Leakage current aware high-level estimation for VLSI circuits

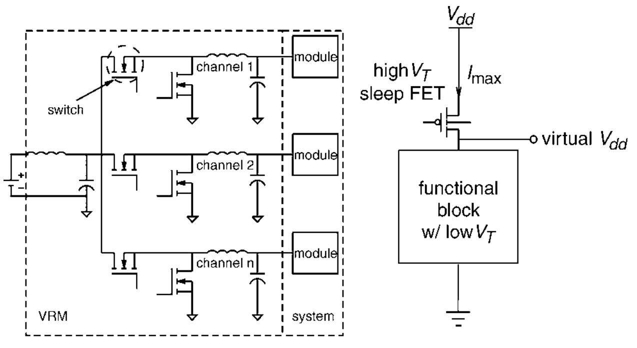
F. Li, L. He, J.M. Basile, R. Patel and H. Ramamurthy

Abstract: The ever-growing leakage current of MOSFETs in nanometre technologies is the major concern to high performance and power efficient designs. Dynamic power management via power-gating is effective to reduce leakage power, but it introduces power-up current that affects the circuit reliability. The authors present an in-depth study on high-level modelling of power-up current and leakage current in the context of a full custom design environment. They propose a methodology to estimate the circuit area, maximum power-up current, and minimum and maximum leakage current for any given logic function. Novel estimation metrics are built based on logic synthesis and gate-level analysis using only a small number of typical circuits, but no further logic synthesis and gate-level analysis are needed during the high-level estimation. Compared to time-consuming logic synthesis and gate-level analysis, the average errors for circuits from a leading industrial design project are 23.59% for area, 21.44% for maximum power-up current, 15.65% for maximum leakage current and 6.21% for minimum leakage current. In contrast, estimation based on quick synthesis leads to an 11× area difference in gate count for an 8-bit adder.

1 Introduction

Power has become one of the primary design constraints for both high-performance and portable system designs. As VLSI technology continues scaling down, leakage power becomes an ever-growing power component because of (i) increase of device leakage current due to the reduction in threshold voltage, channel length, and gate oxide thickness [1, 2], and (ii) the increasing number of idle modules in a highly integrated system. For current high-performance design methodologies, the contribution of leakage power increases at each technology generation [3]. The Intel Pentium IV processors running at 3 GHz already have an almost equal amount of leakage and dynamic power [4]. Dynamic power management (DPM) [5] via power gating at system and circuit levels is effective to reduce both leakage and dynamic power. Figure 1*a* shows a system with a multichannel voltage regulation module (VRM). The VRM channels can be configured to supply power independently for individual modules. Therefore, modules can be turned on or off at appropriate times for power reduction but still maintain the desired functionality and performance. Power gating at circuit level is also called MTCMOS in [6] (see Fig. 1*b*). A PMOS sleep transistor with a high threshold voltage connects the power supply to the virtual V_{dd}. The sleep transistor is turned on when the function block is needed, and is turned off otherwise. (Instead of the PMOS sleep transistor, an NMOS sleep transistor can be inserted between the ground and virtual ground and I_p to be presented later becomes the discharging current in this case. For simplicity of presentation, we assume PMOS sleep transistors in this paper.) With the growing leakage power, power gating at either system level or circuit level is a viable alternative to clock gating, as clock gating only reduces dynamic power. We use MTCMOS to study power gating in this paper, and the idea can be extended to VRM design and DPM at system level.

Key questions in applying power gating include: (i) How to estimate the leakage reduction by power gating and how to decide the area overhead of power gating? The answer determines whether power gating is worthwhile for a given design. Leakage current and power estimation has been studied [7–9] at transistor, gate and system level.



a *b*

Fig. 1 *Power gating*
a System level
b Circuit level

© IEE, 2005
IEE Proceedings online no. 20045165
doi:10.1049/ip-cdt:20045165
Paper received 3rd December 2004

F. Li and L. He are with the Electrical Engineering Department, University of California, Los Angeles, CA 90095, USA
J.M. Basile, R. Patel and H. Ramamurthy are with Intel Corporation, Santa Clara, CA, USA
E-mail: feil@ee.ucla.edu

Most estimation methods consider the input vector-dependent property of leakage current. The MTCMOS technique for leakage reduction further introduces power-up transient current when turning on or off a circuit module. As shown in [10], all nodes in a power-gated module are at logic '0' state. They must be brought to valid logic states by power-up current (I_p) before useful computation can begin. Similar to leakage current, power-up current I_p also depends on the input vector. Its maximum value must be known to design reliable sleep transistors, and evaluate their area overhead. (ii) How to answer the above question and make a decision at an early design stage without performing time-consuming logic synthesis and gate level analysis? Early decision making is needed to deal with time-to-market pressure. High-level estimation for circuit area [11, 12], dynamic power [13, 14], and transient switching current [15] have been studied. However, no previous work has studied high-level modelling and estimation of power-up current.

In this paper we propose a method to estimate the gate count for a given logic function without performing logic synthesis. We show that the quick synthesis leads to an 11× difference for a simple adder, and further validate and improve an area estimation technique that was originally developed for a library with a limited number of cells [14]. The improved estimation method has an average error of 23.59%. This paper then presents an in-depth study of a unified high-level modelling for power-up current and leakage current by using commercial synthesis tools such as Design Compiler in the pre-characterisation stage. We propose a high-level metric to estimate the maximum I_p without performing logic synthesis and gate-level I_p analysis. We verify this metric by a newly developed gate-level analysis for accurate I_p. We then further extend this high-level estimation methodology to leakage power estimation. We use the design environment of a leading industrial high-performance CPU design project. There are hundreds of cells with various sizes (1× to 65×) in the library. All experiments are carried out on a number of typical circuits. The circuits are specified in Verilog and synthesised by Design Compiler to verify our high-level estimations. Owing to the need for IP protection, we report normalised values for currents in this paper.

2 Area estimation

2.1 Overview

Table 1 presents synthesis results for adders where synthesis 1 uses logic functions with intermediate variables, and synthesis 2 uses equivalent logic functions without intermediate variables. An 11× difference in gate count is observed for an 8-bit adder. It shows that quick synthesis using Verilog specified at a higher abstraction level does not necessarily lead to a good estimation. Instead of using quick synthesis, we apply and improve the high-level area estimation in [14].

We summarise the estimation flow from [14] in Fig. 2. It contains a one-time pre-characterisation, where gate-count

Table 1: Area count based on quick synthesis

Circuit	Synthesis 1	Synthesis 2
1-bit adder	3	3
4-bit adder	20	16
8-bit adder	42	490

$\mathcal{A}$ is pre-characterised as a function $\mathcal{F}$ of the linear measure $\mathcal{L}$ and output entropy $\mathcal{H}$. Then, a multi-output function (MOF) is transformed into a single output function (SOF) by adding a m-to-1 MUX, where m is the number of outputs in the original MOF. $\mathcal{L}$ and $\mathcal{H}$ are calculated for the SOF to look up the pre-characterised table and obtain gate count. Removing the MUX from this gate count leads to $\mathcal{A}$ for the original MOF.

We improve the original estimation method in two ways. First, it is claimed in [14] that SOFs with the same output entropy $\mathcal{H}$ and same linear measure $\mathcal{L}$ have the same $\mathcal{A}$. However, we find that it may not be true for VLSI functions implemented with a rich cell library. Functions with smaller output probability of logic '1' have a lower gate count under the same linear measure. Therefore, we have pre-characterised $\mathcal{A}$ as a function $\mathcal{F}(\mathcal{L}, \mathcal{P})$, where $\mathcal{P}$ is the output probability. Since complementary probabilities lead to the same entropy, our pre-characterisation is more detailed compared to that in [14]. Further, we have developed an output-clustering algorithm to partition the original MOF into sub-functions (called sub-MOFs) with minimum support set overlap, and have improved the efficiency and accuracy of the high-level estimation. We summarise our estimation flow with the difference highlighted in Fig. 2, and describe each step and our implementation details in the following Sections.

2.2 Linear measure

Linear measure $\mathcal{L}$ is determined by on- and off-sets of an SOF as $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_0$, where $\mathcal{L}_1$ and $\mathcal{L}_0$ are the linear measure for the on-set and off-set, respectively. $\mathcal{L}_1$ is further defined as $\mathcal{L}_1(f) = \sum_{i=1}^{N} c_i p_i$ ($\mathcal{L}_0$ can be defined similarly). N is the number of different sizes of all the

Estimation of gate count *$\mathcal{A}$ in [14]:*

Pre-characterisation

1. Pre-characterise $\mathcal{A}$ as $\mathcal{F}(\mathcal{L}, \mathcal{H})$ using randomly generated SOFs. $\mathcal{H}$ is the output entropy

Mapping

1. Partition the MOF into sub-MOFs randomly;
2. Transform each sub-MOF into an SOF by adding MUX;
3. Compute $\mathcal{L}$ and $\mathcal{H}$;
4. Obtain gate count $\mathcal{A}'$ of trasformed SOF from $\mathcal{F}(\mathcal{L}, \mathcal{H})$;
5. Remove MUX from $\mathcal{A}'$ to get gate count of original MOF;
6. Obtain final estimate by adding up gate-count for all the sub-MOFs.

Estimation of gate count *$\mathcal{A}$ in this paper:*

Pre-characterisation

1. Pre-characterise $\mathcal{A}$ as $\mathcal{F}(\mathcal{L}, \mathcal{P})$ using randomly generated SOFs. $\mathcal{P}$ is the **ouput probability**.

Mapping

1. Partition the MOF to **minimise support-set overlap**;
2. Transform each sub-MOF into an SOF by adding MUX;
3. Compute **ouput probability** $\mathcal{P}$ and linear measure $\mathcal{L}$;
4. Obtain gate count $\mathcal{A}'$ of transformed SOF from $\mathcal{F}(\mathcal{L}, \mathcal{P})$;
5. Remove MUX from $\mathcal{A}'$ to get gate count of original MOF;
6. Obtain final estimate by adding up gate-count for all the sub-MOFs.

Fig. 2 *Estimation of gate count $\mathcal{A}$*

prime implicants in a minimal cover of function *f*. The size of a prime implicant is the number of literals in it. c_i is one distinct prime implicant size. p_i is a weight of prime implicants with size c_i and can be computed in the following way. Suppose all the input vectors to the logic function can occur with the same probability. Let $c_1, c_2, \ldots, c_N$ be sorted in a decreasing order, and weight p_i be the probability that one random input vector matches all the prime implicants with size c_i but not by the prime implicants with size from c_1 to c_{i-1}, $1 < i \leq N$. For $i = 1$, p_1 is just the probability that one random input vector matches prime implicants with size c_1. Here 'a matching' means that the intersection operation [16] between the vector and the prime implicant is consistent. Note that p_i satisfies the equation $\sum_{i=1}^{N} p_i = \mathcal{P}(f)$, where $\mathcal{P}(f)$ is the probability to satisfy function *f*.

The minimum cover of an SOF can be obtained by two-level logic minimisation [17]. To compute the weight p_i, a straightforward approach is to make the minimum cover disjoint and compute the probability exactly. However, in practice, this exact approach turns out to be very expensive. In our experiments, when the number of inputs is larger than 10, the program using the exact approach does not finish within reasonable time. But with p_i defined as the probability, $\mathcal{L}_1(f)$ can be viewed as a random variable $\mathcal{L}_1(f)$ with certain probability distribution. For each random input vector, the variable $\mathcal{L}'_1(f)$ takes a certain value 'randomly'. With probability of $1 - \mathcal{P}(f)$, $\mathcal{L}_1(f)$ takes the value '0'. Then $\mathcal{L}_1(f)$ becomes the mean of the random variable $\mathcal{L}'_1(f)$. By assuming that the variable $\mathcal{L}'_1(f)$ takes a Gaussian distribution, we use the Monte Carlo simulation technique to estimate the mean value efficiently.

2.3 Output probability and gate-count recovery

The output probability can be obtained as a byproduct of Monte Carlo simulation. Since weight p_i satisfies $\sum_{i=1}^{N} p_i = P(f)$, we can keep record of all the p_i during the Monte Carlo simulation. When the simulation process satisfies the stopping criteria, the output probability can be obtained easily. To recover the gate count of the original MOF, the estimated gate count for the transformed SOF is subtracted by $\alpha\mathcal{A}_{mux}$. $\mathcal{A}_{mux}$ is the gate count of the complete multiplexer we have inserted, and α is the coefficient to obtain the reduced multiplexer gate count due to the logic optimisation.

2.4 Output clustering

As the number of primary outputs increases, the time to calculate the minimum cover of a function increases non-linearly. To make the two-level optimisation more efficient, one may partition the original MOF into sub-MOFs by output clustering, and then estimate for each sub-MOF individually. The gate count of the original MOF is the sum of gate counts for all the sub-MOFs. However, estimation errors may be introduced due to the overlap of the support sets of the sub-MOFs. We propose to partition the outputs with minimum support set overlap (see Fig. 3). A PO-graph is constructed with vertices representing the primary outputs (POs). If two POs have support set overlap, there is an edge connecting the two corresponding vertices. The edge weight is the size of the common support set. The vertex weight is the sum of the weights of all edges connected to this vertex. There are two loops in the algorithm. In each iteration of the inner loop, the vertex with the minimum weight is deleted and the weights are updated for edges and vertices that connect the deleted vertex. It continues until the number of remaining vertices is less or equal to the pre-specified cluster size. The PO-graph is then re-constructed with all the POs that have not been clustered. The algorithm continues until all the outputs are clustered and the PO-graph becomes empty.

```
Output Clustering Algorithm:
Construct the PO-graph;
While (PO-graph is not empty)
begin
        While (# of remaining vertices > CLUSTER_SIZE)
        begin
                Delete the vertex v with the minimum weight;
                Update weights for edges and vertices connecting v;
        end
        Obtain one cluster of POs using remaining vertices;
        Re-construct PO-graph excluding POs already in clusters;
end
```

Fig. 3 *Output clustering algorithm*

2.5 Experimental results

We compare area estimation methods in Fig. 4, where the *x*-axis is the circuit ID number and the *y*-axis is the gate count. During the Monte Carlo simulation to calculate the linear measure, we choose the parameters of confidence and error as 96% and 3%, respectively. The actual gate count is obtained by the synthesis using Design Compiler. The method with random output clustering has an average absolute error of 39.36%. By applying our output clustering algorithm to minimise support set overlap, we reduce the average absolute error to 23.59%. Such estimation errors are much smaller compared to the 11× gate-count difference in Table 1. Note that different descriptions of a given logic function do not change the $\mathcal{L}$ and $\mathcal{P}$, and therefore do not affect the estimation results by our approach. High-level estimation costs over 100× less runtime compared to logic synthesis.

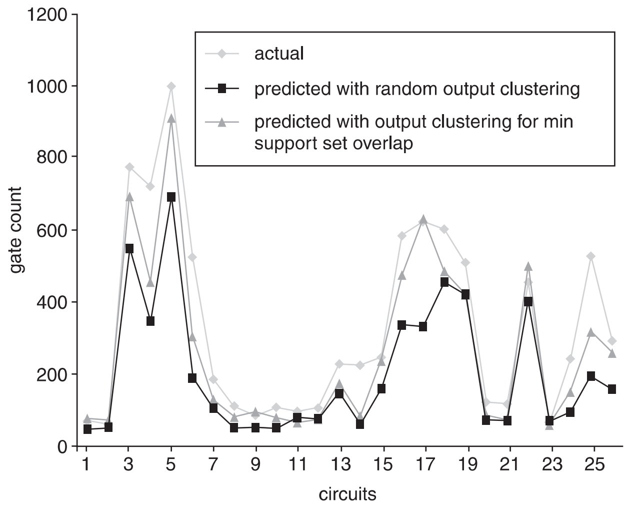


Fig. 4 *Comparison between actual and predicted gate-count*

3 Power-up current estimation

Given the Boolean function f of a combinational logic block and the target cell library, our high-level estimation finds the maximum power-up current $I_p(f)$ when the logic block is implemented with the given cell library for power gating. A Boolean function can be implemented under different constraints, but we assume the min-area implementation in this paper.

We propose the following high-level metric $\mathcal{M}_p$ for $\mathcal{I}_p(f)$:

$$I_p(f) \propto \mathcal{M}_p(f) = \mathcal{I}_{avg} \cdot \mathcal{A} \tag{1}$$

where $\mathcal{A}$ is the gate count estimated using the method in Section 2, and $\mathcal{I}_{avg}$ is the weighted average I_p to be discussed in Section 3.2. Because an accurate gate-level estimator is required for the calculation of $\mathcal{I}_{avg}$ and verification of $\mathcal{M}(f)$, we introduce our gate-level estimation in the next Section.

3.1 Gate-level estimation

3.1.1 Background knowledge: The following observation has been shown in [10]:

Observation 1: All the internal nodes in a circuit with PMOS sleep transistors are at logic ‘0’ after the circuit stays in the power-off state for a long enough time.

Power-up current (I_p) occurs when the power supply is turned on for a circuit module and it is different from the normal switching current (I_s). I_s depends on two successive circuit states S_1 and S_2, which are determined by two successive input vectors V_1 and V_2 for combinational circuits. As discussed in Section 1, I_p can be viewed as a special case of I_s where the state S_1 before power-up is logic ‘0’ for all the nodes. Because no input vector leads to a circuit state with all nodes at logic ‘0’ for nontrivial circuits, the maximum I_p is, in general, different from the maximum I_s. Moreover, the I_p of a circuit is solely decided by the circuit state S_2, and therefore decided by a single input vector when the circuit is powered up. To illustrate that I_p depends on the input vectors, we present the I_p obtained by SPICE simulation for an 8-bit adder under two different input vectors in Table 2. The difference of the maximum I_p is about 24%. I_p is greatly affected by the input vector when the circuit is powered up. We define I_p element to be the power-up current generated by an individual gate, and give the following observation related to timing:

Observation 2: If a set of gates are controlled by one single sleep transistor, all these gates are powered up simultaneously, i.e. all the I_p elements for these gates have the same starting time.

Further, we study the effects of the turn-on time (i.e. the time to turn on the sleep transistor in a MTC-MOS circuit) by simulating a five-stage inverter chain and an eight-bit adder. We use random SPICE simulation with large enough number of vectors for different turn-on times from 0.1 ns to 10 ns (see Table 3). Based on the results, we conclude:

Observation 3: I_p is very sensitive to turn-on time; I_p reduces when turn-on time increases.

Even though a large turn-on time can help reduce the power-up current, a small turn-on time is preferred for high-performance designs. A careful study is needed to achieve the best trade-off between performance and reliability/cost related to a small turn-on time.

Table 2: Maximum I_p of an 8-bit adder

Circuit	Vector1	Vector2	Difference
Adder8	1830	2260	23.50%

Table 3: Turn-on time against power-up current

Turn-on time (ns)	Power-up current (μA) Inv chain (5-stage)	Adder8
0.1	60	2400
1	11.2	629
10	4.49	260

ATPG-based algorithms have been proposed in [10]. It is assumed that the power-up current is proportional to the total charge in the circuit after power-up, and the charge for one single gate with output value ‘1’ is proportional to its fanout number. Therefore, the gate fanout number is used as the figure of merit of the power-up current (I_p) for the gate with output value ‘1’. The ATPG algorithm is performed to find the logic vector that maximises the figure of merit. However, this algorithm does not take the current waveform in the time domain into account. The vector obtained by ATPG algorithm has to be further used in SPICE simulation to obtain the I_p value.

To achieve a more accurate estimation and obtain the I_p value directly, we need a current model that can capture the current waveform. We apply the piecewise linear (PWL) function to model the I_p element. SPICE simulation is used to get the power-up current waveform, and the waveform is linearised at different regions to build the PWL model for each cell in the library (see Fig. 5). Our PWL model considers the following dimensions: gate type, input pin number, gate size, fanout number, turn-on time, and post-power-up output logic value. Note that a much simplified PWL model, the right-triangle current model, has been successfully used in [18] for maximum switching current estimation.

3.1.2 Genetic algorithm: Since exhaustive search for the input vector that generates the maximum I_p is infeasible, we apply genetic algorithm (GA) in our gate-level estimation. We encode the solution (i.e., input vector) into a string so that the length of the string is equal to the number of

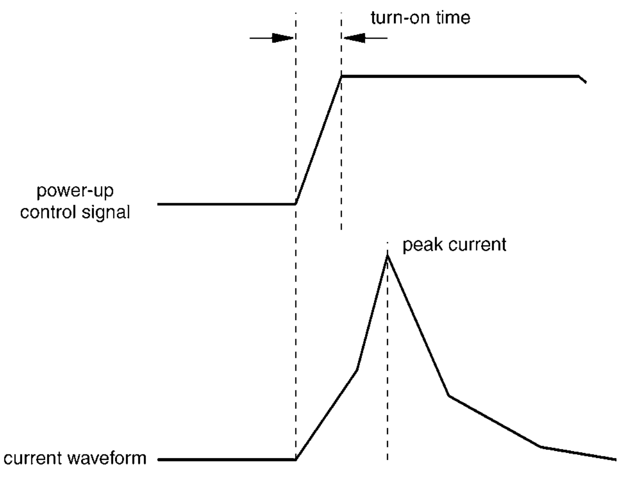


Fig. 5 *PWL current model*

primary inputs. Each bit in the string is either '1' or '0'. The initial population is randomly generated. The population size is proportional to the number of primary inputs. The fitness value is chosen as the maximum I_p value under the input vector represented by the string. The I_p value is obtained by waveform simulation with the PWL current model.

Tournament selection [19] is used in our selection process. From the current generation, we randomly pick two strings and select the one with the higher fitness value. After that, the two strings are removed from the current generation. We repeat this procedure until the current generation becomes empty. By doing this, we divide the original strings into inferior and superior groups. We keep record of the strings in the superior group and put these two groups together to carry out tournament selection again. The two superior groups generated in the two tournaments are combined to go through crossover and mutation, and produce the new generation. The string with the highest fitness will be selected twice so that the best solution so far will stay in the next generation. Since strings with lower fitness have higher probability of being dropped, the average fitness tends to increase by each generation.

The crossover scheme we use is the one-point crossover algorithm. One bit position is randomly chosen for two parent-strings and they are crossed at that position to get the two child-strings. After crossover, we further use a simple mutation scheme that flips each bit in the string with equal probability. The new generation is produced after crossover and mutation, and is ready to go through a new iteration of natural selection. The algorithm stops after the number of generations exceeds a pre-defined number. We summarise the algorithm in Fig. 6.

We carry out experiments and compare the results of genetic algorithm to that by simulations with 5000 random vectors (called 'random 5000') in Table 4. Under the same PWL current model, GA achieves up to 27% estimation improvement to approach the upper bound of power-up current. The average improvement for all the circuits is 6%.

Moreover, we compare the I_p obtained by SPICE simulation of the entire benchmark circuit with the best vector from genetic algorithm to the I_p calculated by our PWL model in Fig. 7(a). Even though the I_p by PWL model is different from that given by SPICE simulation, there is a close correlation between these two currents. Therefore, our PWL model has a high fidelity versus SPICE simulation and gives a conservative estimation. Owing to the high fidelity, we propose to scale the I_p from the PWL model by a constant K, and compare the I_p values by the new

```
GenerationNumber = 1;
Randomly generate the initial generation;
While (GenerationNumber < MAX_GENERATION)
begin
        Evaluate the fitness value of each string;
        Apply tournament selection to get parent-strings;
        Apply crossover and mutation to get child-strings;
        Produce the new generation with the child-strings;
        GenerationNumber++;
end
return the peak $I_p$ and its correspondent input vector
```

Fig. 6 *Generic algorithm for gate-level estimation*

Table 4: Experiments for gate-level estimation

Circuit ID	Random 5000	Genetic algorithm	Estimate improvement (%)
1	12234.00	12234.00	0.00
2	11756.12	11756.12	0.00
3	147989.14	150437.28	+1.65
4	138497.98	143767.61	+3.80
5	193818.20	193818.20	+0.00
6	80758.16	92024.79	+13.95
7	26190.60	28170.35	+7.56
8	13124.87	13975.08	+6.48
9	15205.17	16349.61	+7.53
10	30649.70	30964.78	+1.03
11	16687.65	16687.65	+0.00
12	18042.84	18577.41	+2.96
13	42066.13	42565.20	+1.19
14	**20261.80**	**25710.94**	**+26.89**
15	42982.51	49502.92	+15.17
16	84997.98	94697.20	+11.41
17	105486.86	113573.73	+7.67
18	123394.24	124298.63	+0.73
19	100145.88	100798.74	+0.65
20	18250.63	18907.54	+3.60
21	20560.78	21486.25	+4.50
22	85656.30	85981.01	+0.38
23	11911.55	12599.41	+5.78
24	38598.95	41482.89	+7.47
25	81358.29	92199.21	+13.32
26	35530.71	40449.54	+13.84
Average improvement			+6.06

The turn-on time is 0.1 ns for results in columns 2 and 3

scaled PWL model and SPICE simulation. As shown in Fig. 7(b), the difference between I_p values is greatly reduced. The derivation of the scaling constant K will be discussed in Section 3.2.

3.2 Calculation of $\mathcal{I}_{avg}$ and experimental results

$\mathcal{I}_{avg}$ is not simply the average I_p element for all cells in a library. The frequency of cells used in logic synthesis should be taken into account. We assume that the logic synthesis results for a few typical circuits (or random logic

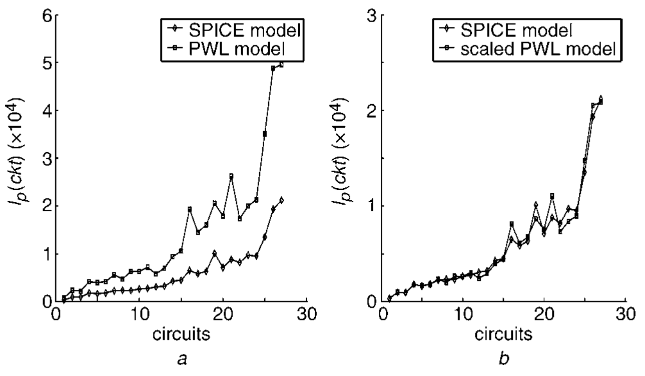


Fig. 7 *Maximum I_p obtained by genetic algorithm*
a Under PWL model and SPICE model
b Under-scaled PWL model and SPICE model

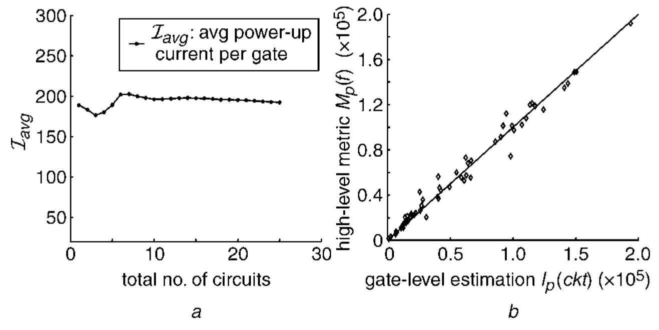


Fig. 8 *$\mathcal{I}_{avg}(PWL)$ and $I_p(ckt)$*

a $\mathcal{I}_{avg}(PWL)$ against number of circuits
b Comparison between gate-level estimation $I_p(ckt)$ and high-level metric $\mathcal{M}_p$, where accurate gate count is assumed

functions) are available. We calculate $\mathcal{I}_{avg}$ in a regression-based way as follows. We compute the average maximum $\mathcal{I}_p$ per gate for n typical circuits by applying the gate-level estimation. We then increase n until the resulting value becomes a 'constant'. We treat this constant value as $\mathcal{I}_{avg}$. In Fig. 8*a*, we plot $\mathcal{I}_{avg}$ with respect to the number of circuits used to calculate $\mathcal{I}_{avg}$. The Figure shows that the change of the $\mathcal{I}_{avg}$ value is relatively large when the number of circuits is small (less than 10 in the Figure). After the number of circuits increases to 20, the value of $\mathcal{I}_{avg}$ becomes very stable and can be used as our high-level metric $\mathcal{M}_p$.

To validate our regression-based $\mathcal{I}_{avg}$, we use the computed value of $\mathcal{I}_{avg}$ under the PWL model and the accurate gate count to obtain the high-level metric $\mathcal{M}_p$. We compare the gate-level estimation $\mathcal{I}_p(ckt)$ by the genetic algorithm to the metric $\mathcal{M}_p$ in Fig. 8*b*. The average absolute error between $I_p(ckt)$ and $\mathcal{M}_p$ is 12.02%. Note that the circuits in Fig. 8*b* are different from those used to compute $\mathcal{I}_{avg}$ for the purpose of the verification of metric $\mathcal{M}_p$.

Our high-level estimation methodology is also directly applicable for SPICE current model. We run SPICE simulation using the best vector obtained by the genetic algorithm to calculate $\mathcal{I}_{avg}(SPICE)$. As shown in Fig. 9, a stable $\mathcal{I}_{avg}(SPICE)$ is also reached quickly. In addition, we can apply $\mathcal{I}_{avg}$ to calibrate our gate-level estimation.

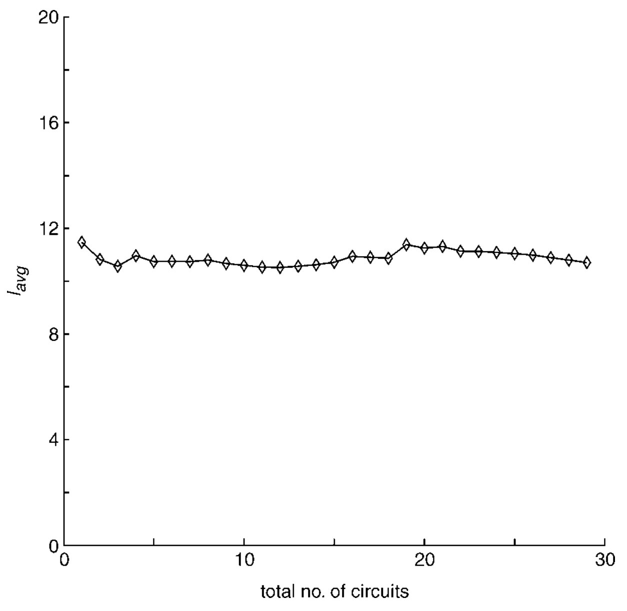


Fig. 9 *$\mathcal{I}_{avg}(SPICE)$*
Turn-on time = 0.1 ns

Table 5: Results of high-level I_p estimation

PWL current model			
Circuit id	Gate-level est. $I_p(ckt)$	High-level est. $I_p(ckt)$	Abs. err(%)
1	12234.00	14626.98	19.56
2	11756.12	13857.14	17.87
3	150437.28	133374.96	11.34
4	143767.61	87569.42	39.09
5	193818.20	175523.76	9.44
6	92024.79	58700.38	36.21
7	28170.35	24442.45	13.23
8	13975.08	15589.28	11.55
9	16349.61	18283.73	11.83
10	30964.78	15011.90	51.52
11	16687.65	12702.38	23.88
12	18577.41	13664.68	26.44
13	42565.20	33488.08	21.33
14	25710.94	15781.74	38.62
15	49502.92	45420.62	8.25
16	94697.20	91418.62	3.46
17	113573.73	121442.42	6.93
18	124298.63	93343.23	24.90
19	100798.74	80833.31	19.81
20	18907.54	16166.66	14.50
21	21486.25	14049.60	19.81
22	85981.01	96422.59	12.14
23	12599.41	10970.23	12.93
24	41482.89	28676.58	30.87
25	92199.21	61009.90	33.83
26	40449.54	49847.21	23.23
Average			21.44

Let $I_p(PWL)$ and $I_p(PWL)$ be the I_p values based on PWL and SPWL current model, respectively. Then we have $I_p(SPWL) = I_p(PWL)/K$, where K is the scaling constant defined in Section 3.1.2 and Fig. 7*b* can be calculated as $\mathcal{I}_{avg}(PWL)/\mathcal{I}_{avg}(SPICE)$.

Furthermore, we compare the maximum I_p using estimated $\mathcal{I}_{avg}$ and estimated $\mathcal{A}$ to the maximum I_p obtained via logic synthesis followed by gate-level analysis. Table 5 shows that the average estimation error is 21.44%. We measure gate-level analysis runtime as the time for logic synthesis and genetic algorithm, and measure the high-level estimation runtime as the time for area estimation and application of the formula $\mathcal{M}_p(f) = \mathcal{I}_{avg} \cdot \mathcal{A}$ (pre-characterisation only has one-time cost and is ignored in the runtime comparison). Our high-level estimation achieves more than 200$\times$ run-time speedup for large test circuits.

4 Leakage current estimation

High-level estimation of leakage current is necessary for evaluating the feasibility of various leakage reduction techniques at a very early design stage. The fact that leakage current also depends on one single input vector means that leakage current shares similar properties with the power-up current that we have studied. Therefore, we believe that a similar high-level metric $\mathcal{I}^{lkg}_{avg}$ can also be applied to high-level leakage current estimation. We

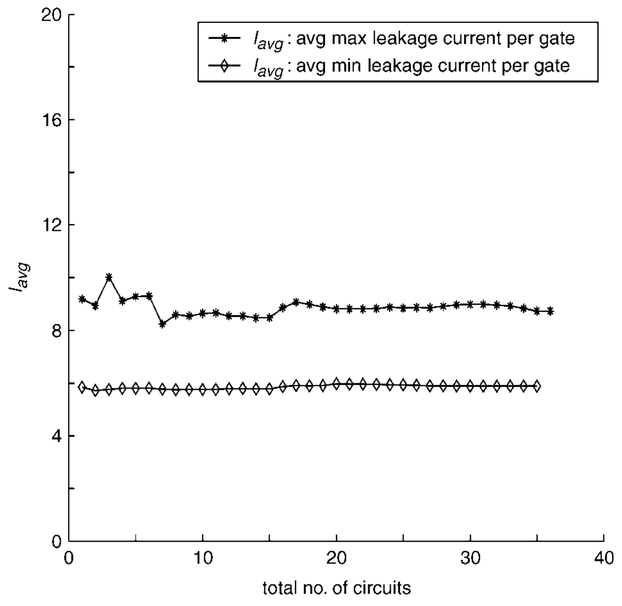


Fig. 10 *Verification of high-level metric for maximum and minimum leakage current*

calculate the high-level metric $\mathcal{I}_{avg}^{lkg}$ for leakage current in a similar way. We apply the genetic algorithm in the gate-level estimation for a few typical circuits, and obtain the

Table 6: Leakage current: gate-level estimation $I_{lkg}(ckt)$ against high-level metric $\mathcal{M}_{lkg}$ for both maximum and minimum leakage current

Gate no.	Max leakage current		Min leakage current	
	$\mathcal{I}_{lkg}(ckt)$	$\mathcal{M}_{lkg}$	$\mathcal{I}_{lkg}(ckt)$	$\mathcal{M}_{lkg}$
136	1079.05	1187.92	731.64	800.16
134	1202.57	1170.45	709.90	788.39
1351	10015.90	11800.62	8264.34	7948.64
1361	10045.15	11887.96	8215.79	8007.47
1758	11174.98	15355.65	10552.79	10343.23
865	7792.39	7555.54	4913.03	5089.24
273	2649.79	2384.58	1516.68	1606.20
159	1609.78	1388.82	922.25	935.48
151	1511.27	1318.94	895.70	888.41
247	1809.43	2157.48	1266.02	1453.23
145	1569.37	1266.54	860.62	853.11
95	1075.89	829.80	539.92	558.93
389	3191.86	3397.81	2274.26	2288.69
377	2516.43	3292.99	1708.89	2218.09
350	3389.71	3057.15	2206.68	2059.23
863	8046.87	7538.07	4604.20	5077.48
1078	12735.66	9416.04	5923.57	6342.43
680	5009.85	5939.61	4116.00	4000.79
764	5370.14	6673.33	4584.77	4495.01
191	1907.51	1668.33	1083.49	1123.75
217	2221.62	1895.44	1267.53	1276.72
485	4245.67	4236.34	3331.65	2853.51
108	1159.54	943.35	635.71	635.42
311	3300.93	2716.50	1772.74	1829.77
861	7090.33	7520.60	4389.23	5065.71
452	5433.82	3948.10	2530.96	2659.35
	Average abs. error (%):			
	15.65		6.21	

metric $\mathcal{I}_{avg}^{lkg}$ for both maximum and minimum leakage current. Our gate-level estimation uses an input-pattern-dependent leakage current model built by SPICE simulation.

In Fig. 10, we show the metric $\mathcal{I}_{avg}^{lkg}$ with respect to the number of circuits used to calculate $\mathcal{I}_{avg}^{lkg}$. Characterising only a few typical circuits (less than 20) is enough to obtain the stable value of $\mathcal{I}_{avg}^{lkg}$. This justifies the application of our high-level metric to leakage current estimation. With the metric $\mathcal{I}_{avg}^{lkg}$ and area estimation, we apply high-level leakage current estimation to our test circuits using the formula:

$$P_{lkg} = N_{gate} \cdot I_{avg}^{lkg} \cdot V_{dd} \tag{2}$$

The estimation results are presented in Table 6. The average estimation error is 15.65% for the maximum leakage current and 6.21% for the minimum leakage current. Again, the circuits used in Table 6 are different from those in Fig. 10 for the purpose of verification.

A simple leakage power model at the architectural level has been proposed in [20]. They modelled leakage power by the equation $P_{static} = V_{cc} \cdot N \cdot k_{design} \cdot \hat{I}_{lkg}$, where V_{cc} is the supply voltage, N is the number of transistors. k_{design} is an empirically determined parameter representing the average characteristics of library cells. $\hat{I}_{lkg}$ is a technology-dependent parameter representing the per-device subthreshold leakage. However, it is unclear how these parameters are determined for different technologies and cell libraries. Our high-level metric $\mathcal{I}_{avg}^{lkg}$ with a well-defined calculation mechanism can be viewed as a combination of parameters k_{design} and $\hat{I}_{lkg}$. In addition to its simplicity, our calculation of $\mathcal{I}_{avg}^{lkg}$ can take into account how frequently the library cells are used by the synthesis tool and the fact that leakage current is input pattern dependent.

4.1 *Temperature and V_{dd} scaling*

Considering that leakage power also depends on supply voltage V_{dd} and temperature, we further characterise the temperature and voltage scaling of I_{avg}^{lkg} based on the following SPICE BSIM4 model. We distinguish subthreshold leakage and gate leakage due to their different temperature scaling trend. The BSIM4 subthreshold leakage current model [3] is as follows:

$$I_{sub} = A \exp\left\{\frac{(V_{GS} - V_T - \gamma' V_{SB} + \eta V_{DS})}{nV_{TH}}\right\} \times \left(1 - \exp\left\{\frac{V_{DS}}{V_{TH}}\right\}\right) \tag{3}$$

$$A = \mu_0 C_{ox} \frac{W}{L_{eff}} V_{TH^2} e^{1.8} \tag{4}$$

where V_{GS}, V_{DS} and V_{SB} are the gate-source, drain-source and source-bulk voltages, respectively; V_T is the zero-bias threshold voltage, V_{TH} is the thermal voltage kT/q, γ' is the linearised body-effect coefficient, η is the drain induced barrier lowering (DIBL) coefficient, μ_0 is the carrier mobility, C_{ox} is gate capacitance per area, W is the width and L_{eff} is the effective gate length.

From (3) we can see the temperature scaling for subthreshold leakage current is $T^2 e^{1/T}$, where T is the temperature, and the voltage scaling for leakage current is $e^{V_{dd}}$. Based on these observations, we propose the following formula for subthreshold leakage metric I_{avg}^{sub} considering temperature

Table 7: Comparison between our high-level leakage model and SPICE simulation

Circuit	Temperature (°C)	V_{dd}	I_{avg}^{lkg} (μA) Formula	SPICE	Abs. err. (%)
Logic circuits including adder,	100	0.95	23.44	23.56	0.49
multiplier, and shifter	100	1.05	29.56	29.63	0.23
	80	0.95	19.44	19.54	0.56
	80	1.05	25.14	25.21	0.27
	60	0.95	16.00	16.11	0.65
	60	1.05	21.33	21.39	0.31

and voltage scaling:

$$I_{avg}^{sub}(T, V_{dd}) = I_s^{sub}(T_0, V_0) \cdot T^2 \cdot \exp\left\{\frac{\alpha_{s1} \cdot V_{dd} + \beta_{s1}}{T}\right\} \quad (5)$$

where I_s^{sub} is a constant current at the reference temperature T_0 and voltage V_0. α_{s1} and β_{s1} in (5) are empirical constants decided by circuit designs.

In the BSIM4 model, gate leakage current is modelled as gate direct tunnelling current – including tunnelling current between gate and substrate (I_{gb}) and current between gate and channel (I_{gc}). The formulas for both I_{gb} and I_{gc} are:

$$I_{gb} = W_{eff} \cdot L_{eff} \cdot X_1 \cdot (EXP_{acc} + EXP_{inv}) \quad (6)$$

$$\begin{aligned} I_{gc} &= W_{eff} \cdot L_{eff} \cdot X_2 \cdot \exp\{-B_3 \cdot T_{ox} \\ &\quad \cdot (\alpha_3 - \beta_3 \cdot V_{oxdepinv}) \cdot (1 + \gamma_3 \cdot v_{oxdepinv})\} \end{aligned} \quad (7)$$

where

$$X_1 = A_r \cdot T_{oxRatio} \cdot V_{gb} \cdot V_{uax} \quad (8)$$

$$\begin{aligned} EXP_{acc} &= \exp\{-B_1 \cdot T_{ox} \\ &\quad \cdot (\alpha_1 - \beta_1 \cdot V_{oxacc}) \cdot (1 + \gamma_1 \cdot V_{oxacc})\} \end{aligned} \quad (9)$$

$$\begin{aligned} EXP_{inv} &= \exp\{-B_2 \cdot T_{ox} \cdot (\alpha_2 - \beta_2 \cdot V_{oxdepinv}) \\ &\quad \cdot (1 + \gamma_2 \cdot V_{oxdepinv})\} \end{aligned} \quad (10)$$

$$X_2 = A_2 T_{oxRatio} V_{gse} V_{uax} \quad (11)$$

$A_1, A_2, B_1, B_2, B_3, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2$ and γ_3 are all empirical constants given by the BSIM4 gate leakage model, W_{eff} and L_{eff} are the channel width and length, respectively; $T_{oxRatio}$, V_{uax} are defined in BSIM4 gate leakage model.

Combining subthreshold leakage and gate leakage, we still keep the format of the formula in (2), but take into account the different scaling feature for subthreshold leakage and gate leakage. The total leakage for a logic circuit can be modelled:

$$P_{lkg} = N_{gate} \cdot I_{avg}^{lkg} \cdot V_{dd} \quad (12)$$

$$I_{avg}^{lkg}(T, V_{dd}) = I_s(T_0, V_0) \cdot f_{avg}(T, V_{dd}) \quad (13)$$

where P_{lkg} is the total leakage power for a logic circuit, I_{avg} is the total leakage current per gate, I_s is the I_{avg} at given temperature T_0 and supply voltage V_0, $f_{\text{avg}}(T, V_{dd})$ is the scaling function to characterise temperature and V_{dd} scaling considering both subthreshold and gate leakage. It can be expressed as follows:

$$f(T, V_{dd}) = \\ A \cdot T^2 \cdot \exp\left\{\frac{\alpha \cdot V_{dd} + \beta}{T}\right\} + B \cdot \exp\{\gamma \cdot V_{dd} + \delta\} \quad (14)$$

where A, B, α, β, γ, and δ are empirical constants for different circuit types, technologies and designs.

We obtain the constants in (5) and (14) empirically by determining the power consumption for different circuit types at multiple temperatures using SPICE simulations and then applying curve fitting. Table 7 compares I_{avg}^{lkg} by SPICE simulation at different temperature and V_{dd} to our I_{avg}^{lkg} calculated by our temperature and V_{dd} scaling formulae. We use the average leakage current for data-path circuits: adder (4-bit, 16-bit and 32-bit), shifter (8-bit, 16-bit and 32-bit), and multiplier (4-bit, 5-bit and 6-bit). The estimation error for I_{avg}^{lkg} at scaled temperature and V_{dd} is less than 1%. This high-level leakage model considering temperature and V_{dd} scaling can be used in the architectural and system-level simulations. We have applied our high-level leakage power model in a coupled thermal and power microarchitecture simulator, PT_{scalar} [21], which studies the interdependence between leakage and temperature and impact on processor performance [22].

5 Conclusions and discussions

Using design examples and design environment of a leading industrial CPU project, we have presented an improved high-level area estimation method. The estimation has an average error of 23.59% for designs using a rich cell library. We have also proposed a high-level metric to estimate the maximum power-up current due to power gating for leakage reduction. Compared to time-consuming logic synthesis followed by gate-level analysis, our high-level estimation has an average error of 21.44% for power-up current. We further extend our high-level estimation methodology to leakage current and the average estimation error is 15.65% and 6.21% for maximum and minimum leakage current, respectively. We also develop the high-level metric for leakage current considering temperature and supply voltage (V_{dd}) scaling. The estimation error for the metric is less than 1% at different temperatures and supply voltages.

Our high-level estimation method can be readily applied to estimate the area overhead due to the sleep transistor insertion in power gating. There are two primary constraints for the sleep transistor. One constraint is the IR voltage drop that introduces a performance penalty. Appropriate sizing of the sleep transistor can be performed to satisfy this constraint based on the maximum switch current. The high-level estimation of maximum switch current has been studied in [15]. There is a reliability constraint for sleep transistors (i.e. avoidance of damaging the sleep transistor by a large transient current). We can obtain the maximum transient current as the larger one between the maximum power-up current and maximum switching current, and size the sleep transistor to satisfy the reliability constraint. In addition to that, our high-level leakage model considering temperature and V_{dd} scaling can also be applied in architectural and system-level simulations. One of the applications is that our high-level leakage model has been successfully used in a coupled thermal and power microarchitectural simulator PT_{scalar} [21, 22].

6 Acknowledgments

This research was partially supported by the NSF CAREER Award 0093273, SRC grant 2002-HJ-1008 and a grant from Intel Design Science and Technology Committee.

7 References

1 Taur, Y., and Ning, T.H.: 'Fundamentals of modern VLSI devices' (Cambridge University Press, 1998)

2 Thompson, S., Packan, P., and Bohr, M.: 'MOS scaling: Transistor challenges for the 21st century', *Intel Technol. J.*, 3rd quarter 1998

3 Chandrakasan, A., Bowhill, W.J., and Fox, F.: 'Design of high-performance microprocessor circuits' (IEEE Press, 2001)

4 Grove, A.S.: 'Changing vectors of Moore's law'. Keynote speech, Int. Electron Devices Meeting, Dec. 2002

5 Benini, L., Bogliolo, A., and Micheli, G.D.: 'Dynamic power management of electronic systems'. Proc. Int. Conf. on Computer Aided Design, Nov. 1998, pp. 696–702

6 Kao, J.T., and Chandrakasan, A.P.: 'Dual-threshold voltage techniques for low-power digital circuits', *IEEE J. Solid-State Circuits*, 2000, **35**, pp. 1009–1018

7 Cheng, Z., Johnson, M., Wei, L., and Roy, K.: 'Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks'. Proc. Int. Symp. on Low Power Electronics and Design, Aug. 1998

8 Johnson, M.C., Somasekhar, D., and Roy, K.: 'Models and algorithms for bounds on leakage in CMOS circuits', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1999, **18**, pp. 714–725

9 Jiang, W., Tiwari, V., de la Iglesia, E., and Sinha, A.: 'Topological analysis for leakage prediction on digital circuits'. Proc. 7th Asia and South Pacific Design Automation Conf., joint with 15th Int. Conf. on VLSI Design, 2002

10 Li, F., and He, L.: 'Maximum current estimation considering power gating'. Proc. Int. Symp. Phys. Design, 2001, pp. 106–111

11 Cheng, K.T., and Agrawal, V.: 'An entropy measure for the complexity of multi-output Boolean functions'. Proc. 27th ACM/IEEE Design Automation Conf., 1999, pp. 302–305

12 Nemani, M., and Najm, F.: 'High-level area prediction for power estimation'. Proc. IEEE Custom Integr. Circuits Conf., 1997, pp. 483–486

13 Nemani, M., and Najm, F.N.: 'Towards a high-level power estimation capability', *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 1996, **15**, pp. 588–598

14 Nemani, M., and Najm, F.: 'High-level area and power estimation for VLSI circuits', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1999, **18**, pp. 697–713

15 Bodapati, S., and Najm, F.N.: 'Energy and peak-current per-cycle estimation at RTL', *IEEE Trans. VLSI*, 2003, **11**, (4), pp. 525–537

16 Abramovici, M., Breuer, M.A., and Friedman, A.D.: 'Digital systems testing and testable design (IEEE Press, 1990)

17 Micheli, G.D.: 'Synthesis and optimization of digital circuits (McGraw-Hill, New York, 1994)

18 Krstic, A., and Cheng, K.-T.: 'Vector generation for maximum instantaneous current through supply lines for CMOS circuits'. Proc. Design Automation Conf., June 1997, pp. 383–388

19 Miller, B.L., and Goldberg, D.E.: 'Genetic algorithms, tournament selection and effects of noise', *Complex Syst.*, 1995, **9**, pp. 193–212

20 Butts, J.A., and Sohi, G.S.: 'A static power model for architects'. Proc. 33rd Int. Symp. on Microarchitecture, Nov. 2000, pp. 191–201

21 'PTscalar', http://eda.ee.ucla.edu/PTscalar/, 2004

22 Liao, W., He, L., and Lepak, K.: 'Temperature and supply voltage aware performance and power modeling at microarchitectural level', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2005, **24**, (7), pp. 1042–1053