A Fast Methodology for Setup/Hold Time Characterization of Analog IPs

DD: Peter H. Chen, Peter Pong, K.C. Wu, and Y.W. Chen
MSD: Alvin Chen, Y.C. Chu and J.J. Huang
IRDC: Jim Wang

Faraday Technology Corporation
{pchen, pong, kcwu, ywchen, alvin, ycchu, jhwang }@faraday-tech.com

ABSTRACT

This paper proposes a fast methodology to characterize Setup/Hold Time for analog IPs. Partial circuits of clock and data paths are simulated instead of the simulation of entire IPs. The partial circuits include all those paths of clock pin and data input pins before reaching the DFF. This methodology includes algorithms for multi-path searching of hierarchical SPICE netlist for the path of clock and input pins, so as to reduce the circuit subset, merge the paths of clock vs. corresponding input pins, and then characterize the setup/hold time for an analog IP. The paths of input pins and clock before DFF are used for Setup/Hold Time Characterization. The results show the accuracy of partial path extraction of circuits is 100% as in full chip simulation, while the run time is cut from 3-8 hours to within seconds.

Table of Contents

Table of Figures

1.0 Introduction

In previous works [1-10], we have successfully implemented very fast methodologies and algorithms for basic characterizations, i.e., Input Cap, Output Cap, and Static Power Characterization for analog IPs, with more than 2000 IPs verified for the previous processes (0.25 um, 0.18 um, and 0.13 um). These methodology flows are officially used for IP release and logged into Faraday's central database. Each new IP is released within minutes/hours without the need for manual key in of the data or the need to wait weeks/months for the SPICE result [1-14].

The work done for this paper is based on strong request from customers. When given an IP, the IP user would like to have the timing constraint before reaching the first level of DFF, i.e., the earliest (setup time) and latest (hold time) signal arrival requirement for an analog IP. For designers and IP users, the timing constraint from the second or deeper levels of DFF is of no concern. What matters to them is the timing constraint from input pin to the first level of DFF. It is the IP designer's responsibility to provide the timing constraint information (Setup/Hold Time) for the IP users. The other levels of DFF timing constraints are related to internal timing requirements of IPs and must be met within the specification. External IP users do not need to know these constraints.

Figure 1 shows the input clock and pins to the various levels of DFF for an analog IP.

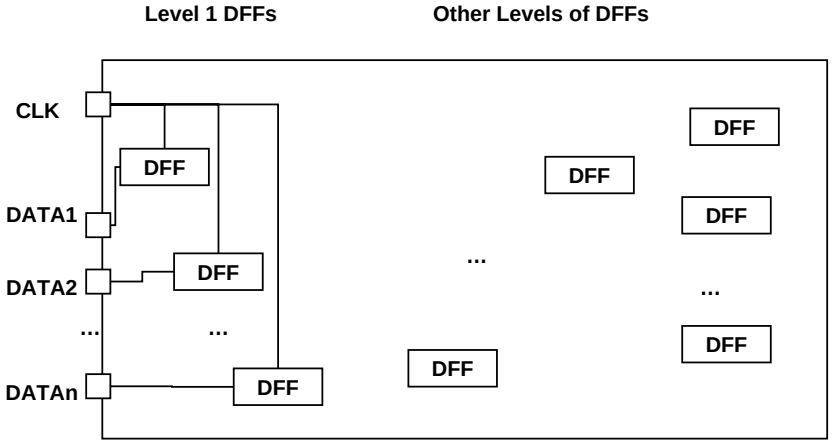

Figure 1. Input Clock and Data Pins to Various Levels of DFF for an Analog IP

As an IP vendor, we are obligated to provide the timing constraint (Setup/Hold Time) from the input pin to the first level of DFFs. However, characterizing the Setup/Hold time from the input pin to the first level of DFFs based on entire IPs is inefficient since we are only interested in a very small part (from input pin to first level of DFF) of IPs while doing characterization. Each IP

takes days/weeks/months to run entire IP characterization, which is very time consuming. On the other hand, setting up unnecessary input parameters (voltage level, operating conditions, logic patterns, etc.) is too tedious.

With the foregoing in mind, we started investigating the flattened netlist. For IPs, a flattened netlist with 500K+ lines of SPICE codes is very common. Some IPs may even have 200M+ lines of SPICE codes. It is simply awful and impossible with the current way to provide Setup/Hold time for these IPs, either by conducting a simulation or merely by observing the size of the individual SPICE codes. The hierarchical netlist is normally much smaller, i.e. in the range of 1K to 5K. This size sounds good; however, the coding presentation for the hierarchical netlist can be very complicated. Besides, hierarchical path searching is very difficult to implement. The multi-path searching for mixed signals (at Device level and XMacro level) appears impossible.

This paper proposes a generic methodology and algorithm to simplify the tedious input parameter setup while cutting the run time from weeks to seconds with the results remaining the same. The generic algorithm extracts the multiple paths of clock and individual pins, thus significantly reducing the simulation circuit and dispensing with the need to set up parameters for different IPs.

In order to speed up the netlist processing, the methodology uses the hierarchical netlist instead of the flattened netlist. A flattened netlist for an IP may for example contain 500K lines (or devices). In the hierarchical netlist, the size can be reduced to 5K lines. Some of the Macros (or SUBCKT) are used (or called) many times. Reducing the size from 500K to 5K can facilitate the path search from 15 minutes to within seconds per our experiences. Path searching involves an NP problem. The time needed would grow exponentially. The algorithm for hierarchical path searching is more complicated to implement but much easier to verify. In order to reduce the searching complexity, the non-directed graphs for flattened devices (for example, M, R, C, Q devices) are automatically classified into directed graphs for each analog gate (for example, A_NAND, A_INV, A_NOR, ESD, etc.).

The path searching algorithm is one of the hardest part in CAD industry. This paper proposes a Macro based searching for path finding. It expands the multi-path searching by pushing unfinished netlist to the stack during the search. The result turns out very good for both single and multiple searches.
After each single/multiple path for input pin and clock is found, the clock and the corresponding clock net have to be merged.

The merged paths do not use the circuits of the entire IPs. Circuits along the paths are extracted from the circuits of the entire IPs. These path circuits are the Subset Circuits of IPs.
The merged paths then have to match the subset of the Subset Circuit.

1.1 Problem Formation

Once given an IP, we are supposed to find the Setup/Hold Time for input pins and clock.
The Setup/Hold Times are the timing constraints from the input pins and clock before they reach DFF.

1.1.1 Input Parameter:

Input Pins: R0-R9, G0-G9, B0-B9
Clock Pin: CK
Destination Block: DFF

1.1.2 Output

Extract SPICE critical paths (from input pins to DFF and from clock to DFF) for SPICE Setup/Hold Time simulation and generate Synopsys Liberty Library.

2.0 Overall Flow

Figure 2 shows the Setup/Hold Time characterization flow for an analog IP.

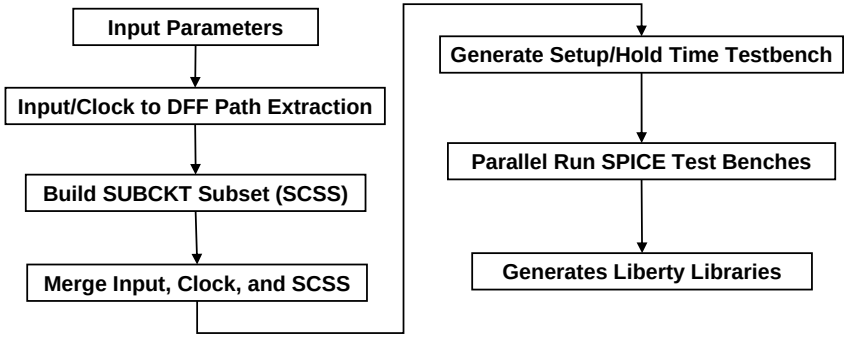

Figure 2. Overview Flow of Setup/Hold Time Characterization

3.0 Building the SPICE Hierarchical Netlist

Why we choose the hierarchical netlist instead of the flattened netlist? The single path search for a flattened netlist is easier to code. However, it is too slow and too difficult for verification. For example, a typical IP containing 500K (lines) of flattened SPICE netlist takes 5 minutes to find a path. The same IP can be represented by 1K (lines) of SPICE hierarchical netlist. It is much faster to do the path finding using the 1K SPICE code. The total path searching time for 30 input pins (R0-R9, B0-B9, G0-G9) and clock pin (CLK) is less than 1 second.

Since the SPICE hierarchical netlist is represented by XS and DEV structure, it is very efficient and fast to perform traversal, Macro/SUBCKT levelization, path finding, XI-Macro finding, SUBCKT Subset generation, path merging, non-input pin signal's tie high/low, and hierarchical signal measurement. All these tasks can be done within a second for 1K lines of SPICE hierarchical netlist.

The entire SPICE hierarchical netlist is passed and represented by DEV and XS structure. Figure 3 shows the DEV data structure.

```
struct DEV {
   string dn; // device name
   vector <string> param; // passing parameters
   string stmt; // statement
};
```

Figure 3. DEV Data Structure for Hierarchical SPICE Netlist

The structure DEV is used to store device information (Device Name, Parameter, and SPICE Statement).

Figure 4 shows an A_INV example.

```
.SUBCKT A_INV OUT SUB VDD VSS IN
MN0 OUT IN VSS SUB N_18_G2 W=WN L=LN M=MN
MP0 OUT IN VDD VDD P_18_G2 W=WP L=LP M=MP
.ENDS A_INV
```

Figure 4. A_INV Example

The A_INV is stored in dn. The OUT, SUB, DVV, VSS, and IN are stored in param. The MN0…, MP0…, and .END … are stored in stmt.

Figure 5 shows the structure XS.

```
struct XS { // X: XI and SUBCKT
  string xsn; // SUBCKT name or XInstance name.
  vector <string> param;
  string stmt; // statement
  vector <string> XDstmt; // XInst or DEV statement
  int hLevel; // hierarchical level.
  bool leafNode; // true: leaf node, false: non-leaf node.
  string macro; // subckt macro. Macros for loading subckt.
  vector <XS> XInst;
  vector <DEV> DInst;
};
```

Figure 5. XS Data Structure for Hierarchical SPICE Netlist

The structure data field explanation:
String xsn: stores SUBCKT name or XInstance name.

Vector <string> param: stores the passing parameters of SUBCKT or XInstances.
string stmt: stores entire SUBCKT statement.
Vector <string> XDstmt: stores either calling XInstances or Device statement.
Int hLevel: stores the hierarchical level of the current XInsance.
Bool leafNode: stores current XInstance as a leaf node or non-leaf node.
string macro: stores the Marco name of the calling XInstance.
Vector <XS> XIns: stores the calling XInstances.
Vector <DIns> DInst: stores Device Insances.

Figure 6 shows a SUBCKT FXDAC030HA0A example.

```
.SUBCKT FXDAC030HA0A OPR .. OPB COMP .. REFSEL B0 .. B9 G0 .. G9 R0 ..
R9
...
XI20 NET0157 ..  NET0148 GNDK VCC18K G0 .. G9 BUS_ESD
XI23 GNDK VCC18K PD_ESD XSC_LOCAL LN=0.18U WN=10U
..
XICORE VCC33A GNDA ..  NET0221 .. NET0273  DAC_3C_CORE_LB
MN0 GNDK VCC18K GNDK GNDK N_18_G2 W=3U L=3U M=280
MP0 VCC18K GNDK VCC18K VCC18K P_18_G2 W=3U L=3U M=224
…
.ENDS FXDAC030HA0A
```

Figure 6. SUBCKT FXDAC030HA0A Example

The xsn stores the FXDAC030HA0A.
The param stores the OPR .. OPB COMP .. REFSEL B0 .. B9 G0 .. G9 R0 .. R9.
The XIns stores XI20 ... and XI23.
The DIns stores MN0… and MP0.
The stmt stores all the statements inside the SUBCKT FXDAC030HA0A.

4.0 SPICE Hierarchical Netlist Extraction

Path search algorithm is one of the most difficult tasks in computer algorithm. Dijkstra's algorithm finds the shortest path for directed graph with non-negative edge. Lee's Algorithm [11] is the original algorithm for single path and shortest path finding. Both algorithms are provided for single path finding. Dijkstra's algorithm can only find the path for the directed graph. Modern router or path finding algorithms are all based these two algorithms with the same constraint.

The SPICE code with non-directed graph for MOS devices or three-terminal resistors cannot fit the directed graph requirement. The non-directed graph of these devices causes the go back loop in the SPICE to traverse backward and diversify everywhere. It is impossible to find multiple paths for SPICE code to use the current router approach.

4.1 SPICE Code Hierarchy

Figure 7 shows the SPICE Code hierarchy for IP FXDAC030HA0A.

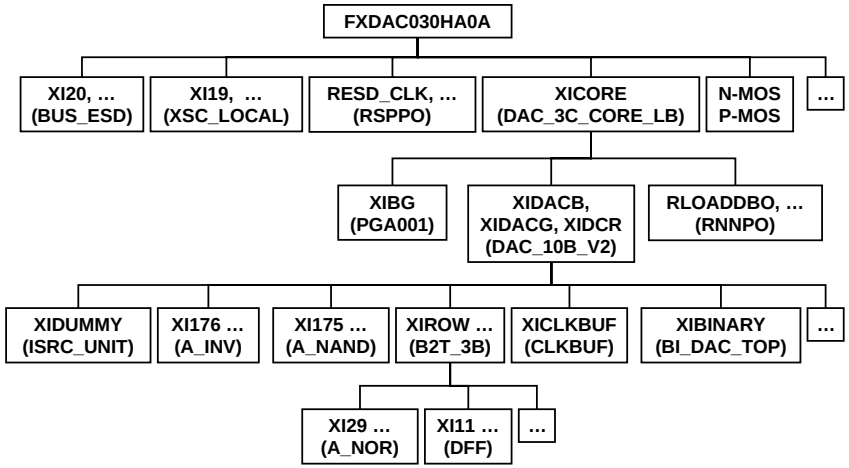

Figure 7. SPICE Code Hierarchy: FXDAC030HA0A Example

4.2 Macro and XMacro

The SUBCKT or Macro is classified into several categories:

Analog Macro or Analog Gate: They are the leaf node in SPICE code, such as A_INV, A_NAND, and A_NOR. These Gates contain MOS devices and three-terminal devices which are non-directed graphs. The Analog Gates are transformed into an edge of directed graph. Logically, they are either one-to-one or n-to-one devices; however, their output can drive multiple outputs and can be one-to-many from the graph point of view.

ESD/Antenna Macro: They are the leaf node in SPICE code. These devices are used in manufacturing and are dummy devices in normal operation, i.e., in SPICE Code. These dummy devices are an edge of directed graph (directly mapped from input port to output port).

XMacro call or NLXM (Non-Leaf XInstance Macro) call: This is a recursive call for further depth traversal in order to reach the leaf node. The XMacros, such as XICORE, IXDACB, XCLKBUF, etc., are not Leaf Node. During the traversal, the order of passing net need to be memorized in order to dive into the next level or pop up to the upper level of XMacro. The XMacro serves as the overhead for function grouping purpose.

In Figure 7, the BUS_ESD, SXC_LOCAL, ISRC_UNIT, RSPPO, A_INV, A_NAND, A_NOR, and DFF are leaf nodes. The XICORE, XIDACB, XIROW, XICLKBUF, and XININARY which are non-leaf nodes are called XMacros.

The leaf nodes BUS_ESD, XSC_LOCAL, and ISRC_UNIT are ESD/Antenna Macros. The leaf nodes RSPPO, A_INV, X_NAND, A_NOR, and DFF are Analog Gates.

In this paper, the non-directed graphs in the Analog Gate and ESD (BUS_ESD and XSC_LOCAL) are converted into an edge of directed graph. Some floating net with non-graph type in the XMacros (XICORE, XIDACB, etc.) is done by heuristic/exhausted search. The stack is used to store the net during the heuristic search of path finding. The stored net will be popped out from the stack if the path is found, i.e., once one path is found, the net on the stack for corresponding path will be cleared (or removed).

The multi-path searching algorithm can be done by recursive call (via condition statement and recursive calls) or flattened looping algorithms (flatten the recursive calls by loops and control them by semaphore flags). Since the recursive call is very difficult to debug, this paper uses the flattened algorithm approach. The path searching algorithm loops for all the sources, i.e., input pins (R0-R9, G0-G9, and B0-B9) and clock pin. It stops once it finds the destination block DFF. However, inside each Macro, the source to destination is not unique, i.e., many paths might exist. This paper uses a stack to store the candidates of multiple paths. Depth first algorithm is used to find the expected path. Once the path is found, the candidate is popped out from the stack. When all the candidates have gone through the source to destination finding, the multiple path finding is done for one source to destination, i.e., input pin to DFF or clock to DFF.

4.3 Automatic Macro Recognition (AMR)

The Automatic Macro Recognition (AMR) algorithm [9] is used to classify the Analog Gates (such as A_INV, A_NAND, A_NOR, etc.) for searching complexity and ESD/Antenna Protection structure reduction [1-2] (such as BUS_ESD, XSC_LOCAL, etc.) and for terminal determination (to reduce both searching complexity and the multi-path search dimension).

4.3.1 A_INV, A_NAND, and A_NOR

Figure 8 to Figure 13 show the Analog Gates (A_INV, A_NAND, A_NOR). They serve the same functionality as the Logic Gates (INV, NAND, NOR, etc.) except the driving strengths are greatly different. Their device types (INV, NAND, and NOR), input passing parameter indexes, and output parameter indexes are recognized by the Automatic Macro Recognition (AMR) algorithm.

Figure 8 shows the SPICE code for A_INV gate. Figure 9 shows the Input Port (IN) passing index (4) and the output port (OUT) passing index (0) as recognized by the AMR. A_INV features one NMOS and one PMOS in parallel.


```
.SUBCKT A_INV OUT SUB VDD VSS IN MP=1 LP=0.35U WP=3.2U …
MN0 OUT IN VSS SUB N_18_G2 W=WN L=LN M=MN
MP0 OUT IN VDD VDD P_18_G2 W=WP L=LP M=MP
.ENDS A  INV
```

Figure 8. SPICE A_INV Gate

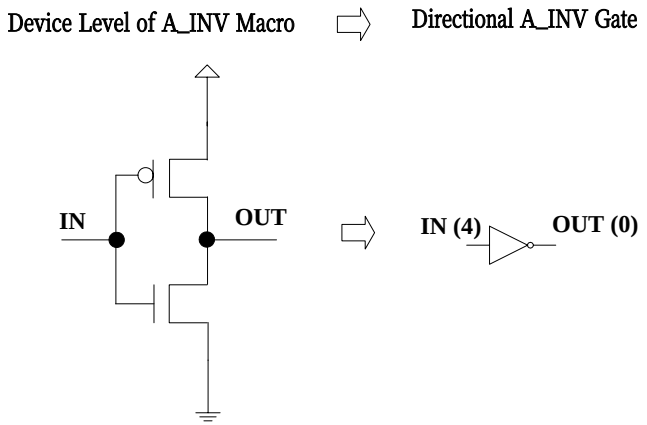

Figure 9. AMR A_INV Gate

```
.SUBCKT A_NAND2 Y SUB VDD VSS A B MP=1 LP=0.35U WP=3.2U …
MN1 NET24 B VSS SUB N_18_G2 W=WN L=LN M=MN
MN0 Y A NET24 SUB N_18_G2 W=WN L=LN M=MN
MP1 Y A VDD VDD P_18_G2 W=WP L=LP M=MP
MP0 Y B VDD VDD P_18_G2 W=WP L=LP M=MP
.ENDS A_NAND2
```

Figure 10. SPICE A_NAND Gate

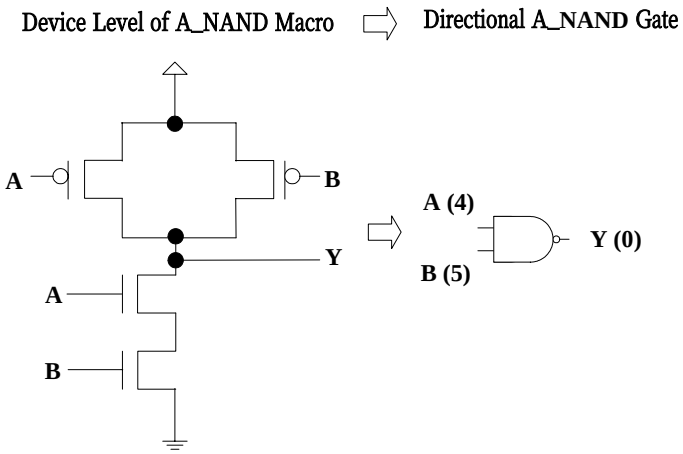

Figure 11. AMR A_NAND Gate

```
.SUBCKT A_NOR2 Y SUB VDD VSS A B MP=1 LP=0.35U …
MN1 Y B VSS SUB N_18_G2 W=WN L=LN M=MN
MN0 Y A VSS SUB N_18_G2 W=WN L=LN M=MN
MP1 NET35 B VDD VDD P_18_G2 W=WP L=LP M=MP
MP0 Y A NET35 VDD P_18_G2 W=WP L=LP M=MP
.ENDS A_NOR2
```

Figure 12. SPICE A_NOR Gate

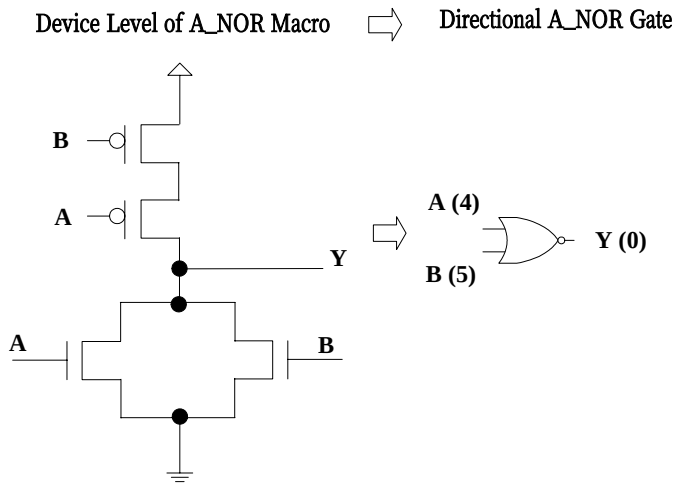

Figure 13. AMR A_NOR Gate

4.3.2 BUS_ESD and XSC_LCOAL

The ESD/Antenna protection structures are used to provide a reverse diode function and to guard against the antenna effect during the magnification and prevent the Electro-Static Discharge (ESD). These structure can be implemented by the MOS or the reversed Diode [1-2].

The BUS_ESD and XSC_LOCAL shown in this paper are implemented by MOS. BUS_ESD is a bus (multiple bits I0-I9/O0-O9) while XSC_LOCAL is a single bit.

During the manufacturing mode, the NWELL serves as drainage for plasma charge against antenna effect/ESD. Since the output port has very low impedance, the plasma and/or electro-static charge flows directly to the NWELL (floating or grounded). During the normal operation mode, the NWELL is tied to high and forms a reversed diode (diode is OFF) for open circuit (disconnected).

The BUS_ESD and XSC_LCOAL are used in manufacturing only. They are dummy devices with very little effect in capacitance during normal operation. The input signals (I0-I9) are directly transmitted to the output pins (O0-O9) in BUS_ESD. The XSC_LOCAL is an open circuit and ties the output gate parallelly.

```
.SUBCKT BUS_ESD O0 … O9 GNDSUB NWELL I0 … I9 LN=0.18U WN=10U
RESD_BUS0 I0 O0 20 $[RSPPO]
...
MPDIODE_0 O0 NWELL NWELL NWELL P_18_G2 W=WN L=LN M=1
…
MNDIODE_0 O0 GNDSUB GNDSUB GNDSUB N_18_G2 W=WN L=LN M=1
…
.ENDS BUS_ESD
```

Figure 14. SPICE BUS_ESD Gate

Figure 15 shows the BUS_ESD during normal operation. The input signal (I0) is directly transmitted to the output pin without bias since both P-Diode and N-Diode are open (OFF).

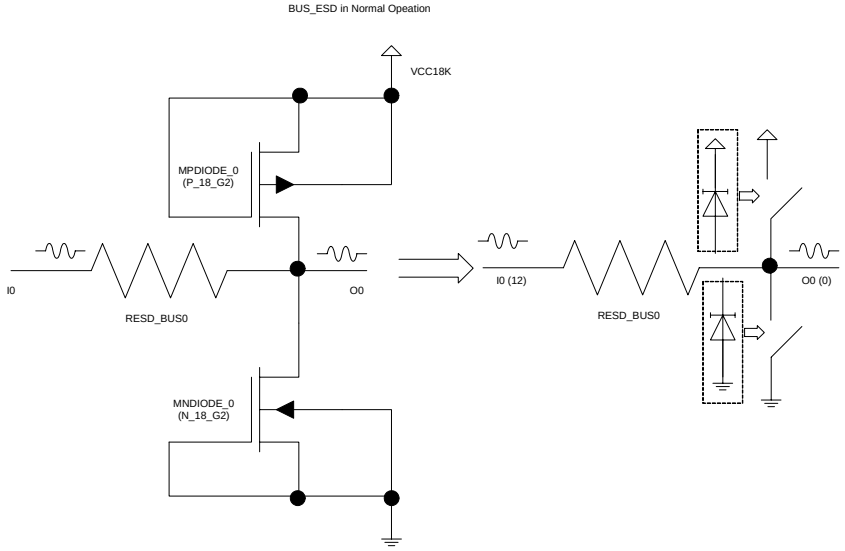

Figure 15. AMR BUS_ESD Gate

```
.SUBCKT XSC_LOCAL GNDSUB NWELL O LN=0.18U WN=10U
MPDIODE O NWELL NWELL NWELL P_18_G2 W=WN L=LN M=1
MNDIODE O GNDSUB GNDSUB GNDSUB N_18_G2 W=WN L=LN M=1
.ENDS XSC_LOCAL
```

Figure 16. AMR XSC_LOCAL Gate

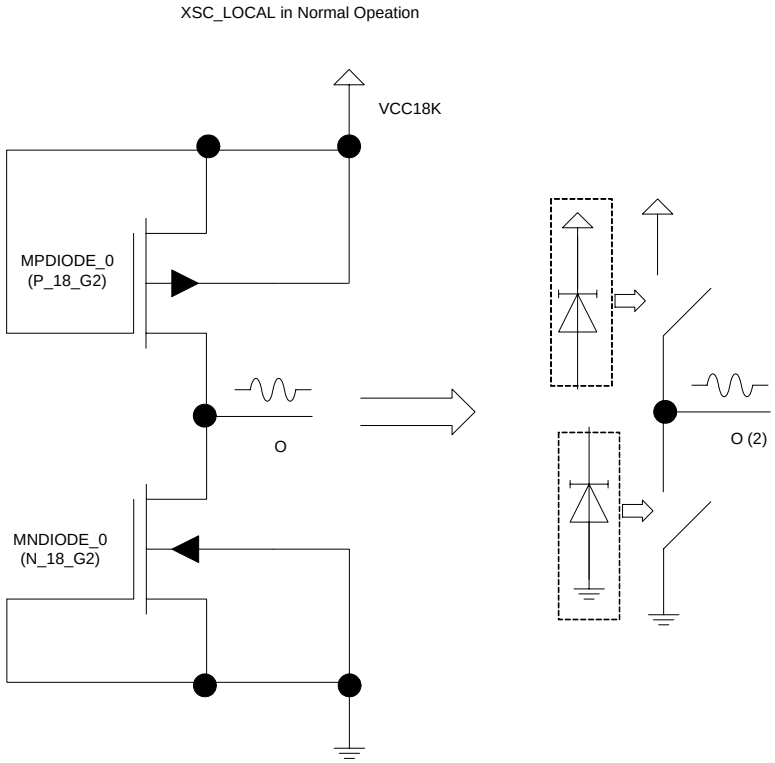

Figure 17. AMR XSC_LOCAL Gate

4.4 From Non-Directed Path to Directed Path

Two-terminal SPICE devices, such as resistors, diodes, and capacitors, are one-to-one graphs. One input terminal maps to one output terminal. These devices are not difficult for the path finding algorithm. They are directed graphs (from Input Pin to Output Pin) with 1-to-1 mapping.

For MOS devices, three-terminal resistors, and three-terminal diodes, these devices are not 1-to-1, i.e., they are non-directed graphs with 1-to-2 (or more) diverse path finding. The diverse path finding is harder to implement and very difficult to verify. These devices appear very often in analog gates such as INV (Inverter) gate, NAND gate, NOR gate, and ESD. These three-terminal devices can be embedded into the analog gates. The analog gates are classified by the AMR algorithm. They are directed graphs, i.e., the input pin can be directly transformed into the output

pin, which means the path is found immediately for these devices. The AGC is used to reduce the searching complexity and implementation effort.

4.5 Multi-Path Finding Algorithm

The Path Finding uses a Macro as a repeat unit.
Figure 18 shows the multi-path searching algorithm.

Each net in the Marco is connected to external inputs. Top-Macro is a special Macro that each net connected to the associated input pin.
For a selected net in each Macro (SUBCKT),

1. If the selected net is connected to DFF (destination), the path is found. Stop path finding for the selected net.
2. Push the selected net to the stack. Repeat steps 3-5 until the stack is clear.
3. If stack is not empty, find Analog Gate path. Pop the net out of the stack if destination in Macro found.
4. If stack is not empty, find the devices (MOS, Resistor, Diode, and Capacitor) path. Pop net out of stack if destination in Macro found.
5. If stack is not empty, find the XMacro (XICORE, XIDACB, XIDACR, etc.) path. Pop net out of stack if destination in XMacro found.

Figure 18. Multi-Path Searching Algorithm

Steps 3, 4, and 5 are used to find the path for Analog Gates, devices, and XMacro. The XMacro is a SPICE function call to SPICE Libraries, which can be recursively called until the leaf node (Analog Gate or devices) is found.
The path finding includes the paths from input pins (R0-R9, G0-G9, and B0-B9) and from clock pin (CLK) to destination (DFF).

Figure 19 shows multiple paths from clock (CLK) to DFF. Figure 20 shows single paths from Input Data (B8) to DFF.

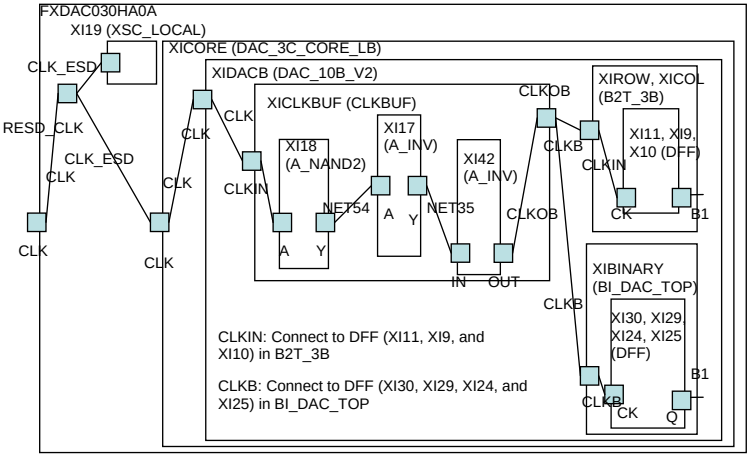

Figure 19. Multiple Paths for Clock (CLK) to DFF

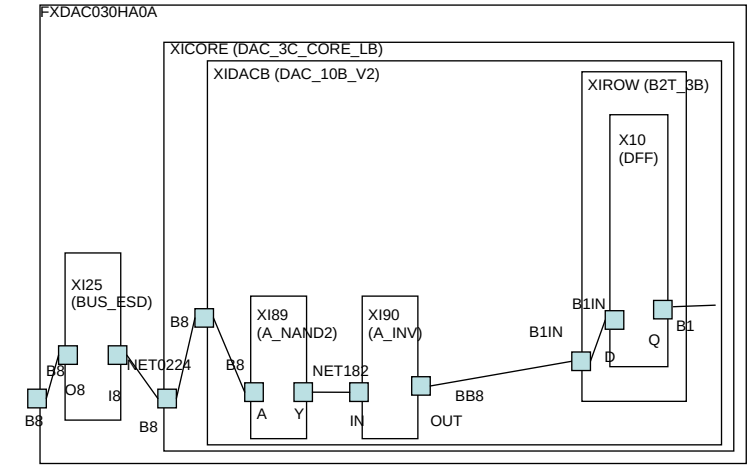

Figure 20. Single Path for Input Data (B8) to DFF

4.6 Building SUBCKT Subset (SCSS)

The SUBCKT (or Macro) Subset (SCSS) contains the SUBCKT (or Macro) connected on the source to destination path. SCSS is a subset of entire IPs.

4.7 Tie High/Low for NAND/NOR Gate

In order to let the signal propagate, the extracted input pin path and clock path that contain the floating net (non-signal net) must be either tied high or tied low. In the previous works [7, 10], we proposed a parallel circuit and serial circuit merger algorithm for both NAND and NOR gates for flattened netlist. In this work, we allow NAND gate to be tied high and NOR gate to be tied low for signal transfer.

Figure 21 shows the Tie High for NAND gate. A global VCC node (VCCXX) is used to Tie HIGH the floating node for NAND Gate.

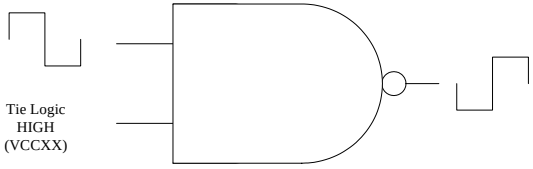

Figure 21. Tie High for NAND Gate

Figure 22 shows the Tie High for NOR gate. A global GND node (VGNDXX) is used to Tie LOW the floating node for NOR Gate.

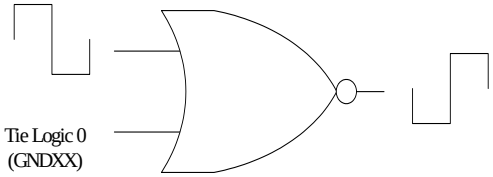

Figure 22. Tie Low for NOR Gate

4.8 Merging Input Pin Path, Clock Path, and SCSS

The Input Pin Path and the Clock Path may traverse different hierarchal paths. Note that the Input Pin Path and the Clock Path are properly tied high and tied low, respectively. Some of the paths may appear in both Input Pin and Clock Path. The duplicated path should be avoided in the final merged hierarchical netlist.

It is very hard to merge/unite two different hierarchical sets into a final netlist. SUBCKT Subset (SCSS) is a subset of original IPs. SCCS is a good reference since it keeps the original hierarchy for both input pin path and clock path.

Figure 23 shows the merger algorithm for SPICE Hierarchical Netlist. It uses the SCSS as the reference hierarchical structure to find the paths in the Macro of Input Pin Path and Clock Path.

1. For each statement in the SCSS
2. If the statement is either contains NAND or NOR statement, then find the associated statement from Input Pin Path and Clock Path.
3. If the statement is not NAND or NOR statement, then directly merge the matched statement from Input Pin Path and Clock Path.

Figure 23. SPICE Hierarchical Netlist Merger Algorithm

Figure 24 shows the merged paths for clock (CLK) to DFF and Data (B8) to DFF.

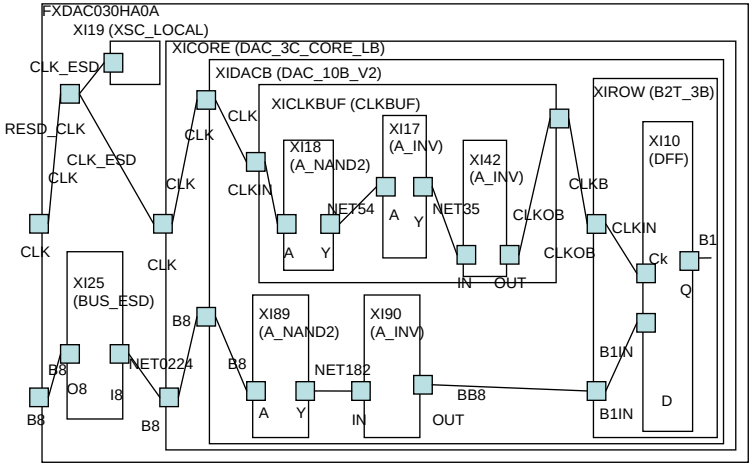

Figure 24. Merged Paths for Clock (CLK) to DFF and Data (B8) to DFF

4.9 Finding the Hierarchical Signal

In order to measure the Q inside the SUBCKT DFF, the hierarchical order of signal Q in DFF should be ascertained. For example,

For data pin B8 with clock CLK:

Measured Tran delay Trig v(CLK) Val = '0.5*(VP)' td = 6n Rise = 1

+ Targ v(X1.XICORE.XIDACB.XIBINARY.NET156) Val = '0.5*(VP)' td = 6n Rise = 1

In order to measure the signal inside the module DFF (which is instantiated by XI10 in the above example), the following hierarchical order need be found out:

X1->XICORE->XIDACB->XIBINARY->NET156 (Q on DFF).

Similar hierarchical traversal algorithms mentioned previously are used to traverse the hierarchical order of the signal. X1 is the top most cell name, for example, FXDAC030HA0A. The others including XICORE, XIDACB, and XIBINARY are the X-Macros. All the fundamental gates in the path, such as BUS, ESD, XSC_LOCAL, A_INV, A_NAND, A_NOR, etc. are ignored. The NET156 (Q) on the instance XI10 (DFF) is considered the measurement point. This signal should be found from the previous path search.

5.0 Common Setup for Delay and Setup/Hold Time

Once we have the Merged Paths of Clock and Data with tie high/low, the rest of the characterization tasks are similar to the DFF characterization. The Setup/Hold Time characterization includes two steps, i.e., measure the delay time and then measure the Setup/Hold Time. HSPICE provides the optimization algorithms and examples. Sections 5.1 through 5.4 show the common setup used for measurement of both Delay and Setup/Hold Time.

5.1 Include Path and Top Macro Instantiation

Figure 25 shows the header section of the SPICE code.

```
1. ************************************************
2. Include SPICE Circuits: fxdac030ha0a
3. Delay Measurement
4. ************************************************
5. .inc '. /sh_time_tb/FXDAC030HA0A_R0_DFF_CLK.MPath'
6. ************************************************
7. X1 Instantiation
8. ************************************************
9. X1 OPR OPG OPB COMP RSET VREF CLK PD STBY REFSEL B0 ... B9 G0 ...
   G9 R0 ... R9 FXDAC030HA0A
```

Figure 25. Header Section of SPICE Code

The header section includes:
1. Include Path (line 5): The Include Circuit which contains the merged SPICE Code of Input Pin and Clock Paths.
2. Top Macro Instantiation (line 9): X-Instantiate the Top Macro (FXDAC030HA0A) with passing parameters (OPR/G/B, RSET..REFSEL, B0-B9, G0-G9, R0-R9).

5.2 Global Power Section

Figure 26 shows the global power section of the SPICE code.

```
1. ************************************************
2. * Global Power definition
3. ************************************************
4. .GLOBAL VCC33A GNDA GNDA_BIAS VCC18K GNDK
5. VDD33_GL_0 VCC33A 0 VP3V
6. …
7. VGND_GL_2 GNDK 0 DC 0
8. ************************************************
9. * Tie High/Low Global Power Setup
10. ************************************************
11. .GLOBAL GVXX GNDXX
12. VDDGVXX GVXX 0 VP
13. VGNDXX GNDXX 0 DC 0
```

Figure 26. Global Power Section of SPICE Code

The Global Power Section includes:
1. The Global Power definition for IPs: The IP contains the global analog power (VCC22A) and the global digital power (VCC18K).
2. The Tie High/Low Global Power definition for IPs: The global power (GVXX) is used to tie high for NAND gate and the global power (GNDXX) is used to tie low for NOR gate to let the signal propagate properly.

5.3 Local Power and Parameters

Figure 27 shows the local power and parameter setup. The local power sets up the power level for the passing parameters for the top Macro. The parameters set up the power levels used by the SPICE testbench. These parameters are process related and are defined in the environment databases.

```
1.  ************************************************
2.  * Local Power Setup
3.  ************************************************
4.  VPD PD 0 DC 0
5.  VSTBY STBY 0 DC 0
6.  VREFSEL REFSEL 0 DC 0
7.  VR0 R0 0 DC 0
8.  ...
9.  VR9 R9 0 DC 0
10. VG0 G0 0 DC 0
11. ...
12. VG9 G9 0 DC 0
13. VB0 B0 0 DC 0
14. ...
15. VB9 B9 0 DC 0
16. ************************************************
17. * Parameters
18. ************************************************
19. .param VP123 = 1.23V
20. .param VP = 1.98V
21. …
```

Figure 27. Local Power and Parameter Setup

5.4 Model and Temperature

Figure 28 shows the Model and Temperature setup of SPICE testbench. The model is pfnf (PMOS in Fast Mode and NMOS Fast Mode). The temperature is -40 degree C in this example.

```
1.  ************************************************
2.  * Models
3.  ************************************************
4.  .prot
5.  .lib '<library-path> /faa0b_v.mod' pfnf
6.  .unprot
7.  ************************************************
8.  * Temperature
9.  ************************************************
10. .temp -40
```

Figure 28. Model and Temperature Setup

6.0 Delay Time

Figure 29 shows the Input Waveform setup for Input Data and Input Clock, Hierarchical Signal Measurement, and Parameter setup for Clock Slew Rate.

```
1.  ************************************************
2.  * Input Data and CLK Waveform
3.  ************************************************
4.  VB8 B8 0 PWL(0n VP 4N VP 4.001N 0)
5.  VCLK CLK 0 PWL(0n VP 1n VP 1.01n 0 6n 0 6.01n VP 9n VP '9n+ckslew' 0)
6.  .tran 0.005n 30n
7.  ************************************************
8.  * Measure Delay.
9.  ************************************************
10. .measure Tran delay Trig v(CLK) Val = '0.5*(VP)' td = 6n Fall = 1
11. + Targ v(X1.XICORE.XIDACR.XIBINARY.NET156) Val = '0.5*(VP)' td = 6n
    Cross = 1
12. ************************************************
13. * Set Clock Slew Rate: clk[0-2].
14. * Data is inifinity earlier/far.
15. * clk[0] = min, clk[1] = 0.5*max, clk[2] = max.
16. ************************************************
17. .param ckslew = '0.015n*1.25'
18. .alter 'bc slew_1'
19. .param ckslew = '0.695n*1.25'
20. .alter 'bc slew_2'
21. .param ckslew = '1.39n*1.25'
```

Figure 29. Waveforms, Hierarchical Signal Measurements, and Parameter Setup for Delay

The Input Clock can be user defined as Positive Trigger or Negative Trigger. Before the clock triggers the measurement, there is a preset signal for DFF.
The Input Data Waveform can be LH (Low to High) or HL (High to Low). The Input Data Waveform (at 4 nsec) needs to be ready (far ahead) before the triggered measurement (9 nsec). The delay measurement is triggered by the Negative Trigger Clock (Fall = 1 at 9 nsec) and measured by Cross (Cross = 1) command (reaching 50% of VP after 6 nsec). Whether the output is from low to high (Rise = 1) or from high to low (Fall =1) is of no consequence to the measurement.

Since there are two trigger modes (positive and negative triggers) for clocks and two input waveforms (LH and HL) for data (the output measurement mode is fixed at the Cross statement), only four combinations for clock and input data waveforms are considered in the delay measurement. Figure 30 to Figure 33 show the four combinations: CK (Positive Trigger)/Data

(LH), CK (Positive Trigger)/Data (HL), CK (Negative Trigger)/Data (LH), and CK (Negative Trigger)/Data (HL).

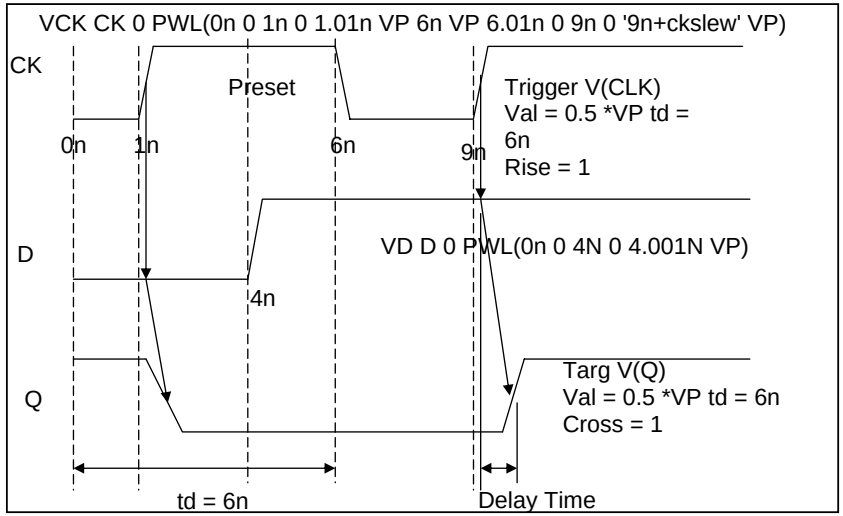

Figure 30. Delay Time Measurement: CK (Positive Trigger) / Data (LH)

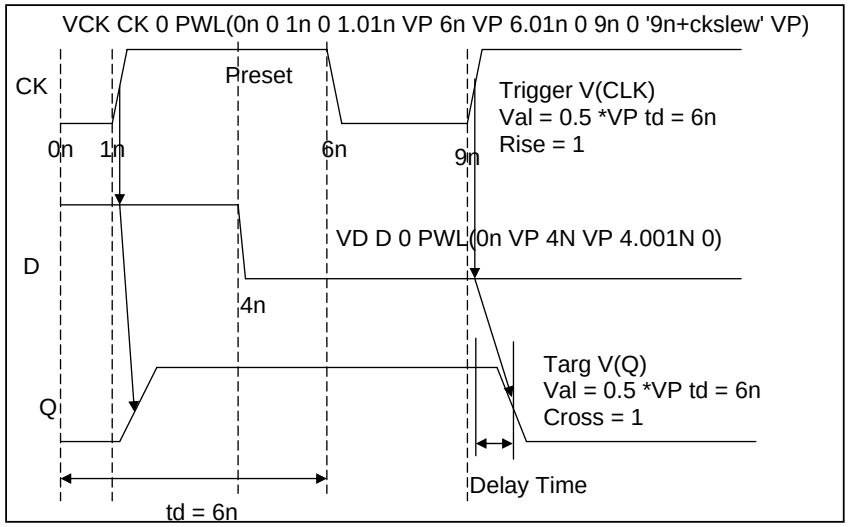

Figure 31. Delay Time Measurement: CK (Positive Trigger) / Data (HL)

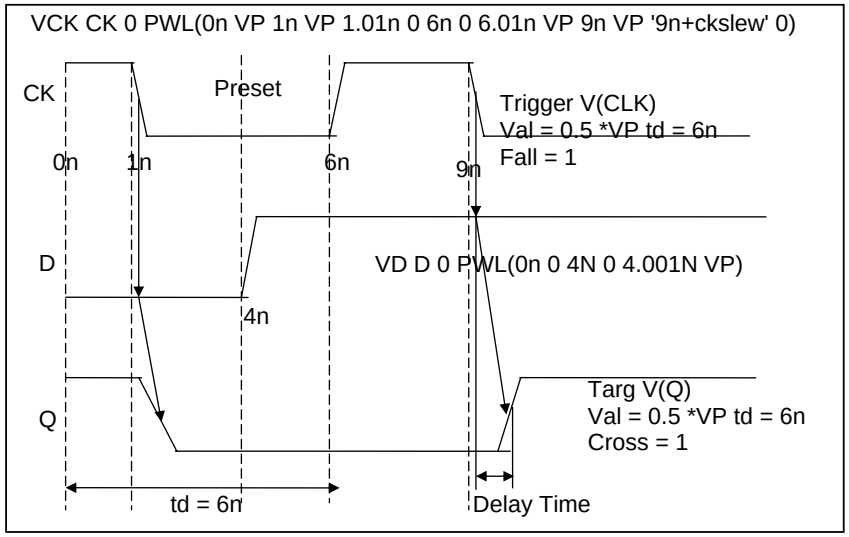

Figure 32. Delay Time Measurement: CK (Negative Trigger) / Data (LH)

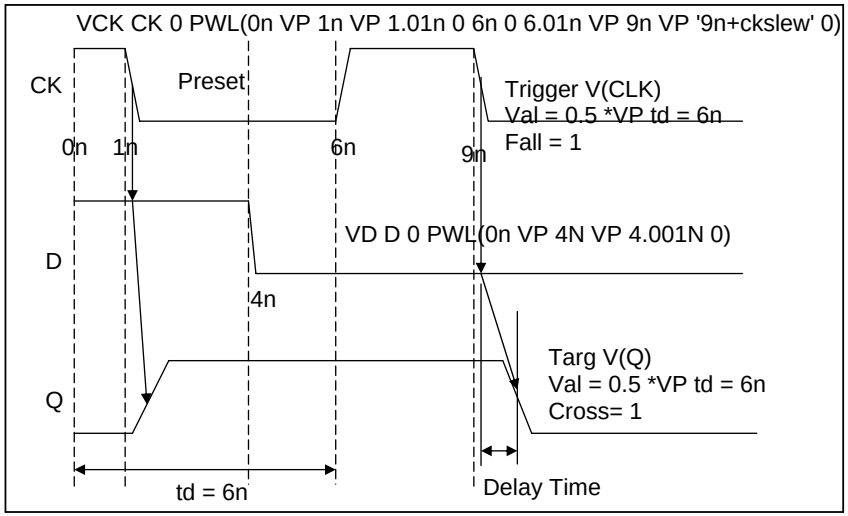

Figure 33. Delay Time Measurement: CK (Negative Trigger) / Data (HL)

In addition, there are three slew rates for clock and three corners (BC, TC, and WC) for processes. The number of delay time results is 2 x 2 x 3 x 3 = 36 (or corresponding test benches running parallelly). For the 10 input pins (R0-R9, G0-G9, B0-B9), there will be 30 x 36 = 1080 test benches.

7.0 Setup Time

The setup time is the amount of time the synchronous input (D) must be stable before the active edge of the clock (CLK). From the SPICE characterization, we measure the output waveform just reaching above half of the input waveform, i.e., 0.5 * VP, as the input (D) waveform to be stable. With this measurement criterion, the time interval between the synchronous input (D) and the active clock (CLK) is called the Hold Time.

Figure 34 shows Input Data Waveform LH and Negative Clock Trigger for Setup Time Measurement. The Input Data (D) is expressed in R8 in this SPICE code.


```
1.  ************************************************
2.  * Optimization Setup.
3.  ************************************************
4.  .param delayStart = -1.9n
5.  .param delayRange = 10.1n
6.  .param delayStop = '9n*1.5+delayRange+daslew+ckslew'
7.  .param delaytime = Opt1 (delayStart, delayStart, delayRange)
8.  .param t0 = '9n+daslew+ckslew*0.5+delay-5p'
9.  .param t1 = '9n+daslew+ckslew*0.5+delay+tcDelta'
10. .param tcDelta = 0.0765n
11. ************************************************
12. * Options
13. ************************************************
14. .PROBE TRAN V
15. .Options POST=1 NoMod AUTOSTOP GMINDC=1E-10
16. .Options WL NOMOD NOPAGE INGOLD=2
17. ************************************************
18. * Input Data Waveform:  Setup Time, LH
19. ************************************************
20. VB8 B8 0 PWL(0n 0 '6n+delaytime' 0 '6n+delaytime+daslew' VP)
21. ************************************************
22. * Input CLOCK Waveform: Negative Clock Trigger Waveform
23. ************************************************
24. VCLK CLK 0 PWL(0n VP 1n VP 1.01n 0 6n 0 6.01n VP '9n+daslew' VP
    '9n+daslew+ckslew' 0)
25. .tran 0.005n delayStop Sweep Optimize = Opt1 Result = voltout Model = OptMod
26. .Measure Tran voltout max v(X1.XICORE.XIDACR.XIBINARY.NET156) from=t0 to=t1
    Goal = 'VP*0.5'
27. .Measure Tran SetupTime Trig v(B8) Val = '0.500*VP' Rise = 1
28. + Targ v(CLK) Val = '0.500*VP' td = 6n Fall = 1
29. .Model OptMod Opt Itropt=15 Method = Bisection relin=1e-6
```

Figure 34. Input Data LH and Negative Clock Trigger for Setup Time Measurement

Line 7 sets the delay time for the bisectional search. The bisectional search is between the -1.9 nsec (1.9 nsec before the clock trigger) and the +10.1 nsec (10.1 nsec after the clock trigger).

Lines 8 and 9 set the t0 and t1 (output measurement window). The parameter delay is measured from the Delay Time Measurement described previously. The delay is the propagation delay of clock from DFF/CLK to DFF/Q.

In line 10, the tcDelta (= 0.0765 nsec) is the input parameter from the IP designer. It is used to set DFF output measurement window (from t0 to t1). This measurement window prevents the low output waveform that causes SPICE to be unable to get any measurement output. The tcDelta should be bigger than half of the waveform slope (0.075 nsec > 0.5 * 0.1 n = 0.05 nsec).

Lines 14-16 are the SPICE Options. The AUTOSTOP option is used to automatically stop (or speed up) the simulation once the result is obtained.

Line 20 sets the input data waveform (LH). The input data (B8) waveform is a moving window controlled by the delay time. The rise time (delay time) is in bisectional pattern [13], i.e., the delay time tried from initial value (-1.9 nsec), then maximum value (10.1 nsec), and then half of previous trial 0.5 * (-1.9 nsec + 10.1 nsec), and so on. It iterates 15 times, until the converged interval (relin = 1.0E-6) is met [13].

Line 24 sets the input clock waveform. The clock contains a negative preset signal and a negative trigger for setup time measurement.

Line 25 performs the transient analysis bisection with .TRAN statement. The transient analysis is performed inside the delayStop with steps of 0.005 nsec. The input clock (CLK) waveform is fixed and the input data (B8) waveform is moving with delay time. The Result (pass or fail) is stored in measurement name (voltout) by line 26.

Line 26 performs the transient analysis with .Measure statement. This statement measures the maximum voltage of output net (X1.XICORE.XIDAR.XIBINARY.NET156) in the interval between t0 and t1. The analysis passes if the maximum output is greater than 0.5*VP. It fails if the maximum output is less than 0.5*VP. The result (pass or fail) is stored in measurement name (voltout).

Lines 27-28 measure the Setup Time between the input data (B8) and the Clock (CLK). The setup time is triggered by first positive (Rise) input data (B8) to the first negative (Fall =1) target clock (CLK).

Line 29 sets the maximum iteration count for bisectional search, bisectional method, and bisectional converged interval (relin = 1.0E-6) [13].

The Setup Time is calculated from the external input Data (B8) and external Clock (CLK). The Clock net may have longer traveling time than the Data net, i.e., the clock has to trigger earlier than data in order to compensate for the traveling delay. If this is the case, we must have Clock input ahead of Data, i.e., the negative setup time for IP is possible.

7.1 Input Data Triggering Window

The input data triggering window is set up by parameter delaytime. The delaytime is determined by the delayStart and delayRange.

The trigger window has to be started a safe range after the preset clock signal. The safe range is used to avoid sampling of unwanted preset clock. It also has to be a certain period after the approaching/triggering data signal with existing clock signal.

7.2 Output Measuring Window

The output measuring window is set up by parameters t0 and tl. The t0 is currently offset by 5 psec. The t1 containing the tcDelta is about 76.5% of the input waveform slope.

7.3 Input Data and Clock Waveform

The Input Waveform D can be LH or HL and the clock can be triggered by positive or negative. Figure 35 to Figure 38 show the four combinations.

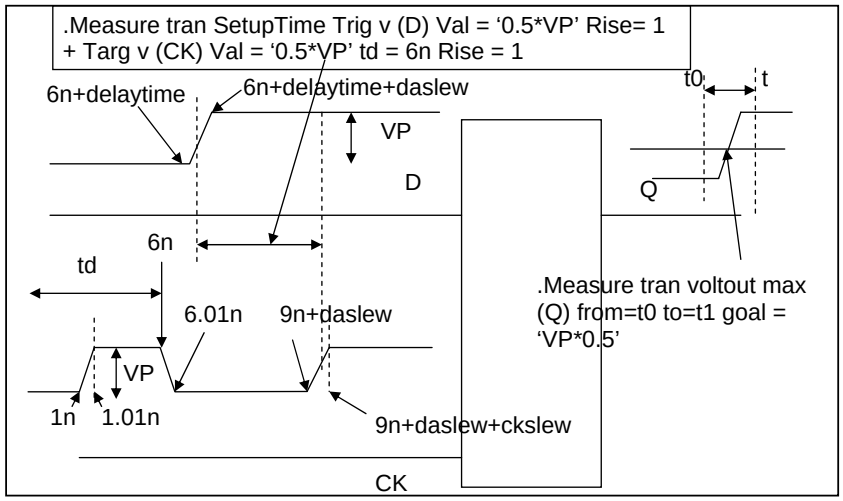

Figure 35. Setup Time (I): Input Data LH and Positive Clock Trigger

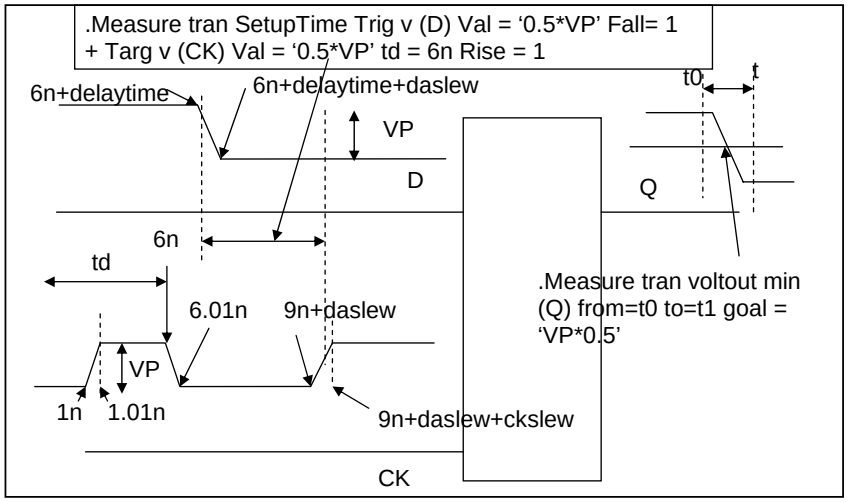

Figure 36. Setup Time (II): Input Data HL and Positive Clock Trigger

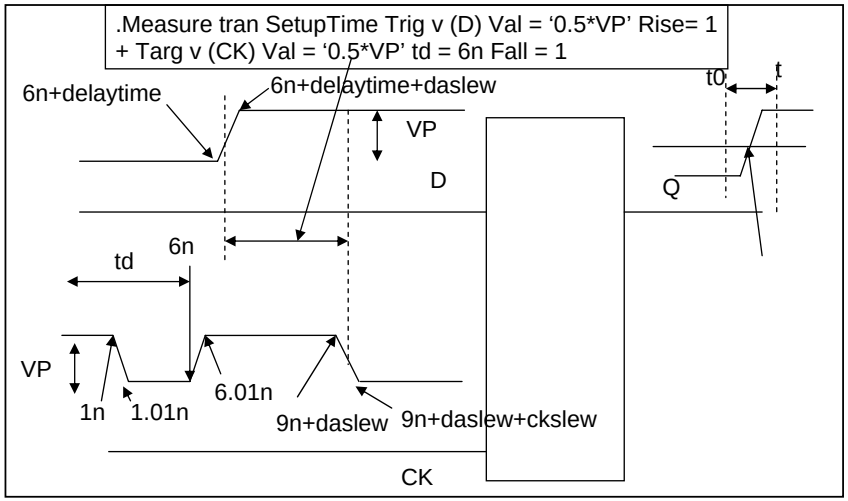

Figure 37. Setup Time (III): Input Data LH and Negative Clock Trigger

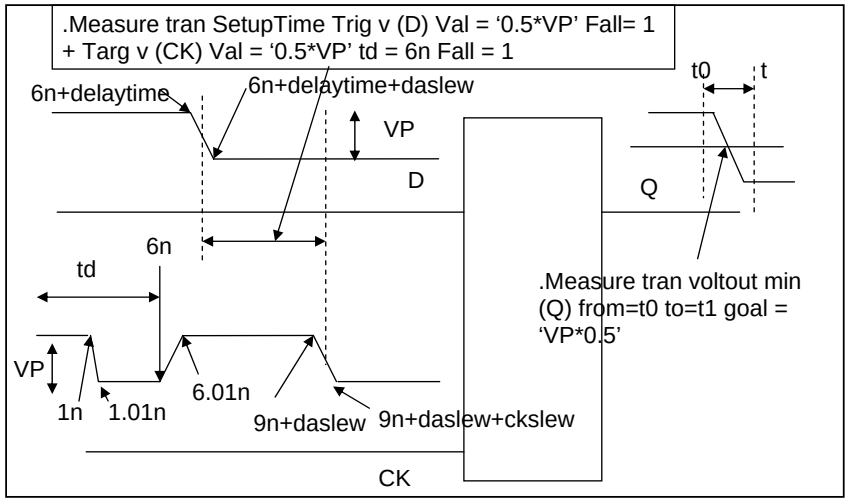

Figure 38. Setup Time (IV): Input Data HL and Negative Clock Trigger

8.0 Hold Time

The Hold Time is the amount of time that the synchronous input (D) must be stable after the active clock (CLK). From the SPICE characterization, we measure the output waveform just reaching below half of the input waveform, i.e., 0.5 * VP, as the input (D) waveform to be stable. With this measurement criterion, the time interval between the synchronous input (D) and the active clock (CLK) is called the Hold Time.

Figure 39 shows the SPICE for a Hold Time measurement.

```
1.  ************************************************
2.  * Input Data Waveform:  Hold Time, LH
3.  ************************************************
4.  VB8 B8 0 PWL(0n 0  0.001N VP '6n+delaytime' VP '6n+delaytime+daslew' 0)
5.  ************************************************
6.  * Input CLOCK Waveform: Negative Clock Trigger Waveform
7.  ************************************************
8.  VCLK CLK 0 PWL(0n VP 1n VP 1.01n 0 6n 0 6.01n VP '9n+daslew' VP
    '9n+daslew+ckslew' 0)
9.  .tran 0.005n delayStop Sweep Optimize = Opt1 Result = voltout Model = OptMod
10. ************************************************
11. * Measure Hold Time: Triggered by Input Data LH
12. ************************************************
13. .Measure Tran voltout min v(X1.XICORE.XIDACR.XIBINARY.NET156) from=t0
    to=t1 Goal = 'VP*0.5'
14. .Measure Tran HoldTime Trig v(CLK) Val = '0.500*VP' td = 6n Fall = 1
15. + Targ v(B8) Val = '0.500*VP' Fall = 1
```

Figure 39. Input Data LH and Negative Clock Trigger for Hold Time Measurement

Line 4 sets the input data (B8) waveform Low to High (LH) with moving window controlled parameter delaytime.

Line 8 sets the clock waveform (CLK).

Line 9 sets the transient analysis bisection with .TRAN statement. It contains the same setting of the setup time as shown in Figure 34.

Line 13 measures the minimum output. It passes if the measurement reaches VP*0.5. It fails if the measurement fails to reach VP*0.5.

Line 14 measures the hold time. The hold time starts from the negative (Fall =1) triggered clock (CLK) and stops at the falling edge of the input data (B8) waveform LH.

Figure 40 to Figure 43 show the four combinations of Input Data and Trigger Clock Waveforms.

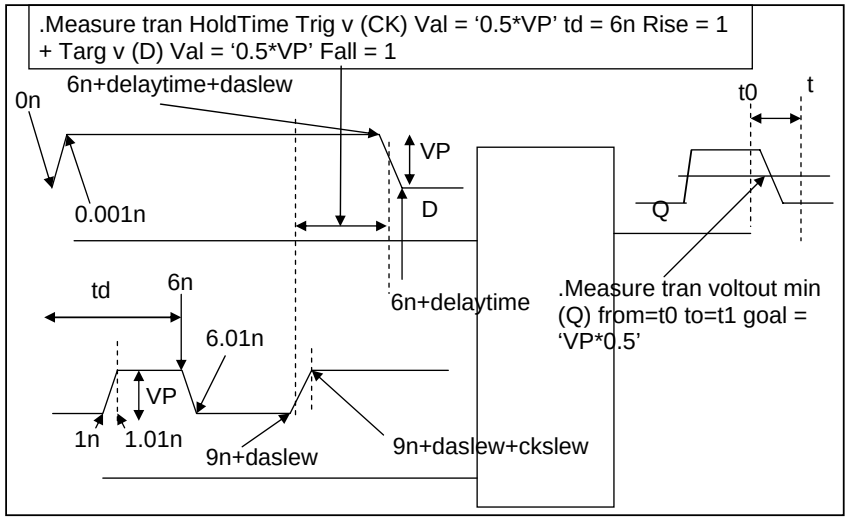

Figure 40. Hold Time (I): Input Data Waveform LH / Positive Triggered Clock

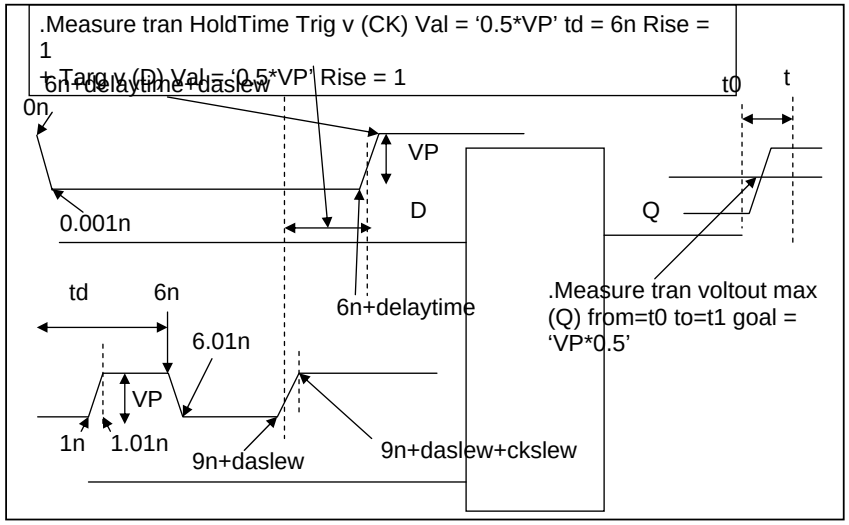

Figure 41. Hold Time (II): Input Data Waveform HL / Positive Triggered Clock

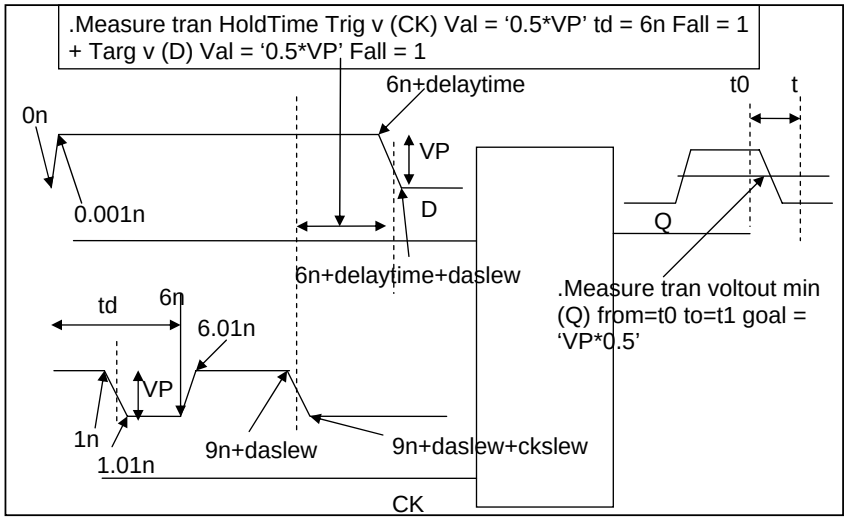

Figure 42. Hold Time (III): Input Data Waveform LH / Negative Triggered Clock

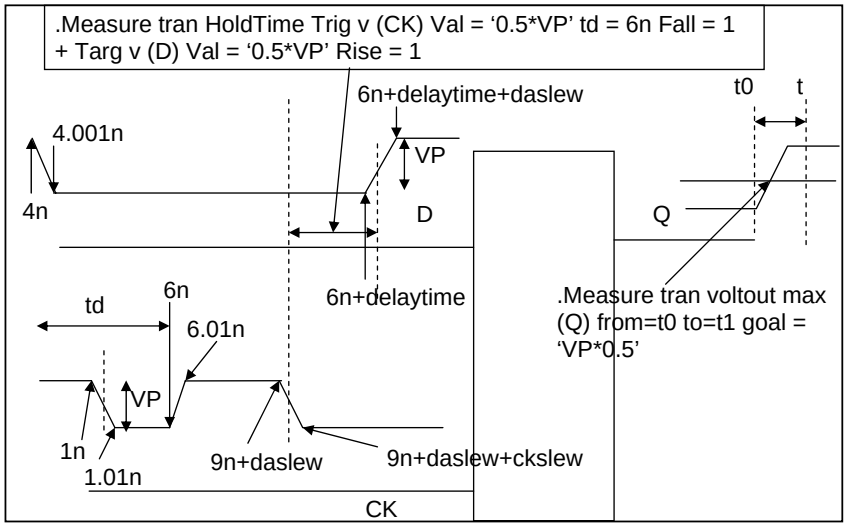

Figure 43. Hold Time (IV): Input Data Waveform HL / Negative Triggered Clock

9.0 Result Comparison

Table 1 shows the Run Time and some typical setup/hold time accuracy comparison of FXDAC030HA0A.

Table 1. Hierarchical Netlist for FXDAC030HA0A

	Full IP Hierarchical Netlist	Clock and Input Pin Partial Path of Hierarchical Netlist	Accuracy
Run time	2.5 hours	Less than 5 seconds	
Setup Time	0.56 nsec	0.55 nec	98%
Hold Time	0.45 nsec	0.44 nsec	98%

10.0 Liberty Library Generation

Figure 44 shows the Synopsys (Liberty) setup/hold time example for input pin (R8) vs. Clock (CLK). The negative clock triggers for setup and hold time are expressed by "setup_falling" and "hold_falling", respectively. The positive clock triggers for setup and hold time are expressed by "setup rising" and "hold_rising", respectively.

The index_1 is a vector with definition of {min, 0.5*max, max} for data slew rate. The index_2 is a vector with definition of {min, 0.5 * max, max} for clock slew rate. The values contain the 3 (index_1) x 3 (index_2) arrays for setup and hold time.
The rise_constraint and the fall_constraint are used to express the Input Data Waveform LH and HL, respectively.

```
1.    pin (B8) {
2.      direction : "input";
3.      is_pad : false;
4.      capacitance : 0.01863;
5.      timing () {
6.        related_pin : "CLK";
7.        timing_type : "setup_falling";
8.        rise_constraint ("CONST_3x3") {
9.          index_1("0.020000,1.000000,2.000000");
10.         index_2("0.020000,0.500000,1.000000");
11.         values("-0.004199,-0.188600,-0.287200", \
12.         "0.045800,-0.082910,-0.210800", "0.046780,-0.090720,-0.218700");
13.       }
14.       fall_constraint ("CONST_3x3") {
15.         index_1("0.020000,1.000000,2.000000");
16.         index_2("0.020000,0.500000,1.000000");
17.         values("-0.033500,-0.212000,-0.339900", \
18.         "0.101500,-0.071190,-0.158100", "0.178600,0.032320,-0.119000");
19.       }
20.     }
21.     timing () {
22.       related_pin : "CLK";
23.       timing_type : "hold_falling";
24.       rise_constraint ("CONST_3x3") {
25.         index_1("0.020000,1.000000,2.000000");
26.         index_2("0.020000,0.500000,1.000000");
27.         values("0.033500,0.212000,0.348700," \
28.         "-0.101500,0.071190,0.196200", "-0.172800,-0.032320,0.119000");
29.       }
30.       fall_constraint ("CONST_3x3") {
31.         index_1("0.020000,1.000000,2.000000");
32.         index_2("0.020000,0.500000,1.000000");
33.         values("0.004199,0.188600,0.304800", \
34.         "-0.066310,0.082910,0.210800", "-0.067290,0.090720,0.209900");
35.       }
36.     }
37.   }
```

Figure 44. Synopsys Setup/Hold Time Liberty Example

11.0 Discussions

This The work in this paper is based on the pre-sim SPICE, i.e., only parameters of Width (W) and Length (L) of the gates are considered. The post-sim SPICE generated by the layout has more parameters, i.e., Area of Source (AS), Area of Drain (AD), Perimeter of Drain (PD), Perimeter of Source (PS), Stress Parameters A, B, C (SA, SB, SC) are additionally given. The post-sim SPICE is more accurate. The pre-sim SPICE can be expanded to include post-sim SPICE for accuracy reason. The methodology discussed in this paper should be the same.

12.0 Acknowledgements

The authors would like to thank Faraday for the full sponsorship of this project and technical writer Ryenne Shih for the careful review of this paper.

13.0 References

[1] Peter H. Chen, Sunil Malkani, Chun-Mou Peng, and James Lin, “Fixing Antenna Problem by Dynamic Diode Dropping and Jumper Insertion,” *Proc. Of ISQED*, pp., 275-282, 2000.

[2] Peter H. Chen, “Beat The Competition: A Knowledge-Based Design Processing Addressing the Antenna Effect and Cell Placement, “*IEEE Circuits and Devices*, pp. 18-27, 20(3)c, June, 2004.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Peter H. Chen, Jim H. Wang, Peter Pong, K.C. Wu, Harrison Liu, K.H. Huang, and Alvin Chen, “Static Power IP Characterization by Automatic Macro Classification and State Reduction,” Patent Pending, June 2005.

[9] 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.

[10] Peter H. Chen, Harrison Liu, Peter Pong, Jim H. Wang, and KC Wu, “Basic IP Characterization,” *SNUG, San Jose, USA*, March 2006.

[11] E. W. Dijkstra, “A Note on twp Problems in Connection with Graph, “ *Numerical Mathematic*, pp. 269-271 (1959).

[12] C. Y. Lee, “An Algorithm for Path Connections and Its Application,” *IRE Transactions on Electronic Computers*, pp. 346-365, September 1961.

[13] “Chapter 2: Timing Analysis Using Bisection,” *HSPICE Application Manuals*, Synopsys Corporation, March 2005.

[14] K. Anshumali, “ACC: Automatic Cell Characterization,” *Proc. of Euro ASIC '91*, pp. 204-209 (May 1991).