systems. He coordinates the DFG program SPP 1500 Dependable Embedded Systems and is a Site Coordinator of the DFG TR89 collaborative research center on Invasive Computing. He is also the Chairman of the IEEE Computer Society, Germany Chapter. For two consecutive terms, he served as the Editor-in-Chief for the ACM Transactions on Embedded Computing Systems. He is also the Editor-in-Chief of the IEEE Design & Test Magazine. He is/has been an Associate Editor for major ACM and IEEE journals.