A Fast Analytical Approach for Static Power-Down Mode Analysis

Michael Zwerger, Pantelis-Rafail Vlachas, Helmut Graeb
Institute for Electronic Design Automation
Technische Universität München, Munich, Germany

***Abstract*—In this paper, a new method for static analysis of the power-down mode of analog circuits is presented. Floating nodes are detected. The static node voltages are estimated. It can be verified that no current is flowing. The method is based on circuit structure. No numerical simulation is needed. The presented approach solves an integer constraint program. Experimental results show a speed-up of factor 2.5 compared a state-of-the-art voltage propagation algorithm. Furthermore, the presented analytical problem formulation enables fast implementation of the method using a constraint programming solver.**

I. Introduction

Driven by the need for power-efficiency, e.g., to avoid heat dissipation or to enable long battery service, modern integrated circuits are equipped with power-down modes. This also affects the analog circuit blocks. In analog power-down mode the supply voltage is usually not turned off or gated in order to avoid head and foot transistors. Those would have to be sized big in order to carry the total current. Furthermore, they would cost voltage swing. In order to avoid these drawbacks, a typical implementation of the power-down mode is to add small power-down transistors all over the design, to switch off currents and clamp all nets to a defined voltage [1], [2]. Examples for power-down circuits can be seen in the circuits shown in Figs. 2 and 3. Our examples follow the convention that the digital signal n_{pwd} is vdd in power-down mode and gnd in power-up or normal operation mode. The intended behavior of power-down mode is that no currents are flowing. Consequently, many devices are operating in sub-threshold region. This can cause floating nodes, i.e., nodes that only have high resistive connections to the other nets. A proper implementation of the power-down mode should avoid floating nodes, as they can cause problems that are very hard to debug and sometimes even impossible to find with numerical simulation. Furthermore, floating nodes can cause reliability problems due to transistor aging [3].

The voltages of floating nodes depend heavily on the sub-threshold behavior, which is often not modeled as accurately as in the linear or triode region. Moreover, the voltage can heavily depend on parasitic current paths and, to make things even worse, on manufacturing tolerances. As a result, numerical simulations are not trustworthy in power-down mode [2]. In order to enable a reliable analysis of the power-down mode, an alternative approach is required.

This work has been funded in part by the German Federal Ministry of Education and Research in the joint research project RESIST under support code 16ES0307.

In this paper, we are presenting a new analytical approach to detect floating nodes and reliably determine the static node voltages in power-down mode. The presented method solves the following analysis task for the power-down mode of analog circuits:

- Detect all floating nodes.
- Estimate all steady-state voltages of internal nodes using circuit structure only.

By using circuit structure only, the problems of numerical simulation described above can be avoided. Additionally, if no floating nodes are detected, it is verified, that no currents are flowing in power-down mode. In order to solve the task, an optimization problem is formulated which is solved by a constraint programming solver.

The paper is structured as follows: Section II discusses the contributions of this work. In section III an integer constraint program is formulated to solve the analysis task. Section IV presents the experimental results. Section V concludes the paper.

II. State of the Art and Contributions

In order to solve the task described in the introduction, an approach relying on numerical simulation could be considered. Modern simulators apply techniques like gmin-stepping [4] to achieve convergence in the presence of floating nodes. However, relying on correct models for sub-threshold behavior is still not possible in many cases. Furthermore, numerical simulation is computationally more expensive. Fast computation is particularly important, if the algorithm is used for analyzing circuits with many digital inputs and for all possible combinations of input voltages. This is, e.g., done in [3] to find stress conditions in power-down mode.

A dedicated method for detecting floating nodes and currents in power-down mode without using numerical simulation is described in [2] and [1]. The methods use a static voltage propagation approach. Compared to those methods, the analytical approach in this work provides the following advantages:

- Speed-up of computation times by factor 2.5 compared to [2].
- Support of all common analog devices, i.e., mosfet, bipolar, diode, resistor compared to [1].
- The analytical problem formulation allows fast implementation of the method by using an integer constraint program solver.

978-1-5090-0246-7/15/$31.00 ©2015 IEEE

TABLE I. STATE $\sigma(c)$ OF psw AND nsw CONNECTIONS

$\tau(c)$ \ $u_{s(c)}$	gnd	$floating$	vdd	
nsw	off	$floating$	on	$\}\sigma(c)$
psw	on	$floating$	off	

III. INTEGER CONSTRAINT SATISFACTION PROBLEM

The analysis problem described in the introduction will be modeled as an integer constraint satisfaction problem [5], [6] in the following. Therefore, the set of nets of the circuit is defined as $N = \{n_1, n_2, ...n_n\}$. The verification task requires the assignment of a voltage to each net of the circuit $n_i \in N$ taking the voltages of the supply nets as input. Thus, for each net n_i a voltage variable u_{n_i} is defined. The vector of voltages variables is

$$\mathbf{u} = [u_{n_1}, u_{n_2}, ..., u_{n_n}]^T. \tag{1}$$

The domain of the variables, $u_{n_i} \in D$, is defined as

$$\begin{aligned} D = \{0,1,2\}, \text{ where } & 0 \Leftrightarrow floating \\ & 1 \Leftrightarrow gnd \\ & 2 \Leftrightarrow vdd \end{aligned} \tag{2}$$

For static analysis, the following optimization problem is solved:

$$\min_{\mathbf{u}} \; Cost(\mathbf{u}) \quad s.t. \quad CircuitConstr(\mathbf{u}) = 1 \tag{3}$$

The cost function is defined as the sum of all voltage variables:

$$Cost(\mathbf{u}) = \sum_{i=1}^{n} u_{n_i} \tag{4}$$

The constraints $CircuitConstr(\mathbf{u})$ represent the devices of the circuit as well as known external voltages, e.g., at supply nets. The constraints are described in detail in the following sections. According to the equivalence between integers and voltage values defined in (2), the value $floating$ is represented by integer value zero, while gnd and vdd are assigned higher values one and two. Thereby, priorities between the voltage values are established by the cost function. For example, a circuit with no devices and no supplies, i.e., a circuit consisting of unconnected nets would have no constraints. In this example, the solution of (3) would be contain only $floating$ nets, as this minimizes the cost function. If devices and external voltages are added, the voltage variables will be forced to take values with a higher cost by the corresponding constraints. The constraints are described in the following.

A. Definitions

Firstly, the supply voltages and external signals like the digital power-down signal result in constraints. These external voltages are given as input to the method. Therefore, the following definitions are made: The set of nodes which have an input voltage value vdd is defined as N_{vdd}. The set of nodes with gnd input voltage value is N_{gnd}.

Secondly, the devices in the circuit and their connections result in constraints. Each device, e.g., each MOSFET transistor can be directly mapped to set of constraint equations that depend on the voltage variables of the nets connected

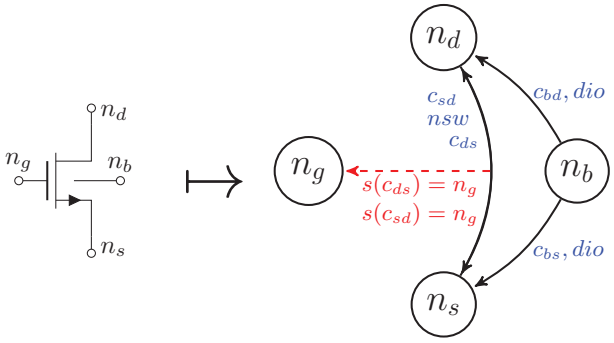

Fig. 1. Example: Connections for a NMOS transistor [2]

to the device. For a clear and systematic description of the constraints, the terminology of the graph model described in [2] can be used. The model describes the static behavior of the circuit. Each edge in the graph corresponds to a constraint. An example for a graph model of an NMOS transistor is given in Fig. 1. In the following, we will define the directed graph $G(N, C, \phi^+, \phi^-, \tau)$, with

$$N = \{n_1, n_2, ...n_n\} \quad \text{(set of nets)} \tag{5}$$

$$C = \{c_1, c_2, ...c_n\} \quad \text{(set of directed connections)} \tag{6}$$

and the incidence mappings

$$\phi^+ : C \to N, \phi^+(c) = n \Leftrightarrow c \text{ starts at } n \tag{7}$$

$$\phi^- : C \to N, \phi^-(c) = n \Leftrightarrow c \text{ ends at } n. \tag{8}$$

Each connection can either be a diode-type connection (dio), a n-switch connection (nsw) or a p-switch connection (psw). Consequently, the type mapping τ is defined as:

$$\tau : C \to T, c \to \tau(c) \quad \text{where} \quad T = \{dio, nsw, psw\} \tag{9}$$

Connections of type nsw and psw model the behavior of drain-source or collector-emitter connections in transistors. Connections of type dio are used to model, e.g., diodes and resistors. Each connection is assigned a state $\sigma(c)$. The state of a connection is either on, off or $floating$ depending on its conductivity. Diode connections are always on. The conductivity of nsw and psw connections is controlled by the voltage variable of a gate or base net. The gate or base which controls the state of c can be obtained with the function $s(c)$. Table I shows $\sigma(c)$.

The definitions are used in Fig. 1 showing the graph model for an NMOS transistor. The bulk node is connected with drain and source with diode connections (c_{bd}, c_{bs}) to model the bulk diodes. Furthermore, the channel is modeled with two connections (c_{sd}, c_{ds}) between drain and source of type $\tau(c_{ds}) = \tau(c_{sd}) = nsw$. These connections are switched by the gate node, i.e., $s(c_{ds}) = n_g$ and $s(c_{sd}) = n_g$. According to Table I, connections c_{ds} and c_{sd} of type nsw are on, when the gate node n_g is vdd, and off otherwise. Similar graph models are defined for other devices like diodes, resistors, and bipolar transistors in [2].

The constraint $CircuitConstr(\mathbf{u})$ from equation (3) will be defined in the following. We will first define constraint equations for the input voltages and then constraint equations for the connections, i.e., for the devices of the circuit.

TABLE II. VALID VOLTAGES ACROSS c WITH $\sigma(c) = on$

state	$\phi^+(c)$	$\phi^-(c)$	valid?
1	vdd	vdd	✓
2	vdd	gnd	X
3	vdd	$floating$	X
4	gnd	vdd	✓
5	gnd	gnd	✓
6	gnd	$floating$	✓
7	$floating$	vdd	✓
8	$floating$	gnd	X
9	$floating$	$floating$	✓

$CircuitConstr(\mathbf{u})$ is a conjunction of those equations, as defined later in (18).

B. Input Voltage Constraints

The constraints for the input voltages are defined as follows:

$$GndInput(u_n) \;\Leftrightarrow\; (u_n = gnd) \tag{10}$$

$$VddInput(u_n) \;\Leftrightarrow\; (u_n = vdd) \tag{11}$$

The constraints (10) and (11) are set for every node in N_{gnd} and N_{vdd}, respectively.

C. Connection Constraints

The connection constraints come from low-impedance connections of the circuit which correspond to connections with state on. As shown in Table II, not all voltage combinations for start and end node of a connection with state $\sigma(c) = on$ are valid. If the start-node $\phi^+(c)$ of an edge c has the value vdd, the voltage value is propagated in the direction of the edge and thus the end-node $\phi^-(c)$ is connected to vdd. As a result $floating$ and gnd are not allowed for $\phi^-(c)$ (states 2, 3). On the other hand, if the end-node $\phi^-(c)$ is gnd then the start-node is connected to gnd, considering that gnd is propagating in the opposite direction. Consequently, $floating$ and vdd are not allowed for $\phi^+(c)$ (states 2, 8). State 2 represents a short-circuit between vdd and gnd. High-impedance connections (state off) do not impose any constraints, since no relation between the connecting nodes holds and all different voltage value combinations are valid.

The constraints from Table II are formulated analytically in the following. Propagation of vdd in direction of a connection and propagation of gnd counter the direction of a connection are described by the following logic predicates:

$$PropVdd(c) \;\Leftrightarrow\; \Big(u_{\phi^+(c)} = vdd \rightarrow u_{\phi^-(c)} = vdd \Big) \tag{12}$$

$$PropGnd(c) \;\Leftrightarrow\; \Big(u_{\phi^-(c)} = gnd \rightarrow u_{\phi^+(c)} = gnd \Big) \tag{13}$$

(12) and (13) can be combined to the propagation constraint for connections with state on:

$$Prop(c) \;\Leftrightarrow\; \Big(PropVdd(c) \wedge PropGnd(c) \Big) \tag{14}$$

The Propagation Constraint $Prop(c)$ can be used to formulate constraints for dio, nsw and psw connections. This will be done in the following. The state of connections of type dio is always on. Consequently, the constraint for diode connections is defined as:

$$ConstrDio(c) \;\Leftrightarrow\; Prop(c) \tag{15}$$

TABLE III. COMPUTATION TIMES

circuit	[2]	this work	speed-up
Oscillator	$3.16\,ms$	$1.147\,ms$	2.75
Miller	$3.44\,ms$	$1.25\,ms$	2.74
LVDS Driver	$85.38\,ms$	$34.16\,ms$	2.49

The state of connections of type nsw or psw is controlled by the base or gate net, namely by $s(c)$. The state is shown in Table I. For the nsw type, the connection is on when the switch node $s(c)$ has the voltage value vdd, otherwise the connection is off. Thus, the constraint for nsw connections is defined as follows:

$$ConstrNsw(c) \;\Leftrightarrow\; \Big(u_{s(c)} = vdd \rightarrow Prop(c) \Big) \tag{16}$$

Similarly, a psw connection is on when $s(c)$ has the voltage value gnd. Therefore, the constraint for psw connections is:

$$ConstrPsw(c) \;\Leftrightarrow\; \Big(u_{s(c)} = gnd \rightarrow Prop(c) \Big) \tag{17}$$

The node constraints and the connection constraints can be combined to the overall constraints:

$$\begin{aligned} CircuitConstr(\mathbf{u}) \;\Leftrightarrow\; \Bigg(& \bigwedge_{n \in N_{vdd}} VddInput(u_n) \;\bigwedge \\ & \bigwedge_{n \in N_{gnd}} GndInput(u_n) \;\bigwedge \\ & \bigwedge_{c \in C, \tau(c) = dio} ConstrDio(c) \;\bigwedge \\ & \bigwedge_{c \in C, \tau(c) = nsw} ConstrNsw(c) \;\bigwedge \\ & \bigwedge_{c \in C, \tau(c) = psw} ConstrPsw(c) \Bigg) \end{aligned} \tag{18}$$

Optimization problem (3) is solved using the branch-and-bound solver of the Gecode C++ Library [7], [5], [6].

IV. EXPERIMENTAL RESULTS

In this section, the method is demonstrated for three example circuits. Two of them are shown in Figs. 2 and 3. The figures show a Miller operational amplifier and an oscillator circuit. Both circuits have a power-down circuit to turn off bias currents with a digital signal n_{pwd}. Additionally, the method was tested on an industrial LVDS driver circuit with 229 transistors. The results are described in the following.

The calculated node voltages and execution times are compared to a state-of-the-art voltage propagation algorithm [2]. The voltage propagation and the constraint programming method described in this paper are implemented in C++ using a common basis of data structures, e.g., to hold the circuit in memory. In order to exclude as many implementation effects as possible, the time measurements where carried out from a common execution point after all input data is read in until the results are available in memory. For the static voltage propagation, the measured time includes the steps of building the graph and executing the propagation. For the constraint propagation, the measured time includes the steps of setting up the constraint program using the C++ interface of Gecode and executing the solver.

TABLE IV. RESULTS FOR THE MILLER AMPLIFIER

net:	n_1	n_2	n_3	n_{in}	n_{out}	n_{bias}	$\overline{n_{pwd}}$
voltage:	gnd	gnd	gnd	vdd	vdd	vdd	gnd

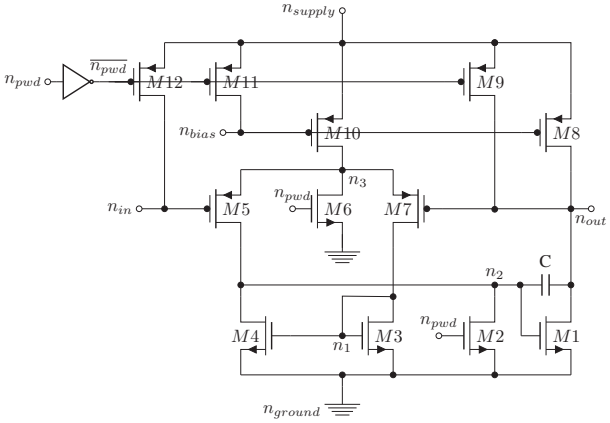

Fig. 2. Miller amplifier

A. Miller Amplifier

The Miller amplifier has five power-down transistors ($M2$, $M6$, $M9$, $M11$, $M12$). They are connecting all nets to n_{supply} or n_{gnd} during power-down mode. The voltage values of nets n_{supply}, n_{gnd} and the power-down signal net n_{pwd} are given as an input. The voltage of n_{pwd} and n_{supply} is vdd. The voltage of n_{ground} is gnd. Solving optimization problem (3) the voltages of all other nodes can be computed. Table IV shows the result. The resulting voltage values are the same as calculated by [2]. Using the result, it can be verified that there are no floating nodes and no current is flowing.

B. Oscillator Circuit

Fig. 3 shows an oscillator with three power-down transistors ($M1$, $M4$, $M11$). The input voltages are vdd for n_{supply} and n_{pwd}, as well as gnd for n_{ground}. The result is shown in the row "with $M4$" of Table V. It is verified that no floating nodes exist and no current is flowing. All voltage values are the same as calculated by [2].

In order to illustrate the capability to discover implementation errors of the power-down mode, we have conducted two additional experiments for the oscillator. Firstly, we have removed the power-down transistor $M4$. This is a design error due to floating nodes n_3, n_4 and n_{out} which can cause potential current flow in the two inverters $M8$, $M5$ and $M7$, $M6$. The result is shown in the row "$M4$ removed" of Table V. The floating nodes have been detected correctly. The result is the same as calculated by [2]. Secondly, we introduced a design-error by connecting the non-inverted power-down signal to the gate of power-down transistor $M1$. The modification is indicated by the dotted connection in Fig. 3. As $M1$ is now conducting in power-down mode, there is now current flowing. The current path in the circuit over $M1$ leads to a contradiction of the constraints in $CircuitConstr(\mathbf{u})$, i.e., problem (3) has no solution. The unwanted current is detected and the situation can, e.g., be examined by numerical simulation.

TABLE V. RESULTS FOR THE OSCILLATOR CIRCUIT

	n_1	n_2	n_3	n_4	n_{out}	$\overline{n_{pwd}}$
with $M4$	vdd	gnd	gnd	vdd	gnd	gnd
$M4$ removed	vdd	gnd	$floating$	$floating$	$floating$	gnd

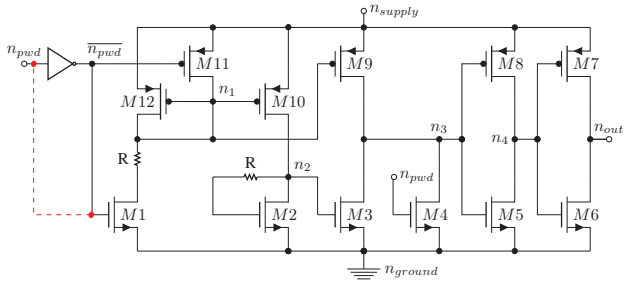

Fig. 3. Oscillator Circuit [1]

C. LVDS Driver

We have solved problem (3) for an industrial LVDS driver circuit with 120 nets and 229 transistors and one power-down mode. The results discovered several floating nodes which were fixed by adding additional power-down switches. For the improved circuit, we could verify that there are no floating nodes and no unwanted current in power-down mode.

D. Speed-up

The computation times for voltage propagation [2] and the presented analytical method are summarized for all three circuits in Table III. It can be seen that the runtimes for all three circuits result in a speed-up factor of approximately 2.5 compared to the state-of-the-art voltage propagation algorithm.

V. CONCLUSION

In this paper, a new analytical formulation of the voltage propagation task as a constraint program has been presented. This constraint program formulation allows the use of generic constraint program solvers for practical implementation. The method is based on circuit structure. No numerical simulation is needed. The results have been compared to a state-of-the-art voltage propagation algorithm. It can be seen that there is a speed-up of 2.5. For all example circuits, it could be verified, that there are no floating nodes and no currents flowing.

REFERENCES

[1] S. Blieck and E. Janssens, "Software Check for Power-Down Mode of Analog Circuits," in *Proceedings ESSCIRC*, Sep. 1996.

[2] M. Zwerger and H. Graeb, "Verification of the power-down mode of analog circuits by structural voltage propagation," *Analog Integrated Circuits and Signal Processing*, Aug. 2013.

[3] ——, "Detection of Asymmetric Aging-Critical Voltage Conditions in Analog Power-Down Mode," in *Design, Automation and Test in Europe (DATE)*, Mar. 2015.

[4] E. Yilmaz and M. Green, "Applying globally convergent techniques to conventional dc operating point analyses," in *Simulation Symposium, 1999. Proceedings. 32nd Annual*, 1999.

[5] B. Roman, "Constraint propagation and backtracing-based search," in *First International summer school on CP*, Jun. 2005.

[6] P. v. B. Francesca Rossi and T. Walsh, *Handbook of Constraint Programming*, Elsevier, Ed., 2006.

[7] G. T. Christian Schulte and M. Z.Lagerkvist, *Modeling and Programing with Gecode*, 2013. [Online]. Available: http://www.gecode.org/doc/4.2.1/MPG.pdf