Basic Analog IP Characterization

Presenter: David Chen
Authors: Peter H. Chen, Harrison Liu, Peter Pong,
Jim Wang, K.C. Wu, and Alvin Chen

Faraday Technology Corporation
No. 5, Li-Hsin Rd. III, Hsinchu Science Park
Hsinchu City, Taiwan 300, ROC
{pchen, hcliu, pong, jhwang, kcwu, alvin}@faraday-tech.com

ABSTRACT

This paper proposes a fast methodology for basic characterization (input cap, output cap, and static power characterization) of analog IPs. An improved graph approach is proposed to reduce the run time and keep the accuracy of the input/output cap characterization. An Automatic Circuit Classification (ACC) and Knowledge-Based (KB) methodology for static power characterization is disclosed, which decreases the amount of test benches required from some million strong to within hundreds.

Table of Contents

Table of Figures

1.0 Introduction

The roles of mixed signal IPs are becoming critical in ASIC design. Mixed signal IPs can be divided into digital and analog parts. The characterization data of digital IPs can be directly read from the digital library; however, analog IPs are custom made and need to be characterized once the design is completed. Since analog IPs are getting bigger and bigger, the simulation time can take several weeks to months. In order to meet the time to market, it becomes imperative to reduce the IP characterization time. Previous literature [4, 5, 6] proposes a non-generic algorithm for partial circuit extraction. This method is based on two kinds of device classification and two levels of extraction for transmission gate and inverter. The method however is not robust enough to handle all IPs. Some IPs with resistors and feedback loops will encounter accuracy problems in the course of QA and verification. In an endeavor to achieve perfect device extraction able to solve the foregoing problem, this paper presents a generic graph approach for partial device extraction of a flattened SPICE netlist with lumped capacitance. The generic graph extraction algorithm can be used for any levels of circuit extraction by given input pins and output pins. Our study shows two-level circuit extraction can achieve an accuracy of 99-100 % in comparison to whole-chip simulation. The simulation time is reduced from weeks/months to seconds/minutes.

Static power characterization is another issue in IP characterization. The static power characterization is state dependent. With the continual growth of pin count of IPs, the number of states grows exponentially. IPs with 20-50 input pins are very common now. The states for a 20-pin IP are around 3 million with three corners (2 to power of 20 multiplied by 3). To generate 3 million test benches is impractical. Hence, an automatic circuit classification (ACC) and knowledge-based (KB) methodology is proposed to minimize the growth of states thus reducing the number of test benches required. By applying this methodology, we can reduce the millions of test benches to less than hundreds. As the Liberty Parser provides *don't care* and fixed voltage capabilities, the static power can be expressed in Liberty.

The design has been used for methodology verification of more than 2000 IPs implemented in 0.25-micron, 0.18-micron, 0.13-micron, and 65-nanometer process technologies with both regular and lower-leakage power. It proves to be a well designed methodology at Faraday.

2.0 Overview

2.1 Block Diagram

Figure 1 shows the overall diagram for Basic IP characterization.

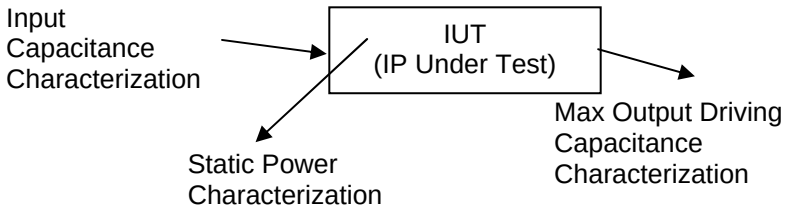

Figure 1 - Block Diagram of Input Cap, Output Cap, and Static Power Characterization

2.2 Overall Flow of Basic IP Characterization

Figure 1 shows the overall flow of Basic IP Characterization.

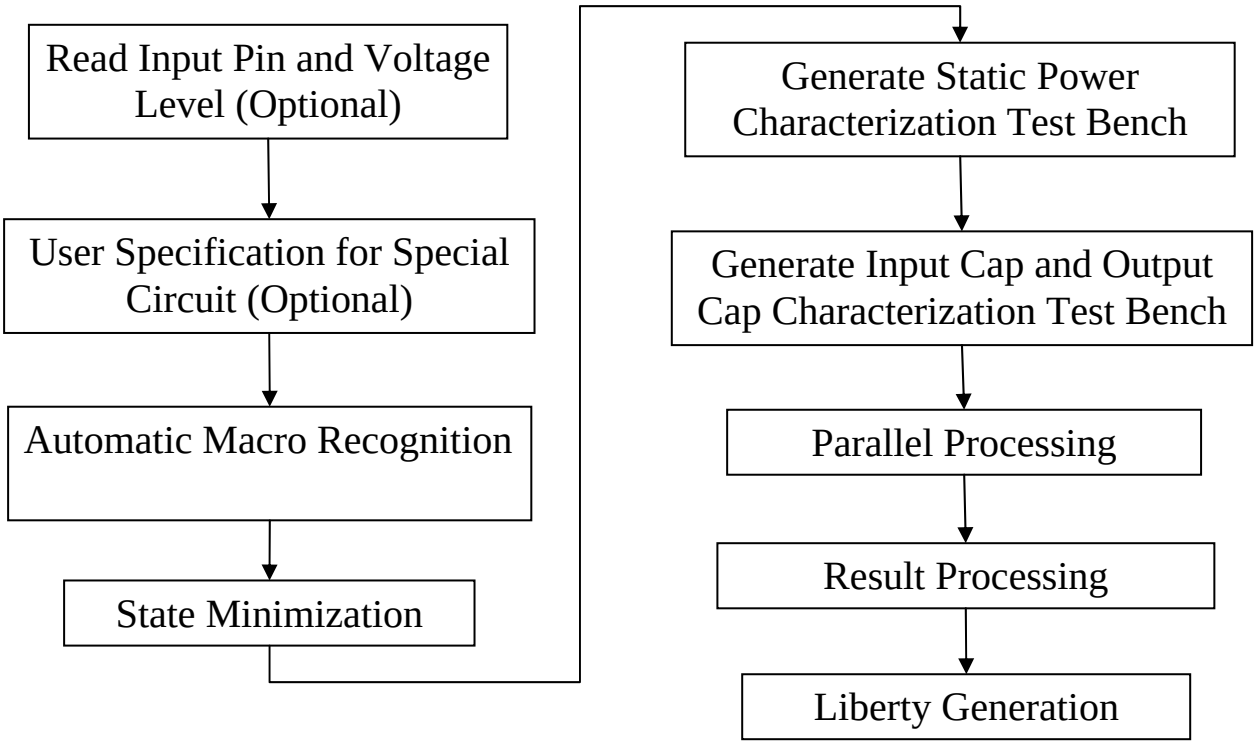

Figure 1 - Overall Flow of Basic IP Characterization

3.0 Input and Output Cap Characterization

This section describes a generic partial circuit extraction algorithm that traverses the analog SPICE code of an IP. After the SPICE code extraction, the number of devices greatly drops. Take DAC for example, the number of devices is reduced from 500K to 1K. The algorithm is used for SPICE simulation code reduction for both Input and Output Cap characterization. Due to this reduction, the simulation time is cut from months down to seconds/minutes. In previous literature [4, 5, 6], the partial extraction algorithm recognizes only the double inverter and

transmission gate. For input pins connected with resistors or capacitors, this algorithm is not flexible. Therefore, it is critical to craft a robust and generic algorithm to do partial circuit extraction without device type specification and to recursively traverse both parallel and serial circuits. To tie high/low floating nodes, another generic algorithm is used to merge the multiple levels of parallel/serial circuits into a single loop of circuit. A tie-up/down is then performed for the floating nodes.

3.1 Generic Graph Algorithm for Circuit Extraction

A generic graph algorithm is used for partial circuit extraction of both input cap and output cap. The extracted circuit will be used for characterization (SPICE simulation) instead of full IP characterization. It is SPICE Model based and is capable of characterization of any corners (for example, Best Case (BC), Typical Case (TC), or Worst Case (WC) from Layout Parasitic Extraction (LPE) which is based on dimension of layout approximation and can be used in Typical Corner (TC).

To start using the algorithm, select input/output pin and specify the level of extraction. The algorithm then extracts partial circuits from the specified input/output pin and at the specified level count of MOS Gate, Power, or Ground.

3.2 Definition of Level of Circuit Extraction

The gate is used to differentiate the level of circuit extraction. Each time the algorithm visits the gate of MOS devices (NMOS or PMOS), the level will increase by one. A MOS device contains three terminals Gate, Source, and Drain (Bulk is not used in the netlist traversal). Diodes, resistors, and capacitors which do not contain the MOS Gate are not counted for the circuit level, *i.e.*, they will be extracted and will not increase the level of circuit extraction. While it is noted that MOS devices may be part of the Transmission Gate, INV, NAND, NOR, ESD, etc, the proposed algorithm is only concerned with the MOS Gates for level of circuit extraction, the type of devices being of no interest here. This algorithm is generic as opposed to the previous algorithm which only handles double inverter and transmission gate.

Each time the algorithm traverses through a Gate (Input Gate only, Source or Drain is not counted), the order of level increases by one. Parallel traversal and serial traversal are both counted. For example, if we traverse a resistor, diode, or capacitor, the order of level will not increase by one since these devices are two-terminal devices which do not contain a Gate.

In a typical analog design containing 500K devices, for instance,

Level 0 (or order 0) extraction: may contain resistors, capacitors, and diodes only. Their typical extracted size is around 50 devices, *i.e.*, around 50 lines of SPICE code with extra lumped parasitic information.

Level 1 extraction: may contain resistors, capacitors, diodes, and ONE level of NMOS or PMOS device (parallel or serial). There are typically around 500-1000 devices.

Level 2 extraction: contains typically about 1K-3K devices, parallel and serial together.

Level 3 extraction: contains typically about 2K-5K devices, parallel and serial together.

3.3 Generic Algorithm for Circuit Extraction (GACE)

There are two steps for the Generic Algorithm for Circuit Extraction:

1. Parsing the SPICE Netlist: This step builds a graph data structure for purpose is to build an efficient data structure for devices and nodes of SPICE. Each device and node is indexed by a unique ID, respectively. The device and node information can be accessed through the unique ID (or unique index). The complexity of parsing MOS netlist is O (N * M).
2. Circuit Extraction (CE): This step builds a graph of circuit extraction. The complexity of extracting the MOS netlist is O (N + M), where N = number of MOS nodes, and M = number of MOS devices = line of MOS netlist.

3.3.1 Parsing the SPICE Netlist

This section describes the parsing of a SPICE netlist. The steps are as follows:

1. Build a structural array named DEVS for storing devices: Each structural unit in DEVS has three memory spaces to store the node connections of devices. Those three memory spaces are front node, center node, and end node. If the device is a MOS, store the drain, gate, and source node number into the memory spaces. The node number is the ID of the node array (NODES). If the device is a resistor or a capacitor, store the two-terminal node's ID in the front node and end node and set the center node as NULL.
2. Build a structural array named NODES for storing nodes: Each structural unit in NODES has a sub-array to store the device IDs. Those devices (expressed in device ID) are connected to the node. The device ID is the index of the device array (DEVS). If the node is the gate node of a MOS, set a marker at the MOS in the sub-array. This marker is for dividing a spice netlist into different groups in the Circuit Extraction.
3. Start parsing of SPICE Netlist: The DEVS and NODES are empty in the beginning.

Figure 2 shows the algorithm for netlist construction.

```
1. For each line (device) of SPICE netlist {
2.      Put the device into the DEVS
3.      For the two or three terminal nodes of the device {
4.          Search the nodes existing in NODES {
5.              if (the terminal node has not existed in NODES) {
6.                  Put the node into the NODES
7.              }
8.              Put the device ID into the sub-array of the terminal node
9.              if the terminal node is the gate node of the device {
10.                 Set a marker on the device in the sub-array of the node
11.             }
12.             Save the node ID into the device node connections
13.         }
14.     }
15. }
```

Figure 2 - Algorithm for Netlist Construction

3.3.2 Circuit Extraction

This section describes the circuit extraction. The steps are as follows:

1. Specify the extraction order and the input pin.
2. Build a queue named *snode* for storing devices temporarily.
3. Build a queue named *sdev* for storing nodes: The device of next order will be stored in sdev temporarily.
4. Start circuit extraction: In the beginning, set every node and device to belong to a big (infinity) order (such as 99999) and put the input pin into the *snode*.

Figure 3 shows the Algorithm for Circuit Extraction.

```
1.    Set onum = 0
2.    while (onum <= order) {
3.       while (snode != empty) {
4.            Pop a node from snode
5.            For each devices in the sub-array of the node {
6.                 if  the node is not the gate node of the device {
7.                      if the node is not a VCC or GND {
8 .                          Put the terminal node of the device into snode
9 .                          Set the order of the terminal node to onum
10.                     }
11.                } else {
12 .                    Put the device into sdev
13 .               }
14 .               Set the order of the device be onum
15 .          }
16 .       }
17 .      onum++
18 .      if (onum <= order) {
19 .           while (sdev != empty) {
20 .               Pop a device from sdev
21.                If the front node (end node) of the device is not a VCC or GND {
22 .                    Put front node (end node)  of the device into snode
23 .                    Set the order of  front node  (end node)  to onum
24 .               }
25.           }
26 .       }
27 . }
```

Figure 3 - Algorithm for Circuit Extraction

3.3.3 Circuit buftehd Example

This section shows an example for circuit buftehd. It includes the schematic diagram, netlist, and associated MOS device ID/node ID.

Figure 4 shows the circuit buftehd example. The input pins are I and E and the output pin is O.

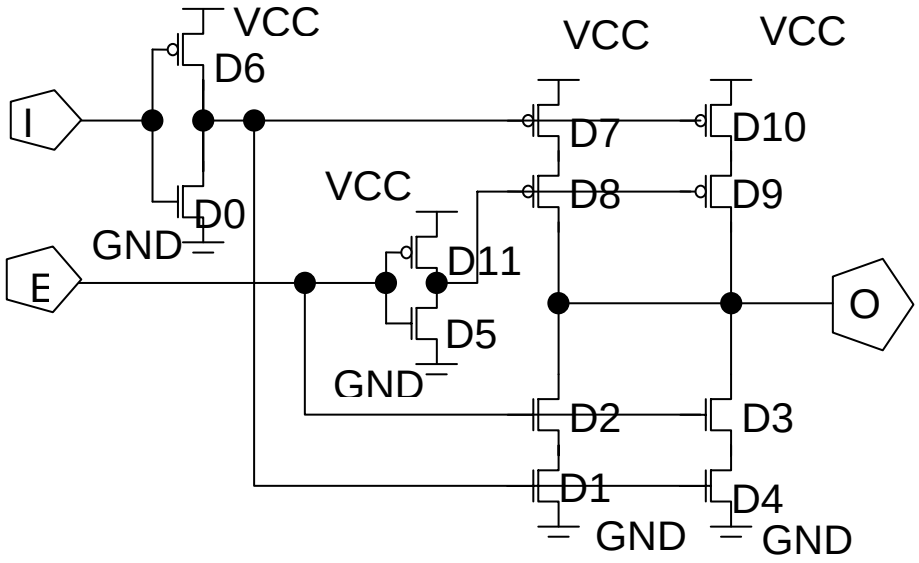

Figure 4 - Circuit buftehd Example

Figure 5 shows the SPICE Code of circuit Buftehd.

NMOS netlist:
D0 N_1 I GND GND NMOS
D1 N_4 N_1 GND GND NMOS
D2 O E N_4 GND GND NMOS
D3 O E N_3 GND GND NMOS
D4 N_3 N_1 GND GND NMOS
D5 N_2 E GND GND NMOS

PMOS netlist:
D6 N_1 I VCC VCC PMOS
D7 N_6 N_1 VCC VCC PMOS
D8 O N_2 N_6 VCC VCC PMOS
D9 O N_2 N_5 VCC VCC PMOS
D10 N_5 N_1 VCC VCC PMOS
D11 N_2 E VCC VCC PMOS

Capacitance NMOS netlist:
D12 I GND 0.451668F
D13 O GND 0.541840F
D14 N_1 GND 1.777830F
D15 E GND 1.164280F
D16 N_2 GND 1.513090F
D17 N_6 GND 0.047442F

D 18 N_5 GND 0.047771F
D 19 N_4 GND 0.028807F
D 20 N_3 GND 0.027364F

Figure 5 - SPICE Code of Circuit buftehd

Table 1 shows the replacement of Net Name and New Net ID.

Table 1. Node ID to Substitute Net Name

Net Name	New Net ID
N_1	N0
I	N1
GND	N2
N_4	N3
O	N4
E	N5
N_3	N6
N_2	N7
VCC	N8
N_6	N9
N_5	N10

Figure 6 shows the new SPICE Code after the Device ID and Net ID replacement. The above Net Name is replaced by the Node ID (N0, N1, .., N10). In the real code, the N0, N1, …N10 are represented by the unique ID, *i.e.*, 0, 1, ..,10, respectively. Those integer numbers are easier to access the node information. In addition, the Devices (D0, D1, D2, .., D15) are accessed by the unique device ID (or corresponding index), *i.e.*, 0, 1, 2,..,15, respectively.

NMOS netlist:

D 0 N0 N1 GND GND NMOS

D 1 N3 N0 GND GND NMOS

D 2 N4 N5 N3 GND GND NMOS

D 3 N4 N5 N6 GND GND NMOS

D 4 N6 N0 GND GND NMOS

D 5 N7 N5 GND GND NMOS

PMOS netlist:

D 6 N0 N1 VCC VCC PMOS

D 7 N9 N0 VCC VCC PMOS

D 8 N7 N9 VCC VCC PMOS

D 9 N7 N10 VCC VCC PMOS

D 10 N10 N0 VCC VCC PMOS

D 11 N7 N5 VCC VCC PMOS

Capacitance NMOS netlist:

D 12 N1 GND 0.451668F

D 13 N4 GND 0.541840F

D 14 N0 GND 1.777830F

D 15 N5 GND 1.164280F

D 16 N7 GND 1.513090F

D 17 N9 GND 0.047442F

D 18 N10 GND 0.047771F

D 19 N3 GND 0.028807F

D 20 N6 GND 0.027364F

Figure 6 - New SPICE Code of Circuit buftehd after Device and Net ID Replacement

Figure 7 shows the Device ID that is associated with each node.

N0 (N_1): 0 1 4 6 7 10 14
N1 (I): 0 6 12
N2 (GND): 0 1 4 5 12 13 14 15 16 17 18 19 20
N3 (N_4): 1 2 19
N4 (O): 2 3 8 9 13
N5 (E): 2 3 5 11 15
N6 (N_3): 3 4 20
N7 (N_2): 5 8 9 11 16
N8 (VCC): 6 7 10 11
N9 (N_6): 7 8 17
N10 (N_5): 9 10 18

Figure 7 -Device ID that is Associated with Each Node

After the Device ID and Node ID replacement, the original SPICE netlist can be represented by ID naming. Those IDs are used as the array index to access the device and node structure. Figure 8 shows the Device ID and Node ID for the circuit buftehd example.

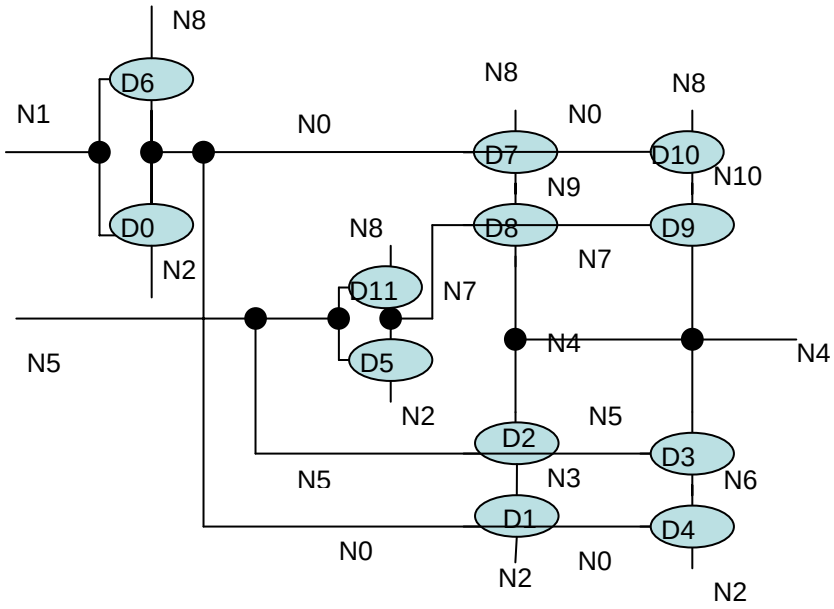

Figure 8 - Device ID and Node ID for Circuit buftehd

3.3.4 Graph Representation of Netlist

This section describes the graph representation for level 0, level 1, and level 2 of circuit buftehd. The steps are as follows:

1. Set SPICE Device as a two-terminal net in the graph. The SPICE devices include resistor, capacitor, MOS, and diode. The resistor, capacitor, and diode are two-terminal devices. The MOS is a three-terminal device. In order to translate the three-terminal device into a two-terminal device, the net on the gate is ignored. A net on the gate is used to control the signal from source to drain or vice versa; therefore, a net on the gate can be ignored. Only a net from drain to source is considered; however, the gate is used to differentiate the level of circuit.
2. Set a SPICE node as a node in the graph.

Graph Example of buftehd

1. Device: D1 D2 D3 …; Node: NMOS1 NMOS2 NMOS3 …
2. Input Pins I and E are cut by gate and become independent nodes.

3.3.5 Graph Construction

Figure 9 shows a graph example of circuit buftehd. Note that each node is represented by a circle and each device is represented by an edge. VCC and GND are the termination of graph.

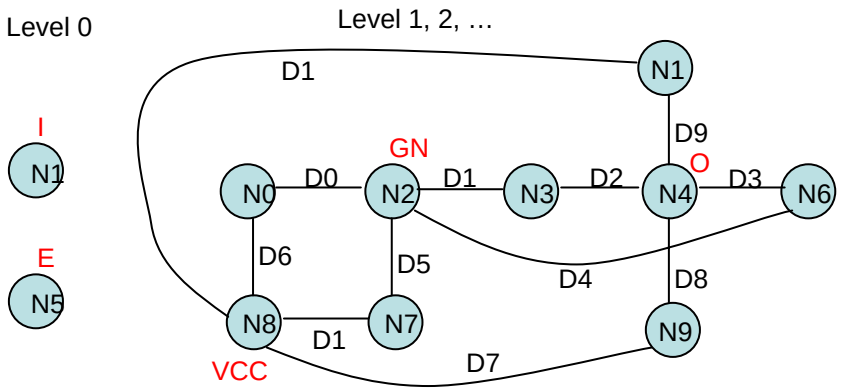

Figure 9 - Graph Example of Circuit buftehd

Figure 10 shows the regrouping by VCC and GND as the Cutting Nodes.

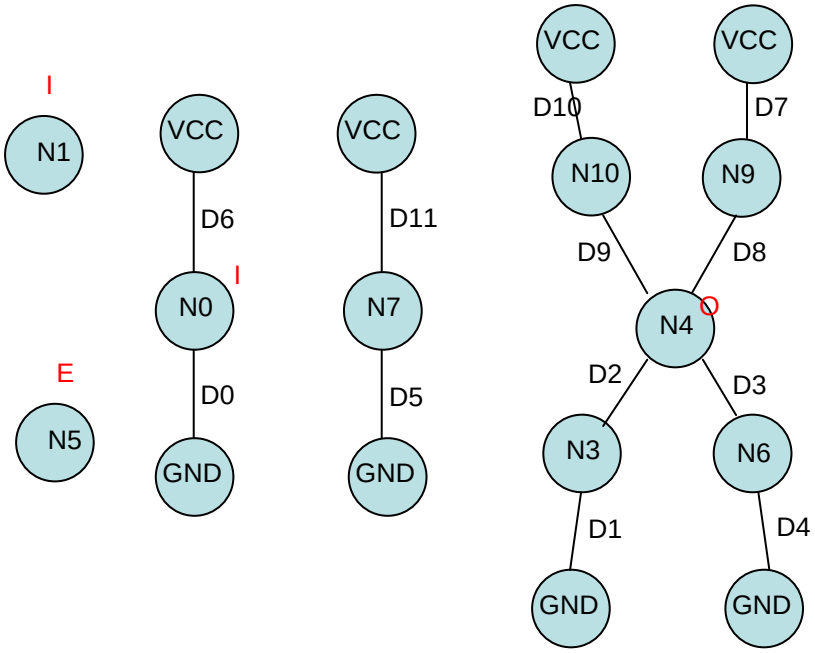

Figure 10 - Regrouping by VCC and GND as Cutting Nodes of Circuit buftehd

3.3.6 Rebuilding Netlist

The Netlist is cut by MOS gate, because the source to drain of MOS is connected in the graph representation. Figure 11 shows the netlist cut by the MOS gate.

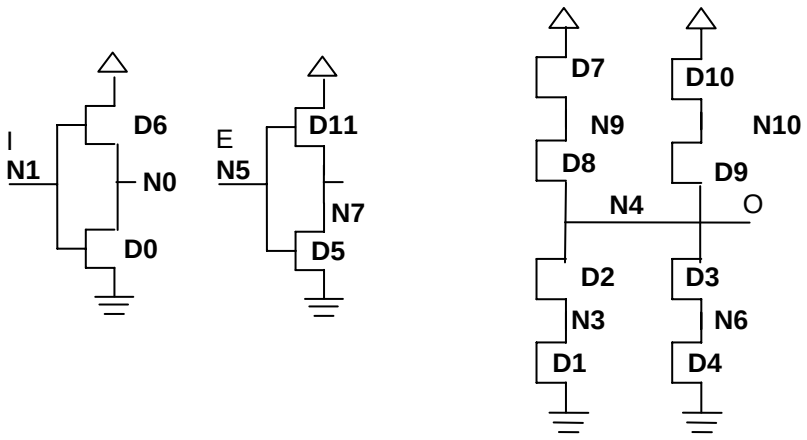

Figure 11 - Rebuilding Netlist Cut by MOS Gate of Circuit buftehd

3.4 Level 1 and Level 2 of Circuit Extraction

To do circuit extraction, specify the input pin, input pin N1 for example, and the desired level (0, 1, 2) of extraction. Figure 12 shows the Partial Circuit Extraction for Level 1 and Level 2 of Circuit buftehd.

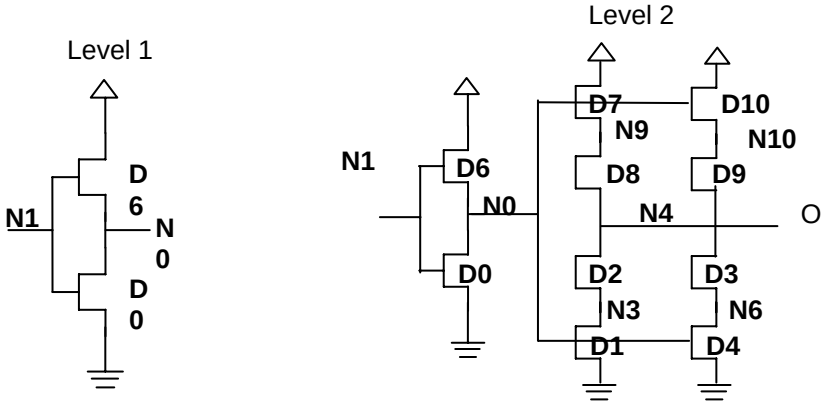

Figure 12 - Partial Circuit Extraction for Level 1 and Level 2 of Circuit buftehd

3.5 Opening/Shorting the Floating Netlist

After the circuit is extracted, some of the MOS's gates are cut and become floating nodes. If these floating nets do not open or short properly, the signal propagation will not be terminated due to the circuit extraction. In order to propagate the signal, opening or shorting the extracted circuit is necessary.

The following rules are used to make the signal propagate. Floating Net only occurs at the MOS gate and the other input pins.

- NMOS in Parallel form: If the gate is not the input pin associated, *i.e.*, it is a floating node/gate, then turn off the circuit by tying the NMOS low. The signal will be controlled by the input pin associated.
- PMOS in Parallel form: If the gate is not the input pin associated, i.e., floating node/gate, then turn off the circuit by tying the PMOS high.

Figure 13 shows an example of circuit turned on/off for a single NMOS/PMOS in parallel form.

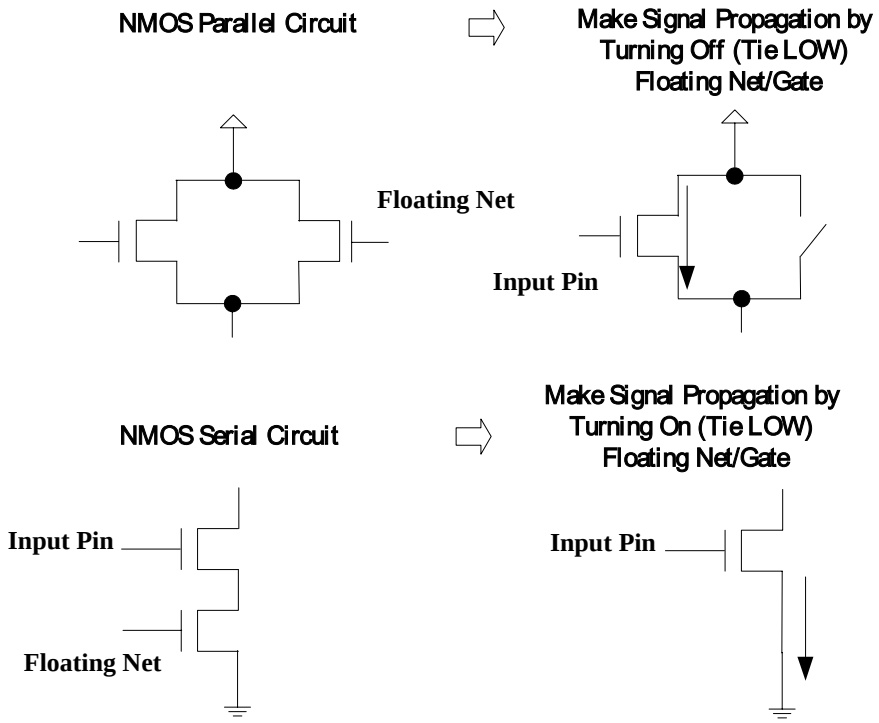

Figure 13 - Circuit Turned On/Off for a Single NMOS/PMOS in Parallel Form

- NMOS in Serial form: If the gate is not the input pin associated, *i.e.*, it is a floating node/gate, then turn on the circuit by tying the NMOS low.
- PMOS in Serial form: If the gate is not the input pin associated, *i.e.*, it is a floating node/gate, then turn on the circuit by tying the PMOS high.

Figure 14 shows an example of circuit turned on/off for a single NMOS/PMOS in serial form.

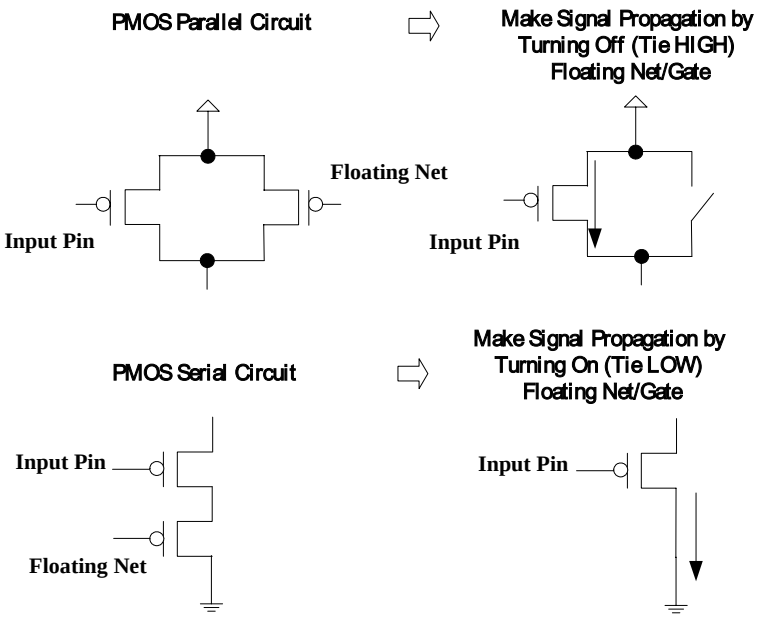

Figure 14 - Circuit Turned On/Off for a Single NMOS/PMOS in Serial Form

- MOS not in Parallel/Serial form: Tie NMOS to VCC, and PMOS to GND.
- If the circuit contains multiple parallel and serial sub-circuits, follow the above rules to merge the parallel/serial circuits one by one to the NMOS or the PMOS.

3.5.1 Circuit Merge

For a complicated circuit, for example, a sequence of serial and parallel circuits instead of a single NMOS/PMOS in serial/parallel form, two steps are proposed to merge the complicated circuit into a single circuit. Figure 15 shows the algorithm for merging a sequence of serial circuits and parallel circuits. An iterative method is proposed to solve the merge problem. The serial and parallel merges are used to merge the complicated circuit into a single NMOS/PMOS circuit.

A stick is a branch of circuit with an NMOS or PMOS element.

1. Set each MOS as a stick in the beginning.
2. Serial Merge:
   a. If two sticks are serially connected and the connected node does not connect to another stick, start the serial merge.
   b. The two sticks are merged into one stick.
   c. If one of the sticks has a signal input pin (related), turn on the MOS gate of all other sticks that are serially connected. If both sticks do not have a signal input MOS, set the MOS condition to be temporarily unknown.
   d. Serially merge all the sticks.
3. Parallel Merge:
   a. If two sticks are parallel connected, start the parallel merge. After the parallel merge, the two sticks are parallelly merged into one stick.

b. If one of the sticks has a signal input pin (related), turn off the MOS of the stick for the other stick. If both sticks do not have a signal input, set the MOS condition to be temporarily unknown.
c. Parallelly merge all sticks.

4. Iterate serial merge and parallel merge until the all sticks are merged into one stick.
5. The floating nets are set depending on the method used.

Figure 15 - Serial and Parallel Merge of Extracted Circuit

3.5.2 A Serial and Parallel Merge of Extracted Circuit

Figure 16 shows a serial and parallel merger of the extracted circuit. The gate nodes of D2 D5 D6 D7 are floating. The D5 and D4, D2 and D3, D6 and D7 are serially connected, respectively. By a serial merge, D5 and D2 are turned on. D6 and D7 are temporarily unknown. These sticks (D6 and D7) are turned off during the parallel merge.

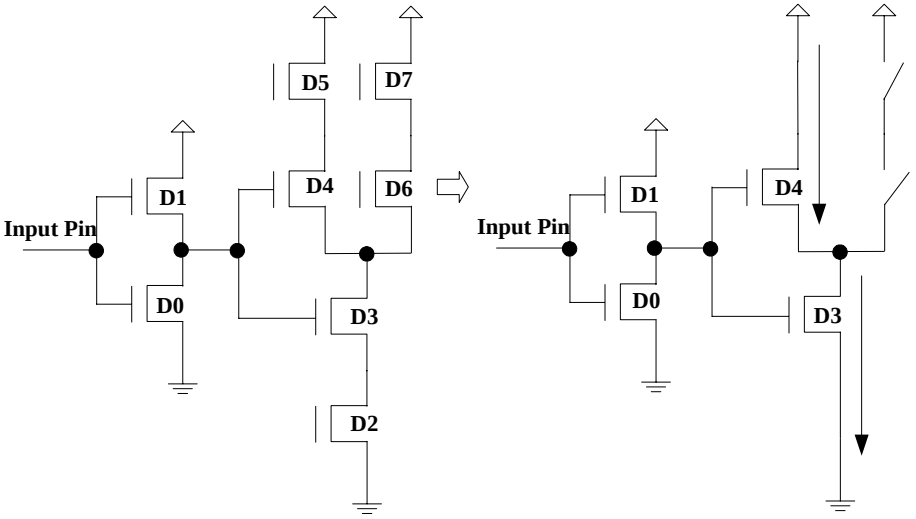

Figure 16 - A Serial and Parallel Merge of Extracted Circuit

Figure 17 shows the open and short of a NAND gate structure which contains a parallel PMOS and a serial NMOS in IP FXDAC030HA0A.

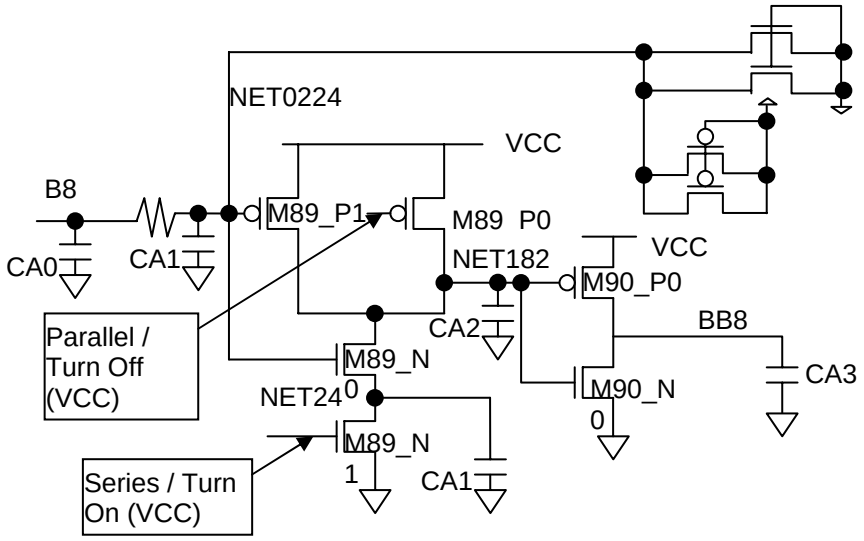

Figure 17 - Partial Circuit Extraction Result for IP FXDAC030HA0A

4.0 Results of Input and Output Cap Characterization

This section uses DAC as an example to show the results of input and output cap characterization.
Table 2 and Table 3 show a comparison of input and output characterization result between Levels 1, 2, 3 of Partial Circuit Extraction and Full IP Characterization.

Table 2. Comparison between Partial Circuit Extraction and Full IP Characterization for Input Capacitance

	Level 0 Circuit Extraction Characterization	Level 1 Circuit Extraction Characterization	Level 2 Circuit Extraction Characterization	Level 3 Circuit Extraction Characterization	Full IP Characterization
FIN1	2.446e-10	2.151e-10	2.149e-10	2.149e-10	2.149e-10
FB1	6.341e-12	6.236e-12	6.234e-12	6.234e-12	6.234e-12

Table 3. Comparison between Partial Circuit Extraction and Full IP Characterization for Maximum Capacitance

	Level 0 Circuit Extraction Characterization	Level 1 Circuit Extraction Characterization	Level 2 Circuit Extraction Characterization	Level 3 Circuit Extraction Characterization	Full IP Characterization
DRDY	4.37253	4.28219	4.28216	4.28216	2.149e-10
D0	3.89214	3.72358	3.72356	3.72356	6.234e-12

The characterization results showed that, after levels 1 and 2 of circuit extraction, the accuracy of both input capacitance and maximum output capacitance reached 90-100 %. The simulation time was cut down from months to seconds as well.

Table 4 shows the simulation result of Typical Case (TC) of two/three levels of Partial Circuit Extraction Characterization (PCEC).

Table 4. Comparison between Partial Circuit Extraction Characterization and LPE Extraction of Input Capacitance

	TC of Two/Three Levels of PCEC	Input Cap by LPE	Difference in Ratio (PCEC – LPE) / LPE
LIN1	2.149e-10	2.1489e-10	4.64982e-05
LIN2	1.009e-10	1.24396e-10	-0.188881
FIN1	2.158e-12	2.4029e-12	-0.101919
FIN2	2.177e-12	2.40229e-12	-0.101919
FB1	6.234e-12	3.01725e-12	1.06612
FB2	6.272e-12	3.0188e-12	1.07765
ENB	3.393e-14	2.13751e-14	0.587361
SEL	9.856e-14	1.15322e-13	-0.14535

As shown above, two- and three-level extractions have identical results in this test case. Here, we took the Typical Case as the basis of comparison since commercial extraction tools (X-Calibre or RC/XT) can only estimate Input Cap (Electrical Behavior) of Typical Case by the layout information. Note that a commercial tool can only view the first level of capacitance. The tool converts the dimension into the Input Capacitance for Typical Corner by estimation. Since it is not model based, it cannot cover the other corners (BC and WC) or change into other scenarios. If the input pin is connected to the resistors, the LPE Input Cap will be ZERO since the resistance has zero capacitance and the LPE tool cannot see the second level of devices and will hence get a wrong answer (Input Cap 0). The input cap of an IP can never be zero.

With reference now to Layout Parasitic Extraction (LPE), there are two types of layout parasitic extractions: one is parasitic coupling capacitance (PCC-LPE) and the other is Lumped Layout Parasitic Extraction (L-LPE). The L-LPE combines all the parasitic coupling capacitances (fringe capacitance, layer-to-layer coupling capacitance, and bilateral coupling capacitance) of each associated node. This paper employs L-LPE for circuit extraction.

5.0 Static Power Characterization

This paper proposes an automatic methodology for static power characterization of analog IPs. below. More than 2000 IPs implemented on processes ranging from 0.25-micron, 0.18-micron, 0.13-micron, to 65-nanometer were used for validation. The static (or leakage) power provides standby mode information to IPs and applications. The static power characterization is part of the ASIC/IP design and synthesis flows; therefore, it is essential to have a generic, fully

automatic, reliable, and accurate characterization flow that embraces all IPs across diverse processes [1-7].

The static power characterization can be achieved using either AC/transient characterization or DC characterization. Characterizing static power via AC characterization involves several steps. First, IP designers have to spend hours running SPICE tools to generate and store the initial conditions. The test benches, once provided with the generated initial conditions (ICs), take a couple of days/weeks to perform the SPICE simulation. It is difficult to maintain the database of the initial conditions for each IP. Moreover, since AC takes several time steps for simulation, it needs to take an average to obtain the measured results. The DC characterization, on the contrary, does not need any IC generation and needs one single time step to acquire the same level of accuracy of static power attainable by the AC characterization. In this paper, the DC characterization is used for static power characterization. The run time is cut significantly from days/weeks to seconds [7].

Static power is state dependent, *i.e.*, it hinges on the states of the voltage level (0 or 1) of the input pin. For example, for an IP with a pin count of 20, the number of test benches for static power characterization at three corners (Best Case, Typical Case, and Worst Corner) is $3 \cdot 2^{20} \cong 3$ million. It is impractical to perform the entire 3 million test benches and to write millions of static power attributes to each corner of the library.

A Reference Circuit is used to provide a stable voltage and driving capacity, *i.e.*, the voltage state is *fixed*. The Reference Circuit is a *Fixed State* Circuit, *i.e.*, the state is fixed and does not switch. The Reference Circuit can be identified by Differential Circuit and OP-Amp, respectively. A multiplexer (Mux) is used to re-route the input signal. The power consumption remains the same among the routes, *i.e.*, the Mux is a *Don't Care* Circuit. This Mux is a single-state, *i.e.*, the static power does not change with state switched.

In hierarchical SPICE, the Reference and Mux are hierarchically represented as Sub-circuits (or Macros). In this paper, an Automatic Macro Recognition (AMR) algorithm is used to classify the *Fixed State* Circuits (Differential or OP-Amp) and the *Don't Care* (Multiplexer) Circuits. Once these circuits are classified, an automatic path finding algorithm is used to identify the connection of input pins and these (*Fixed State* and *Don't Care*) circuits. Input pins connected to the Reference Circuit can be represented by a single (fixed) state. Input pins connected to the *Don't Care* Circuit can be represented by a single (independent) state.

Based on their design concept, IP designers can also use an alternative method, known as Knowledge-Based State Reduction (KBSR), to minimize the states. The KBSR can be used in two situations: 1) the IP designs use other circuits that the AMR recognizes to provide the stable voltage, driving capacity, and signal re-route; and 2) the IP designers intend to specify a special loading (capacitor or resistor) and a special state for input pins. Designers are required to use their knowledge to map the input pins to their own Fixed Reference and Don't Care Circuits.

Both AMR and KBSR can be used to minimize the states. In order to use the AMR, the input SPICE netlist has to be hierarchical for macro recognition and path finding. Each state is associated with one SPICE test bench. This methodology automatically reduces the states (test benches) from millions to within hundreds.

With the *Fixed* and *Don't Care* capability of the Liberty Parser, the static power attribute in the Liberty Library can represent millions of states in hundreds of entries either explicitly or implicitly. The Liberty Library is used by synthesis tools for system integration.

5.1 Algorithm for Static Power Characterization

The algorithm for static power characterization is as follows:

1. Read all the general input pin and voltage level information.
2. Parse the analog voltage level definition (VCCA, VCC33A, VCCK, etc.), special analog pin voltage level information, and switch pin and switch state information.
3. Perform the Automatic Reference recognition.
4. Generate the input pin state table based on the switch pin and switch state information.
5. Generate the test benches for static power characterization.
6. Parallel process the static power characterization and store the results.
7. Process the characterization results.
8. Write the Liberty Library.

5.2 Automatic Reference Recognition

The Automatic Macro Recognition (AMR) algorithm is used to classify the Sub-Circuits (or Macros) into Analog Gates, Reference Macros, and *Don't Care* Macros. The smaller analog gates include Analog Inverter (A_INV), Analog NAND (A_NAND), Analog ESD (BUS_ESD), and Analog DFF (A_DFF). The larger macros include the Fixed Reference Circuit and the *Don't Care* Circuit. The purpose of the AMR is to translate the three-terminal MOS devices and the non-directional gates (or macros) into directional (input to output) gates and macros. These directional analog gates or macros are similar to digital gates. The complexity of path searching for pin-to-macro connection is greatly reduced because the device's terminals and direction are minimized or fixed.

5.3 Reference Macro

The Reference Circuits can be implemented by Differential Circuits which provide the stable voltage and the OP-Amp buffers with driving capacity. The Reference Circuits comprise a first stage (Differential Circuit and loading), a second stage, and a feedback loop on the negative input (INN). The Differential Circuits comprise two input signals (negative input (INN) and positive input (INP)), and one Differential Pair (NMOS pair or PMOS pair). Therefore, the Fixed Circuits can be generically identified by the Differential Circuits and OP-Amp.

Figure 18 and Figure 19 show the NMOS and PMOS Reference Macros, respectively.

The AMR algorithm recognizes a Macro as a Fixed Reference if the Macro contains the NMOS/PMOS Differential Circuit with two differential inputs (INN and INP) and one feedback loop in the negative input (INN).

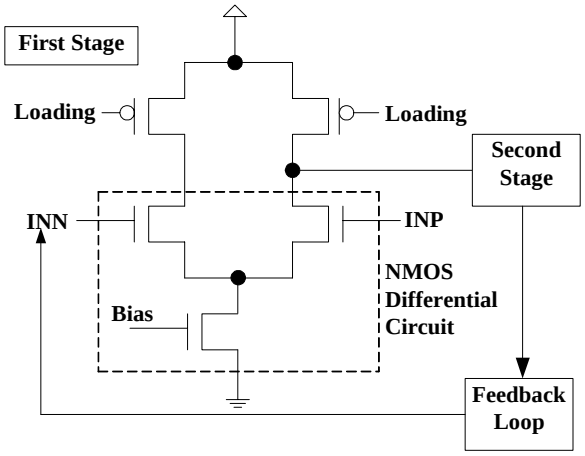

Figure 18 - NMOS Reference Macro

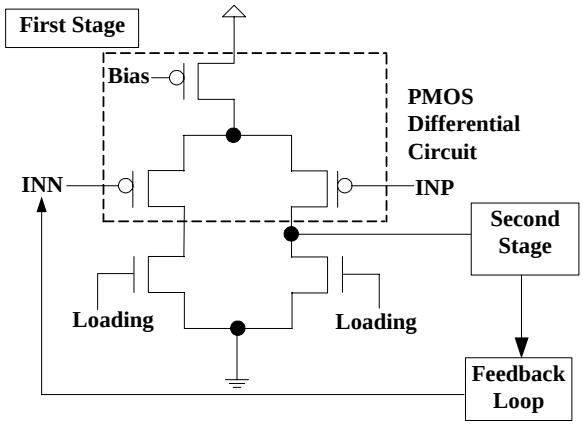

Figure 19 - PMOS Reference Macro

5.4 *Don't Care* Macro

The AMR algorithm automatically classifies a Sub-circuit (or Macro) as a *Don't Care* Macro if the Sub-circuit contains one Transmission Gate and two opposite selection signals, *i.e.*, an inverter is connected to one of the selection signals. Figure 20 shows a typical *Don't Care* Macro.

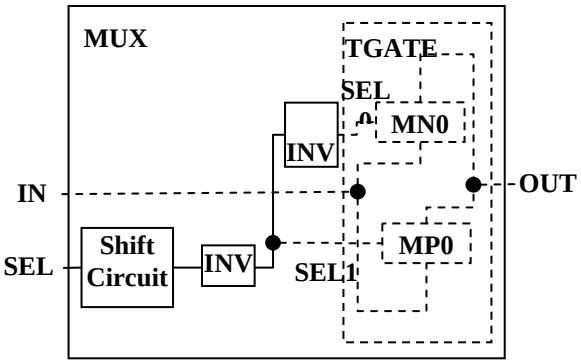

Figure 20 - Don't Care Macro

5.5 Path Connection Identification

The path connection algorithm is used to ensure the connection between the input pins and the *Fixed*/*Don't Care* Macros.

The hierarchical netlist contains Macros. The analog devices in the Macros are classified by the AMR as Analog Gates such as A_INV, A_NAND, A_DFF, etc. The purpose of this transformation is to minimize the path searching necessitated by a three-terminal MOS structure (one input and two outputs). The behaviors of the Analog Gates are similar to those of regular digital gates with directional and one-to-one features.

There are two infinity search loops:

1) An outer infinity loop to search in the range from input pins to the target Macro; and
2) An inner infinity loop to check the connection search inside each Macro. This loop contains netlist connection searching and matching in X Instances, Analog Gates, and devices (graph search for MOS, Resistors, Capacitors, and Diodes).

If the net inside the Macros cannot find any more connections, it is considered the end of the net connection search in Macros. The net is required to find the next upper-level Macro until the target Macro (Fixed Reference/Mux) is located.

5.6 Static Power Characterization Example

This section demonstrates the static power characterization of a Digital-to-Analog Converter (DAC) design.

Example: DAC010HA0L

Only input pins are related to the state of DC leakage power. The input pins for DAC010 are: COMP, RSET, VREF, CLKIN, DACIN0-9, PD, and VSL. There are 16 input pins, *i.e.*, totally $3(BC/TC/WC)\times 2^{16} \cong 1.92\cdot 10^{8}$ test benches, *i.e.*, 192 million test benches or 64 million states,

for static power characterization. This requires 64 million entries of cell_leakage_power attribute in a Liberty Library.
COMP is made a Voltage Compensation (in Fixed state) by the AMR. RSET is made a Fixed Reference Voltage (Differential Macro) by the AMR. DACIN0-9 are connected to *Don't Care* Macro by the AMR.

The COMP and RSET are *Fixed States*. The DACIN0-9 are *Don't Care* States. The input pins for *Switched States* are VREF, CLKIN, PD, and VSL. They contain 16 states (or $3 \times 2^4 = 48$ test benches). Table 5 shows the 16 logic states.

Table 5. Logic States for DAC010

Pin / State	VREF	CLKIN	PD	VSL
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

6.0 Result of Static Power Characterization

6.1 AC and DC Comparison

This section uses DAC as an example to show the result of static power characterization.
Table 6 shows the comparison between AC and DC Static Power Characterization.

Table 6. Comparison of AC and DC Static Power Characterization for DAC

	ICRT (sec)	RT (sec)	SR (Watts)
AC	65.3 by ADIT	88.6 / 98.16 / 116.9	3.12E-9 / 1.38E-8 / 1.64E-6
DC	None	8.1 / 11.2 / 23.6	3.11E-9 / 1.37E-8 / 1.63E-6

Abbreviations:
AC: Transient/AC Static Power Characterization

DC: DC Static Power Characterization
ICRT: Initial Condition Run Time
RT: Run Time for Best Corner / Typical Corner / Worst Corner
SR: Simulation Result for Best Corner / Typical Corner / Worst Corner

6.2 Test Bench Reduction

Table 7 shows the test bench reduction result for DAC.

Table 7. Test Bench Reduction for DAC

Methodology	Input Pin	Total Test Benches	Execution Time
Traditional AC with Initial Conditions	COMP, RSET, VREF, CLKIN, DACIN0-9, PD, VSL	192 million	More than 3 hours for a test bench. More than millions of hours to complete.
Knowledge-Based Data Reduction	VREF, CLKIN, PD, VSL	48	Parallel run. Finished within 1 minute for all test benches of DAC.

6.3 Liberty Generation

The following shows the static power entry of Liberty Library for the DAC example. The static powers are required by a Synopsys synthesis tool. The static power of an analog IP is converted into digital format.

Note that Liberty Library is a digitized format which does not care about the power level. So, only HIGH or LOW level is shown below. The analog power level is snapped into HIGH/LOW due to Liberty's digital format, *i.e.*, RSET is set to fixed (tied to HIGH). The pins DACIN0-9 and COMP (floating) are *don't care* pins that are bypassed by Liberty Parser. The pins VREF, CLKIN, PD, GNDK, and VSL are pins for switched states.

```
leakage_power () {
   when : "RSET * !VREF * !CLKIN * !PD * !GNDK * !VSL";
   value : 3.2169e+07;
}
leakage_power () {
   when : "RSET * VREF * !CLKIN * !PD * !GNDK * !VSL";
   value : 10.974;
}
…
```

7.0 Acknowledgements

The authors would like to thank Faraday for the full and valuable support of this project and Ms. Valerie Shih for the careful review and proofreading of this paper. The comments and input from Chris Kiegle of IBM are also appreciated.

8.0 References

[1] Peter H. Chen, Steven Chien, and Jim J. Wang, "Automation of IP Characterization and Function Test by HSPICE and Liberty-API," SNUG, Hsinchu, Taiwan, May 2004.

[2] Peter H. Chen, Steven Chien, Jim J. Wang, and Steve Tsai, "Method for IP Characterization and Path Finding, and Computer Readable Recording Medium for Storing Program," Taiwan Patent No. I232948 (July 2005), US Patent Pending.

[3] Peter H. Chen, Steven Chien, Jim J. Wang, and Steve Tsai, "Method for IP Characterization and Path Finding, and Computer Readable Recording Medium for Storing Program," Patent Pending.

[4] Peter H. Chen, Harrison Liu, Jerry Hong, Jim H. Wang, and Sam Lee "Hybridization Methodology for Finding Maximum Capacitance of Mixed Signal Design," SNUG, San Jose, USA, March 2005.

[5] Peter H. Chen, Harrison Liu, Peter Pong, Jim H. Wang, and Jerry Hong, "A Fast Algorithm for IP Input Cap Characterization," Patent Pending, January 2005.

[6] Peter H. Chen, Harrison Liu, Peter Pong, Jim H. Wang, and Jerry Hong, "A Fast Methodology Flow for IP Input Cap and Max Cap Characterization," SNUG, Hsinchu, Taiwan, May 2005 (best academic paper and most popular paper awards).

[7] Peter H. Chen, Jim H. Wang, Peter Pong, K.C. Wu, Harrison Liu, K.H. Huang, and Alvin Chen, "A Methodology for Static Power IP Characterization," DesignCon, International Engineering Consortium, Santa Clara, California, USA, February, 2006.