Gate Sizing in the Presence of Gate Switching Activity and Input Vector Control

Nathaniel A. Conos, Saro Meguerdichian, and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{conos, saro, miodrag}@cs.ucla.edu

***Abstract*—We introduce a novel gate sizing approach that considers both the gate switching activity (SA) and gate input vector control leakage (IVC). We first extract SA using simulation and find promising input vectors. Next, in an iterative framework, we interchangeably conduct gate sizing and refining the IVC. As dictated by the new objective function, our algorithm conducts iterative gate freezing and unlocking with cut-based search for the most beneficial gate sizes under delay constraints. We evaluate our approach on standard benchmarks in 45 nm technology, showing promising improvement, achieving up to 62% (29% avg.) energy savings compared to the traditional objective function.**

I. Introduction

Gate sizing is a powerful optimization technique used to minimize power and/or area under strict timing constraints by altering the widths of transistors in gates. Gate sizing has been extensively studied over the past three decades [1][2][4] and several approaches have been proposed. Previous approaches, however, do not consider switching activity (SA) and the impact of input vector control leakage (IVC), which greatly impact the overall optimization strategy. Furthermore, gate sizing is often combined with dual or multi-threshold techniques which further increases the importance of accurate power and delay modeling. As a result, the modern design flow imposes a number of modeling and optimization challenges

A major challenge is the simplification of timing and power models, which may lead to suboptimal solutions when mapping out to real designs. Accounting for accurate gate and interconnect delay and its dependencies on capacitive load slew are often ignored [4]. Additionally, nominal gate switching activity and/or average gate leakage are generally assumed in previous works limiting the potential improvements by accounting for realistic operating conditions. Moreover, previous approaches are either dynamic or leakage power-centric in their optimization flows, which do not address the varying application usage characteristics present in high-performance systems (e.g., data-centers, super-computing) to energy constrained mobile devices (e.g., tablets, smart-phones).

Cell library-based optimization has emerged as the de facto standard for modeling power and delay of a circuit design. Many previous approaches, however, utilize simplified timing models by assuming convex and/or linear delay and power models [3]. Empirical analysis has shown that accurate timing models are non-linear/non-convex. Furthermore, optimizing circuit designs using a discrete cell library, however, leads to solving an NP-Hard problem [5]. As a result, many heuristics have been developed in order to address the huge problem search space. A major drawback of these methods is that they require heavy parameter tuning and are difficult reproduce, since they are technology dependent, and are rooted to a set of sensitivity functions. These methods often perform iterative per-gate or per-group improvement are too compute intensive and are impractical to be applied on modern IC sizes, even with incremental updates. Furthermore, these approaches mainly perform local optimization (i.e., local-moves) and are susceptible to be trapped in local minimas [6].

The usage of modern cell libraries, however, have enabled the support various supply/threshold voltages, and drive strengths, enabling a rich performance and energy trade-off to address the potentially vastly differing device usage characteristics. However, current tools do not account for realistic conditions into their objective functions (e.g., gate activity, duty cycle, input vector control), with respect to their applications, potentially impacting obtained results.

We improve state-of-the-art sizing methodologies by simultaneously considering gate switching activity (SA) and gate input vector control (IVC). One of our key contributions is that we demonstrate significant benefits of incorporating actual gate SA and gate IVC in the objective function over the equivalent approach that only uses nominal and average values for switching and leakage weights, and show how the obtained solution varies when accounting varying duty cycles.

The focal point of our approach is a scalable gate sizing algorithm that considers gate SA and IVC leakage. The steps are to: 1) extract the SA of gates based on simulation of real workloads; and 2) conduct IVC to obtain the input vector that induces the lowest total leakage energy across all gates, and 3) an iterative gate sizing approach freezes maximally-constrained gates (ones that are at high-power states as determined by SA and IVC) while searching for a sizing option that best improves the current picture. The objective function in step 3 to be considered at the iteration depends on the types of options available and their impacts on both delay and energy. The algorithm prevents the algorithm from reaching a local minima by freezing gates as they are sized until all gates have been frozen, then unfreezes all gates, re-conducts IVC (since new gates may be energy-dominant), and reiterates steps 2 and 3 until the solution converges or the delay constraint cannot be met.

We evaluate our approach on benchmarks included in ISCAS-85/89, ITC99 and arithmetic units. Our results indicate over 62% (29% avg.) energy improvement over a method that

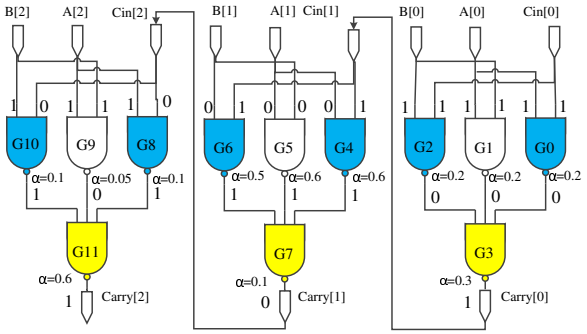


Fig. 1: Carry propagation for 3-bit carry-ripple adder.

TABLE I: NAND gate leakage values (nW) for two sizes (X1, X2) based on input vector control (IVC) from a single threshold 45-nm cell library [7], where min and max leakage states are represented by bold and italicized fonts, respectively.

	NAND-3	
IVC	X1	X2
000	3.32	13.28
001	18.18	72.73
010	4.21	16.84
011	39.49	157.97
100	**1.29**	**5.15**
101	18.78	75.13
110	3.76	15.04
111	*55.8*	*223.22*

	NAND-2	
IVC	X1	X2
00	**3.48**	**13.93**
01	24.8	99.2
10	4.09	16.34
11	*37.21*	*148.83*

assumes nominal SA and IVC, demonstrating that gate SA and IVC play an major role in the guiding sizing decisions.

II. Motivation

We begin by providing a small realistic example demonstrating the importance of considering both SA and IVC in the gate sizing optimization process. Consider the carry propagation of a 3-bit carry-ripple adder, shown in Figure 1. Assume that 2- and 3-input NAND gates have input-dependent leakage power consumption values for two possible sizes, small (X1) and large (X2), shown in Table I. Also assume that the given input vectors ($A = 101$, $B = 101$, and $C_{in} = 1$) are realized throughout the entire duration of the application. Figure 1 shows the input vectors to each gate. Therefore, the overall leakage power of the circuit is 288 nW. For simplicity of the example, ignoring load and slew dependencies, assume that all gates have delay of 10 ps at size X1 and 5 ps at size X2. Finally, assume that at the beginning of the optimization process, all gates are nominally sized to size X1. Therefore, there are eight nominal critical paths (colored), $\{G0, G2\} \rightarrow G3 \rightarrow \{G4, G6\} \rightarrow G7 \rightarrow \{G8, G10\} \rightarrow G11$, with nominal delay 60 ps. Consequently, total leakage energy of the circuit is 1.73×10^{-17} J.

As an example, consider a delay constraint of 55 ps, it is clear that one of gates G3, G7, or G11 should be sized up to X2, as all critical paths pass through these bottleneck gates and decrease the delay of each of these gates will decrease the overall delay. A traditional approach to gate sizing would consider these gates equally. In other words, increasing the size of either would decrease delay and increase switching and leakage power by the same amounts. However, from Table I, we see that the leakage power of a gate, due to transistor stacking, strongly depends on its applied inputs, with up to a 43X difference between the lowest-leakage state (input vector "100": 1.29 nW) and highest-leakage state (input vector "111": 55.8 nW) of a 3-input NAND gate. Furthermore, switching energy of a gate is directly proportional to its activity factor, or the likelihood that the gate will switch. Therefore, because the gates have both different applied input vectors and different activity factors, sizing up each one will have a different effect on overall power and energy consumption, so they should not be weighted equally in the optimization process.

First, consider the case where the duty cycle of the adder is low and therefore leakage energy dominates. We can determine from Table I that increasing the size of gates G3, G7, or G11 will increase leakage power by 9.96 nW, 167.42 nW, or 56.35 nW, respectively, while decreasing the overall delay by 5 ps. Therefore, the optimal decision is to increase the size of gate G3, which will have minimal impact on leakage energy, increasing leakage power to 298 nW and *decreasing* leakage energy to 1.64×10^{-17} J. Increasing the size of G7 would instead increase leakage power to 455 nW, *increasing* leakage energy to 2.50×10^{-17} J. Thus, considering IVC in this example in the optimization algorithm can improve the energy by roughly 60%.

Now, consider the high duty cycle scenario, where switching energy is the dominant factor. Again, for simplicity, assume that all gates consume 10 nJ and 20 nJ of switching energy at nominal activity factor 1.0 for a given application at sizes X1 and X2, respectively. Figure 1 shows the activity factors (α) for each gate. Therefore, overall switching energy consumption at the nominal size is 35.5 nJ. In this case, increasing the size of gate G7 is the optimal decision, since it has the lowest activity factor and consumes less switching energy than when up-sizing either G3 or G11. In fact, this decision results in a switching energy of 36.5 nJ, whereas increasing the size of G11 would result in a switching energy of 41.5 nJ. Therefore, the decision that considers SA performs roughly 14% better.

To present these motivations, we have made a number of assumptions that when relaxed make the optimization much more complex in practice. It is reasonable to assume that additional information (gate switching, input vector state) can be readily obtained by modern CAD tools and/or by implementing a simple gate-level simulator. Such information is beneficial since it enables the simultaneous consideration of low duty cycle and high duty cycle scenarios, as in real use cases at current and pending deep-submicron feature sizes, leakage and switching may both have significant impacts on overall energy. For example, sizing up G7 in the high duty cycle scenario may in reality not be optimal, since its input gates have higher values for α than, the input gates of G3, and thus their switching energies would increase by larger factors. Thus, this IVC depends on how the circuit is sized and its duty-cycle. Therefore, a feedback loop exists between gate sizing and IVC that must be addressed simultaneously during the optimization. The simple example here demonstrates that both IVC and SA are crucial considerations in gate sizing for energy optimization in the presence of delay deadlines.

III. Related Work

We now cover a set of related gate sizing approaches that have considered a variant of SA or IVC. Several approaches exist that address continuous and discrete gate sizing. Common methods to solve the gate sizing problem have been convex optimization [3], Lagrangian Relaxation [1][2][18], and gradient and sensitivity-based optimization [8][19].

Gate sizing methods have also been combined with V_{dd} and V_{th} assignment to minimize power under various gate SA ratios [9][10]. These works, however, have only considered average leakage values when accounting for leakage and have not explored real application activity factors when considering gate switching activity. Leakage minimization using IVC is a popular technique for due to its strong dependency on the input vector state [11]. IVC and gate replacement techniques have also been combined [12] by replacing gates at their worse-case leakage state with equivalent gates with lower leakage power. The technique is further combined with circuit aging in pre-and-post silicon phases. [13][20]. IVC has also been explored in the presence of uncertainty [14].

To the best of our knowledge, we are the first to consider gate sizing in the presence of both SA, IVC, and duty cycle. Prior approaches have at most considered one or two terms accurately [17], and/or do not differentiate between the duty cycle with respect to switching and leakage energy weights, leaving many approaches to be either dynamic or leakage power-centric. For example, the state-of-the-art gate sizing contest considers only nominal leakage power [4]. Our technique minimizes total energy, such that both the switching and leakage energy components are accurately accounted for in accordance to their usage or duty cycle.

IV. Cell Library Energy and Delay

The total energy of a CMOS integrated circuit can be characterized into two main components: 1) dynamic (switching) energy due to charging of input pin/output load capacitance's; and 2) static (leakage) energy, which we model from the dominant sub-threshold leakage and gate leakage currents. Thus, the total energy consumed can be computed as:

$$E_{total} = E_{switch} + E_{leak} \quad (1)$$

$$E_{switch} = \sum_{i}^{N} es(g_i), \qquad E_{leak} = \sum_{i}^{N} el(g_i) \quad (2)$$

es and el represent the switching and leakage energies, respectively, for gate g_i. es is the product between probability that a gate's input pin j will switch, α (SA), and the estimated full-cycle (e_{fc}) power consumed from propagating a signal from input pin j to output pin k. el is the sum of leakage energies consumed at each possible leakage state of a gate, which is also dependent the ratio of the total time spent at each leakage state for both *active* and *standby* (idle) periods. The total time (T) is directly proportional to product of the circuit delay (D) and total cycles, where D represents the critical output-pin arrival time ($rise$ or $fall$) of a primary output gate $ot^{r,f}(g_i)$:

$$D = max(ot^{r,f}(g_i)) \qquad s.t.\ g_i \in G_{out} \quad (3)$$

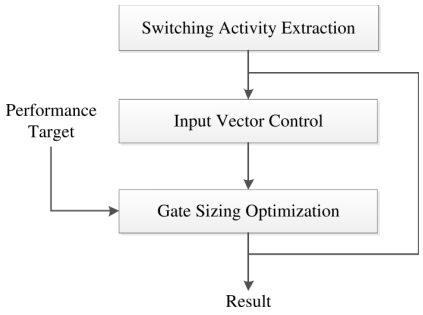


Fig. 2: Gate sizing optimization flow.

G_{out} represents a circuit's set of primary output gates. Therefore, the delay of a circuit can be determined by solving:

$$ot^{r,f}(g_i) = dl^{r,f}(g_i) + max(ot(fin_j^{f,r})) \quad s.t.\ fin_i \in FI_i \quad (4)$$

$fin_j^{f,r}$ is the $fall$, $rise$ arrival time of a fan-in gate j in the set FI_i of gate g_i. Note that the propagation of delay depends on the unateness assumption. For simplicity, we assume all cells are negative unate, thus, rise (r) and fall (f) gate delays are propagated as assumed to the next stage.

We use a cell table library look-up as [4] to model gate $rise$ and $fall$ delay ($dl^{r,f}$) as a function of its input slew (transition time), and driving load. However, we use an alternate 45-nm cell library (Nandgate) [7] to account for switching and input vector dependent leakage power, which are obtained in a similar look-up table fashion, provided per-input pin accurate switching, and input vector state probabilities, which can be obtained using gate-level simulation.

V. Technical Approach

Our gate sizing procedure is composed of three major phases (Figure 2). The first phase extracts gate switching activity factors (SA) for a given circuit by performing event-driven gate simulation from a set of input bit vectors. Figure 3 illustrates an example SA extraction for a carry-look-ahead unit (cla4) from two applications (mpeg2enc/dec). The second step identifies a primary input bit vector that places gates in low leakage states in order to minimize the total energy of the circuit, which accounts for leakage consumption for both active (obtained from SA) and idle periods. IVC techniques range from random simulation to satisfiability (SAT) and model counting-based formulations. The final component is the gate sizing algorithm, where the goal is to minimize total energy consumption under a delay constraint. The approach is iterative; at each iteration, gates are either frozen or unlocked based on their leakage (IVC) and switching (SA) impact, while a search is conducted for the most beneficial current *move*.

Our algorithm is sensitivity-based in nature in terms of determining which move or set of moves to perform. A gate sizing move can have 1 of 3 effects (increase, decrease, have no effect) on 2 parameters (energy and delay), leading to a total of 9 separate possible classes for a move. The algorithm classifies each move to a class and enforces a priority in terms of selecting a move that has higher precedence. There are three precedence levels, where level 1 is the highest priority. Moves

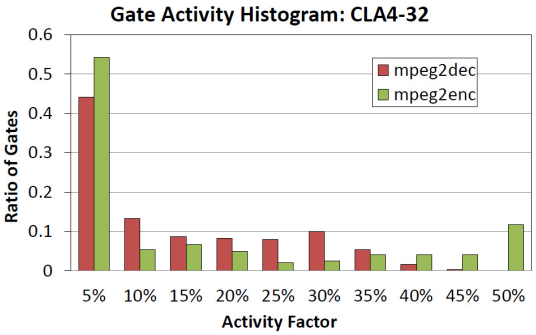
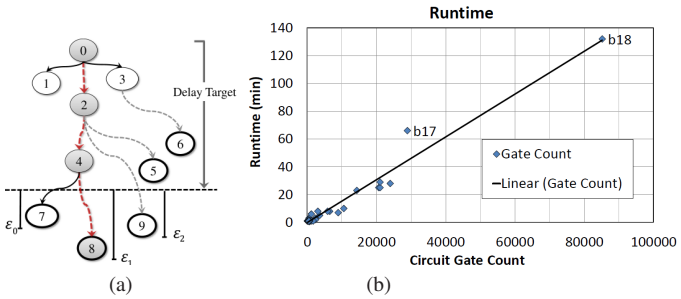


Fig. 3: Gate switching activity for a 32-bit CLA circuit when using real mpeg2enc/dec application input stimulus. Shown are varying distribution of gate activity within a circuit and across two applications.



(a) (b)

Fig. 4: (a) An example of ϵ critical path (ϵ_{path}); the critical path in red; transitive fan-out output nodes in bold outlines; and ϵ_i corresponds to the absolute delay difference w.r.t to the target delay used for estimating delay cost of a move. (b) The linear run-time of the new gate sizing approach.

that improve both parameters are at precedence 1, moves that improve just one parameter and do not affect the other are at precedence 2, and moves that improve one parameter at the expense of degrading another are of precedence 3. Note that moves that degrade both parameters are never selected. Each precedence level has its own objective function for selecting the best move: 1) the product of the respective improvements; 2) the single improvement; and 3) the normalized ratio of improvement and degradation.

The algorithm considers a cut of M gates at a time and restricts one gate to be sized per group visit. Once a size move is committed, the gate is locked and is no longer considered within that phase. The completion of a phase is defined as having locked all gates, or having no more acceptable moves among sizable gates that improve the objective function. The algorithm terminates after the solution converges or if a target delay (D_{target}) can not be met after a number of phases. All gates are unlocked before the start of each new sizing phase.

The algorithm initially freezes the top K energy-critical gates by setting them to their minimal sizes at the beginning of the phase. We note that this initial set potentially restricts some delay critical gates, as improving the delay of these gates may be required in meeting a deadline. To relax this constraint (i.e., if a solution cannot be obtained), K is relaxed through a locking threshold ratio γ, where a new K is computed (e.g., $K = K^{'} \cdot \gamma$), thereby, enabling potentially more delay critical gates to be reduced. It is crucial to identify the top K energy-critical gate, which in turn depends on both SA and IVC; this maximally-constrained gate locking is one of the key innovations of the approach, and prevents being trapped at a local minima by encouraging global circuit optimization.

We utilize an epsilon tree to minimize circuit delay updates (ϵ_{path}), which consists of gates that were on the critical path during the last accurate delay computation (Figure 4a). Since the critical path may change during optimization, we also include the immediate fan-out gates of each critical gate (e.g., nodes 1 and 3), fan-in nodes may be added for greater accuracy as their slews may also impact timing propagation. The figure shows bold-outlined nodes (e.g., 7, 8, 9, 5 and 6) are the primary outputs (G_{out}), and are transitively connected to at least one node belonging to the critical path (e.g., 0, 2, 4, 8). Thus, the delay cost of sizing a gate on the ϵ_{path} can be estimated by the sum of its δ_i respect to the target delay (D_{target}) is used to estimate the delay impact of each move via a delay cost formula, as shown below:

$$D_{cost} = \sum_i^{|G_{out}|} (\delta_i - D_{target})^2 \quad (5)$$

This formulation enables very efficient delay estimation by only considering the delay impact of a small subset of gates at a time. A drawback of this approach, however, is that a potentially new critical path may emerge. This remains to be a major challenge for existing gate sizing techniques that attempt to maintain delay accuracy during optimization [8][18][19]. To address this issue, the frequency of delay updates can be increased by adjusting M and γ to be larger values, as we have done. These parameters can be adjusted, to trade-off accuracy vs run-time. Our used values of M and γ achieved a delay accuracy to be within 5%, while achieving linear run-time scaling with respect to circuit size (Figure 4b).

VI. Experimental Results

We evaluated our gate sizing approach on a set of benchmarks in ISCAS-85/89, ITC-99 suites, as well as integer arithmetic units consisting of adders (carry ripple, carry-look-ahead, Kogge-Stone) and multipliers (array, Dadda). All units were synthesized using a single threshold (HVt) 45-nm open cell library from [7] under the typical cell configuration. An in-house timing/power engine was implemented in C++ and was correlated to an industrial tool, Synopsys PrimeTime, to be within 10^{-3}ps. All results were optimized using identical rules such as ensuring no slew or load violations exists in the final design, as presented in [4]. The only differences in our framework is the choice in cell library, which was done in order to enable accurate IVC computations, as well as the choice of circuit benchmarks. In handling slew and load violations, we adopt an iterative approach as proposed in [19].

The SA of gates and IVC for each circuit were obtained from simulation of random input vectors. However, real application switching activity factors were obtained from mpeg2enc/dec benchmarks from recorded operand values from each unit type, running ARM7TDMI-ISA mpeg2-enc/dec traces [15][16]. The initial simulation parameters set were, K = 25%, M = twice the length of average critical path, γ = 0.2, and were fixed across all benchmarks. The delay target for each circuit was set as the median between the achieved delay when all gates were set to their maximal size, and the

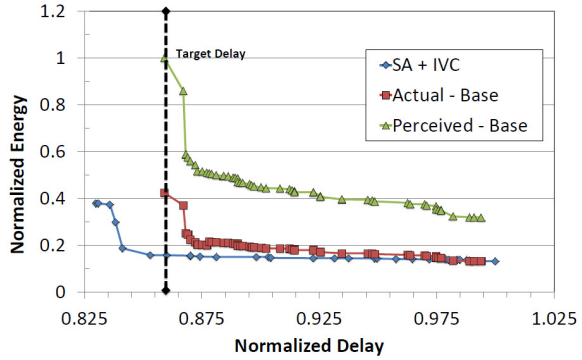


Fig. 5: Energy vs delay plot of c2670. The SA+IVC approach consistently outperforms the $Base$ method.

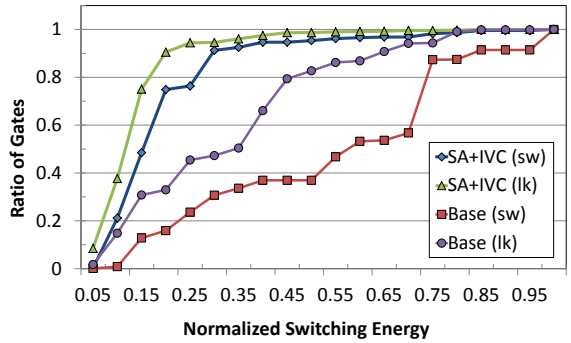


Fig. 6: Cumulative distribution of leakage and switching energy after sizing for gates in c2670. The accurate SA+IVC approach results in a higher percentage of gates at lower energy.

achieved delay when all gates are at their minimal size. Five duty cycle scenarios (D0=10%, D1=20%, D2=33%, D3=50%, and D4=100%) were considered.

We evaluate our sizing algorithm under two gate sizing assumptions: 1) SA+IVC, which considers gate switching activity factors and input vector control in the objective function; and 2) $Base$, where the objective function uses only nominal gate switching (50%) and average gate leakage values for total energy computation. Table II compares the two methods, where Max (%) savings corresponds to the maximum energy improvement achieved over the $Base$ method across the five duty cycle cases (D0 to D4) for each circuit under the same timing constraint. As expected, the maximum improvement observed varies across duty cycles and circuits, motivating the advantage of utilizing accurate power and delay knowledge.

Table IV provides overall energy improvements across the benchmarks suite. The results generated by the new approach achieved a max energy improvement of 62% for circuit c2670 and 29% average overall for the same delay. Figure 5 provides a normalized energy and delay plot for c2670, which illustrates the advantage of using more accurate power and delay information. A delay of 0.87 shows that the $Perceived$ (green) energy deviates from the trend of the $Actual$ (red) energy plot. In performing move-trace analysis, we noted that $Base$ method caused the algorithm to over-size a few selected critical paths and encountered a timing wall much earlier, where as SA+IVC was able to efficiently trade-off delay for an additional 0.05 delay units, as shown.

TABLE IV: Overall energy savings with respect to benchmark suite.

Benchmark Suite	Max Tot (%)	Avg. Tot (%)	Avg. Sw (%)	Avg. Lk (%)
ISCAS-85	62.64	29.70	30.58	26.74
ISCAS-89	53.20	28.33	29.99	26.96
ITC-99	58.83	29.23	30.90	24.15
Arith	57.19	30.48	33.23	33.15

Figure 6 shows a cumulative distribution of gate switching and leakage energies of the max improved result for SA + IVC over $Base$ for circuit c2670. Our approach shows that accurate knowledge enabled the algorithm to efficiently guide the circuit to a lower energy state, as shown with higher percentage of gates falling under lower energy profiles for both leakage and switching energy. This is important to note since due to the difficulty of comparing gate sizing algorithms, many existing algorithms are sensitivity-based in nature, thus, the ability to guide an algorithm to determine more promising "moves" greatly impacts the optimization procedure.

Table III presents results comparing the minimal configuration found by SA + IVC and the perceived minimal configuration obtained by $Base$. The minimum energy configuration determined was cla432 and dad8, optimized under the same timing constraints determined by the the multiplier. For these configurations, our approach shows additional savings in both leakage and switching categories where the majority of the savings for both cases (15% mpeg2enc, 25% mpeg2dec) were achieved by the multiplier circuit.

VII. Conclusion

We present a new gate sizing approach that includes the switching activity (SA) and input vector control (IVC) to minimize overall energy. The new objective function has several ramifications on the optimization procedure, including the need for reiteration between gate sizing and input vector selection and freezing and unlocking of high-power gates. On a comprehensive set of benchmarks, from ISCAS-85/89, ITC-99, and arithmetic units, synthesized using 45 nm technology, we reduce average actual energy consumption by 30%. The approach is generic in the sense that thermal impacts and multi-V_{th} can be easily addressed using the new optimization procedure.

References

[1] H. Shiyan et al., "Gate Sizing For Cell Library-Based Designs," *DAC* pp. 847–852, 2007.

[2] M. M. Ozdal et al., "Gate sizing and device technology selection algorithms for high-performance industrial designs," *ICCAD*, pp. 724–731, 2011.

[3] S. Joshi, "An Efficient Method for Large-Scale Gate Sizing," *TCSI*, pp. 2760–2773, 2008.

[4] M. M. Ozdal et al., "The ISPD -2012 Discrete Cell Sizing Contest and Benchmark Suite," *In Proceedings of ISPD* pp. 161–164, 2012

[5] W. N. Li, "Strongly NP-hard discrete gate-sizing problems," *ICCD*, pp.1045–1051, 1993.

[6] A. Agarwal et al., "Statistical timing based optimization using gate sizing," *DAC*, pp. 400–405, 2005.

[7] Nangate FreePDK45-nm Library, *http://www.si2.org/*, 2011.

[8] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *VLSI*, pp. 465–472, 1997.

TABLE II: Energy improvements when considering gate SA and IVC during the gate sizing procedure over the $Base$ method. The obtained switching and leakage energies are presented for the SA+IVC. The maximum energy deltas (Δ %), corresponds to the max difference in energy profile "perceived" by the $Base$ method during optimization.

		Max Energy Improvement			$SA + IVC$					
Circuit	No. Gates	Total (%)	Sw (%)	Lk (%)	Sw (μJ)	Max. (Δ%)	Lk (μJ)	Max. (Δ%)	Duty Cycle	Delay (ns)
c880	383	8.14	8.16	8.05	125.11	31.22	15.75	3.01	D2	0.73
c1355	554	14.35	15.01	13.26	264.13	31.12	164.79	15.19	D1	0.74
c1908	932	13.41	13.72	9.41	405.7	29.57	32.06	6.75	D4	1.14
c7552	3568	43.65	44.02	41.32	1685	30.27	284.6	3.21	D2	1.20
c5315	2330	58.72	59.1	54.57	855.6	32.93	87.58	3.19	D4	1.34
c432	168	30.15	35.24	22.35	68.54	33.15	53.55	23.67	D1	0.62
c2670	1202	62.64	62.74	60.78	407.6	31.91	24.62	1.39	D4	0.85
c3540	1703	28.84	29.66	24.14	799.1	31.09	152.2	2.47	D2	0.79
s1488	698	46.38	46.49	44.81	182.6	35.54	14.09	12.05	D2	0.42
s1494	692	35.5	36.08	33.84	283.9	36.36	103.3	12.55	D1	0.42
s15850	10547	31.34	30.76	34.48	3803	33.69	662.1	4.98	D3	2.22
s838	473	32.49	38.89	27.17	139.3	31.76	199.8	10.84	D1	1.64
s5378	3054	16.76	30.08	15.06	826.6	36.2	7876	26.34	D0	0.68
s9234	5897	50.19	50.52	47.23	2041	32.51	242.5	4.43	D3	1.50
s38417	23963	53.2	53.85	46.17	6661	30.65	714.2	8.58	D3	1.29
s35932	21035	22.27	22.31	21.25	8317	29.28	332.1	37.26	D3	0.80
s38584	18161	29.01	31.52	28.75	5820	32.24	59720	4.77	D0	1.52
b10	204	44.69	45.06	32.43	40.56	34.45	1.55	54.83	D2	0.36
b11	633	35.53	35.79	31.74	250.9	32.37	18.77	61.52	D2	1.10
b12	1183	22.91	27.89	3.31	314.0	37.03	107.3	60.54	D1	0.61
b13	375	22.05	22.78	15.01	161.23	31.85	18.4	55.33	D1	0.33
b14	6498	28.42	28.65	26.21	3024	33.22	320.7	54.94	D2	1.31
b15	8920	27.62	27.7	26.75	1666	38.99	166.1	54.24	D3	1.58
b17	28911	21.25	25.29	20.79	8.53	38.76	80.58	55.99	D3	1.68
b18	85188	7.02	8.67	5.96	44.54	37.84	71.4	54.72	D1	2.10
b20	14322	27.85	28.57	24.07	3.66	33.2	0.74	55.8	D2	2.08
cra32	225	38.32	45.14	37.76	6.72	44.90	92.22	32.09	D0	2.08
cla432	305	22.25	25.02	12.18	10.18	42.87	3.28	13.62	D2	0.35
ks32	611	24.2	23.18	24.63	17.64	44.76	40.83	14.6	D0	0.30
arr8	512	35.96	36.3	23.18	178.5	34.69	22.84	21.29	D1	0.81
dad8	542	30.35	30.99	36.33	101.3	35.83	9.12	11.98	D2	0.62

TABLE III: Energy improvement of ($SA + IVC$) over $Base$ using extracted gate switching activity and input vector control from mpeg2enc/dec applications assuming a (D2) "33%" duty cycle. The units represent an single-adder (32b) and multiplier (8b) configuration of an ARM7TDMI core [16].

	Energy Improvement			cla432				dad8			
Application	Total (%)	Sw (%)	Lk (%)	Sw (μJ)	Sw Imp. (%)	Lk (μJ)	Lk Imp. (%)	Sw (μJ)	Sw Imp. (%)	Lk (μJ)	Lk Imp. (%)
mpeg2enc	15.31	17.15	8.55	742.7	4.73	628.7	1.09	2267	20.61	498.6	13.92
mpeg2dec	25.10	29.89	6.21	11.02	6.42	24.30	1.20	101.4	30.99	9.24	21.58

[9] A. Srivastava et al., "Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment," *DAC*, pp. 783–787, 2004.

[10] H. Yu-Hui et al., "Switching-activity driven gate sizing and Vth assignment for low power design," *ASPDAC*, pp. 24–27, 2006.

[11] A. Abdollahi et al., "Leakage current reduction in CMOS VLSI circuits by input vector control," *VLSI*, pp. 140-154, 2004.

[12] L. Yuan et al., "A combined gate replacement and input vector control approach for leakage current reduction," *VLSI*, pp. 173–182, 2006.

[13] Y. Wang et al., "Leakage power and circuit aging cooptimization by gate replacement techniques," *VLSI*, pp. 615–628, 2011.

[14] Y. Alkabani et al., "Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability," *DAC*, pp. 606–609, 2008.

[15] C. Lee et al., "MEDIABENCH: a tool for evaluating and synthesizing multimedia and communicatons systems," *MICRO*, pp. 330–335, 1997.

[16] T. Mudge et al., "The SimpleScalar-Arm power modeling project," *http://eecs.umich.edu/panalyzer/*.

[17] C. Tsui et al., "Minimizing the dynamic and subthreshold leakage power consumption using least leakage vector assisted technology mapping," *VLSI Journal*, pp. 76–86, 2008

[18] L. Li et al., "An efficient algorithm for library-based cell-type selection in high-performance low-power designs," *ICCAD*, pp. 226–232, 2012

[19] J. Hu et al., "Sensitivity-guided metaheuristics for accurate discrete gate sizing," *ICCAD*, pp. 233–239, 2012

[20] S. Wei et al., "Aging-based Leakage Energy Reduction in FPGAs," *FPL*, pp. 277–277, 2013