Designing Approximate Circuits using Clock Overgating *

Younghoon Kim, Swagath Venkataramani, Kaushik Roy and Anand Raghunathan
School of Electrical and Computer Engineering, Purdue University
{kim1606,venkata0,kaushik,raghunathan}@purdue.edu

ABSTRACT

Approximate computing is an emerging paradigm to improve the efficiency of computing systems by leveraging the intrinsic resilience of applications to their computations being executed in an approximate manner. Prior efforts on approximate hardware design have largely focused on circuit-level techniques. We propose a new approach, *clock overgating*, for the design of approximate circuits at the Register Transfer Level (RTL). The key idea is to gate the clock signal to selected Flip-Flops (FFs) in the circuit, even during execution cycles in which the circuit functionality is sensitive to their state. This saves power in the clock tree, the FF itself and in its downstream logic, while a quality loss ensues if the erroneous FF state propagates to the circuit output. We develop a systematic methodology to identify an energy-efficient overgating configuration for any given circuit and quality constraint. Towards this end, we develop 3 key strategies — significance-based overgating, grouping FFs into overgating islands, and utilizing internal signals of the circuit as triggers for overgating — that efficiently prune the large space of possible overgating configurations. We evaluate clock overgating by designing approximate versions of 6 machine learning accelerators, and demonstrate energy benefits of 1.36× on average (and upto 1.80×) for negligible (<0.5%) loss in application quality (classification accuracy).

Categories and Subject Descriptors

B.7.1 [**INTEGRATED CIRCUITS**]: VLSI (Very large scale integration)

Keywords

Approximate Computing; Clock Gating; Energy Efficiency

1. INTRODUCTION

Many prevalent and emerging application domains such as multimedia, recognition, mining, data analytics, search and vision among others, possess the characteristic of *intrinsic application resilience*, which enables them to produce outputs of acceptable quality, despite approximations to some of their computations. Intrinsic application resilience arises from several factors: algorithms are designed to be robust to noise, input data contains redundancies, applications only need to produce an acceptable output rather than a unique golden result, *etc.* [1, 2]. Approximate computing leverages intrinsic application resilience and relaxes the need for strict correctness in the execution of computations to improve energy and/or performance.

*This work was supported in part by the National Science Foundation under Award #1423290.

DAC '16, June 05–09, 2016, Austin, TX, USA
 ISBN 978-1-4503-4236-0/16/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2897937.2898005.

Over the last decade, several research efforts have developed techniques for approximate computing in both software [2, 3] and hardware [4–19]. Approximate hardware may be realized by operating circuits under overscaled conditions [4, 5] (*e.g.*, lowering the voltage to a point where timing errors occur), or by designing hardware that contains fewer transistors or gates but deviates to a limited extent from the specification [6–17]. A vast majority of approximate hardware design techniques are at the circuit or logic levels of abstraction. It is desirable to explore approximate design at the higher levels of abstraction, since from the designer's point of view, quality is most naturally defined and evaluated at the application level. For example, it is much more natural to talk about classification accuracy or image quality than to specify accuracy or error constraints at the outputs of individual adders or multipliers within a circuit.

The Register Transfer Level (RTL) remains the most common level of abstraction at which hardware designs are specified in the industry. In addition to inherent advantages in terms of faster simulation and analysis times, RTL descriptions contain higher-level semantic information (data types, bit positions, operators, *etc.*) that can be leveraged in the process of approximation.

Very few research efforts have previously investigated approximate computing at the RTL. Notably, Axilog [19] proposes Verilog HDL extensions that designers can use to specify which signals and operations in the RTL code can be subject to approximations. In contrast, our objective is to develop a new approximate design technique, and hence Axilog [19] is complementary to our proposal. Another effort, ABACUS [18], generates approximate designs from behavioral/RTL descriptions by applying pre-specified approximation transformations to individual operations or operands. A key drawback of this approach is that the approximations are fixed at design time and hardwired into the circuit implementation. This is an issue in practice, since the same hardware may be used to execute computations that can tolerate different levels of approximation, or the application context may dictate the use of different accuracies for the same computation. These considerations strongly suggest that the approximations employed should be reconfigurable at runtime.

We propose a new approximation technique called *clock overgating* for the design of approximate hardware. Clock gating is one of the most widely adopted low-power techniques, in which clock signals to sequential elements (flip-flops or latches) in the circuit are suppressed to reduce power, provided that doing so preserves the exact functionality of the circuit. We extend this concept to propose clock overgating, where we gate the clock signal to a Flip-Flop (FF) *even when doing so may affect the circuit outputs*. Clock overgating may result in incorrect outputs when an FF is supposed to change state but incorrectly retains its previous state, and this error propagates to the circuit outputs. In return, switching power is saved in the clock tree, the FF itself, and the fanout logic cone of the FF. We note that these power savings are over-and-above the savings due to conventional clock gating and other low-power techniques.

Clock overgating has the following desirable properties: (i) It is easily reconfigurable, *i.e.*, the clock overgating signal to each FF can be modulated dynamically in a fine-grained manner, (ii) it preserves the structure of the circuit and is hence minimally-intrusive, and (iii) it leverages the wide support for clock gating that is present in com-

mercial EDA tools and design flows.

Given the RTL description of a circuit, an input testbench and an output quality constraint, we propose a systematic methodology to identify where (in which FFs) and when (in which clock cycles) to perform overgating. The search space for overgating, defined by all possible FFs and all possible execution cycles, is extremely large (*e.g.*, 2^{10000} for a circuit with 100 FFs that operates for 100 cycles), and is even more challenging in cases when designs take a variable number of cycles to complete execution. Rather than explicitly search through this prohibitively large space, our methodology utilizes internal signals from the circuit to trigger clock overgating. Doing so restricts the search space, and also has the added benefit of incurring minimal overhead in the logic that realizes the overgating conditions. Further, we group FFs in a circuit into clock overgating islands based on their location in the circuit and how they impact the overall application output. FFs in each overgating island are constrained to have the same overgating condition, greatly reducing the search space without significantly affecting the energy savings from overgating. We then perform a gradient descent search to identify the set of clock overgating island–trigger signal combinations that maximize the improvement in energy for the specified quality constraint.

In summary, the key contributions of this work are as follows:

- We propose clock overgating, a new technique to design approximate circuits at the RTL, in which sequential elements in the circuit are clock gated even when doing so may affect the circuit outputs.
- We develop a systematic methodology to determine the clock overgating conditions for sequential elements in any given circuit. Our methodology groups sequential elements into overgating islands and uses internal circuit signals as overgating triggers to efficiently prune the space of possible overgating configurations.
- We apply clock overgating to develop approximate versions of accelerators for 6 machine learning applications, and demonstrate 1.36× average improvement in energy for <0.5% loss in quality at the application-level.

The rest of the paper is organized as follows. Section 2 describes related research efforts and highlights the key distinguishing features of our work. Section 3 presents the basic concepts of clock overgating and Section 4 discusses the proposed overgating methodology. The experimental methodology is described in Section 5 and the results are subsequently presented in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

The field of approximate computing has received considerable interest in recent years. Several research efforts have proposed approximate design techniques at different levels of abstraction, including software [2,3], architecture [20,21], and circuits [4–19]. In this section, we focus our discussion on techniques proposed for designing approximate circuits and position our work in their context.

Research on approximate circuits began with manual designs of basic arithmetic components such as adders [6–10] and multipliers [10, 11]. Approximate design techniques such as voltage overscaling [4, 5], truncating carry chains [8], and input operand partitioning [9] were proposed in this context. Modifying the Boolean function realized by the circuit to reduce logic complexity was proposed as an approach to approximate circuit design in [12]. Subsequent efforts broadened this concept to the approximate design of arbitrary circuits using techniques such as redundancy propagation [13], path pruning [14], and don't care based simplification [15], among others. All of the above efforts operate at the logic level of abstraction, and target the design of approximate combinational circuits. How such approximate circuit blocks should be composed to form larger designs remains an open challenge.

In contrast to the aforementioned efforts, we focus on a higher level of abstraction, *viz.* RTL, where very few efforts have explored approximate computing. Axilog [19] provides Verilog HDL extensions, which can be used to specify and identify portions of the design that are safe to approximate. Axilog is complementary to our work as our objective is to develop new approximate design techniques that can be applied to the identified parts. ABACUS [18] applies predefined approximate transformations (*e.g.*, scaling bitwidth, strength reduction) to operators in the RTL description. However, a key limitation of [18] is that the approximations are hardwired into the circuit implementation, and cannot be changed at runtime. In practice, hardware needs to be re-used in different application contexts, operate on different inputs, or execute different operations within an application; all these scenarios require the ability to modulate the degree of approximation at runtime. Clock overgating is inherently quality configurable, since one can enable overgating in different subsets of FFs to realize distinct energy *vs.* quality tradeoffs. Furthermore, it has other desirable attributes such as preserving the overall structure of the circuit and can be easily integrated into existing design flows. Finally, recent work has explored High-Level Synthesis (HLS) of approximate circuits through precision scaling [17]. Notwithstanding significant advances in HLS, we note that hardware is often still designed at the RTL in the industry, hence we focus on the complementary problem of approximating any given RTL design in a quality-configurable manner.

3. CLOCK OVERGATING: DESIGN APPROACH

Given a hardware design described at the RTL and a quality constraint specified at its outputs, our objective is to design an approximate version through clock overgating that is as energy-efficient as possible, while meeting the specified quality constraints. This section describes the basic concepts behind clock overgating, the challenges involved in applying it to the design of approximate circuits, and our approach to addressing these challenges.

3.1 Clock Overgating: Concept

Clock gating is a popular low-power design technique that is widely used to reduce dynamic power dissipation. In clock gating, additional logic is embedded in the clock tree of the circuit, which suppresses the clock signal transitions from reaching one or more sequential elements (FFs or latches) in the circuit under certain conditions. Since the clock signal is suppressed, the target FFs cannot switch states and therefore switching power is saved in the clock tree leading to the FFs, the FFs themselves and their fanout logic cones. To preserve functionality, clock gating is performed only when an FF's state is guaranteed not to switch, or if it can be proven that the circuit's outputs are insensitive to the FF's inputs in the gated execution cycles.

In this work, we propose clock overgating, whose concept is illustrated in Fig. 1. In clock overgating, selected FFs in the circuit are gated even when doing so may result in an incorrect circuit output. In other words, local errors are introduced in the circuit due to incorrect FF states, and they may propagate to the circuit output, potentially causing the output quality to degrade. In return, since FFs are being gated for additional execution cycles beyond conven-

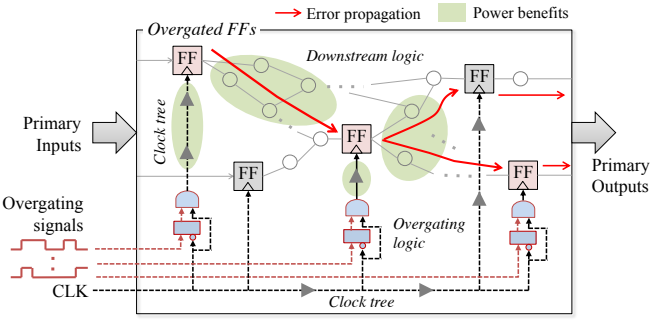

Figure 1: Clock overgating concept

tional clock gating, it results in improved dynamic power and energy. Thus, clock overgating presents designers with a means to tradeoff energy *vs.* quality. The key question that arises, which we discuss next, is how to utilize clock overgating to achieve the best energy benefit for a given output quality?

3.2 Clock Overgating: Design Space

Identifying a *clock overgating configuration* involves defining: (i) which FFs in the circuit should be overgated, and (ii) when (during which clock cycles) the selected FFs should be overgated, or in other words, what conditions should be used to trigger overgating for the selected FFs. We refer to the FFs that are candidates for overgating as Clock Overgating Targets (COTs), and the signals used to trigger clock overgating in them as Clock Overgating Enables (COEs). Fig. 2 shows the design space for clock overgating. If M is the number of FFs in a circuit, which operates for N cycles, then each FF can be either gated or active in each cycle, leading to a total of 2^{M*N} overgating configurations. Clearly, a brute force search of all possible overgating configurations is infeasible for practical designs. Therefore, efficient heuristics are key to exploring this very large design space and to identifying the clock overgating configuration that is most energy-efficient for a given output quality constraint (QC).

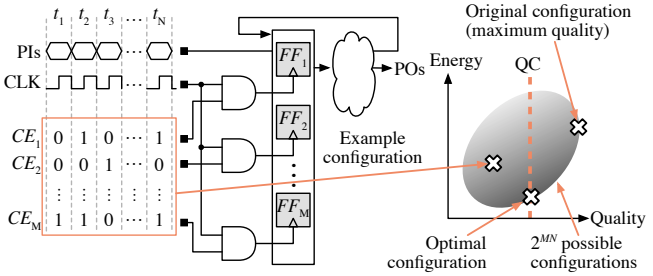

Figure 2: Design space of clock overgating configurations

To address the above challenge, we adopt 3 strategies that leverage the semantic information available at the RTL to significantly reduce the number of COTs as well as the number of possible COEs for each COT. Although these strategies do not guarantee optimality, *i.e.*, they may result in a suboptimal overgating configuration, our experimental results demonstrate that they work well in practice and yield significant improvements in energy even for very tight quality constraints.

Significance constrained overgating: In general, each FF in the circuit can be regarded as a COT and overgated independently. However, in practice, multiple FFs are semantically grouped into registers that together store/represent multi-bit data. We extract this information from the RTL description of the circuit and exploit the bit significance of individual FFs in their respective registers to constrain the sequence in which they are overgated. In typical circuits, errors in the LSBs of registers impact the output quality by an exponentially smaller amount relative to the MSBs. Further, the LSBs typically have the most switching activity and fan out to larger cones of logic; hence, they can lead to higher energy savings when overgated. Therefore, when identifying COEs for FFs in a given register, we proceed in the order of their bit significance, *i.e.*, we identify COEs for the LSBs first before considering the MSBs for overgating. In summary, the first strategy prunes the search space by associating FFs with registers specified at the RTL, and constraining the manner in which they are overgated based on bit-significance.

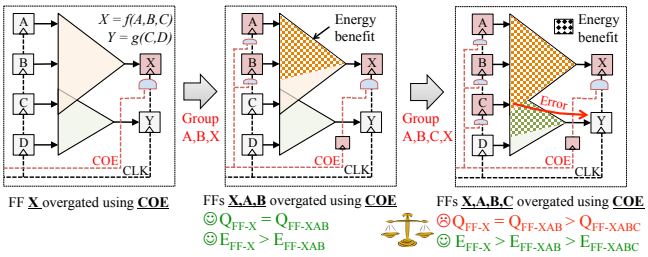

Figure 3: Functional grouping of FFs

Functional grouping of FFs: In this strategy, we reduce the number of COTs by grouping multiple FFs into overgating islands, and overgating all FFs in an island together. We achieve this by leveraging the functional relationship between FFs that can be easily identified at the RTL. Fig. 3 illustrates the opportunity for FF grouping. Consider a scenario where clock overgating in FF X is triggered by signal COE. This enables energy savings in X and in its downstream logic. Now, as shown in Fig. 3, X is functionally related to FFs A, B and C *i.e.* X changes state only in response to a change in A, B or C. Also, note that FFs A and B exclusively fan out to X, whereas C also fans out to other FFs in the circuit. In this case, the key observation is that when COE is used to clock overgate FFs A and B in the cycle before X is overgated, it results in no additional quality loss at the circuit outputs, as any errors introduced at A and B propagate only to X, which itself is overgated in the next cycle. However, this results in additional energy savings in the FFs A and B, and in the combinational logic connecting FFs A and B with X. Therefore, we group FFs X, A and B into a single overgating island, and use the same signal (COE) to overgate all of them. Note that when grouped, the overgating signal to X is delayed by one clock cycle, as the values in A and B propagate to X only after a cycle.

The above strategy can be extended to include an FF in a group even when some of its fanouts lie outside the group. In this case, grouping may result in additional quality degradation, as errors introduced in the FFs due to overgating can propagate to the circuit outputs through its fanouts that lie outside the island. In the example shown in Fig. 3, if C is grouped along with X, A and B, the error introduced in C impacts its fanouts other than X, leading to an additional degradation in quality. This additional quality degradation needs to be compared against the net energy benefits to decide if it is favorable to add an FF to an overgating island. A detailed description of how FFs are grouped into overgating islands is presented in Section 4.

Using internal signals as COEs: The final strategy addresses the large space of possible COE candidates that could be used to trigger overgating in each COT. Since logic to generate COEs needs to be added to the circuit, a key constraint in picking COEs is that they should incur low overhead. To reduce the search space, while also minimizing the energy overheads in the logic that generates COEs, we propose to use internal signals already present within the circuit as COEs. In addition, to ensure that the timing constraints on the gating signal are satisfied, and that no glitches are introduced on the

gated clock signals, we further restrict COEs to be the outputs of FFs in the circuit and their complements. While this restriction greatly prunes the design space, our experiments suggest that it still enables a rich and favorable energy-quality tradeoff. Our expereience suggests that in practice, circuits contain control states such as address counters, FSM state variables *etc.* that naturally predict which registers in the circuit are active and likely to affect the output the most.

In summary, we propose clock overgating, an RTL approximation technique in which clock signals to FFs in a circuit are gated even when the circuit outputs may be affected, thereby resulting in a tradeoff between energy and output quality. We also propose three key strategies to prune the space of possible overgating configurations, enabling clock overgating to be applied to any given input circuit while meeting the desired quality constraint.

4. DESIGN METHODOLOGY

In this section, we present a systematic methodology for clock overgating that realizes the design approach described in Section 3.

Algorithm 1 Approximate design using clock overgating

Input: Original circuit Ckt_{orig},
Application dataset $AppData$, Quality constraint QC
Output: Approximate circuit Ckt_{app}
1: COT_{List} = form_COTs(Ckt_{orig})
2: $COT_{subList}$ = select COTs in LSB position from COT_{List}
3: COE_{List} = FFs and their complements in Ckt_{orig}
4: $Ckt_{opt} = Ckt_{orig}$ with COE=0 $\forall$ COTs in COT_{List}
5: **do**
6: $Ckt_{app} = Ckt_{opt}$; $FOM_{opt} = 0$
7: $COT\text{-}COE_{List}$={<T,E> | $T \in COT_{subList}$, $E \in COE_{List}$}
8: **for** each <COT,COE> in $COT\text{-}COE_{List}$ **do**
9: Ckt_{tmp}=assign COE as ovg. signal for COT in Ckt_{app}
10: $\Delta E = E_{orig}$ - get_energy(Ckt_{tmp},$AppData$)
11: $\Delta Q = Q_{orig}$ - get_quality(Ckt_{tmp},$AppData$)
12: $FOM = \Delta E/\Delta Q$
13: **if** (ΔE>0 **and** ΔQ<QC **and** $FOM > FOM_{opt}$) **then**
14: $COT_{opt} = COT$; $COE_{opt} = COE$
15: $Ckt_{opt} = Ckt_{tmp}$; $FOM_{opt} = FOM$
16: **end if**
17: **end for**
18: remove COT_{opt} from $COT_{subList}$
19: $COT_{subList}$ = add COT containing next significant bit of COT_{opt} from COT_{List}
20: **while** ($Ckt_{app} \neq Ckt_{opt}$)
21: **return** Ckt_{app}

Algorithm 1 shows the pseudocode of the proposed methodology. Given the original circuit (Ckt_{orig}), an input dataset ($AppData$), and a constraint on the output quality (QC), the algorithm produces an energy-efficient approximate version (Ckt_{app}) that is clock overgated and satisfies the specified quality constraint. First, a list of possible clock overgating targets (COT_{List}) and clock overgating enables (COE_{List}) are formed by employing the different design space pruning strategies described in Section 3 (lines 1-3). To this end, the FF grouping strategy is first used to form the COT_{List} (line 1). The details of this process are explained in Algorithm 2. Next, the significance constrained overgating strategy is used to further reduce the list of COTs (resulting in $COT_{subList}$). In this case, only the COTs that are in the least significant bit positions in their corresponding registers are chosen from COT_{List}, and added to $COT_{subList}$ (line 2). Next, since only FF outputs and their complements are considered as possible clock gating enables, COE_{List} is initialized with all FFs and their complements in the circuit (line 3).

Despite significantly reducing the design space using the various strategies, trying out all possible COT-COE combinations becomes computationally expensive for larger circuits. Therefore, we adopt a gradient descent approach (lines 5-20), where in each iteration we select the COT-COE pair that yields the best energy *vs.* quality tradeoff without violating QC. To this end, we first initialize the COE for each COT to logic 0, which corresponds to no overgating (line 4). Then, we form a list of all possible COT-COE pairs ($COT\text{-}COE_{List}$) by pairing each COT in $COT_{subList}$ with each COE in COE_{List} (line 7). Next, for each COT-COE pair we form a candidate overgated circuit by assigning the COE under consideration as the overgating signal for the COT (line 9). Subsequently, we evaluate the energy benefits (line 10) and the corresponding quality loss (line 11). Note that COT-COE pairs that cause unacceptable quality degradation are captured during quality evaluation and filtered out from further $COT\text{-}COE_{List}$ formations. We then compute a figure of merit (FOM) for each COT-COE pair, which is given by the ratio of energy saved to quality lost (line 12). We identify the COT-COE pair that has the best FOM (lines 13-16) and update the approximate circuit by "committing" the COE to the COT (line 15). The committed COT is removed from $COT_{subList}$ (line 18), and the COT containing the next significant bit position is added to $COT_{subList}$ (line 19). This process is repeated until no COT-COE pair yields further improvements in energy without violating QC (line 20), at which point the algorithm terminates and returns the approximate circuit (line 21).

Algorithm 2 Identify overgating candidates

Input: Original circuit Ckt, Application dataset $AppData$
Output: Clock overgating candidate list: COT_{List}
1: COT_{List} = FFs in Ckt
2: G = create_FF_connection_graph(Ckt)
3: W = get_RTL_simulation_waveform(Ckt,$AppData$)
4: $TSC(FF_1,FF_2)$: temporal correlation between FF_1&FF_2 in W
5: **do**
6: **for** each COT in COT_{List} **do**
7: **for** each COT_{In} in fan-in(COT,G) **do**
8: **for** each COT_{In-out} in fan-out(COT_{In},G) **do**
9: $TSC_{tmp} = TSC(COT_{In},COT_{In-out})$
10: ΔE_{local} += TSC_{tmp} × fan-out-size(COT_{In},COT_{In-out})
11: **if** ($COT_{In-out} \neq COT$) **then** ΔQ_{local} += TSC_{tmp}
12: **end for**
13: **if** ($\Delta E_{local}/\Delta Q_{local} > \eta$) **then**
14: COT_{List} = merge(COT,COT_{In})
15: **end if**
16: **end for**
17: **end for**
18: **while** (COT_{List} modified?)
19: **return** COT_{List}

Identifying clock overgating candidates: Algorithm 2 presents the methodology used to obtain the list of clock overgating candidates in line 1 of Algorithm 1. Algorithm 2 essentially implements the FF grouping strategy presented in Section 3. First, COT_{List} is initialized to contain all the FFs in Ckt (line 1). Then COT_{List} is reduced by grouping FFs whose switching activities are highly correlated (lines 2-18). To this end, a FF connection graph (G), whose vertices represent the FFs and whose edges represent a combinational path between two FFs, is constructed from the RTL code (line 2). Then, the circuit RTL is simulated with the given dataset, and the switching activity of each FF is recorded (line 3). Subsequently, the degree of 1-cycle temporal switching correlation (TSC) between FF pairs that have an edge connecting them in G is computed (line 4). We de-

note two FFs to be temporally correlated if they are likely to switch in successive cycles. Next, for each COT and its fan-in (COT_{In}), we compute the local energy savings expected in all the fan-outs of COT_{In} if it is overgated along with COT (line 10). Similarly, we evaluate the local errors expected in the fan-outs of COT_{In} that are not contained within COT, assuming that COT_{In} is overgated along with COT (line 11). We merge COT_{In} with COT if the ratio of the local energy savings to the local error induced is greater than a threshold η (lines 13-15). In our implementation, we determine the value of η empirically as discussed in Section 6.3. This process is repeated until no pair of candidates in COT_{List} can be merged (line 18), and the final COT_{List} is produced (line 19).

5. EXPERIMENTAL METHODOLOGY

This section describes the experimental setup used to evaluate clock overgating.

Benchmarks. Our benchmark circuits consist of algorithm-specific hardware accelerators for 6 popular machine learning applications listed in Table 1. Classification accuracy, or the fraction of application inputs correctly classified, was used as the measure of output quality for all the benchmarks. We used separate datasets for training and testing/validation, and the same testing dataset was used to evaluate the original and approximate circuits.

Table 1: Benchmark applications

Algorithm	Application	Dataset	FFs / Gates
GLVQ	Eye detection (EYE)	Image set from NEC labs	183 / 3100
KNN	Optical digit classification (OPT)	OCR digits	344 / 4128
	Webpage classification (W5A)	LIBSVM	422 / 7348
SVM	Text classification (TEXT)	Reuters	480 / 27760
ANN	Digit classification (DIGT)	MNIST	398 / 29833
	Face detection (FACE)	YUV faces	394 / 31428

Quality and power evaluation. The design methodology and heuristics described in Section 4 were implemented as a custom script that executes the following design flow. First, a custom RTL parser extracts necessary information from the input RTL code and produces an approximate version. The output quality (classification accuracy) of the approximate circuit is obtained by RTL simulation with the testing dataset using Mentor Graphics ModelSim. RTL simulation also produces the switching activity information that is used for power estimation. Power is estimated by synthesizing the approximate circuit to IBM 45nm technology using Synopsys Design Compiler, and estimating the circuit power including the clock tree power using Synopsys PrimeTime. We note that the most aggressive optimization options available by Synopsys Design Compiler (conventional clock gating, gate sizing, use of low leakage cells, *etc.*) were applied to create well-optimized baselines for comparison.

6. RESULTS

In this section, we present the results of various experiments that demonstrate the benefits of clock overgating and underscore the effectiveness of the proposed strategies in exploring the large design space of overgating configurations.

6.1 Energy-quality tradeoff

Fig. 4 shows the energy reduction obtained using clock overgating for various output quality constraints. Note that output quality constraints are expressed in terms of classification accuracy degradation with reference to the baseline, and the energy consumption of each approximate circuit is normalized to the energy consumption of its baseline implementation. As shown in Fig. 4, clock overgating provides significant energy benefits between 1.07×–1.80× (1.36× on average) at the application level with virtually no loss (<0.5%) in classification accuracy. When the quality constraints are relaxed to <2% and <5%, the energy benefits increase to 1.10×–1.80× (1.43× on average) and 1.12×–1.80× (1.50× on average), respectively.

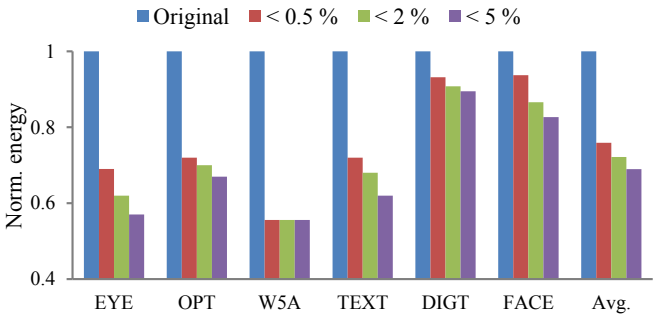

Figure 4: Energy improvements for various quality constraints

The graph shown in Fig. 5 breaks down the energy from Fig. 4 into 3 key components: (i) leakage (static) power, (ii) dynamic power in the registers and the clock tree, and (iii) dynamic power in the combinational logic. We only present the <5% quality constraint case here, but our observations hold for other cases as well. Again, all the energy components are normalized to the baseline implementation. First, we find that the overhead due to the increase in leakage power is negligible (0.21% on average). This suggests that the overhead for clock overgating was kept minimal (cell area increased by 0.28% on average) by using the internal signals as COEs. Next, we observe that clock overgating reduces the power dissipated in the clock tree, the FFs' internal power, as well as the dynamic power consumed by the combinational logic. A general trend shown in Fig. 5 is that the power improvements are more pronounced in the combinational logic than in the clock tree and the FFs. This is due to the fact that only a small fraction (16.9% on average) of the total FFs are clock overgated in the design, and further, these overgated FFs are dominated by LSBs. The parts of the combinational logic that process LSBs typically exhibit higher switching activity and therefore overgating the corresponding FFs results in considerable dynamic power savings without a significant impact on output quality.

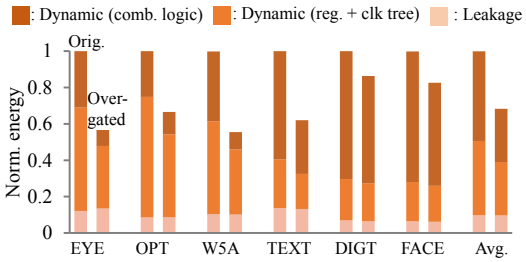

Figure 5: Energy breakdown analysis

6.2 Comparison with Precision Scaling

The graph shown in Fig. 6 compares the energy benefits achieved by clock overgating and precision scaling for various target quality constraints. In the case of precision scaling, different numbers of LSBs were clock gated in all data path registers, throughout the evaluation of the circuit. It can be clearly seen that clock overgating outperforms precision scaling in terms of energy benefits; for example, 1.36× *vs.* 1.19× on average for a quality constraint of <0.5%. This is due to the fact that clock overgating allows for a more fine-grained control over which FFs should be gated in which execution cycles.

6.3 Effectiveness of FF grouping

As described in Section 3, we group multiple FFs into overgating islands, which significantly cuts down the search space of overgat-

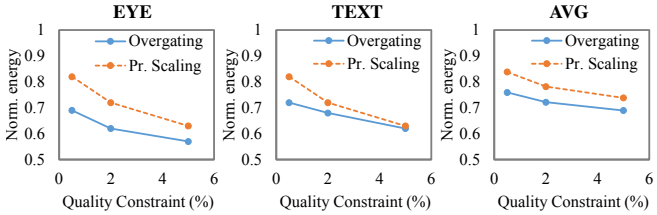

Figure 6: Energy benefits using clock overgating *vs.* precision scaling

ing configurations. However, this renders clock overgating to be more coarse grained, potentially impacting the energy efficiency of the overgated circuit. Fig. 7 quantifies this tradeoff for 2 benchmarks by varying the number of COTs into which the FFs of the circuit are grouped, and observing the runtime of the methodology as well as the energy efficiency of the resulting overgated circuit. Note that, in Fig. 7, energy is normalized to that of the baseline circuit. A quality constraint of <5% classification accuracy was used in these experiments; however, the conclusions hold for other quality constraints as well. We observe that the runtime increases with an increase in the number of COTs. In the case of energy, the benefits of clock overgating are very small when FFs are aggressively grouped. The energy efficiency of the overgated circuit improves drastically once the number of COTs crosses a threshold. However, beyond a point, the benefits saturate and very fine grained overgating does not yield additional improvements. Thus, if carried out to the right extent, FF grouping can achieve significant improvements in overall runtime, without a significant loss in the energy benefits. For instance, in the case of EYE, when its 183 FFs are grouped to form 129 COTs (using η=3.5 in Algorithm 2), the runtime reduces by 5.8× at the cost of a mere 1.8% decrease in energy savings. Based on our experiments, we have observed that η values of 3.2–3.5 work well for all circuits. We note that by combining the strategies described in Section 3, the total runtime was reduced substantially to < 4 hours for a circuit size of ~30K gates. While we believe this is a reasonable runtime for our proof-of-concept implementation, it can be considerably reduced by utilizing static techniques (*vs.* simulation) for switching activity estimation, and parallelizing across multi-cores (for example, the loop over COT-COE combinations in Algorithm 1 can be parallelized) to ensure scalability for larger circuits.

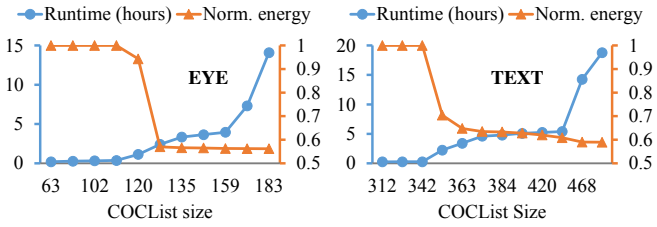

Figure 7: Runtime *vs.* optimality tradeoff analysis

6.4 Illustration of clock overgating in action

Fig. 8 visually illustrates clock overgating in action by plotting the activity of FFs during around 1000 cycles of execution of the EYE application. In Fig. 8, each point on the *Y* axis corresponds to a distinct FF, while the *X* axis represents execution cycles. The color at each (x, y) coordinate indicates the status of FF y at cycle x. In a given cycle, an FF can be either active, clock gated or clock overgated, marked by blue, green and red colors respectively. Also, the length of the segment in the *Y* axis corresponding to each FF is proportional to the energy benefits possible by gating the FF. Therefore, the area in Fig. 8 occupied by the overgated regions roughly corresponds to the overall energy benefits. We observe that only 12% of the FFs (FF index 162–183) are overgated in the circuit, but since they and the combinational logic that they feed contribute up to 50% of the total energy, we achieve ~40% improvement in energy.

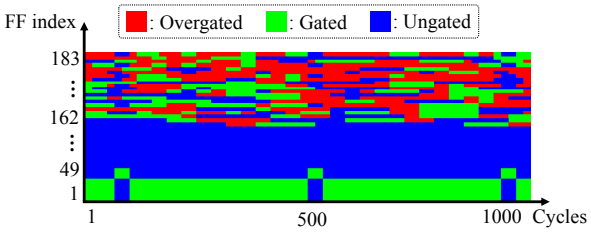

Figure 8: Clock overgating status over time

7. CONCLUSION

The intrinsic resilience in emerging applications provides new opportunities for optimizing hardware through approximate computing. In this work, we propose clock overgating, a register-transfer level approximation technique. The key idea behind clock overgating is to suppress the clock signal to selected FFs in the circuit even when the circuit functionality is affected as a result, thereby creating a trade-off between energy and quality. We developed a systematic methodology that identifies an energy-efficient clock overgating configuration given any arbitrary circuit and output quality constraint. The methodology employs three key strategies that leverage the semantic information available at the RTL to efficiently explore the large space of overgating configurations. Our results across hardware implementations for a range of machine learning algorithms suggest that clock overgating is a promising approach to approximate hardware design and that it enables favorable tradeoffs between energy and output quality.

8. REFERENCES

[1] M. A. Breuer. Multi-media applications and imprecise computation. In *Proc. Euromicro Conf. on Digital System Design*, pages 2–7, 2005.

[2] S. T. Chakradhar and A. Raghunathan. Best-effort computing: Re-thinking parallel software and hardware. In *Proc. DAC*, pages 865–870, June 2010.

[3] Sasa Misailovic et al. Chisel: Reliability-and accuracy-aware optimization of approximate computational kernels. In *Proc. OOPSLA*, pages 309–328, 2014.

[4] P. K. Krause and I. Polian. Adaptive voltage over-scaling for resilient applications. In *Proc. DATE*, March 2011.

[5] Rajamohana Hegde and Naresh R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Proc. ISLPED*, pages 30–35, 1999.

[6] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *Proc. ICCAD*, pages 728–735, 2012.

[7] V. Gupta et al. IMPACT: Imprecise adders for low-power approximate computing. In *Proc. ISLPED*, pages 409–414, Aug. 2011.

[8] N. Zhu et al. An enhanced low-power high-speed adder for error-tolerant application. In *Proc. ISIC*, pages 69–72, 2009.

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

[10] J. Huang et al. A methodology for energy-quality tradeoff using imprecise hardware. In *Proc. DAC*, pages 504–509, 2012.

[11] P. Kulkarni et al. Trading accuracy for power with an underdesigned multiplier architecture. In *Proc. VLSI Design*, pages 346–351, Jan. 2011.

[12] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Proc. DATE*, pages 957–960, March 2010.

[13] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *Proc. DATE*, pages 1–6, March 2011.

[14] A. Lingamneni et al. Energy parsimonious circuit design through probabilistic pruning. In *Proc. DATE*, pages 1–6, Mar. 2011.

[15] S. Venkataramani et al. SALSA: Systematic logic synthesis of approximate circuits. In *Proc. DAC*, pages 796–801, 2012.

[16] A. Ranjan et al. ASLAN: Synthesis of approximate sequential circuits. In *Proc. DATE*, pages 1–6, 2014.

[17] Chaofan Li et al. Joint precision optimization and high level synthesis for approximate computing. In *Proc. DAC*, pages 104:1–104:6, 2015.

[18] K. Nepal et al. ABACUS: A technique for automated behavioral synthesis of approximate computing circuits. In *Proc. DATE*, pages 1–6, 2014.

[19] A. Yazdanbakhsh et al. Axilog: Language support for approximate hardware design. In *Proc. DATE*, pages 812–817, 2015.

[20] Bo Zhai et al. Energy-efficient subthreshold processor design. *IEEE Transactions on Very Large Scale Integration Systems*, 17(8):1127–1137, Apr. 2009.

[21] Bradley Thwaites et al. Rollback-free value prediction with approximate loads. In *Proc. PACT*, pages 493–494, 2014.