A Practical Transistor-Level Dual Threshold Voltage Assignment Methodology

Puneet Gupta* Andrew B. Kahng*‡† Puneet Sharma†
puneet@ucsd.edu abk@ucsd.edu sharma@ucsd.edu

UCSD †ECE and ‡CSE Departments, La Jolla, CA
*Blaze DFM, Inc., Sunnyvale, CA

Abstract

Leakage power has become one of the most critical design concerns for the system-level chip designer. Multi-threshold techniques have been used to reduce runtime leakage power without sacrificing performance. In this paper, we present an effective and scalable transistor-level V_{th} assignment approach and show leakage reduction over standard cell-level V_{th} assignment. The main disadvantage of transistor-level V_{th} assignment is increased cell library size and characterization effort. In comparison to previous approaches, our approach yields better solution quality, requires smaller cell library, is more accurate in considering the impact of V_{th} assignment on propagation delay, slew (transition delay) and capacitance, and is significantly faster.

1. Introduction

Leakage power has become one of the most critical design concerns. While lowered supplies (and consequently lowered V_{th}) and aggressive clock gating can achieve dynamic power reduction, these techniques increase leakage power and therefore cause its share of total power to increase. Leakage has become a significant contributor to total power and its contribution is projected to increase from 18% at 130*nm* to 54% at the 65*nm* node [12]. Leakage is composed of three major components: (1) subthreshold leakage, (2) gate leakage, and (3) reverse biased drain substrate and source-substrate junction band-to-band-tunneling leakage [1]. Subthreshold leakage is the dominant contributor to total leakage at 130*nm* and is forecast to remain so in the future [1]. In this work we target subthreshold leakage reduction.

Leakage reduction methodologies can be divided into two classes depending on whether they reduce *standby* leakage or *runtime* leakage. Standby techniques reduce leakage of devices that are known not to be in operation, while runtime techniques reduce leakage of active devices. Several techniques have been proposed for standby leakage reduction. *Body biasing* or *VTMOS* based approaches [6] dynamically adjust the device V_{th} by biasing the body terminal [1]. *Muti-threshold CMOS (MTCMOS)* techniques [10, 7, 11, 15] use high-V_{th} CMOS (or NMOS or PMOS) to disconnect Vdd or Vss or both to logic circuit implemented using low V_{th} devices in standby mode. *Source biasing*, where a positive bias is applied in standby state to source terminals of off devices, was proposed in [5]. Other techniques such as use of transistor stacks [23] and input-vector control [4] have also been proposed.

Fewer techniques have been proposed for runtime leakage reduction. Recently, [3] proposed a runtime leakage reduction approach that increases the gate lengths of transistors not on critical paths. However, the only mainstream approach to runtime leakage reduction is the multi-V_{th} manufacturing process. In this approach, cells on non-critical timing paths are manufactured to have higher threshold voltages, while cells on critical timing paths have lower threshold voltages. [20] presented a heuristic algorithm for selection and assignment of optimal high V_{th} to cells on non-critical paths. The multi-V_{th} approach has also been combined with several other power reduction techniques [9, 22, 16]. The primary drawback of this technique has been the rise of process costs due to additional manufacturing steps and masks required for each extra V_{th}. However, the substantial leakage savings provided by this approach have outweighed cost increases, and dual-V_{th} processes are now standard and used together with other power reduction techniques.

Today's standard cell library based flows use cell-level V_{th} assignment (CLVA) techniques in which all PMOS and NMOS transistors in a cell are assigned the same threshold voltage. When two threshold voltages are available, the library required for CLVA is only twice the size of a single-V_{th} library since there are two variants of each cell. In this paper, we investigate the benefits and costs associated with transistor-level V_{th} assignment (TLVA). Since different transistors control different timing arcs, TLVA can modify the delays of individual timing arcs unlike CLVA. Asymmetry in timing criticality of different timing arcs of a cell instance in a circuit, as well as in rise and fall transitions, can be utilized by TLVA to yield significant leakage savings. Unfortunately, TLVA requires a larger number of cell variants which translates to a larger library and higher library characterization effort.

Approaches for TLVA have previously been proposed in [17, 21, 8]. [17] proposed a sensitivity-based *upsizing* (i.e., begins with nominal V_{th} assigned to all transistors and assigns low V_{th} iteratively to timing-critical transistors) algorithm which combined transistor sizing and V_{th} assignment. [8] later proposed an enumeration based technique with better quality of results and reduced runtime. However, the enumeration based approach proposed in [8] quickly grows in space and runtime requirements as the input size increases. The approach neglects the effect of V_{th} assignment on capacitance and is inclined to assign low V_{th} more to the transistors near primary inputs. A sensitivity-based *downsizing* (i.e., begins with low V_{th} assigned to all transistors and assigns nominal V_{th} to

[1]Body biasing has also been proposed to reduce leakage of active devices [14].

non-critical transistors) approach was proposed in [21]. However, the technique of [21] has the following shortcomings.

- *Impractical*. The presented technique does not use precharacterized cell delay values but relies on analytical expressions to estimate delays of different timing arcs. Despite the high library characterization costs, analytical delay models are never used in practice due to their unacceptably large inaccuracy. The equations used in [21] assume that each input is controlled by one NMOS and one PMOS transistor, and that there are no transistors that do not control an input. These assumptions fail for all but the simplest cells. The impact of V_{th} on capacitance is ignored and the used transistor delay models and timing analysis ignore delay dependence on slew (transition delay). Also, the impact of switching time of NMOS (PMOS) transistors on *rise* (*fall*) transition is ignored.
- *Large library size*. A standard cell library is required for timing closure in successive design steps. The proposed approach allows freedom to assign V_{th} to individual transistors in a cell separately and hence may require up to 2^T variants of any given cell, where T is the number of transistors in the cell. A cell library with such a large number of cell variants may not be practical.
- *Large runtime*. The proposed algorithm (called PS[2] in [21]) is extremely time-consuming since all sensitivities are recomputed and full static timing analysis run after each downsizing move.

We present an effective, accurate and scalable transistor-level V_{th} assignment technique which is sensitivity-based and performs downsizing. In our studies, we have found downsizing to be significantly more effective for leakage reduction than upsizing irrespective of the delay constraints. An intuitive rationale is that upsizing approaches have dual objectives of delay and leakage while performing the upsizing moves. Downsizing approaches, on the other hand, are bound to meet timing constraints since they only perform downsizing moves if no timing violations are caused, and they hence have the sole objective of leakage minimization[3]. We apply our dual-Vth leakage reduction approach first at cell level and then at transistor level to reduce the runtime of TLVA. The following subsections introduce our ideas in detail.

1.1 Delay Asymmetries in Timing Arcs within Cell

We use the term *timing arc* to indicate an intra-cell path from an input transition to a resulting rise (or fall) output[4] transition. For an n-input gate there are $2n$ timing arcs[5]. Due to different parasitics as well as PMOS/NMOS asymmetries, these timing arcs can have different delay values associated with them. For instance, Table 1 shows the delay values for the same input slew, load capacitance pair for different timing arcs of a NAND2X2 cell from the *Artisan TSMC* 130*nm* library.

[2] [21] proposed two other algorithms, BT and PB, which are inferior in solution quality but significantly faster. We compare with PS due to the similarity of our algorithm with PS and demonstrate the runtime savings due to our engineering optimizations such as the use of incremental timing analysis.

[3] An upsizing approach, however, may be faster when loose delay constraints are to be met since very few transistors have to be upsized. However, delay is almost always the primary design goal and loose delay constraints are rare.

[4] We assume all cells have one output.

[5] There may be four timing arcs corresponding to non-unate inputs (e.g., select input of MUX).

Timing Arc	Propagation Delay (ps)	Transition Delay (ps)
$A \rightarrow Y \uparrow$	99.05	104.31
$A \rightarrow Y \downarrow$	73.07	79.12
$B \rightarrow Y \uparrow$	107.20	112.98
$B \rightarrow Y \downarrow$	70.65	76.37

Table 1: Asymmetry in delays of various timing arcs within a NAND2X2 cell.

1.2 Use of Asymmetry

Pin swapping is a common post-synthesis timing optimization step to make use of the asymmetry in delays of different input pins. To make use of asymmetry in rise-fall delays, techniques such as P/N ratio perturbations have been previously proposed to decrease circuit delay [2]. We propose to exploit these asymmetries using TLVA to "recover" leakage from non-critical timing arcs within a cell. The other technique for runtime leakage reduction, gate-length biasing [3], may also be developed for leakage reduction using the available asymmetry. We use TLVA since it is more mature and appears to yields a more favorable tradeoff between delay penalty and leakage reduction.

The remainder of this paper is organized as follows. In Section 2, we describe the proposed TLVA methodology. Section 3 describes our experiments and presents the results. Finally, Section 4 concludes and gives a brief description of ongoing and future research.

2. Methodology

In this section we describe our methodology for transistor-level V_{th} assignment. Our flow to exploit non-criticality (or presence of slack) of timing arcs involves the following steps:

1. Cell-variant creation
2. Library generation
3. Optimization for leakage

2.1 Cell-Variant Creation

For each cell, our library contains variants corresponding to all subsets of the set of timing arcs. A gate with n inputs has $2n$ timing arcs and therefore 2^{2n} variants (including the original cell). Given a set of critical timing arcs, our goal is to assign nominal V_{th} to some transistors in the cell and low V_{th} to the remaining transistors to meet two criteria: (1) critical timing arcs have a delay penalty of under 1% with respect to the cell in which all transistors are assigned low V_{th}, and (2) cell leakage power is minimized. TLVA in a cell, given a set of critical timing arcs, can be done for simple cells by analyzing the cell topology. However, we automate the process of V_{th} assignment to transistors in a cell in the following manner. For each cell, we enumerate all configurations in which low V_{th} is assigned to some transistors and nominal V_{th} to the others. For each configuration we find the delay and leakage under a canonical load of an inverter (INVX1) using SPICE simulations. Then, for each possible subset of timing arcs that can be simultaneously critical, one V_{th} assignment configuration is chosen based on the two criteria above. We note that approaches proposed in [8, 21] allow freedom to assign V_{th} to individual transistors in a cell separately and hence may require up to 2^T cell variants of each cell, where T is the number of transistors in the cell. Since even one- or two- input cells can have many transistors, the number of variants required by previous approaches can be much larger than the number of variants created with our approach. Figure 1 shows V_{th} assignment to transistors of the simplest NAND cell (NAND2X1) when only the rise and fall timing arcs from input A to the output are critical.

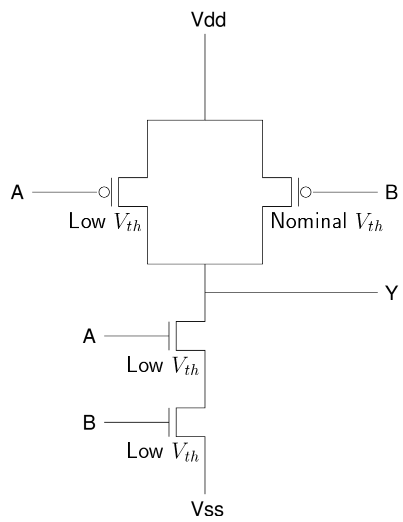

Figure 1: V_{th} assignment in NAND2X1 when only the rise and fall timing arcs from input A to the output are critical.

2.2 Library Generation

We generate a restricted library comprising of variants of the 25 most frequently used cells in our test cases. To identify the most frequently used cells, we synthesize our test cases with the complete *TSMC* 130*nm* library and pick the 25 most frequently used cells. Variants for each cell are created as described in the previous subsection by threshold voltage assignment to transistors in the cell. We use *Cadence SignalStorm* and *Synopsys HSPICE* for library characterization (delays and power). We use *TSMC* 130*nm* SPICE netlists and *IBM* 130*nm* SPICE device models for library characterization. We do not assume or optimize the nominal and low threshold voltages but use the voltages specified with the *IBM* 130*nm* SPICE device models. Our implementation does not require all 2^{2n} variants to be present in the library; any variants that cannot be manufactured due to process restrictions may be omitted from the library.

2.3 Optimization for Leakage

Our sensitivity-based downsizing approach begins with a circuit in which all transistors are assigned low V_{th}. [8] concludes that transistor sizing should be performed separately and before V_{th} assignment. Also, several works [19, 21] perform transistor sizing and V_{th} assignment separately. We use *Synopsys Design Compiler v2003.06-SP1* for transistor sizing prior to V_{th} assignment. Since delay is almost always the primary design goal we use the smallest-delay sizing solution for V_{th} assignment.

Timing Analyzer Implementation

A timing analyzer is an essential component of any delay-aware power optimization approach; it is used to compute delay sensitivity to V_{th} assignment. For an accurate yet scalable implementation, we use three types of timers that vary in speed and accuracy.

- *Standard static timing analysis (SSTA).* Slews and actual arrival times (AATs) are propagated forward after a topological ordering of the circuit. Required arrival times (RATs) are back-propagated and slacks are then computed. Slew, delay and slack values of our timer matches exactly with *Synopsys PrimeTime vU-2003.03-SP2*, and our timer can handle unate and non-unate cells[6].
- *Exact incremental STA (EISTA).* We begin with the fan-in nodes of the node that has been modified. From all these nodes, slews and AATs are propagated in the forward direction until the values stop changing. RATs are propagated in the backward direction from only those nodes for which the slew, AAT or RAT has changed. Slews, delays and slacks match exactly with SSTA.
- *Constrained incremental STA (CISTA).* Sensitivity computation involves temporary modifications to a cell to find changes in its slack and leakage. To make this step faster, we restrict the incremental timing calculation to only one stage before and after the gate being modified. The next stage is affected by drive strength change and the previous stage is affected by pin capacitance change of the modified gate. The ripple effect on other stages farther away from the gate (primarily due to slew changes[7]) is neglected since high accuracy is not critical for sensitivity computation.

We use the phrase *downsizing a timing arc* to mean substitution of a cell instance with a variant that has V_{th} assignment such that the timing arc is slowed down. In our terminology, s_p^i represents the slack on i^{th} timing arc of cell instance p and $s_p'^i$ represents the slack on the arc after it is downsized. ℓ_p and ℓ'_p indicate the initial and final leakages of cell instance p before and after downsizing respectively. P_p^i represents the sensitivity associated with downsizing timing arc i on cell instance p and is defined as:

$$P_p^i = \frac{\ell_p - \ell'_p}{s_p^i - s_p'^i}$$

The pseudocode of our leakage optimization implementation is given in Figure 2. The algorithm begins with SSTA and initializes slack values s_p^i for all timing arcs of all cells in Line 1. Sensitivities P_p^i are computed for all cell instances p and all of their timing arcs i, and put into the set L in Lines 3-5. In Lines 7-8, we select the highest sensitivity $P_{p^*}^i$ corresponding to the i^{th} timing arc of cell instance p^* and remove it from the set L. If the highest sensitivity is negative, we exit the loop in Lines 9-10. In Line 11, the function $SaveState$ saves the V$_{th}$ assignment of all transistors in the circuit and delay, slew and slack values for all timing arcs. The timing arc i of cell instance p^* is downsized and EISTA is run from p^* to update the delay, slew and slack values in Lines 12-13. In Line 14, we check for a timing violation (negative slack on any timing arc) due to downsizing and if timing is violated we restore the state in Line 15. If, however, there is no timing violation then this move is accepted and sensitivities of node p^*, its fan-in nodes, and its fan-out nodes are updated in Lines 17-21. The loop continues until sensitivities become negative or L becomes empty.

Function $ComputeSensitivity(q,i)$ temporarily downsizes the i^{th} timing arc of cell instance q and finds its slack using CISTA. Since high accuracy is not critical for sensitivity computation we choose to use CISTA which is faster but less accurate than EISTA. Table 2 shows a comparison of leakage and runtime when EISTA and CISTA are used for sensitivity computation.

[6]Delay values from our timing analyzer match with PrimeTime only under our restricted use model. Our timing analyzer does not support several important features such as interconnect delay, hold

```
Algorithm V_th Assignment()

Run SSTA to initialize s_p^i  ∀cell instances, p, ∀timing arcs of p, i
L ← {}
forall cell instances, p, and timing arcs of p, i
    P_p^i ← ComputeSensitivity(p, i)
    L ← L ∪ P_p^i
do
    P_{p*}^i ← max(L)
    L ← L − {P_{p*}^i}
    if(P_{p*}^i ≤ 0)
        exit
    SaveState()
    Downsize timing arc i of cell instance p*
    EISTA(p*)
    if(TimingViolated())
        RestoreState()
    else
        N ← p* ∪ fan-in and fan-out nodes of p*
        forall q ∈ N, and timing arcs of q, j
            if(P_q^j ∈ L)
                P_q^j ← ComputeSensitivity(q, j)
                Update P_q^j in L
while(|L| > 0)

function ComputeSensitivity(q, i)

old_slack ← Slack of timing arc i of cell instance q
old_leakage ← Leakage of cell instance q
SaveState()
Downsize timing arc j of cell instance p
CISTA(j)
new_slack ← Slack of timing arc i of cell instance q
new_leakage ← Leakage of cell instance q
RestoreState()
return (old_leakage − new_leakage)/(old_slack − new_slack)
```

Figure 2: Algorithm for transistor-level V_{th} assignment for leakage optimization.

3. Experiments and Results

In this section we describe the experimental flows for validation of our TLVA approach, and then present experimental results. We use the following ISCAS'85 combinational and ISCAS'89 sequential[8] benchmarks: *c5315* (1442 cells), *c6288* (4289 cells), *c7552* (1902 cells), *s9234* (1040 cells), *s13207* (2688 cells), and *s38417* (7826 cells). The test cases are synthesized with the *Artisan TSMC* 130*nm library* using *Synopsys Design Compiler v2003.06-SP1* with low-V_{th} cells only. To limit library characterization runtime, we restrict the library to variants of the following 25 most frequently used cells: CLKINVX1, INVX12, INVX1, INVX3, INVX4, INVX8, INVXL, MXI2X1, MXI2X4, NAND2BX4, NAND2X1, NAND2X2, NAND2X4, NAND2X6, NAND2X8, NAND2XL, NOR2X1, NOR2X2, NOR2X4, NOR2X6, NOR2X8, OAI21X4, XNOR2X1, XNOR2X4, and XOR2X4. The delay constraint is kept very tight so that the post-synthesis delay is very close to minimum achievable delay. *STMicroelectronics* 130*nm* device models are used with two (low and nominal) V_{th} values each for PMOS and NMOS transistors (PMOS: -0.88V and -0.13V; NMOS: 0.11V and 0.15V). We use *Cadence SignalStorm v4.1* (with *Synopsys HSPICE vU-2003.09*) for delay and power characterization of cell variants. *Synopsys Design Compiler v2003.06-SP1* is used to measure circuit delay, dynamic power and leakage power. We assume an activity factor of 0.02 for dynamic power calculation in all our experiments.

Circuit	Leakage (mW)		CPU (s)	
	EISTA	CISTA	EISTA	CISTA
c5315	0.3010	0.2984	14.71	8.90
c6288	1.3493	1.3739	455.12	246.73
c7552	0.5022	0.4977	44.14	24.50
s9234	0.0847	0.0848	2.42	1.46
s13207	0.1630	0.1630	15.08	9.35
s38417	0.5565	0.5562	332.75	162.70

Table 2: Comparison of leakage and runtime when EISTA and CISTA are used for sensitivity computation.

3.1 Power Reduction

Table 3 shows the reduction in leakage, dynamic and total power achieved by our *sensitivity-based downsizing* (SBD) algorithm for TLVA. $62\% - 89\%$ reduction in leakage and $23\% - 63\%$ reduction in total power is achieved in comparison to when all transistors in the circuit are assigned low V_{th}. Dynamic power decreases because of less gate capacitance on nominal V_{th} assignment. We allow no delay penalty, i.e., circuit delay (as reported by *Synopsys PrimeTime*) does not increase after SBD. We observe larger leakage reductions in sequential circuits; this is because circuit delay is determined by the slowest pipeline stage and the percentage of non-critical paths is typically higher in a sequential circuit.

3.2 Comparison with Previous Work

We compare the quality of results and runtime of SBD with the latest previous TLVA approach [8][9]. Our implementation of the SATVA algorithm proposed in [8] sets the pruning factor to allow up to 50 tuples at gate outputs[10]. Reduced pruning improves solution quality but increases runtime and memory requirements. We run experiments on a dual Xeon 1.4GHz computer with 2GB RAM. We use the same library (as described in Subsection 2.2) for both approaches. Table 4 presents the power, runtime and actual delay (as reported by *Synopsys PrimeTime*) after performing SATVA and SBD. Delay penalty is set to 0%, 5% and 10%.

SBD consistently performs better than SATVA and is significantly faster. Since SATVA ignores the impact of V_{th} assignment on capacitance, it generates circuits which do not satisfy tight delay constraints. Runtime of SBD increases with circuit size and when there is less slack on nodes. This is because CLVA assigns low V_{th} to a smaller number of cells, and TLVA must then optimize a larger number of cells. Since SATVA is an enumeration based technique, its runtime and memory requirements increase with the presence of gates having three or more inputs.

3.3 CLVA vs. TLVA

Power reduction achieved using TLVA with respect to CLVA is presented in Table 5. The algorithm is used for both CLVA and TLVA; however, for CLVA the cell library is constrained to have only the cells in which all transistors have the same V_{th}.

For sequential circuits and when delay penalties are large, we find insignificant additional leakage savings for TLVA. The primary

time checks, false paths, multiple clocks, 3-pin SDFs, etc.

[7]There may be some impact due to coupling induced delay also, as the arrival time windows can change; we ignore this effect.

[8]To handle sequential test cases, we convert them to combinational circuits by treating all flip-flops as primary inputs and primary outputs.

[9]Since [21] cannot be extended to use the cells in our library, we do not present a comparison with it.

[10]Memory constraints prevent us from going over 50 tuples.

Circuit	Delay (ns)	Power (mW)								
		Low V_{th}			TLVA			Reduction		
		Leakage	Dynamic	Total	Leakage	Dynamic	Total	Leakage	Dynamic	Total
c5315	0.556	1.4413	1.5345	2.9758	0.2984	1.4199	1.7183	79.30%	7.48%	42.26%
c6288	2.118	3.5994	8.0316	11.631	1.3739	7.5714	8.9453	61.83%	5.73%	23.09%
c7552	0.485	1.8328	2.0813	3.9141	0.4977	1.9144	2.4121	72.84%	8.02%	38.37%
s9234	0.437	0.7074	0.3907	1.0981	0.0848	0.3685	0.4533	88.01%	5.68%	58.72%
s13207	0.904	1.3934	0.6296	2.0230	0.1630	0.5923	0.7553	88.30%	5.92%	62.66%
s38417	0.692	4.9381	4.2069	9.1450	0.5562	3.8305	4.3867	88.74%	8.95%	52.03%

Table 3: Power reduction from SBD with respect to when all transistors are assigned low V_{th}. Delay penalty is set to 0% (i.e., circuit delay is not allowed to increase for TLVA).

Circuit	Delay Constraint (ns)	Actual Delay (ns)		Leakage (mW)		CPU (s)	
		SATVA	SBD	SATVA	SBD	SATVA	SBD
c5315	0.556	0.570	0.556	0.4379	0.2984	254.79	8.90
	0.584	0.586	0.584	0.3264	0.2010	242.49	6.45
	0.612	0.612	0.612	0.1903	0.1673	269.32	4.30
c6288	2.118	2.203	2.118	1.8676	1.3739	572.36	246.73
	2.224	2.235	2.224	1.4948	0.6824	577.41	134.51
	2.330	2.329	2.329	0.6492	0.4215	453.83	56.55
c7552	0.485	0.501	0.485	0.7673	0.4977	262.83	24.50
	0.509	0.511	0.509	0.5753	0.2842	268.06	13.83
	0.533	0.533	0.533	0.2491	0.2097	202.55	7.38
s9234	0.437	0.444	0.434	0.9681	0.8480	72.02	1.46
	0.458	0.458	0.456	0.8949	0.8308	73.17	1.40
	0.480	0.479	0.479	0.8235	0.7881	72.96	1.26
s13207	0.904	0.907	0.903	0.1812	0.1630	57.23	9.35
	0.949	0.942	0.949	0.1571	0.1533	55.81	8.38
	0.994	0.991	0.994	0.1549	0.1530	59.80	7.47
s38417	0.692	0.702	0.692	0.6089	0.5562	546.82	162.70
	0.727	0.723	0.727	0.5668	0.5471	522.54	150.42
	0.762	0.760	0.758	0.5527	0.5456	510.31	156.88

Table 4: Comparison of SATVA[8] and our transistor-level V_{th} assignment approach (SBD). Delay penalty is set to 0%, 5% and 10% for each of the test cases.

Circuit	Delay (ns)	Power (mW)									CPU (s)	
		CLVA			TLVA			Reduction			CLVA	TLVA
		Leakage	Dynamic	Total	Leakage	Dynamic	Total	Leakage	Dynamic	Total		
c5315	0.556	0.3772	1.4298	1.8070	0.2984	1.4199	1.7183	20.90%	0.69%	4.91%	3.93	8.90
	0.584	0.2389	1.4120	1.6509	0.2010	1.4075	1.6085	15.85%	0.32%	2.57%	3.83	6.45
	0.612	0.1750	1.4023	1.5773	0.1673	1.4011	1.5684	4.40%	0.09%	0.56%	3.61	4.30
c6288	2.118	1.8696	7.7094	9.5790	1.3739	7.5714	8.9453	26.51%	1.79%	6.62%	67.92	246.73
	2.224	0.8557	7.4242	8.2799	0.6825	7.3659	8.0484	20.24%	0.79%	2.80%	54.93	134.51
	2.330	0.4422	7.2779	7.7201	0.4215	7.2672	7.6887	4.67%	0.15%	0.41%	45.61	56.55
c7552	0.485	0.6834	1.9342	2.6176	0.4977	1.9144	2.4121	27.18%	1.02%	7.85%	7.94	24.50
	0.509	0.3457	1.8994	2.2451	0.2842	1.8930	2.1772	17.78%	0.34%	3.02%	7.00	13.83
	0.533	0.2157	1.8864	2.1021	0.2097	1.8854	2.0951	2.78%	0.05%	0.33%	6.50	7.38
s9234	0.437	0.0984	0.3697	0.4681	0.0848	0.3685	0.4533	13.81%	0.34%	3.17%	1.23	1.46
	0.458	0.0873	0.3676	0.4549	0.0831	0.3675	0.4506	4.85%	0.02%	0.95%	1.26	1.40
	0.480	0.0788	0.3672	0.4460	0.0788	0.3672	0.4460	0.05%	0.00%	0.01%	1.25	1.26
s13207	0.904	0.1741	0.5929	0.7670	0.1630	0.5923	0.7553	6.37%	0.11%	1.53%	8.34	9.35
	0.949	0.1536	0.5919	0.7455	0.1533	0.5919	0.7453	0.20%	-0.01%	0.03%	8.24	8.38
	0.994	0.1530	0.5921	0.7451	0.1530	0.5921	0.7451	0.00%	0.00%	0.00%	7.45	7.47
s38417	0.692	0.5845	3.8322	4.4167	0.5562	3.8305	4.3867	4.84%	0.04%	0.68%	148.25	162.70
	0.727	0.5504	3.8306	4.3810	0.5471	3.8304	4.37753	0.60%	0.01%	0.08%	147.23	150.42
	0.762	0.5457	3.8303	4.3760	0.5456	3.8303	4.3759	0.02%	0.00%	0.00%	156.65	156.88

Table 5: Leakage, dynamic and total power reduction from transistor-level V_{th} assignment (TLVA) with respect to cell-level V_{th} assignment (CLVA). Delay penalty is set to 0%, 5% and 10% for each of the test cases.

reason is the availability of slack on larger number of paths; CLVA assigns nominal V_{th} to most cells leaving only a few low-V_{th} cells on which TLVA can do optimization. High-performance circuits are desired to operate at the highest possible frequency, and techniques like retiming and clock skew scheduling are used for balancing pipelines. For such circuits, large leakage savings (15%-30%) can be expected from TLVA with reference to CLVA.

4. Conclusions

We have presented a sensitivity-based downsizing approach for TLVA. The approach is effective, accurate, scalable, and is easily usable with the today's design flows. We compare our approach with a recently proposed TLVA approach and find our approach superior in terms of quality of results and runtime.

A comparison between CLVA and TLVA shows a 5%-27% reduction in leakage for tight delay constraints. Even though our approach uses fewer cell variants than previous approaches, the number of cell variants is significantly larger than those required for CLVA. This leads to large cell libraries and increased characterization effort. Therefore, TLVA should be performed for only the most frequently used cells, such as inverters, buffers, NAND and NOR gates. Fortunately, the most frequently used cells typically have one or two inputs, and hence only a small number of variants need to be characterized for any cell. To further reduce library size, only one of the cell variants in which different logically equivalent inputs are fast may be retained, and pin-swapping techniques can be used during leakage optimization.

Our ongoing work includes development of better approaches to reduce cell library size. To increase scalability, we plan to investigate "batched" moves (along the lines of the PB algorithm in [21]) in which several *independent* transistors are assigned nominal V_{th} in every iteration. Recent works conclude that sizing and V_{th} assignment should be performed together [13, 18]. This is in contrast to the previously preferred approach of keeping sizing and V_{th} assignment separate; we plan to combine our approach with other power-reduction techniques such as sizing and dual-V_{dd}.

5. References

[1] A. Agarwal, C. H. Kim, S. Mukhopadhyay and K. Roy, "Leakage in Nano-Scale Technologies: Mechanisms, Impact and Design Considerations," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 6–11.

[2] F. Beeftink, P. Kudva, D. Kung and L. Stok, "Combinatorial Cell Design for CMOS Libraries," *Integration, the VLSI Journal*, vol. 29, no. 4, pp. 67–93, 2000.

[3] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester, "Selective Gate-Length Biasing for Cost-Effective Runtime Leakage Control," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 327–330.

[4] J. Halter and F. Najm, "A Gate-level Leakage Power Reduction Method for Ultra Low Power CMOS Circuits," in *IEEE Custom Integrated Circuits Conference*, 1997, pp. 475–478.

[5] M. Horiguchi, T. Sakata and K. Itoh, "Switched-Source-Impedance CMOS Circuit for Low Standby Sub-Threshold Current Giga-Scale LSI's," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1131–1135, 1993.

[6] I. Hyunsik, T. Inukai, H. Gomyo, T. Hiramoto and T. Sakurai, "VTCMOS Characteristics and its Optimum Conditions Predicted by a Compact Analytical Model," in *International Symposium on Low Power Electronics and Design*, 2001, pp. 123–128.

[7] J. Kao, S. Narendra and A. Chandrakasan, "MTCMOS Hierarchical Sizing Based on Mutual Exclusive Discharge Patterns," in *Proc. ACM/IEEE Design Automation Conference*, 1998, pp. 495–500.

[8] M. Ketkar and S. Saptnekar, "Standby Power Optimization via Transistor Sizing and Dual Threshold Voltage Assignment," in *Proc. IEEE International Conference on Computer Aided Design*, 2002, pp. 375–378.

[9] D. Lee and D. Blaauw, "Static Leakage Reduction Through Simultaneous Threshold Voltage and State Assignment," in *Proc. ACM/IEEE Design Automation Conference*, 2003, pp. 192–194.

[10] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu and J.Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, 1995.

[11] S. Mutoh, S. Shigematsu, Y. Matsuya, H. Fukada, T. Kaneko and J. Yamada, "1V Multithreshold-Voltage CMOS Digital Signal Processor for Mobile Phone Application," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1795–1802, 1996.

[12] S. Narendra, D. Blaauw, A. Devgan and F. Najm, "Leakage Issues in IC Design: Trends, Estimation and Avoidance," in *Proc. IEEE International Conference on Computer Aided Design*, 2003, tutorial.

[13] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson and K. Keutzer, "Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization," in *International Symposium on Low Power Electronics and Design*, 2003, pp. 158–163.

[14] K. Nose, M. Hirabayashi, H. Kawaguchi, S. Lee and T. Sakurai, "V_{th} Hopping Scheme to Reduce Subthreshold Leakage for Low-Power Processors," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 413–419, 2002.

[15] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tabae and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 861–869, 1997.

[16] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda and D. Blaauw, "Duet: An Accurate Leakage Estimation and Optimization Tool for Dual-V_{th} Circuits," *IEEE Transactions on Very Large Scale Integrated Systems*, vol. 10, no. 2, pp. 79–90, 2002.

[17] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda and D. Blaauw, "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing," in *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 436–441.

[18] A. Srivastava, D. Sylvester and D. Blaauw, "Power Minimization using Simultaneous Gate Sizing, Dual-Vdd and Dual-Vth Assignment," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 783–787.

[19] Q. Wang and S. B. K. Vradhula, "Static Power Optimization of Deep Submicron CMOS Circuits for Dual V_T Technology," in *Proc. IEEE International Conference on Computer Aided Design*, 1998, pp. 490–495.

[20] L. Wei, Z. Chen, M. Johnson, K. Roy and V. De, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits," in *Proc. ACM/IEEE Design Automation Conference*, 1998, pp. 489–494.

[21] L. Wei, Z. Chen, K. Roy, Y. Ye and V. De, "Mixed-V_{th} CMOS Circuit Design Methodology for Low Power Applications," in *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 430–435.

[22] L. Wei, K. Roy and C. K. Koh, "Power Minimization by Simultaneous Dual-V_{th} Assignment and Gate-Sizing," in *Proc. ACM/IEEE Design Automation Conference*, 2000, pp. 413–416.

[23] Y. Ye, S. Borkar and V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," in *Proc. Symposium on VLSI Circuits*, 1998, pp. 40–41.