Symbolic Power Analysis of Cell Libraries

Matthias Raffelsieper and MohammadReza Mousavi

Department of Computer Science, TU/Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
M.Raffelsieper@tue.nl, M.R.Mousavi@tue.nl

Abstract. Cell libraries are collections of logic cores (cells) used to construct larger chip designs; hence, any reduction in their power consumption may have a major impact in the power consumption of larger designs. The power consumption of a cell is often determined by triggering it with all possible input values in all possible orders at each state. In this paper, we first present a technique to measure the power consumption of a cell more efficiently by reducing the number of input orders that have to be checked. This is based on symbolic techniques and analyzes the number of (weighted) wire chargings taking place. Additionally, we present a technique that computes for a cell all orders that lead to the same state, but differ in their power consumption. Such an analysis is used to select the orders that minimize the required power, without affecting functionality, by inserting sufficient delays. Both techniques have been evaluated on an industrial cell library and were able to efficiently reduce the number of orders needed for power characterization and to efficiently compute orders that consume less power for a given state and input-vector transition.

1 Introduction

A *cell library* is a collection of logic cores used to construct larger chip designs, consisting of combinational cells (e.g., `and` and `xor`) and sequential cells (e.g., latches and flip-flops). Cell libraries are usually described at multiple levels of abstraction, such as a transistor netlist and a Verilog description. Cells are used repeatedly in many larger designs and hence any minor improvement in their power consumption may lead to considerable power saving in the subsequent designs.

In this paper, we analyze the dynamic power consumption of netlists, as these describe the design that will finally be manufactured. An analysis at the level of Verilog descriptions is also possible but not very promising, because the same functional behavior (as that of the netlist) is often described using very different structures, a prominent example being User Defined Primitives (UDPs). To measure the dynamic power consumed by a netlist we use an abstract measure, namely the number of wires charged, possibly weighted by the node capacitance, if this is additionally available.

From a given transistor netlist, we first build a transition system, which describes the state of each wire in the netlist. In practice, this transition system

is described symbolically by equations in the inputs and values of wires [2]; hence, we use Binary Decision Diagrams (BDDs) for a symbolic representation of the netlist semantics. In order to efficiently analyze the different power consumption of different permutations, we quotient these permutations into equivalence classes. Every such equivalence class contains different orders of applying a transition from one input vector to another, but all of these orders have the same functional effect and consume the same amount of power. Thus, during power characterization, only one of these orders has to be considered.

In practice, the order of evaluating input changes may be controlled efficiently; in such cases, we seek a reduction in the dynamic power consumption by choosing, among functionally equivalent orders, the one that has the minimal power consumption. For this problem, we developed an analysis of functionally equivalent orders. Then, given the current state of the wires and an input vector transition, we can determine the order that consumes the minimal amount of power. Thus, by choosing this order the functionality of the circuit is not altered, but the power consumption is indeed reduced.

Related Work. The work reported in [3] also determines the power consumption of cells. The authors present an empirical algorithm, which also groups together different input vector transitions. However, they group together different values of inputs, whereas our approach groups together different orders of applying the same input vector. Furthermore, their grouping is made manually and afterwards all remaining input vectors and orders are enumerated explicitly, as opposed to our symbolic approach. Another approach that also uses a transition system model of circuits is presented in [8]. This approach builds an explicit representation of the transition system, and hence has to combat the size of these transition systems by simplifying the netlist, something which is not required in our symbolic representation. A symbolic representation of cells for the purpose of power analysis is also used in [1]. There, the symbolic representation is used during simulation of cells to determine the charge for each wire. Our work can be seen as a preprocessing step to theirs, as we first reduce the number of orders that later have to be simulated. Already in [7] it was observed that superfluous transitions (called *glitches*) of signals cause an increased power consumption. In contrast to our work, there glitches are detected by simulations, and only considered at cell outputs. The authors propose a number of techniques to reduce glitches. One of these is the addition of delays to enforce a certain order of events, which is also what we propose to select a low power evaluation. The theoretical basis of this paper builds upon [5] and [6]. Those analyses are extended by also taking power consumption into account and using the results to build symbolic graph structures that represent the equivalence classes of orders in a compact way.

Paper Structure. The rest of this paper is structured as follows. In Section 2, we introduce the notion of vector-based transition system which we use to model the semantics of transistor netlists. Furthermore, the section introduces orders as permutations of inputs and lists as their building blocks. Section 3 then presents our first technique to determine all equivalence classes of orders that are

functionally equivalent and consume the same amount of power. A technique that determines equivalence of orders based on functional equivalence is then presented in Section 4. For each of these equivalence classes, it is furthermore analyzed which of these orders consumes the least amount of power for a given state and input vector transition. We briefly sketch our implementation in Section 5 and report empirical results obtained from applying our implementation on an industrial cell library in Section 6. We conclude the paper in Section 7.

2 Preliminaries

The analysis in this paper is concerned with *transistor netlists*. These consist of a number of transistor instantiations, which for the purpose of this paper are seen as a switch. From such a transistor netlist, a system of Boolean equations is created, using the method of [2]. These equations form a transition system, where states are represented by a vector of Boolean variables. Transitions occur between stable states (i.e., states that are finished evaluating for the current input values) and are labeled with another Boolean vector, representing the new values of the inputs. The structure of the states is irrelevant for the transitions, thus the set of states is kept abstract in our formal treatment. Only in the experiments, state vectors are investigated to determine the number of charged wires. Since a transistor netlist can start up in any arbitrary state, the transition systems we consider do not have an initial state, instead an evaluation can start in any state.

Hence, we consider a *vector-based transition system*, which is a triple $T = (S, I, \rightarrow)$ where S is an arbitrary set of states (usually, as explained above, S is a set of vectors representing the current internal state), $I = U^m$ is the set of *input vectors* for some basic set U (commonly the Boolean values $\mathbb{B}$), and $\rightarrow \subseteq S \times I \times S$ is the transition relation. For a vector $\vec{v} = (v_1, \ldots, v_k)$, we denote its j-th position by $\vec{v}|_j = v_j$ and the update of the vector $\vec{v}$ in coordinate j by value v' is denoted by $\vec{v}[j := v'] = (v_1, \ldots, v_{j-1}, v', v_{j+1}, \ldots, v_k)$.

We are only interested in one-input restricted traces, because they form the common semantic model of hardware description languages (the so-called *single event* assumption in [3]). In order to define *one-input restricted traces*, i.e., traces where the used input vectors differ in at most one coordinate, we use the *Hamming distance* $\mathrm{d_H}(\vec{i}_1, \vec{i}_2) = |\{1 \leq j \leq m : \vec{i}_1|_j \neq \vec{i}_2|_j\}|$. A one-input restricted trace is then a trace consisting of consecutive steps $s_0 \xrightarrow{\vec{i}_1} s_1 \xrightarrow{\vec{i}_2} s_2$ with $\mathrm{d_H}(\vec{i}_1, \vec{i}_2) \leq 1$. We define the transition system $T^I = (S \times I, \rightarrow_{T^I})$ as the system where the states are extended with the input vector used to arrive in that state, and the transitions are only labeled by the input position that was changed. Formally, we have $\rightarrow_{T^I} \subseteq (S \times I) \times \{1, \ldots, m\} \times (S \times I)$ where $s_0;\vec{i}_0 \xrightarrow{j}_{T^I} s_1;\vec{i}_1$ iff $s_0 \xrightarrow{\vec{i}_1} s_1$ and $\vec{i}_1 = \vec{i}_0[j := \vec{i}_1|_j]$. Here and in the following we denote a tuple $(s, \vec{i}) \in S \times I$ by $s;\vec{i}$. It is easy to see that one-input restricted traces can be converted between the two representations. Thus, we use the two representations interchangeably in the rest of this paper and drop the subscript T^I if no confusion can arise.

For an evaluation of a netlist, we want to consider each input exactly once, to determine whether it has changed its value or not. Thus, we are interested in

one-input restricted traces of length m where none of the indices occurs twice. This can be described by means of a *permutation*, that assigns to each step the coordinate of the input to consider. We write Π_m for the set of all permutations over the numbers $\{1, \ldots, m\}$. In the remainder, we will make use of the fact that every permutation can be obtained by a sequence of *adjacent transpositions*, which are swappings of two neighboring elements. To be able to construct permutations from smaller parts, we denote by $\mathcal{L}_m$ the set of all *lists* containing each of the numbers $\{1, \ldots, m\}$ at most once. Then, we interpret the permutations Π_m as those lists containing every number exactly once (i.e., the lists of length m). Given a list $\ell = j_1 : \cdots : j_k \in \mathcal{L}_m$, we define the *length* of this list as $|\ell| = k$ and the *sublist* $\ell[a \,..\, b] = j_a : \cdots : j_b$ for $1 \leq a \leq b \leq k$. The empty list is denoted $\emptyset$ and a singleton list is denoted by its only element. The concatenation of two lists $\ell_1 = j_1 : \cdots : j_a$ and $\ell_2 = j'_1 : \cdots : j'_b$ with $j_c \neq j'_d$ for all $1 \leq c \leq a$ and $1 \leq d \leq b$ is defined as $\ell_1 {+\!+} \ell_2 = j_1 : \cdots : j_a : j'_1 : \cdots : j'_b$.

3 Reducing Input Vector Orders for Power Analysis

3.1 Order-Independence of Power-Extended Transition Systems

Our first improvement in power analysis is achieved by grouping, in equivalence classes, those orders that have the same power characteristics. For such orders, it is sufficient to only consider one member of the equivalence class. Important for such equivalence classes is to result in the same state in order not to affect the functionality of the netlist. This latter problem of determining whether the state reached after applying an input vector in different orders is the same has already been considered in [5] and [6]. Here, we extend those works to also consider the dynamic power consumption. For this purpose, we define a *power-extended vector-based transition system.*

Definition 1. *The* power-extended vector-based transition system $T_p = (S_p, I, \rightarrow_p)$ *of a vector-based transition system* $T = (S, I, \rightarrow)$ *is defined* $S_p = \mathbb{R} \times S$ *and* $w;s \xrightarrow{\vec{\imath}}_p w{+}p(s,s');s'$ *for* $s \xrightarrow{\vec{\imath}} s'$. *The function* $p : S \times S \rightarrow \mathbb{R}$ *computes a weighted number of wire chargings given the source and target states.*

The added first component of the states, a number, is used to sum the weights of the charged wires (which we interpret as the consumed power) during an evaluation. (Our definition indeed allows for weighted wire charging in order to cater for node capacitance in our calculations. However, for presentation purposes throughout the rest of this paper, we assume the weights of all wire chargings to be equal; hence, in the remainder of the paper, the sum of weights denotes the number of wire chargings.) Note that a vector-based transition system does not assume a particular type of the states, so this is still a vector-based transition system.

We initially set the power component (the real number component in the state) to 0, indicating that no chargings have taken place yet. Then, we select two inputs (identified by their position in the input vector, as in the transition

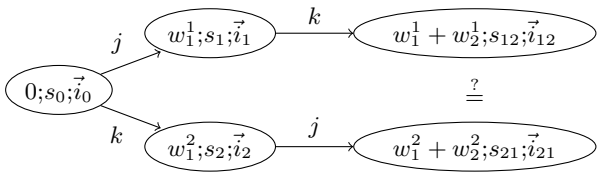


Fig. 1. Evaluation of two input coordinates

system T^I) and change them in both possible orders. Finally we check whether the resulting states for the two orders are equal or not. For example, assume that we initially arrive in state s_0 with an input vector $\vec{i}_0$, denoted by $0;s_0;\vec{i}_0$. Then, we apply the two consecutive input changes, denoted by $[j := v'_j]$ and $[k := v'_k]$ where $j \neq k$, in the two possible orders leading to the two evaluations depicted in Figure 1.

As indicated in Figure 1, it remains to be checked whether the two states $w_1^1 + w_2^1;s_{12};\vec{i}_{12}$ and $w_1^2 + w_2^2;s_{21};\vec{i}_{21}$ are equal or not. First, we note that the input vectors $\vec{i}_{12}$ and $\vec{i}_{21}$ are equal, as they are constructed by updating positions j and k in $\vec{i}_0$ with the same values. Formally, this holds because for $j \neq k$, $\vec{i}_{12} = \vec{i}_0[j := v'_j][k := v'_k] = \vec{i}_0[k := v'_k][j := v'_j] = \vec{i}_{21}$. Thus, we only have to compare the remaining parts of the states. We have already addressed this problem, called *order-independence*, for generic vector-based transition systems in [6]. Checking that the states s_{12} and s_{21} are equal is the same as order-independence, i.e., checking that the order of these two inputs does not affect the functionality. By requiring that $w_1^1 + w_2^1 = w_1^2 + w_2^2$, we additionally require that the order of the two inputs also does not cause different power consumptions.

In this paper, by exploiting the result of [6], we show that for transistor netlists checking order-independence for two inputs is both necessary and sufficient to establish order-independence for traces of full input length. In order to do this we introduce the relation $\twoheadrightarrow$, which denotes traces.

Definition 2. *Let $T = (S, I, \rightarrow)$ be a vector-based transition system with m inputs, let $s;\vec{i} \in S \times I$, and let $\ell = j_1 : \cdots : j_k \in \mathcal{L}_m$.*

We define $\twoheadrightarrow_T \subseteq (S \times I) \times \mathcal{L}_m \times (S \times I)$ as $s;\vec{i} \stackrel{\ell}{\twoheadrightarrow}_T s';\vec{i'}$ iff there exists a state $s_0;\vec{i}_0$ and $1 \leq b \leq m$ such that $s_0;\vec{i}_0 \stackrel{b}{\rightarrow} s;\vec{i} \stackrel{j_1}{\rightarrow} \ldots \stackrel{j_k}{\rightarrow} s';\vec{i'}$.

In the above definition, we additionally require that the initial state is reachable from some other state, which we call *one-step reachability*. This restriction is added to rule out transient initial states that can only occur at boot-up and will never be reached again.

We then call a vector-based transition system T with m inputs *order-independent*, iff for all $\pi, \pi' \in \Pi_m$ it holds that $\stackrel{\pi}{\twoheadrightarrow} = \stackrel{\pi'}{\twoheadrightarrow}$. To relate order-independence, which considers traces of length m, and the check that only considers two inputs, we make use of the following theorem of [6].

Theorem 3 (Theorem 9 in [6]). *Let $T = (S, I, \rightarrow)$ be a deadlock-free vector-based transition system with m inputs having the fixed-point property. Then T is order-independent, iff $j \overset{\downarrow}{\diamond} k$ for all $1 \leq j < k \leq m$.*

In Theorem 3, deadlock-freedom has its intuitive meaning, namely that for every current state and every possible input transition, a next state can be computed. This is the case for the semantics of transistor netlists, our target application, as they always compute a next state for any input vector. The second requirement, the fixed-point property, demands that a reached state is stable, i.e., applying the same input vector twice does not result in a different state from when the input vector is only applied once. It is shown in [6] that vector-based transition systems constructed from a transistor netlist using the algorithm of [2] always have the fixed-point property. Intuitively, this holds because the states are stable and hence cannot distinguish the stable situation from applying the same inputs again. Since an unchanged state means that also the number of wire chargings does not change, this also holds in our power-extended vector-based transition system. Therefore, without mentioning this explicitly in the remainder of this paper, we assume these two hypotheses of Theorem 3 to hold.

The relation $\overset{\downarrow}{\diamond}$, called the *one-step reachable commuting diamond*, relates two input positions $1 \leq j \neq k \leq m$, if for every one-step reachable state $s_0;\vec{i}_0 \in S \times I$, $s_0;\vec{i}_0 \xrightarrow{j} \circ \xrightarrow{k} s_{12};\vec{i}_{12}$ iff $s_0;\vec{i}_0 \xrightarrow{k} \circ \xrightarrow{j} s_{12};\vec{i}_{12}$. This is similar to the situation depicted in Figure 1, assuming that the states s_{12} and s_{21} and their power consumptions are equal.

To summarize, also for power-extended vector-based transition systems we only have to analyze pairs of inputs, instead of complete sequences, to determine order-independence and therefore equivalent power consumption. This drastically decreases the number of required checks from $m!$, i.e., the number of all permutations, to $\frac{m^2-m}{2}$, the number of all pairs of inputs.

3.2 Equivalence Relation on Orders

Full order-independence of a power-extended vector-based transition system would mean that all orders always have the same power consumption. Of course, this is neither expected in any useful transistor netlist, nor is it of much practical relevance. Therefore, we want to create an equivalence relation on orders that groups together those subsets of orders having the same number of wire chargings. This relation, formally defined below, is called *power independence*.

Definition 4. *Let $T = (S, I, \rightarrow)$ be a vector-based transition system with m inputs.*

We define a relation $\leftrightarrow_T$ on $\mathcal{L}_m$, where $\ell \leftrightarrow_T \ell'$ iff the lists are equal except for swapped positions $\ell'[j+1] = \ell[j]$ and $\ell'[j] = \ell[j+1]$, for which the one-step reachable commuting diamond property holds (i.e., $\ell[j] \overset{\downarrow}{\diamond} \ell[j+1]$).

Using relation $\leftrightarrow_T$, we define the equivalence relation $\equiv_T$ on $\mathcal{L}_m$ as the reflexive transitive closure of $\leftrightarrow_T$. If $\ell \equiv_T \ell'$, then we also call ℓ and ℓ' (power-) independent.

Note that the above definition is using a general vector-based transition system. If this is a power-extended one, then we call the relation power-independence, otherwise we only talk about independence of lists. For the power-independence relation, we have the following result, showing that it indeed groups together those orders that have equal (functional and power consumption) behavior.

Lemma 5. *Let $T_p = (S_p, I, \rightarrow_p)$ be a power-extended vector-based transition system with m inputs and let $\pi, \pi' \in \Pi_m$ be power-independent (i.e., $\pi \equiv_{T_p} \pi'$).*

Then for traces $0;s_0;\vec{i}_0 \xrightarrow{\pi}\!\!\!\!\rightarrow_{T_p} w_1;s_1;\vec{i}$ and $0;s_0;\vec{i}_0 \xrightarrow{\pi'}\!\!\!\!\rightarrow_{T_p} w_2;s_2;\vec{i}$ it holds that $w_1 = w_2$ and $s_1 = s_2$.

Proof. Follows by an induction on the number of swapped input coordinates to reach π' from π. □

Thus, to characterize a cell, one only has to choose one representative from each power-independent equivalence class and measure the power consumption for this order. All other orders in this equivalence class will have the same power consumption and hence do not have to be considered.

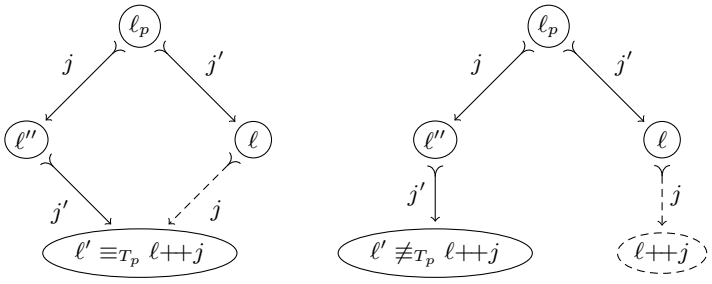
To obtain the different orders that have to be considered, we build the *power-independence DAG* (directed acyclic graph) that enumerates all equivalence classes of the power-independence relation $\equiv_{T_p}$.

Definition 6. *Let $T_p = (S_p, I, \rightarrow_p)$ be a power-extended vector-based transition system with m inputs.*

The power-independence DAG *$G_i = (V_i, \rightarrowtail_i)$ of T_p is defined as $V_i \subseteq \mathcal{L}_m$ with root $\emptyset \in V_i$ and for $1 \leq j \leq m$ with $j \notin \ell$, $\ell \rightarrowtail_i \ell'$ for some unique $\ell' \equiv_{T_p} \ell{+\!+}j$.*

We did not label the edges with inputs in the above-given DAG since the input corresponding to each edge is always the single element by which the two lists of the start and the end node of the edge differ.

To construct the power-independence DAG G_i, we start with the single root of this DAG, $\emptyset$, which indicates that initially no inputs have been considered yet. The construction of the DAG then proceeds in a breadth-first fashion: for each *leaf* ℓ (which is a node without outgoing edges) and every input j that has not yet been considered (i.e., is not in ℓ), an edge is added to that leaf. The target node of this edge is determined by looking at the parent nodes of the currently considered leaf. If there exist a parent node ℓ_p reaching the current node ℓ with input j', a node ℓ' reachable from the parent node ℓ_p with list $j : j'$, and inputs j and j' are exchangeable, i.e., $j \overset{\downarrow}{\diamond} j'$, then the edge is drawn to the existing node ℓ'. This is depicted in Figure 2 (a), where the dashed edge is added. Otherwise, if one of the above conditions is violated (i.e., either the inputs cannot be exchanged or the node ℓ' has not been generated yet), a new node $\ell{+\!+}j$ is created and an edge drawn there. As an example, the case where for all $\ell_p \overset{j:j'}{\rightarrowtail}{}^*_i \ell'$ we have $j \overset{\downarrow}{\not\diamond} j'$ is depicted in Figure 2 (b). There, the dashed edge and the dashed node are added to the DAG. This process finishes at leaves for which the list of



(a) $j \downarrow \diamond\, j'$ (b) $j \not\downarrow \diamond\, j'$

Fig. 2. Construction of the power-independence DAG

considered inputs contains every input exactly once. It can furthermore be shown that the above construction always yields the power-independence DAG. Next, we prove that this DAG exactly distinguishes between the equivalence classes of the power-independence relation $\equiv_{T_p}$.

Theorem 7. *Let T_p be a power-extended vector-based transition system with m inputs and let $G_i = (V_i, \rightarrowtail_i)$ be its power-independence DAG.*

Then, for all orders $\pi_1, \pi_2 \in \Pi_m$, π_1 and π_2 are power-equivalent, iff there exist paths $\emptyset \overset{\pi_1}{\rightarrowtail}{}_i^* \pi$ and $\emptyset \overset{\pi_2}{\rightarrowtail}{}_i^* \pi$ in G_i for some order $\pi \in \Pi_m$.

Proof. To prove the "if" direction, we observe that due to the definition of the power-independence DAG in Definition 6, all nodes on a path, when appending the remaining considered inputs, are power-independent. This directly entails $\pi_1 \equiv_{T_p} \pi \equiv_{T_p} \pi_2$.

The "only-if" direction is proved by an induction over the number of swappings needed to reach π_1 from π_2, cf. Definition 4. If there are none, then $\pi_1 = \pi_2$ and the theorem trivially holds. Otherwise, we can apply the induction hypothesis to π_2 and the order π', resulting from π_1 by undoing the last swapping of j and $j{+}1$. This gives two paths $\emptyset \overset{\pi'}{\rightarrowtail}{}_i^* \pi$ and $\emptyset \overset{\pi_2}{\rightarrowtail}{}_i^* \pi$ in the graph. Since the two swapped positions j and $j+1$ are power-independent, and the rest of the orders π_1 and π' are the same, the two paths induced by these two orders must have the diamond shape due to the requirement in Definition 6, proving that also a path $\emptyset \overset{\pi_1}{\rightarrowtail}{}_i^* \pi$ exists. □

In summary, to determine the power-independent orders of a given power-extended vector-based transition system, we construct its power-independence DAG. Due to the above theorem, the lists contained in the leaves of the power-independence DAG are representatives of the different equivalence classes of orders that have to be considered for power characterization, i.e., only one of these orders has to be measured to obtain the real power consumption of all equivalent orders. Therefore, the number of leaves compared to the number of all possible orders is a measure for the reduction obtained by our method.

4 Selecting Orders to Minimize Power Consumption

The technique introduced in the previous section is useful in characterizing the power consumption of a cell. However, in order to minimize the power consumption, we develop a slightly different technique. Namely, contrary to the previous section, where we identify orders that always have the same dynamic power consumption, we now want to identify orders that are functionally independent (i.e., they do not influence the computation of a next state) but may have different power consumption. Then, by taking the order (one representative order among the equivalent orders) that consumes the least amount of power, the dynamic power consumption of computing the next state can be reduced.

For this purpose, we again define a DAG structure describing the different possible orders, but now we identify nodes that are computing the same next state, i.e., the inputs leading to such a shared node only need to have the diamond property regarding the functionality and not necessarily regarding the power consumption. Furthermore, we add to each node a back-pointer that determines which input leads to less power consumption. Then, by traversing the DAG from some leaf to the root following these back-pointers, one can construct the order that computes the same next state but uses minimal power. Next, we formalize this intuition.

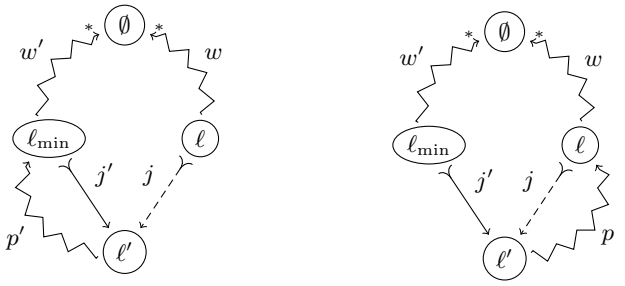
Definition 8. *Given a vector-based transition system $T = (S, I, \rightarrow)$ with m inputs and its power-extended vector-based transition system $T_p = (\mathbb{R} \times S, I, \rightarrow_p)$, we define the* power-sum DAG *$G_s = (V_s, \rightarrowtail_s, \rightsquigarrow_s)$, where $V_s \subseteq \mathcal{L}_m$, $\rightarrowtail_s \subseteq V_s \times V_s$, and $\rightsquigarrow_s \subseteq V_s \times (S \times I \times I) \times V_s$. The root is defined to be $\emptyset \in V_s$.*

The transition relation *$\rightarrowtail_s$ is defined for every $\ell \in V_s$ and every $1 \leq j \leq m$ as $\ell \rightarrowtail_s \ell'$ for some unique ℓ' such that $\ell +\!\!+ j \equiv_T \ell'$.*

The back-pointer relation *$\rightsquigarrow_s$ is defined for every $\ell \in V_s$, $s;\vec{i} \in S \times I$, and $\vec{i'} \in I$ as $\emptyset \not\rightsquigarrow_s^{s;\vec{i};\vec{i'}} \ell$ and, if $\ell \neq \emptyset$, $\ell \rightsquigarrow_s^{s;\vec{i};\vec{i'}} \ell'$ for some unique $\ell' \in V_s$ with $\ell' \overset{j'}{\rightarrowtail}_s \ell$, $\ell' = j_1 : \cdots : j_h$, and $0;s;\vec{i} \xrightarrow{j_1}_p \ldots \xrightarrow{j_h}_p \circ \xrightarrow{j'}_p w;s';\vec{i'}$ for which $w \in \mathbb{R}$ is minimal.*

Note that in the definition of the transition relation of this DAG, we use independence based on equal states, not the extended power-independence which also checks for equal power consumption. Finally, we remark that again labels of edges are left out, but the transition relation $\rightarrowtail_s$ can be understood as labeled by an input position $1 \leq j \leq m$, indicating the added input coordinate that has been considered. This was already made use of in the definition of the back-pointer relation, but this position can again be recovered as the single input coordinate by which the two lists differ.

The construction of the power-sum DAG works similarly to the construction of the power-independence DAG. For it, we use the auxiliary function $w_{\min}$, which assigns to every node and state and input transition the minimal weight that the resulting state can be reached with, i.e., for $\ell \in V_s$ and $s;\vec{i};\vec{i'} \in S \times I \times I$, $w_{\min}(\ell, s;\vec{i};\vec{i'}) = w$ if $0;s;\vec{i} \overset{\ell'}{\twoheadrightarrow}_{T_p} w;s';\vec{i'}$ and w is minimal among all $\ell' \equiv_T \ell$. This



(a) Before adding $\ell \overset{j}{\rightarrowtail}_s \ell'$ (b) $w + p < w' + p'$

Fig. 3. Construction of the back-pointer relation in the power-sum DAG for some state and input vector transition $s;\vec{i};\vec{i'} \in S \times I \times I$

can be efficiently read from the back-pointer relation. To complete the function, we define $w_{\min}(\ell, s;\vec{i};\vec{i'}) = \infty$ if no $\ell' \in V_s$ exists such that $\ell \overset{s;\vec{i};\vec{i'}}{\rightsquigarrow}_s \ell'$.

We start with the root node $\emptyset$ and add nodes in a breadth-first fashion. At each step, for each leaf ℓ of the DAG, we add an edge for every input position $1 \leq j \leq m$ that is not yet contained in ℓ. If there exists a node ℓ' such that $\ell_p \overset{j'}{\rightarrowtail}_s \ell$, $\ell_p \overset{j:j'}{\rightarrowtail}{}^*_s \ell'$, and $j \downarrow\!\!\diamond_T j'$, then the edge $\ell \overset{j}{\rightarrowtail}_s \ell'$ is added. Otherwise, a new node $\ell{+\!+}j$ is added to the DAG, and the edge $\ell \overset{j}{\rightarrowtail}_s \ell{+\!+}j$ is added. This is the same construction that was used for the power-independence DAG G_i, illustrated in Figure 2, only here the commuting diamond property does not take the power consumption into account.

For the back-pointer relation, we use that sub-paths of a path with minimal weight also are of minimal weight, since otherwise a sub-path could be replaced by a smaller one. So, when adding an edge $\ell \overset{j}{\rightarrowtail}_s \ell'$, we define $\ell' \overset{s;\vec{i};\vec{i'}}{\rightsquigarrow}_s \ell$ if $s;\vec{i} \overset{\ell}{\twoheadrightarrow} s_0;\vec{i_0} \overset{j}{\rightarrow} s';\vec{i'}$ and $w_{\min}(\ell, s;\vec{i};\vec{i'}) + p(s_0, s') < w_{\min}(\ell', s;\vec{i};\vec{i'})$, otherwise we leave $\rightsquigarrow_s$ unchanged. Note that if ℓ' is a new node, then always the first case is applied, since the sum is always smaller than ∞.

An illustration of the back-pointer construction is shown in Figure 3, where the dashed edge $\ell \overset{j}{\rightarrowtail}_s \ell'$ is to be added. Initially, we assume that there already is a node $\ell_{\min}$ such that $\ell' \overset{s;\vec{i};\vec{i'}}{\rightsquigarrow}_s \ell_{\min}$, i.e., the power consumption is minimal if taking the minimal path from the root $\emptyset$ to node $\ell_{\min}$, which is assumed to have weight w', and then extending it by considering coordinate j', whose power consumption we assume to be p'. This situation is depicted in Figure 3 (a). Next, the node ℓ is considered. Note that $\ell' \equiv_T \ell{+\!+}j \equiv_T \ell_{\min}{+\!+}j'$, as otherwise the edge $\ell \overset{j}{\rightarrowtail}_s \ell'$ would not be drawn. We assume the weight of the minimal path from the root $\emptyset$ to the node ℓ to be w and the power consumption of the step from ℓ to ℓ' to be p. In case $w + p < w' + p'$, then we have found a new minimal

path for ℓ', thus we update the back-pointer relation as shown in Figure 3 (b). Otherwise, the previous path of the back-pointers is still giving the minimal path even after adding $\ell \overset{j}{\rightarrowtail}_s \ell'$, so in that case the back-pointer relation remains as depicted in Figure 3 (a).

It can be shown that the above construction yields exactly the power-sum DAG G_s of a power-extended vector-based transition system. In the following theorem, it is shown that G_s identifies all orders that lead to the same state and a construction of the order consuming the minimal amount of power is given.

Theorem 9. *Let $T_p = (\mathbb{R} \times S, I, \rightarrow_p)$ be a power-extended vector-based transition system with m inputs and power-sum DAG $G_s = (V_s, \rightarrowtail_s, \rightsquigarrow_s)$. Furthermore, let $\pi, \pi' \in \Pi_m$ be some orders and $s;\vec{i};\vec{i'} \in S \times I \times I$ be some state together with previous and next input vectors.*

If $\emptyset \overset{\pi}{\rightarrowtail}_s^* \pi'$, then a path $\pi' = \ell_m \overset{s;\vec{i};\vec{i'}}{\rightsquigarrow}_s \ldots \overset{s;\vec{i};\vec{i'}}{\rightsquigarrow}_s \ell_0 = \emptyset$ exists and $\pi \equiv_T \pi' \equiv_T \pi''$ for $\pi'' = j_1 : \cdots : j_m \in \Pi_m$ defined by $\ell_r = \ell_{r-1} +\!\!+ j_r$ for all $1 \leq r \leq m$ such that $0;s;\vec{i} \overset{\pi''}{\twoheadrightarrow} w;s';\vec{i'}$ and w is minimal.

Proof. Existence of the back-pointer path and hence of π'' is guaranteed by the (unique) existence of a successor w.r.t. $\rightsquigarrow_s$ for every node that is not the root and since $\ell_r \neq \emptyset$ for every $1 \leq r \leq m$. The property $\pi \equiv_T \pi' \equiv_T \pi''$ directly follows from the definition of the transition relation of G_s. Finally, minimality of w follows from the definition of the back-pointer relation of G_s. □

Given a cell and its power-sum DAG G_s, one can obtain the order consuming the least amount of power for a given state, input vector transition, and order π in which the inputs are to be changed. This works by first traversing the DAG G_s according to the order π, which will result in a leaf π' of the DAG. From the leaf, the back pointer relation is followed upwards to the root, giving another order π'' with $\pi'' \equiv_T \pi' \equiv_T \pi$ that consumes the least amount of power, as shown in Theorem 9. Enforcing this order π'' can for example be done by adding delays, which is also proposed in [7].

5 Implementation

The techniques presented in Sections 3 and 4 were implemented in a prototype tool. This tool first parses a SPICE netlist and builds a symbolic vector-based transition system from it using the algorithm of [2], where states consist of a vector of formulas, computing values from the set $\{\mathsf{0}, \mathsf{1}, \mathsf{Z}\}$. The values $\mathsf{0}$ and $\mathsf{1}$ correspond to the logic values false and true, respectively, and represent an active path from a wire in the netlist to the low and high power rails, respectively. The third value, Z, represents a floating wire that has neither a path to the low nor to the high power rail. As the initial state of the netlist, we allow arbitrary values for all of the wires. The inputs are restricted to the binary values $\mathsf{0}$ and $\mathsf{1}$.

The power consumption of a transition is computed by the function p in Definition 1. In our implementation, this function is defined as $p(\vec{s}, \vec{s'}) =$

$\left|\{1 \leq j \leq n : \vec{s}|_j = 0, \vec{s'}|_j = 1\}\right|$ for a netlist consisting of n wires, i.e., it counts the number of wires that transition from 0 to 1.

Building the power-independence DAG is then performed by first computing the diamond relation for all pairs of inputs (also taking power consumption into account). This is done symbolically using BDDs, requiring a total of $O(n \cdot m^2)$ BDD comparisons for m inputs and n state variables. Here, for every pair of input variables (of which there are $O(m^2)$ many) and every of the n state variables, two BDDs are constructed. The first computes the next state function after applying the two inputs in one order, the second BDD computes the next state function after applying the inputs in the other order. The currently considered pair of inputs has the power-extended diamond relation, if and only if these pairs of BDDs are equal for all state variables and the total number of wire chargings is the same. Finally, we construct the power-independence DAG as described in Section 3.2.

If the power-sum DAG is to be constructed, as described in Section 4, then we first need to compute the functional independence relation for all pairs of inputs. This also requires $O(n \cdot m^2)$ BDD comparisons. Furthermore, we need to keep track of the state to which a list of input coordinates leads, to be able to construct the back-pointer relation. For this purpose, we unroll the symbolic transition relation, i.e., we create a new transition relation that computes, given a starting state and input vector, the state and input vector after changing the inputs in the order of the currently considered node. This is used to create a symbolic formula computing the number of wires charged when adding another input to the list. Among these formulas we finally compute a symbolic minimum that indicates which parent node leads to minimal power consumption.

6 Experimental Results

We applied our technique to reduce the number of considered orders, which was presented in Section 3, and the technique to select an equivalent order that consumes less power, presented in Section 4, to the open-source Nangate Open Cell Library [4]. For each of the contained netlists, the SPICE source was parsed, a transition system created, and the power-independence DAG or power-sum DAG built and traversed to enumerate all equivalence classes. All of our experiments were conducted on a commodity PC equipped with an Intel Pentium 4 3.0 GHz processor and 1 GB RAM running Linux.

6.1 Reducing Input Vector Orders

Our results for reducing the number of considered orders with different functional or power consumption behavior are presented in Table 1, where the first column gives the name of the cell, the second column gives both the number of inputs and the number of wires, the third column the number of all possible orders, and the fourth column shows the number of different equivalence classes returned by our approach together with the time it took to compute these. Finally, the

last column demonstrates the achievable power reduction, to be explained in Section 6.2.

For combinational cells, marked with "(c)" in Table 1, the results show that our approach cannot reduce the number of orders that have to be considered. This is usually due to situations in which wires are in one order first discharged only to be finally charged, whereas evaluating them in another order keeps the wire charged during the whole evaluation. Thus, all possible orders have to be considered during power characterization of these cells.

For sequential cells however, we can observe some larger savings especially for the larger cells. For example, in case of the largest cell in the library, the cell `SDFFRS`, we could reduce the number of orders to consider from 720 to only 288, which is a reduction by 60 %. Especially for sequential cells these savings have an effect, since for these cells the characterization not only has to take the possible input combinations into account, but also the current internal state. Overall, when summing up the absolute number of orders that have to be considered for the sequential cells, we get a reduction by more than 47 %. This is especially advantageous for the large cells, as witnessed by the average of the reduction rates of sequential cells, which is only slightly above 16 %. So especially for large sequential cells with lots of possible orders, our approach can reduce the number of orders that have to be considered significantly.

6.2 Selecting Input Vector Orders

We also evaluated the technique presented in Section 4, which computes the functionally equivalent orders and a path back that uses the minimal amount of power, using the open-source Nangate Open Cell Library [4]. The results are shown in the last column of Table 1, where the number of equivalence classes w.r.t. $\equiv_T$, the average number of maximal differences in wires chargings, and the amount of time for constructing the DAG and computing the result are given.

The results show that for combinational cells all orders lead to the same final state, which is expected as the state is completely determined by the new input values. For the sequential cells we observe that not all orders lead to the same final state, as there are multiple leaves in the power-sum DAG. This happens because the computation can depend on internally stored values, which might have different values when applying the input changes in different orders.

We illustrate the selection of orders by means of an example. Consider the scan logic of the cell `SDFFRS` (which is also the same in the cells beginning with `SDFF`), which is a multiplexer (mux) that selects, based on the value of the scan enable signal, between the data input and the scan input. In case the scan enable signal changes from `0` to `1` and the data input changes, then the power-sum DAG tells us that it is more power-efficient to first change the scan enable signal and then change the data input, than vice versa. This can be explained intuitively by the fact that while the scan enable signal is `0`, the mux is transparent to changes in the data input, so also wires connected to transistors controlled by the mux output are affected. This is not the case anymore if we first change the scan enable to `1`, so that the change in the data input cannot be observed at the

Table 1. Results for the Nangate Open Cell Library

Cell	**#I / W**	**#Π$_m$**	**|G$_i$| / t [s]**	**|G$_s$| : Avg / t [s]**
AND2 (c)	2 / 3	2	2 / 0.37	1 : 2.5 / 0.37
AND3 (c)	3 / 4	6	6 / 0.46	1 : 3.5 / 0.47
AND4 (c)	4 / 5	24	24 / 0.60	1 : 4.5 / 0.66
AOI211 (c)	4 / 4	24	24 / 0.62	1 : 4.0 / 0.64
AOI21 (c)	3 / 3	6	6 / 0.44	1 : 2.5 / 0.45
AOI221 (c)	5 / 5	120	120 / 0.97	1 : 7.0 / 1.14
AOI222 (c)	6 / 6	720	720 / 1.79	1 : 9.5 / 5.57
AOI22 (c)	4 / 4	24	24 / 0.68	1 : 5.0 / 0.66
BUF (c)	1 / 2	1	1 / 0.33	1 : 0.0 / 0.27
CLKBUF (c)	1 / 2	1	1 / 0.25	1 : 0.0 / 0.29
CLKGATETST	3 / 13	6	6 / 0.68	4 : 3.7 / 0.71
CLKGATE	2 / 11	2	2 / 0.54	2 : 0.0 / 0.54
DFFRS	4 / 24	24	24 / 1.20	12 : 1.1 / 1.82
DFFR	3 / 19	6	4 / 0.78	4 : 0.0 / 0.90
DFFS	3 / 19	6	6 / 0.82	4 : 1.0 / 0.90
DFF	2 / 16	2	2 / 0.63	2 : 0.0 / 0.68
DLH	2 / 9	2	2 / 0.50	2 : 0.0 / 0.52
DLL	2 / 9	2	2 / 0.53	2 : 0.0 / 0.52
FA (c)	3 / 14	6	6 / 0.71	1 : 3.0 / 0.76
HA (c)	2 / 8	2	2 / 0.46	1 : 1.0 / 0.48
INV (c)	1 / 1	1	1 / 0.24	1 : 0.0 / 0.25
MUX2 (c)	3 / 6	6	6 / 0.52	1 : 4.0 / 0.53
NAND2 (c)	2 / 2	2	2 / 0.35	1 : 1.5 / 0.35
NAND3 (c)	3 / 3	6	6 / 0.44	1 : 2.5 / 0.44
NAND4 (c)	4 / 4	24	24 / 0.58	1 : 3.5 / 0.61
NOR2 (c)	2 / 2	2	2 / 0.35	1 : 1.5 / 0.36
NOR3 (c)	3 / 3	6	6 / 0.47	1 : 2.5 / 0.45
NOR4 (c)	4 / 4	24	24 / 0.58	1 : 3.5 / 0.63
OAI211 (c)	4 / 4	24	24 / 0.59	1 : 4.0 / 0.64
OAI21 (c)	3 / 3	6	6 / 0.47	1 : 2.5 / 0.45
OAI221 (c)	5 / 5	120	120 / 1.07	1 : 7.0 / 1.22
OAI222 (c)	6 / 6	720	720 / 1.80	1 : 9.5 / 5.61
OAI22 (c)	4 / 4	24	24 / 0.62	1 : 5.0 / 0.66
OAI33 (c)	6 / 6	720	720 / 1.77	1 : 8.0 / 4.79
OR2 (c)	2 / 3	2	2 / 0.37	1 : 2.5 / 0.38
OR3 (c)	3 / 4	6	6 / 0.46	1 : 3.5 / 0.47
OR4 (c)	4 / 5	24	24 / 0.59	1 : 4.5 / 0.66
SDFFRS	6 / 30	720	288 / 2.99	48 : 6.4 / 13.01
SDFFR	5 / 25	120	96 / 1.61	16 : 6.4 / 3.07
SDFFS	5 / 25	120	36 / 1.49	16 : 6.0 / 2.77
SDFF	4 / 22	24	18 / 1.04	8 : 6.0 / 1.33
TBUF (c)	2 / 5	2	2 / 0.38	1 : 3.0 / 0.39
TINV (c)	2 / 4	2	2 / 0.39	1 : 3.0 / 0.38
TLAT	3 / 12	6	6 / 0.63	2 : 3.0 / 0.67
XNOR2 (c)	2 / 5	2	2 / 0.41	1 : 2.0 / 0.41
XOR2 (c)	2 / 5	2	2 / 0.44	1 : 2.0 / 0.40

output of the mux. In case of the cell `SDFFRS`, choosing the first order can cause 7 more wires to be charged.

Note that some correlation exists between the size of an equivalence class and the achievable power reduction: The more possible orders there are the more likely it is that another equivalent order with less power consumption exists. This can also be observed in results of Table 1, where the largest differences occur for combinational cells, which always have exactly one equivalence class.

7 Conclusions

This paper presented a technique to group together orders of applying input vectors which affect neither the functional behavior nor the power consumption. Such a technique is useful for power characterization, where it is sufficient to only choose one order of each of these equivalence classes, as all other elements exhibit the same behavior. Additionally, we presented a technique to select an order that uses the minimal power among functionally equivalent orders. Such a technique is useful when the order can be controlled, e.g., by means of the addition of delays as proposed in [7]. Then, one can force the evaluation to use an order consuming the minimal amount of power, without affecting the result of the computation. Both techniques were evaluated on the Nangate Open Cell Library and provided reductions within reasonable amounts of time.

Acknowledgments

We would like to thank (in alphabetical order) Alan Hu, Hamid Pourshaghaghi, and Shireesh Verma for their valuable input on this paper's topic. Also, we would like to thank the anonymous referees for their fruitful remarks.

References

1. A. Bogliolo, L. Benini, and B. Ricco. Power Estimation of Cell-Based CMOS Circuits. In *Proc. of DAC'96*, pages 433–438. ACM, 1996.
2. R. Bryant. Boolean Analysis of MOS Circuits. *IEEE Transactions on Computer-Aided Design*, 6(4):634–649, 1987.
3. M. Huang, R. Kwok, and S.-P. Chan. An Empirical Algorithm for Power Analysis in Deep Submicron Electronic Designs. *VLSI Design*, 14(2):219–227, 2000.
4. Nangate Inc. Open Cell Library v2008_10 SP1, 2008. Downloadable from `http://www.nangate.com/openlibrary/`.
5. M. Raffelsieper, M.R. Mousavi, J.-W. Roorda, C. Strolenberg, and H. Zantema. Formal Analysis of Non-Determinism in Verilog Cell Library Simulation Models. In *Proc. of FMICS'09*, volume 5825 of *LNCS*, pages 133–148. Springer, 2009.
6. M. Raffelsieper, M.R. Mousavi, and H. Zantema. Order-Independence of Vector-Based Transition Systems. In *Proc. of ACSD'10*, pages 115–123. IEEE, 2010.
7. A. Raghunathan, S. Dey, and N. K. Jha. Glitch Analysis and Reduction in Register Transfer Level Power Optimization. In *Proc. of DAC'96*, pages 331–336. ACM, 1996.
8. W.-Z. Shen, J.-Y. Lin, and J.-M. Lu. CB-Power: A Hierarchical Cell-Based Power Characterization and Estimation Environment for Static CMOS Circuits. In *Proc. of ASP-DAC'97*, pages 189–194. IEEE, 1997.