

Interval Arithmetic Based Input Vector Control for RTL Subthreshold Leakage Minimization

Shilpa Pendyala and Srinivas Katkooi
Computer Science & Engineering
University of South Florida, Tampa, Florida 33620
Email: spendya2@mail.usf.edu, katkooi@cse.usf.edu

Abstract—Applying appropriate minimum leakage vector (MLV) to each RTL module instance results in a low leakage state with significant area overhead. For each RTL module, via Monte Carlo simulation, we identify a set of MLV intervals such that maximum leakage is within (say) 10% of the lowest known leakage. We can reduce area overhead by choosing PI MLVs such that resultant inputs to internal nodes are also MLVs. Otherwise, control points can be inserted. Based on interval arithmetic, given a DFG, we propose a heuristic for Primary Input (PI) MLV identification with minimal control points. Experimental results for DSP filters implemented in 16nm technology are promising.

I. INTRODUCTION AND MOTIVATION

Digital CMOS circuits implemented in deep-submicron technology nodes can consume subthreshold leakage power that is 50% or more of the total power consumption [1]. Therefore, subthreshold leakage optimization is an active research area with several techniques reported in the literature [2], [3], [4], [5], [6], [7], [8], [9], [10]. Of these techniques, Input Vector Control (IVC) is attractive due to its low latency overhead. In this work, we report a Register Transfer Level (RTL) leakage minimization approach where in given the data-flow-graph representation we identify low leakage vector using interval arithmetic.

Subthreshold leakage is a strong function of current input vector applied to a digital CMOS circuit [11]. Therefore, under idle conditions the circuit can be put in a low leakage mode by applying a minimum leakage vector (MLV). Once the circuit becomes active, it can readily process the new input vectors without any latency overhead. Several MLV identification algorithms [6], [7], [12] have been proposed at the logic-level.

A straightforward way for input vector control at RTL is to apply appropriate minimum leakage vector (MLV) to each RTL module instance thus resulting in a low leakage state. However this requires significant area and control overhead. The additional area is in terms of multiplexors at the inputs of the RTL modules using which we select MLV.

In this work, we propose an interval arithmetic based input vector control for RTL leakage minimization with very little area overhead. We characterize each module in the library (Section III-C) for leakage power based on two-phased Monte-Carlo simulation. We identify a set of low Leakage data intervals such that maximum leakage is within (say) 10% of the lowest known leakage. We can reduce area overhead by choosing PI MLVs such that resultant inputs to internal nodes are also MLVs. Otherwise, control points can be inserted.

We report experimental results for five datapath-intensive benchmarks implemented in 16nm technology (PTM Model

[13]). On average, we obtain 74.4% leakage savings, 59% dynamic power savings, and 70% total power savings. These savings are obtained at the expense of only 1.9% area overhead.

The rest of the paper is organized as follows: Section II presents background, related work, and terminology. Section III describes the proposed approach. Section IV reports experimental results. Finally, Section V draws conclusions.

II. BACKGROUND, RELATED WORK, AND TERMINOLOGY

We will first give a brief overview of the subthreshold leakage minimization techniques. As this work is related to input vector control technique, we will survey IVC techniques reported in the literature. After providing a brief overview of interval arithmetic, we establish the terminology used throughout this paper.

A. Subthreshold leakage minimization

Subthreshold leakage control can be in standby mode or active mode. Standby techniques include Multi Threshold CMOS (MTCMOS), Power gating, and Super Cutoff CMOS (SCCMOS). Active mode leakage control techniques include input vector control, force stacking, sleepy stack, and power gating with stacking.

Standby techniques – MTCMOS [2] and Power gating [4] involve disconnecting supply voltage and/or ground to the circuit through sleep transistors which need to be appropriately sized to reduce delay penalty (in active mode) and wake-up time (to restore circuit to active mode). Thus, these techniques incur both area and delay overheads. In Power gating, both sleep and logic transistors have low threshold voltage, while in MTCMOS, the sleep transistors are high V_t . In SCCMOS style [14], the sleep transistor is driven into super cutoff mode resulting in an order of magnitude leakage reduction in sleep transistor. These savings are at the expense of complex controller design and large delay penalty.

Active techniques – As the leakage depends only on the current input vector, during the idle mode, we can apply the minimum leakage vector (MLV). Thus, MLV needs to be determined *a priori* and incorporated into the circuit. This technique is known as the Input Vector Control (IVC). For an n -input module, as the input space grows exponentially (2^n), MLV determination heuristics have been proposed [6], [7]. Transistor stacking is effective in reducing leakage power. Hence, several design techniques [15], [16], [17] that favor increased transistor stacking are proposed.

Compared to the standby techniques, the main advantage of active techniques is that the circuit can switch idle to active mode with no delay penalty. For IVC technique the area penalty occurs due to additional hardware needed to incorporate MLV into the circuit. Delay penalty is incurred if this additional hardware is in the critical path of the circuit.

B. Input vector control technique

As IVC has little delay penalty, many researchers have proposed using IVC leakage power minimization. To the best of our knowledge, these techniques are at the logic level.

Abdollahi, Fallah, and Pedram [6] propose gate-level leakage reduction with two techniques. The first technique is an input vector control wherein SAT based formulation is employed to find the minimum leakage vector. The second technique involves adding nMOS and pMOS transistors to the gate in order to increase the controllability of the internal signals. The additional transistors increase the stacking effect leading to leakage current reduction. The authors report over 70% leakage reduction at the expense of up to 15% delay penalty.

Gao and Hayes [18] present integer linear programming and mixed integer linear programming approaches for leakage reduction by means of input vector control. MILP performs better than ILP and is 13 times faster. Average leakage current is about 25% larger than minimum leakage current.

IVC technique does not work effectively for circuits with large logic depth. Yuan and Qu [7] have proposed a technique to replace the gates of worst leakage state with other libraries in active mode. A divide-and-conquer approach is presented that integrates gate replacement, an optimal MLV searching algorithm for tree circuits, and a genetic algorithm to connect the tree circuits. Compared with the leakage achieved by optimal MLV in small circuits, the gate replacement heuristic and the divide-and-conquer approach can reduce on average 13% and 17% leakage, respectively.

C. Interval arithmetic

Interval arithmetic (IA) [19] is concerned with arithmetic operations such as addition and subtraction on intervals. The intervals can be either discrete or continuous. IA has been extensively applied in error bound analysis arising in numerical analysis.

In this work, we are concerned with integer arithmetic therefore we restrict our discussion to integer intervals. An interval $I = [a, b]$ represents all integers $a \leq i \leq b$. Further, the above interval is a *closed* interval as it includes both extremal values. We can have an *open* interval, such as $I = (a, b)$ where $a < i < b$. The *width* of an interval is the difference between the extremal values $|b - a|$. If the interval width is zero, then the interval is referred to as a *degenerate* interval (for example $[1, 1]$). We can represent a given integer, say a , as a degenerate interval $[a, a]$.

Given two intervals $U = [a, b]$ and $V = [c, d]$, the following

equations hold:

$$\begin{aligned} U + V &= [a, b] + [c, d] \\ &= [a + c, b + d] \end{aligned} \quad (1)$$

$$\begin{aligned} U - V &= [a, b] - [c, d] \\ &= [a - d, b - c] \end{aligned} \quad (2)$$

$$\begin{aligned} U * V &= [a, b] * [c, d] \\ &= [\min(a * c, a * d, b * c, b * d), \\ &\quad \max(a * c, a * d, b * c, b * d)] \end{aligned} \quad (3)$$

$$\begin{aligned} U \div V &= [a, b] \div [c, d] \\ &= [\min(a \div c, a \div d, b \div c, b \div d), \\ &\quad \max(a \div c, a \div d, b \div c, b \div d)] \end{aligned} \quad (4)$$

(assuming $0 \notin [c, d]$)

An interval $[c, d] \subseteq [a, b]$ if and only if $a \leq c \leq d \leq b$. As we will deal with binary operations (addition and multiplication) we talk in general of an ordered pair of intervals i.e., $([a, b], [c, d])$ where the interval $[a, b]$ is for the first input and $[c, d]$ for the second.

D. Definitions and Terminology

Definition 1: Leakage Power Function, $\mathcal{P}(V, t, w)$ returns the leakage power of a module (of type t and bitwidth w) when input vector V is applied.

The above function is defined for the purpose of presenting our idea. For example $\mathcal{P}((2, 3), +, 8)$ will return the leakage value of an 8-bit adder with inputs 2 and 3.

Definition 2: Estimated Lowest Leakage, $\mathcal{P}_{low}(t, w)$, is the lowest leakage value of a functional unit i.e., minimum $\mathcal{P}(V, t, w)$

$\mathcal{P}_{low}(t, w)$ can be obtained by either lower bound leakage analysis [20] of the underlying circuit or by simulation.

Definition 3: Optimization Tolerance, ε , is the permissible deviation from the estimated lowest leakage in any module.

ε is a user-specified constant. For example, if $\varepsilon = 0.1$, we can tolerate upto 10% increase in the leakage of any module in the design.

Definition 4: Low Leakage Vector, $\mathcal{V}(t, w)$, is an input vector v of a module of type t and width w , such that

$$\mathcal{P}(v, t, w) \leq (1 + \varepsilon)\mathcal{P}_{low}(t, w).$$

Definition 5: Low Leakage Interval, L , is an input interval such that for every vector $v \in L$, v is a low leakage vector.

Definition 6: Low Leakage Interval Set, $\mathcal{L}(t, w)$, is the set of all low leakage intervals of a given module of type t and w .

Definition 7: Data Flow Graph, $G(V, E)$, is a directed graph such that $v_i \in V$ represents an operation and $e = (v_i, v_j) \in E$ represents a data transfer from operation v_i to v_j .

We also assume two functions at our disposal, $\mathcal{T}(op)$ and $\mathcal{W}(op)$, that return the type and width of a given operation op , respectively.

III. PROPOSED APPROACH - INTERVAL ARITHMETIC BASED INPUT VECTOR CONTROL

We first present two motivating examples, and then formulate the problem. We describe a Monte Carlo based characterization technique to extract low leakage interval set of a

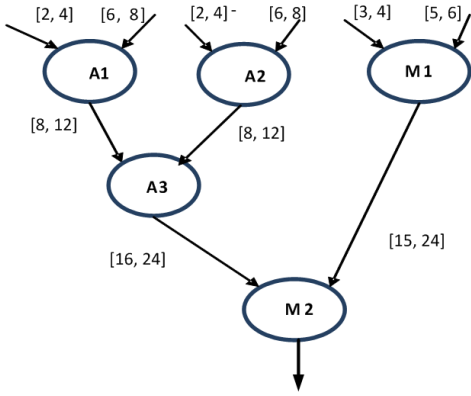


Fig. 1. Example 1 - Interval propagation with no control points

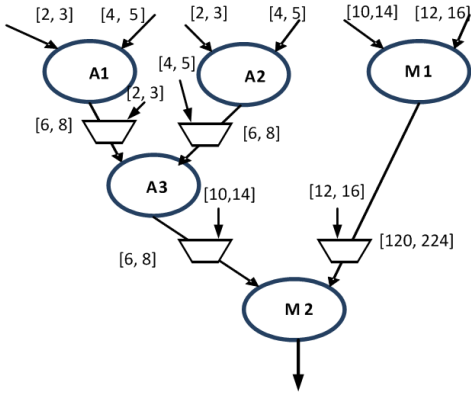


Fig. 2. Example 2 - Interval propagation with control point insertion

functional unit. We then present the heuristic that accepts an input DFG and determines MLVs at the primary inputs and internal nodes.

A. Motivating Examples

Example 1: In Figure 1, we show an example DFG with three adders (A1–A3) and two multiplier (M1, M2). Let us say the low leakage vector sets are: $\mathcal{L}(+, 8) = \{([2, 4], [6, 8]), ([8,12], [8,12]), ([14, 20], [14,24])\}$ and $\mathcal{L}(*, 8) = \{([3, 4], [5, 6]), ([9, 10], [12, 12]), ([13, 24], [20, 24])\}$. We start applying low leakage vectors for A1 and A2 i.e., $[2, 4]$ on the first input and $[6, 8]$ on the second input. Using interval arithmetic (Eqn. 1), we compute A1's and A2's output range to be $[8, 12]$. Thus, the input interval of A3 is $([8, 12], [8, 12]) \in \mathcal{L}(+, 8)$. Therefore, it puts A3 in a low leakage mode. Similarly the computed input interval for M2, i.e., $([16, 24], [20, 24]) \subset ([13, 24], [20, 24]) \in \mathcal{L}(*, 8)$, therefore puts M2 also in low leakage mode. From this example we see that applying low leakage vectors at the primary inputs can put all the internal nodes in the DFG in a low leakage state.

Example 2: Now consider the same DFG, however with different low leakage vector sets: $\mathcal{L}(+, 8) = \{([2, 3], [4, 5])\}$ and $\mathcal{L}(*, 8) = \{([10, 14], [12, 16]), ([20, 22], [20, 26])\}$. We again apply low leakage vectors on primary inputs (Figure 2). However, the computed input interval of A3, $([6,8], [6, 8])$ does not overlap with the interval in $\mathcal{L}(+, 8)$. Therefore, we will have to introduce a *control point* at the inputs of A3. The control point consists of a multiplexor that can be used to force a low leakage vector in idle state. Similarly, M2 needs a

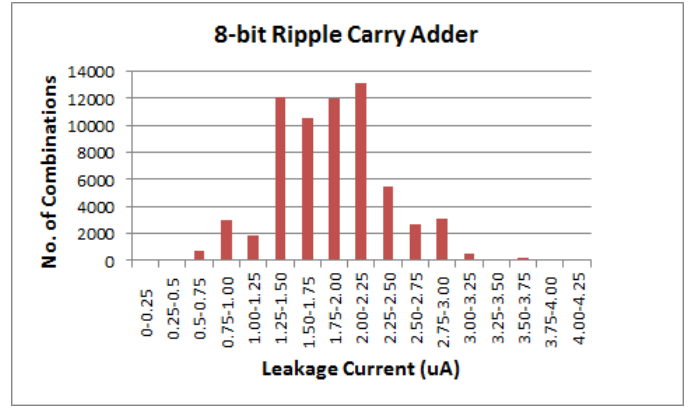


Fig. 3. Leakage current distribution for 8b adder implemented in 16nm node

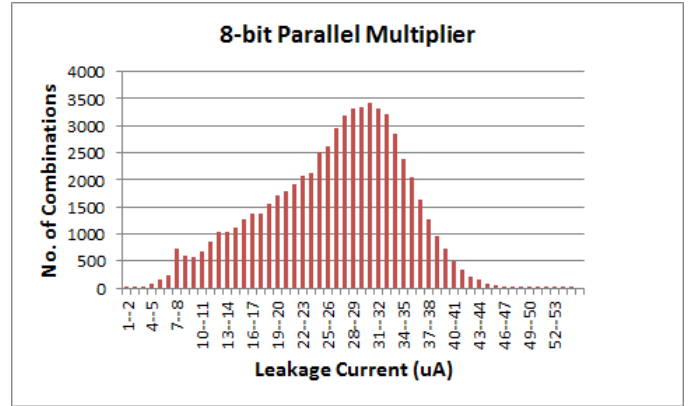


Fig. 4. Leakage current distribution for 8b multiplier implemented in 16nm node

control point. Thus, with two control points we put all modules in a low leakage state.

As seen from the above two examples, it is possible to achieve low leakage stage for the entire design by applying low leakage vectors only at the primary inputs. Our proposed heuristic attempts to maximize the leakage power savings while keeping the control overhead to as small as possible.

B. Problem Formulation

Given the following two inputs:

- a data flow graph $G(V, E)$ and
- set of low leakage interval sets, $\bigcup_{t,w} \mathcal{L}(t, w)$, for all distinct operations of type t and width w ,

identify low leakage vectors on primary inputs and a set of control points \mathcal{C} such that the following objective functions are minimized:

$$\sum_{v_i \in V} \mathcal{P}(\mathcal{V}, \mathcal{I}(v_i), \mathcal{W}(v_i)) \quad (5)$$

$$|\mathcal{C}| \quad (6)$$

C. Module Library Characterization

Figures 3 and 4 show the leakage current distribution of an 8-bit ripple carry adder and an 8-bit parallel multiplier, respectively. The data for both module instances have been generated by exhaustive simulation of the layouts implemented in 16nm technology node. The leakage power values are

measured using nanosim with PTM [13], [21] technology parameter values for 16nm node.

The leakage current range for 8b adder is $[0.084\mu\text{A}, 4.3\mu\text{A}]$ and that of the 8b multiplier is $[1.4\mu\text{A}, 56\mu\text{A}]$. Thus, the approximate max-to-min ratio for adder and multiplier are 51 and 40 respectively. For $\varepsilon=0.1$, the number of distinct low leakage vectors for the adder is 119. The percentage of input space that puts the adder in a low-leakage state ($\varepsilon=0.1$) is $(119/(2^8 * 2^8))*100 = 0.18\%$. These vectors can be merged to obtain low leakage intervals. The number of such intervals is 80. Similarly, for multiplier, number of low leakage vectors is 490 and the size of the interval set is 329. The percentage of input space that puts the multiplier in a low-leakage state ($\varepsilon=0.1$) is $(490/(2^8 * 2^8))*100 = 0.74\%$.

As the input space of a n -bit module instance grows exponentially, exhaustive simulation based low leakage interval extraction is not feasible. Therefore, we propose a Monte-Carlo (MC) based approach.

Typically, an MC based approach has four steps: (a) input space determination, (b) input sampling based on a probability distribution, (c) computation of property of interest, and (d) result aggregation.

Given the SPICE-level model of an n -bit module instance, we perform two successive MC runs. The property of interest is the leakage power. Both runs are subject to a user-specified time-limit.

- *Run I - Coarse grained MC run:* The input space under consideration is the entire input space, i.e., 2^{2n} input vectors. We uniformly sample the input space and then simulate the layout with samples to obtain leakage power values. These power values are sorted in ascending order and then $(\varepsilon \times 100)\%$ of the values are used to identify *low leakage* regions in the input space.
- *Run II - Fine grained MC run:* In this run, the input space is restricted to the regions identified in Run I. Further, the sampling is biased in the neighborhood of low leakage vectors identified previously. For a given sample, we accept the sample only if the corresponding leakage power falls within the $[\mathcal{P}_{low}, (1+\varepsilon)*\mathcal{P}_{low}]$ range. The result aggregation step involves merging input samples to create set of low leakage intervals.

The proposed MC based approach is scalable to larger circuits. In case of a parallel multiplier, the time taken by MC simulation is 6 hrs and 9 hrs for 8b and 16b instances, respectively. The simulations were carried out on a SunOS workstation (16 CPUs, 96GB RAM). For 8 bit instance, we collected 800 samples out of 65536. For a 16 bit instance, we collected 800 samples out of 4294967298 (65536×65536) for a 16b instance, respectively. We chose to simulate 800 samples because of the reasonable simulation time. In case of ripple carry adder, the time taken by the simulation is 5 hrs and 6 hrs for a 8b and 16b instances, respectively. Hence, the input space explored in 8b and 16b instances are 1.2% and $1.86 \times 10^{-4}\%$, respectively with a sampling interval of 32 and 1024, respectively.

The tradeoff incurred due to increased sample interval is suboptimal minimum leakage vector. The MC approach can be further sped up by increasing the sampling interval. Let the compilation time for a single Nanosim run in a simulation

```

1 Algorithm find_LLW
2 Inputs: (a) Graph  $G(V,E)$ ; (b) Low Leakage Vector Sets
3 Outputs: Low leakage vectors and control points
4 begin
5    $L \leftarrow \text{Topological\_Sort}(G)$  /*  $L$  is a sorted list */
6    $\mathcal{C} \leftarrow \emptyset$  /* internal control points */
7   foreach  $v_i \in L$  do
8     Let  $a$  and  $b$  denote input edges of  $v_i$ 
9      $c$  the output edge of  $v_i$ 
10    if  $v_i$  is a PI node
11    then
12       $I_{a,b} \leftarrow \mathcal{L}(\mathcal{T}(v_i), \mathcal{W}(v_i))$ 
13       $\mathcal{C} \leftarrow \mathcal{C} \cup \{a, b\}$ 
14    end if
15     $I_c \leftarrow \text{Interval\_Propagate}(I_{a,b}, \mathcal{T}(v_i))$ 
16    Let  $v_j$  be the successor of  $v_i$ 
17    Let  $d$  be the second input of  $v_j$ 
18    /* check for interval containment */
19     $\text{contains} \leftarrow \text{FALSE}$ 
20    foreach  $L \in \mathcal{L}(\mathcal{T}(v_j), \mathcal{W}(v_j))$  do
21      if  $L \cap I_c \neq \emptyset$  then
22         $\text{contains} \leftarrow \text{TRUE}$ 
23        break
24      end if
25    end for
26    if( $\text{contains} == \text{FALSE}$ ) then
27      /* insert a new control point */
28       $I_{c,d} \leftarrow \mathcal{L}(\mathcal{T}(v_j), \mathcal{W}(v_j))$ 
29       $\mathcal{C} \leftarrow \mathcal{C} \cup \{c, d\}$ 
30    end if
31  end for
32 end Algorithm

```

Fig. 5. Algorithm to determine Low Leakage Vector

be t , bit width be n and sample interval be s , simulation time is inversely proportional to the square of s and directly proportional to t . Hence, the time complexity of MC based approach is $O((2^{2n}/s^2)t)$.

D. Low Leakage Vector Determination

Figure 5 shows the pseudo-code of the proposed heuristic for low leakage vector determination. It accepts an input DFG (directed acyclic graph) and low leakage vector sets for distinct types and operations obtained from the characterization procedure as described in Section III-C. Breadth First Search is used to cover all the nodes of the graph.

First the graph is topologically sorted (line 5) to yield a sort list L . A set \mathcal{C} that collect the control points, is initialized (line 6). The for loop in lines 7–31 visits each node in the order specified by L . If a node is a PI node (i.e., both inputs to the node are primary), then the intervals on both inputs are initialized to the appropriate low leakage vector sets (line 12). If there are multiple intervals that result in low leakage, then an interval is chosen randomly. Both inputs are added to the set \mathcal{C} (line 13). On line 15, we call a function *Interval_Propagate()* that accepts an ordered interval pair and the operation type of the node (i.e., $\mathcal{T}(v_i)$). *Interval_Propagate()* implements the interval arithmetic equations (Eqns. 1–4) and returns an appropriate output interval I_c . In lines 19–25, we check if the

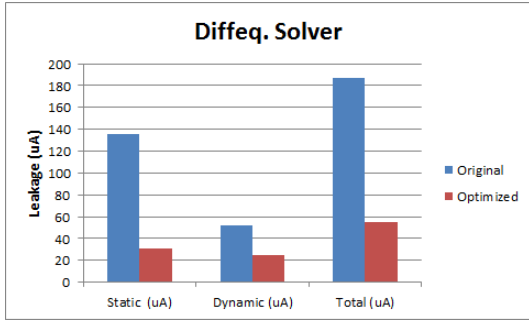


Fig. 6. Differential Equation Solver

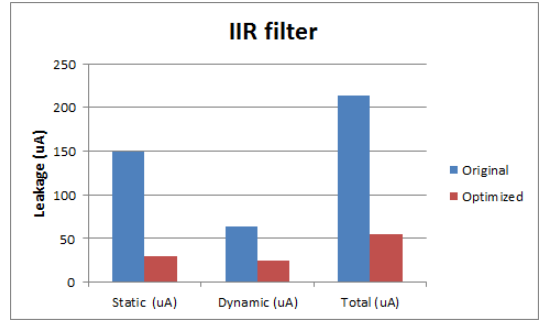


Fig. 9. IIR Filter

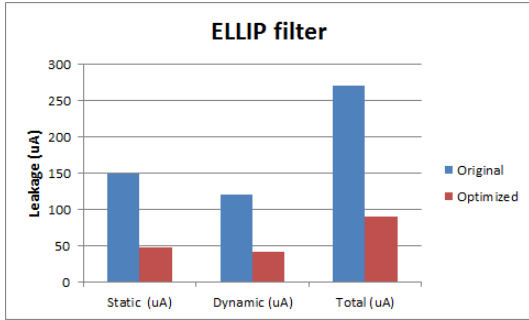


Fig. 7. Elliptic Wave Filter

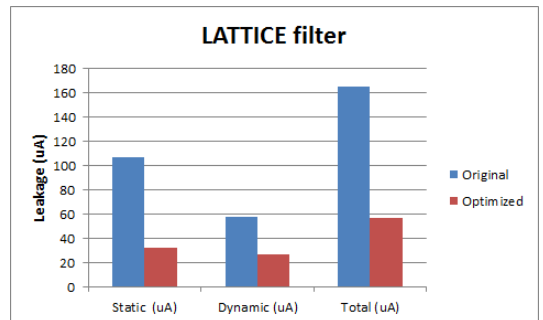


Fig. 10. Lattice Filter

computed interval is contained in low leakage vector set of the successor v_j . If the check succeeds, then we move onto the next node in the list. If the check fails, then a new control point is inserted by resetting the inputs of node v_j to its low leakage vector set (line 28) and adding the inputs of v_j to control point set.

In the above algorithm we assume only one successor for each node. This assumption is made to simplify the presentation of the algorithm. It is straightforward to extend the algorithm to multiple successors.

IV. EXPERIMENTAL RESULTS

We report the experiment results obtained by applying the proposed input vector control technique on five datapath-intensive benchmarks, namely, IIR, FIR, Elliptic, Lattice, and Differential Equation Solver. As described in Section III-C the library is characterized *a priori* and the low leakage vectors sets saved. The DFG of each filter is then processed to identify a low leakage vector. For the experimental results we assume $\epsilon=0.1$ (i.e., we can tolerate up to 10% leakage increase in any

module instance).

The leakage power values are measured at the layout level using nanosim. We employ the PTM models for 16nm technology node generated by the online model generation tool available on the ASU PTM website [13]. We simulate each layout with 1000 random vectors and measure the dynamic and subthreshold leakage currents. Figures 6–10 compare the average dynamic and leakage currents. We can observe that both leakage and dynamic components are significantly reduced. The savings in dynamic power are the side-effect of holding the inputs stable to the circuit. We can also observe the dominance of subthreshold leakage in deep-sub-micron nodes (16nm).

Table I tabulates the percentage power savings. We can observe that significant savings are obtained in leakage, dynamic, and total power (columns 2,3, and 4). The average savings in leakage, dynamic, and total powers are 74.4%, 59%, and 70% respectively.

Table II reports the area penalty and the number of control points. The area is measured in terms of the number of transistors in the HSPICE netlist. As we can observe the area overhead (second column) is reasonable in the range of 0.48% to 3.93% with an average of 1.9%. The last column reports the number of control points inserted in the intermediate nodes.

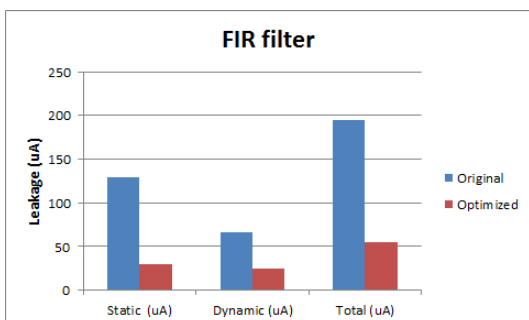


Fig. 8. FIR Filter

Design	Leakage Savings (%)	Dynamic Savings (%)	Total Savings (%)
Diffeq (2+, 5*)	77.29	52.79	70.51
EWf (26+, 8*)	68.13	65.11	66.79
FIR (4+, 5*)	77.57	62.52	71.81
IIR (4+, 5*)	79.83	61.47	74.31
Lattice (8+, 5*)	69.46	54.07	65.26

TABLE I
PERCENTAGE POWER SAVINGS

Design	Area	Control
	Overhead(%)	Points
Diffeq (2+, 5*)	0.48	1
EWf (26+, 8*)	2.16	8
FIR (4+, 5*)	1.77	4
IIR (4+, 5*)	1.33	3
Lattice (8+, 5*)	3.93	9

TABLE II
AREA PENALTY AND CONTROL POINTS

Design	Leakage Savings (%)			Area Overhead (%)	
	[7] Over MLV	[7] Random	Ours	[7]	Ours
Diffeq	21.79	32.15	77.29	6.89	0.48
EWf	24.6	34.69	68.13	6.0	2.16
FIR	21.07	32.1	77.57	6.8	1.77
IIR	21.09	32.24	79.83	6.8	1.33
Lattice	23.6	33.33	69.46	6.7	3.93

TABLE III
EXPERIMENT 1 - COMPARISON WITH [7] - MLV AT PIS

The number of control points depends on the percentage of input space in consideration to create the low leakage intervals.

Our technique is implemented at register transfer level of abstraction. To the best of our knowledge, there is no work done at RTL that uses input vector control for leakage reduction. We, therefore, compared our technique with a technique from prior work done at gate level. Out of the two relevant works from literature [6] and [7], [7] is an improvement over the technique used in [6]. Therefore, we compare with [7]. As [7] proposes a gate-level replacement algorithm, we synthesized gate level circuits for our benchmarks and then ran the algorithm.

We conducted two experiments using the gate replacement algorithm.

Experiment 1: In this experiment, MLV is provided to each module instance that receives primary inputs in the circuit followed by the gate replacement algorithm. Table III shows the comparison of leakage savings and area overhead in both the techniques when MLV is applied each module instance that receives primary inputs. Column 2 shows the improvement by gate replacement algorithm over a circuit where MLV is applied at primary inputs. Column 3 shows the improvement by application of MLV to primary inputs and gate replacement algorithm over a circuit where a random input vector is applied to primary inputs.

Experiment 2: In this experiment, MLV is applied to every module instance in the circuit and gate replacement algorithm from [7] is applied. Table IV reports similar results.

From these experiments, we demonstrate that the proposed algorithm yields better leakage savings over the gate-replacement algorithm [7] with smaller area overhead.

V. CONCLUSION

As leakage power is a strong function of current inputs applied to a circuit, we have formulated the low leakage vector

Design	Leakage Savings (%)			Area Overhead (%)	
	[7] Over MLV	[7] Random	Ours	[7]	Ours
Diffeq	20.88	32.24	77.29	9.6	0.48
EWf	22.43	33.8	68.13	12.25	2.16
FIR	21.14	32.51	77.57	10.2	1.77
IIR	21.14	32.51	79.83	10.2	1.33
Lattice	21.39	32.75	69.46	10.6	3.93

TABLE IV
EXPERIMENT 1 - COMPARISON WITH [7] - MLV AT ALL DFG NODES

identification in terms of identifying low leakage data intervals. We have successfully leveraged the value propagation in a data flow graph so that applying low leakage vectors at the primary inputs will drive most of the internal nodes to a low leakage state. We conclude that interval arithmetic based input vector control for RTL leakage minimization is feasible and significant subthreshold leakage savings can be obtained.

REFERENCES

- [1] "International technology roadmap for semiconductor," 2010.
- [2] S. Mutoh *et al*, "A 1 V multi-threshold voltage CMOS DSP with an efficient power management technique for mobile phone applications," in *Proceedings of the IEEE ISSC Conference*, Feb 1996, pp. 168–169.
- [3] K. Roy, "Leakage Power Reduction in Low-Voltage CMOS Design," in *Proceedings of the IEEE International Conference on Circuits and Systems*, 1998, pp. 167–173.
- [4] M. Powell *et al*, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in *Proceedings of ISLPED*, 2000, pp. 90–95.
- [5] H. Singhet *et al*, "Enhanced leakage reduction techniques using intermediate strength power gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 11, pp. 1215–1224, nov. 2007.
- [6] A. Abdollahi *et al*, "Leakage current reduction in cmos vlsi circuits by input vector control," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 140–154, feb. 2004.
- [7] L. Yuan and G. Qu, "A combined gate replacement and input vector control approach for leakage current reduction," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 2, pp. 173–182, feb. 2006.
- [8] L. Yan *et al*, "Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, nov. 2003, pp. 30–37.
- [9] X. He *et al*, "Adaptive leakage control on body biasing for reducing power consumption in cmos vlsi circuit," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, march 2009, pp. 465–470.
- [10] Y. Lee and T. Kim, "A fine-grained technique of nbt-aware voltage scaling and body biasing for standard cell based designs," in *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, jan. 2011, pp. 603–608.
- [11] J. Halter and F. Najm, "A gate-level leakage power reduction method for ultra-low-power cmos circuits," in *Custom Integrated Circuits Conference, 1997., Proceedings of the IEEE 1997*, may 1997, pp. 475–478.
- [12] R. Rao *et al*, "A heuristic to determine low leakage sleep state vectors for cmos combinational circuits," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, nov. 2003, pp. 689–692.
- [13] "Asu predictive technology model website." [Online]. Available: <http://ptm.asu.edu/>
- [14] H. Kawaguchi *et al*, "A super cut-off cmos (scmos) scheme for 0.5-v supply voltage with picoampere stand-by current," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 10, pp. 1498–1501, oct 2000.
- [15] S. Narendra *et al*, "Scaling of stack effect and its application for leakage reduction," in *Low Power Electronics and Design, International Symposium on, 2001., 2001*, pp. 195–200.
- [16] M. C. Johnson *et al*, "Leakage control with efficient use of transistor stacks in single threshold CMOS," *ITVLSI*, vol. 10, no. 1, pp. 1–5, February 2002.
- [17] N. Hanchate and N. Ranganathan, "Lector: a technique for leakage reduction in cmos circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 196–205, feb. 2004.
- [18] F. Gao and J. Hayes, "Exact and heuristic approaches to input vector control for leakage power reduction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 11, pp. 2564–2571, nov. 2006.
- [19] Ramon E. Moore, "Methods and applications of interval analysis". Philadelphia, PA : Siam., 1979.
- [20] M. C. Johnson and K. Roy, "Models and Algorithms for Bounds on Leakage in CMOS Circuits," *ITCAD*, vol. 18, no. 6, pp. 714–725, June 1999.
- [21] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *Electron Devices, IEEE Transactions on*, vol. 53, no. 11, pp. 2816–2823, nov. 2006.