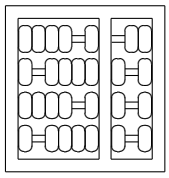

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

FDPE (Fast and Accurate Dynamic Power Estimation): Implementation and User Guidelines

D. Vidal *M. L. Côrtes*

Technical Report - IC-15-01 - Relatório Técnico

February - 2015 - Fevereiro

FDPE (Fast and Accurate Dynamic Power Estimation): Implementation and User Guidelines

Daniel Vidal Mário Lúcio Côrtes

Abstract

This Technical Report presents the implementation details and usage guidelines of FDPE framework. FDPE provides fast and accurate dynamic power estimation for gate-level digital circuits using Liberty characterization data and Verilog simulator.

This framework enables accurate power estimation and achieves a speed-up of three orders of magnitude over Spice. FDPE also enables dynamic power estimation for large scale circuits that are unfeasible to run in Spice simulations. In addition, FDPE has the following advantages over similar tools: *ease of use*, *flexibility, low-cost*, and *public availability.*

Contents

1 Introduction

Analog simulation with tools such as Spice has been used for decades in integrated circuits to accurately estimate current and voltage signals, including dynamic power. However, due to its intrinsic computational complexity, it is unfeasible to run Spice simulations in medium to large circuits. Another simulation approach Fast-Spice [1] was developed, using simplified transistor models and alternative numerical solution methods, resulting in faster simulation at the cost of lower accuracy.

The use of higher level circuit models may produce simulation speed-ups even better than the one achieved with Fast-Spice. One typical high-level circuit model is known as *gate-level*. A gate-level circuit representation describes a circuit as basic logic cells (AND, OR, NAND, etc.) and allows the use of efficient event-driven logic simulation.

This technical report presents the implementation details of a novel framework, named as FDPE [2], which models the dynamic power behavior of digital systems in gate-level simulations and has the following attributes:

- *Accuracy:* as compared to Spice and existing solutions;
- *Speed:* three orders of magnitude speed-up with respect to sign-off quality Spice simulation;
- *Ease of use:* can be easily integrated to the existing design flows; uses standard Liberty format [3] for describing the cells; does not require individual cell characterization;
- *Flexibility and low-cost:* can use any logic simulator, including open source solutions, as long as it is compatible to Verilog and VPI [4];
- *Public availability:* as opposed to other published solutions, our proposal is fully available for public use.

In order to build the dynamic power behavior, a power model is added to the typical behavioral cell model, and physical characteristics (energy and timing) are taken into account.

The report is organized as follows: Section 2 contains a brief explanation about logic cells data formats and modeling technology used by FDPE, Section 3 describes the power model used in FDPE, Section 4 presents the implementation of power model in FDPE framework, Section 5 presents the framework usage guidelines and known limitations, and Section 6 presents conclusions.

2 Logic cells data formats and modeling technology

2.1 Liberty library and characterization data of logic cells

Liberty is an open source gate-level modeling technology, widely accepted by standard cell library vendors and/or foundries, and EDA software. It contains the needed information for circuit implementation such as area, timing (propagation delays and transition times), and power. This section briefly introduces Liberty capabilities that are relevant for FDPE.

Reference [5] presents a guide to cell timing and power Liberty modeling, and [3] provides the full description of all available Liberty modeling and characterization capabilities.

Timing and power can be represented as LUTs (look-up tables) known as NLDM (*Nonlinear delay model)* and NLPM (*Nonlinear power model)*, or as current source know as CCS (*Current Composite Format).* FDPE supports only NLDM/NLPM data, which are widely used for mature technology nodes such as 180 nm or 130 nm.

The most relevant Liberty building blocks are *groups* and *attributes* (Figure 1). A *group* is a named collection of statements that defines a library, a cell, a pin, a timing arc, etc. A pair of braces ({}) encloses the contents of the group. Groups can contain other groups, e.g. a library group contains cells groups. *Attribute* statement defines characteristics of specific objects in the library. Attributes applying to specific objects are assigned within a group statement defining the object, such as a cell group or a pin group [3].

```
group_name (name){
  attribute_name1 : attribute_value1;
  attribute_name2 : attribute_value2;
  group_name2 (name2){
    attribute_name3: attribute_value3;
    attribute_name3: attribute_value3;
  };
};
```

Figure 1: Generic Liberty syntax

A Liberty file is usually composed by a header section, where physical units, characterization conditions, and timing/power look-up tables are defined, and another section where all cells are described. Figure 2 shows an example of Liberty unit description.

```
  /* unit attributes */
  time_unit : "1ns";
  voltage_unit : "1V";
  current_unit : "1mA";
  pulling_resistance_unit : "1kohm";
  leakage_power_unit : "1mW";
  capacitive_load_unit (1.0,pf);
```

Figure 2: Liberty example – description of physical units

Figure 3 shows characterization/operation conditions examples where PVT condition is process *typical* for both p and n MOS transistors, supply voltage of 1.8V, and temperature is 25C.

Attributes `slew_(lower/upper)_threshold_pct_(rise/fall)` specify lower / upper trip points used for rise/fall time characterization. In this example, rise transition time is the time taken by the signal voltage to go from 30% to 70% of the supply voltage. Similarly, fall transition time is the time taken by the signal voltage to go from 70% to 30% of the supply voltage.

Attributes `(input/output)_threshold_pct_(rise/fall)` specify input/output trip points for propagation delay characterization when the signal is rising/falling. In this example propagation delay is the time required for the output to reach 50% of the supply voltage

when the input changes to 50% of the supply voltage.

Attribute `slew_derate_factor` specifies how the transition times recorded in the library need to be derated to match the measured transition times. In this example all transition times are measured as 30% $\leftrightarrow$ 70% of the supply voltage and represented as 10% $\leftrightarrow$ 90% of the supply voltage. The scaling is calculated as:

$$\texttt{slew\_derate\_factor} = \frac{t_{measured}}{t_{represented}} \implies \texttt{slew\_derate\_factor} = \frac{t_{30\%\leftrightarrow 70\%}}{t_{10\%\leftrightarrow 90\%}} \equiv \frac{0.7 - 0.3}{0.9 - 0.1}$$

In NLDM modeling, any physical parameter y (e.g. output transition time, propagation delay, power) is modeled as a function of multiple input variables $x_1, x_2, \ldots, x_n$ (e.g. input transition times, output capacitance). Each modeled parameter is represented in Liberty file as an n dimension table.

In order to build the tables, many combinations of $x_1, x_2, \ldots, x_n$ inputs are simulated and for each simulated combination, y is measured and recorded in a table.

Figure 4 shows NLDM/NLPM table definitions. Both tables in example are 2D, where one index is the input transition time and the other index is total output capacitance. In this section, the index values are meaningless and are used only as placeholders to specify the table dimension, which is 7×7 in this case.

Figure 5 shows an inverter cell definition as an example of Liberty cell description section. In timing and power modeling groups, output pins usually have a `related_pin` which specify the input pin reference, e.g. a AND cell may have many similar timing and power groups: one for each input related pin.

```
/* operation conditions */
nom_process  : 1;
nom_temperature : 25;
nom_voltage  : 1.8;
operating_conditions(tt) {
  process : 1;
  temperature   : 25;
  voltage : 1.8;
  tree_type  : balanced_tree;
}
default_operating_conditions : tt;

/* threshold definitions */
slew_lower_threshold_pct_fall : 30.0; slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0; slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall   : 50.0   ; output_threshold_pct_fall  : 50.0;
input_threshold_pct_rise   : 50.0   ; output_threshold_pct_rise  : 50.0;
slew_derate_from_library   : 0.5    ;
```

Figure 3: Liberty example – description of characterization conditions

```
  lu_table_template(delay_template_7x7) {
    variable_1 : input_net_transition;
    variable_2 : total_output_net_capacitance;
    index_1 ("1000, 1001, 1002, 1003, 1004, 1005, 1006");
    index_2 ("1000, 1001, 1002, 1003, 1004, 1005, 1006");
  }
  power_lut_template(energy_template_7x7) {
    variable_1 : input_transition_time;
    variable_2 : total_output_net_capacitance;
    index_1 ("1000, 1001, 1002, 1003, 1004, 1005, 1006");
    index_2 ("1000, 1001, 1002, 1003, 1004, 1005, 1006");
  }
```

Figure 4: Liberty example – description of NLDM/NLPM tables

```
cell (INVX1L) {
  area : 6.585600;
  pin(Y) {
    direction : output;
    capacitance : 0.0;
    function : "(!A)";
    internal_power() {
      related_pin : "A";
      rise_power(energy_template_7x7) {
     index_1 ("0.040000, 0.080000, 0.158000, 0.318000, 0.638000, 1.280000, 2.570000");
     index_2 ("0.000500, 0.001475, 0.004351, 0.012837, 0.037871, 0.111722, 0.329590");
     values ( \
        "0.009863, 0.009935, 0.010058, 0.010426, 0.011575, 0.014093, 0.020857", \
        "0.010466, 0.010421, 0.010275, 0.010551, 0.011397, 0.013713, 0.020707", \
        "0.011892, 0.011927, 0.012080, 0.011195, 0.011463, 0.012660, 0.019826", \
        "0.015458, 0.015221, 0.014154, 0.012779, 0.012949, 0.013447, 0.020182", \
        "0.022368, 0.022121, 0.021192, 0.019504, 0.017239, 0.009529, 0.011902", \
        "0.037524, 0.036958, 0.035722, 0.032299, 0.029018, 0.023659, 0.024747", \
        "0.069801, 0.069726, 0.069057, 0.059942, 0.055479, 0.046033, 0.039835");
      fall_power(energy_template_7x7) {index_1(...); index_2 (...); values (...);}
    }
    timing() {
      related_pin : "A";
      timing_sense : negative_unate;
      cell_rise(delay_template_7x7) {
        index_1(...); index_2 (...); values (...);}
      rise_transition(delay_template_7x7) {
        index_1(...); index_2 (...); values (...);}
      cell_fall(delay_template_7x7) {
         index_1(...); index_2 (...); values (...);}
      fall_transition(delay_template_7x7) {
         index_1(...); index_2 (...); values (...);}
    }
    max_capacitance : 0.329590;
  }
  pin(A) {
    direction : input;
    capacitance : 0.003324;
  }
  cell_leakage_power : 0.000000024258;
}
```

Figure 5: Liberty example – inverter cell

2.2 Verilog cell behavioral modeling

Verilog is a hardware description language widely used in industry for description, simulation, and synthesis of digital systems in many abstraction levels. The abstraction level of interest in FDPE is named *gate-level,* where the system is represented by cells of a standard logic cell library. This section presents a brief introduction to Verilog cell modeling. Detailed explanations and examples can be found in [6], and a full description of Verilog capabilities can be found in [4].

In order to simulate a circuit mapped to a standard cell library, each used cell must have a behavioral model. A Verilog cell model is usually built using 3 basic functions: *built-in primitives, user defined primitives*, and *timing description.*

Verilog has built-in *gates* and *transmission gates* primitives that can be used to describe basic logic functionality or to build complex logic functions (Figure 6 and Table 1). Complex logic functions (e.g. full-adders, and-or cells, muxes) and memory elements (e.g. latches and flip-flops) are usually described by *user defined primitives* that are primitive logic functions described by logic tables.

Timing description is a group of expressions that defines logic paths and their timing values, and timing checks (e.g. setup, hold, minimum pulse). Figure 7 shows a typical example of an inverter cell Verilog model.

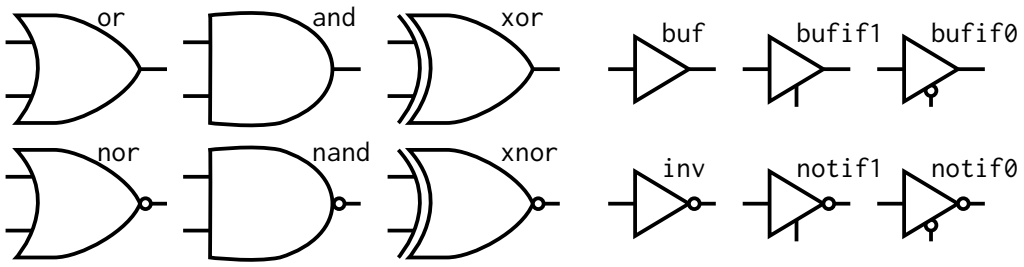

Figure 6: Verilog built-in gates and transmission gates primitives

Table 1: Description of Verilog built-in primitives

gate	description	gate	description
`and`	N-input AND gate	`not`	N-output inverter
`nand`	N-input NAND gate	`buf`	N-output buffer
`or`	N-input OR gate	`bufif0`	Tri-state buffer, active low enable
`nor`	N-input NOR gate	`bufif1`	Tri-state buffer, active high enable
`xor`	N-input XOR gate	`notif0`	Tri-state inverter, active low enable
`xnor`	N-input XNOR gate	`notif1`	Tri-state inverter, active high enable

```
module INVX1L ( Y, A );
   output Y;
   input A;

  //gate primitive
  not I0  (Y , A);

  // timing section
  specify
    // delay parameters
    specparam
      tplh$A$Y = 1,
      tphl$A$Y = 1;
    // path delays
    (A *> Y) = (tplh$A$Y, tphl$A$Y);
  endspecify
endmodule
```

Figure 7: Typical Verilog model for an inverter cell

3 Power model design and description

FDPE power model design decisions and rationale are described as follows:

- *Isosceles Triangular power model* aiming at *accuracy* and *efficiency*. Triangular shape is close to CMOS gates real current waveform [7] and it can be represented efficiently in an event driven simulator [8]. A more accurate model is an asymmetrical triangle shape as used in [8] and [9]. Nevertheless, FDPE approach uses a simpler model that achieves reasonable accuracy.

- *Power estimation at simulation time*, like in [10, 8]. This feature provides *ease of use* and *flexibility*, since post simulation processing and circuit activity database are not required.

- *Liberty format based* as in [9]. This is a widely used format, usually supplied by standard cell library vendors and foundries. It contains all necessary data for transient power estimation, namely timing and energy. This feature provides ease of use as no additional cell characterization is needed.

- *Verilog based*. It is a standard hardware description language and it provides an interface mechanism (VPI) that allows read/write access to internal structures of the circuit [4]. VPI enables communication between models and Liberty data. This feature provides *flexibility* and *low-cost*, as the workflow is similar to typical gate level simulations and any Verilog simulator that supports VPI can be used, including freeware simulators.

- *Leakage neglected*. In modern deep sub-micron CMOS technology leakage power cannot be neglected as it is a significant component of overall power and also can exhibit dynamic behavior. However, this simplification is acceptable for mature technology nodes such as 180nm, which are still widely used.

The following subsections present the FDPE power model design.

3.1 Triangle model discrete representation

The triangle power model is a continuous time signal. To represent it efficiently in an event driven simulation, FDPE uses its second order derivative as in [8], enabling the triangle waveform to be represented by 3 impulse functions. Given the derivative operation linearity, the total power is calculated by superposing all impulse functions occurred during simulation.

Figure 8 shows an example of a composition of two power components, and their derivatives.

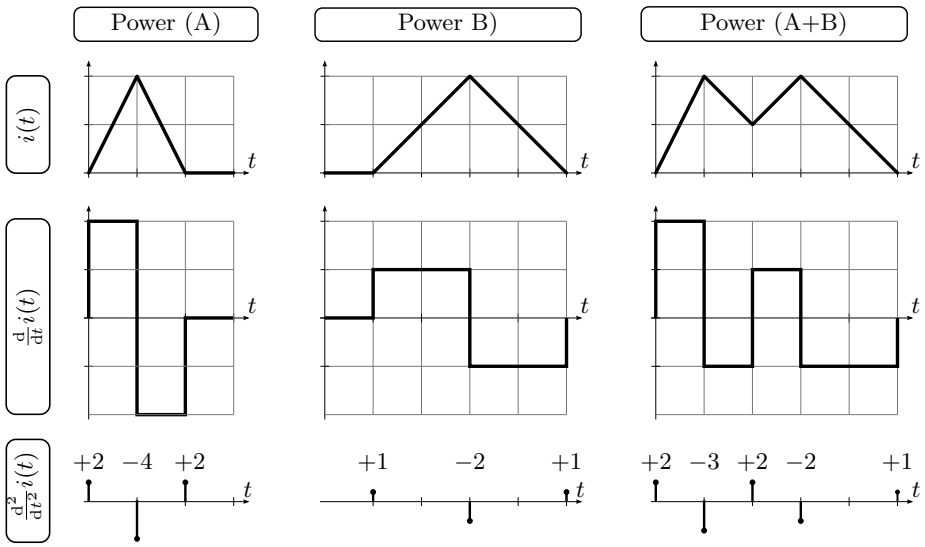

Figure 8: Power composition and representation by discrete second order derivative

3.2 Power model

In order to implement a triangle power signal, the following parameters are needed:

- $T_{\ddot{p}_0}$, $T_{\ddot{p}_1}$, $T_{\ddot{p}_2}$: start time, peak time and stop time respectively;
- $\ddot{p}_0$, $\ddot{p}_1$, $\ddot{p}_2$: second order derivative value at times $T_{\ddot{p}_0}$, $T_{\ddot{p}_1}$, $T_{\ddot{p}_2}$ respectively;
- P_{pk}: peak power value;
- E: total energy drained from power supply during transition, including the energy needed to charge all the output pin loads;
- C_L: Total output pin load capacitance.

Liberty library format provides the following parameters:

- τ_A, τ_Y: input, output transition time respectively. Liberty data usually present 10-90% signal transition time. FDPE uses a 0-100% transition time so care must be taken to scale it properly;

- t_{pd}: input to output propagation delay;
- E_{INT}: energy drained from power supply during transition, considering only MOS short circuit current and energy required to charge internal nodes;
- C_A: input pin capacitance.

In addition, the event time of an input transition, T_A is needed to trigger the power model.

The implemented power model follows an intuitive approach, which can be considered a simplified version of [9]. FDPE power models were developed for two categories of cells: combinational and sequential which are presented as follows.

3.2.1 Combinational model

Figure 9 shows the power model and its parameters. Model events and parameters relations are detailed as follows.

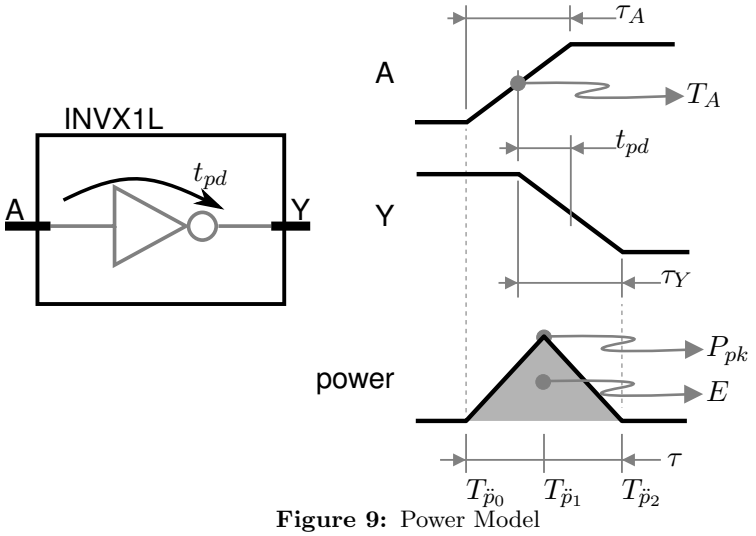

Figure 9: Power Model

Event T_A is detected by an event monitor that is embedded into Verilog behavioral models. As soon as T_A is detected, all other events $T_{\ddot{p}_0}$, $T_{\ddot{p}_1}$ and $T_{\ddot{p}_2}$ are scheduled based on the actual simulation time. Scheduling $T_{\ddot{p}_0}$ after T_A is equivalent to schedule an event in the past. To overcome this difficulty in event driven simulators, all events are delayed by a fixed time and, as the time shift is known, the delay can be removed later for data analysis.

As energy related to dynamic power is consumed during signal transitions, the power triangle start time $T_{\ddot{p}_0}$ is equal to the time that input signal crosses V_T (for rising edge) or VDD-V_T (falling edge) as stated in (1); the stop time $T_{\ddot{p}_2}$ is equal to output pin stop time (2), and the peak time $T_{\ddot{p}_1}$ is calculated as in (3).

$$
\begin{aligned}
T_{\ddot{p}_0} &= T_A - (\tau_A/2) \times \frac{VDD - V_T}{VDD} && (1)\\
T_{\ddot{p}_2} &= T_A + t_{pd} + (\tau_Y/2) && (2)\\
T_{\ddot{p}_1} &= (T_{\ddot{p}_0} + T_{\ddot{p}_2})/2 && (3)
\end{aligned}
$$

In case an input transition does not cause an output toggle, e.g. a transition at one input of an AND cell with the other input at zero, the stop time is considered to be the input slope stop time (4).

$$T_{\ddot{p}_2} = T_A + (\tau_A/2) \quad (4)$$

The energy drained from power supply is equal to the area of its power signal waveform [7]. By geometry, P_{pk} can be calculated as in (5).

$$E = \frac{P_{pk} \times \tau}{2} \implies P_{pk} = \frac{2 \times E}{\tau} \quad (5)$$

If an input signal transition does not cause an output toggle, or it causes a high to low transition on cell output, total energy is equal to internal energy only (6).

$$E = E_{INT} \quad (6)$$

If an input signal transition causes an output transition from low to high, total energy is equal to internal energy plus the energy needed to charge the total load capacitance (7).

$$E = E_{INT} + E_L \quad (7)$$

$$E_L = \frac{V_{DD}^2 \times C_L}{2} \quad (8)$$

Load capacitance C_L of an output pin is computed as the sum of input capacitance C_A of all inputs connected to that output net. Second order derivatives $\ddot{p}_0$, $\ddot{p}_1$,$\ddot{p}_2$ are calculated based on triangle geometry and symmetry:

$$
\begin{aligned}
\ddot{p_0} &= \frac{P_{pk}}{\tau/2} && (9)\\
\ddot{p_1} &= -2\ddot{p_0} && (10)\\
\ddot{p_2} &= \ddot{p_0} && (11)
\end{aligned}
$$

3.2.2 Sequential model

The power model for sequential cells is built similarly to combinational cells: the triangle start, peak and stop times are determined by signal transition time. However, in this case there is one power model for each pin as each pin has its own related energy information in Liberty data. Each pin type is explained as follows.

Clock Start time is considered to be as in equation (1) and stop time is considered to be as in equation (12), according to [9]. T_{CK} is the clock pin transition time, $t_{CK\rightarrow Q}$is the clock to output delay and τ_Q is output transition time.

$$T_{\ddot{p}_2} = T_{CK} + t_{CK\rightarrow Q} - (\tau_Q/2) \quad (12)$$

Data Input Start, stop and peak time are assumed to be similar to the combinational cell, when the input does not cause an output transition, as stated in equations (1), (4), and (3).

Data output Start and stop time are considered to match the transition start and end, and are calculated as in equations (13) and (14).

$$T_{\ddot{p}_0} = T_Q - (\tau_Q/2) \quad (13)$$
$$T_{\ddot{p}_2} = T_Q + (\tau_Q/2) \quad (14)$$

The internal energy related to data output pin, E_{INT_Q}, stored in Liberty includes the energy related to clock pin, $E_{INT_{CK}}$. Therefore, it is necessary to subtract $E_{INT_{CK}}$ from E_{INT_Q} in order to calculate the energy considered to build the triangle model as shown in equation (15) for rising transition, and as in equation (16) for falling transition.

$$E = E_{INT_Q} - E_{INT_{CK}} + E_L \quad (15)$$
$$E = E_{INT_Q} - E_{INT_{CK}} \quad (16)$$

4 FDPE implementation

4.1 Verilog model modifications

In order to include the power model of section 3, Verilog cell models had to be modified. The parameters stated in section 3.2.1 were implemented as Verilog variables, mapped according to Table 2. Additionally, two Verilog variables and two Verilog parameters were needed for transition annotation (Section 4.2.1):

- `n_in` : total number of input pins;
- `t_count` : number of input pins that had transition times annotated;
- `t_ready` : output pin transition time annotated (`1`: annotated, `0`: not annotated)
- <pin>`_sense` : signal propagation unateness[1] relation between output and input <pin> (`1`: positive unate, `-1`:negative unate, `0`: non-unate)

Table 2: Variables and parameters to be added to a typical Verilog cell model

Verilog Variable	**Parameter** (section 3.2.1)	**Description**
<pin>_cap	C_L	total fanout capacitance connected to <pin>
e<pin>_load	E_L	energy needed to charge the fanout capacitance of <pin>
e<pin>_<rpin>_<r|f>	E_{INT}	Internal energy on a rising edge of <pin> due to a <rpin> transition
d<pin>_<rpin>_<r|f>	t_{pd}	propagation delay
t<r|f><pin>	τ_{IN}	rise|fall transition time of <pin>
t<r|f><pin>_<rpin>	τ_{OUT}	rise|fall time of <pin> due to a <rpin> transition
Ts_<pin>_<rpin>_<r|f>	$T_{\ddot{p}_0}$	power model start time for a rise|fall transition at <pin> due to a <rpin> transition
Te_<pin>_<rpin>_<r|f>	$T_{\ddot{p}_2}$	power model stop time for a rise|fall transition at <pin> due to a <rpin> transition
T_<pin>_<rpin>_<r|f>	τ	power model duration for a rise|fall transition at <pin> due to a <rpin> transition

```
real Y_cap, eY_load;
real eY_A_r, eY_A_f ;
real dY_A_r, dY_A_f;
real trA, tfA;
real trY_A, tfY_A;
real Ts_Y_A_f, Te_Y_A_f, T_Y_A_f,
real Ts_Y_A_r, Te_Y_A_r, T_Y_A_r;

parameter A_sense = -1;
parameter n_in = 1;
integer t_count;
integer t_ready;
```

Figure 10: Extra Verilog variables and parameters for an inverter cell with output pin Y and input pin A

Figure 10 shows a Verilog declaration example of additional variables and parameters needed for the inverter cell of Figure 7.

The model must be able to predict when an output pin toggles, therefore the logic primitive (section 2.2) is duplicated with zero delay and buffers are inserted to isolate each input pins as illustrated in Figure 11.

[1]Unateness specifies how the edges (transitions) can propagate through a cell and how they appear at the output of the cell. The signal propagation is *positive unate* if a rising transition on an input causes the output to rise (or not to change) and a falling transition on an input causes the output to fall (or not to change). A *negative unate* signal propagation is one where a rising transition on an input causes the output to have a falling transition (or not to change) and a falling transition on an input causes the output to have a rising transition (or not to change). In a *non-unate* signal propagation, the output transition cannot be determined solely from the direction of change of an input but also depends upon the state of the other inputs. [5]

```
// input module port isolation
buf (Ax,A);

// cell primitive
not I0  (Y , Ax);

// cell primitive copy with zero delay
not I0x (Yx, Ax);
```

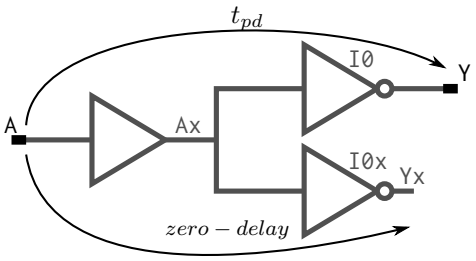

Figure 11: Verilog primitive duplication and input isolation for an inverter cell

In order to trigger the power models, events monitors are needed to analyze input/output pins values and their relationship.

Figure 12 and 13 shows the event monitor of an inverter cell and a two input NAND cell, respectively. FDPE power models are implemented as Verilog macros as represented in figure 14, where the system task `$fdpe_power_update` updates the power waveform at times $T_{\ddot{p}_0}$, $T_{\ddot{p}_1}$, $T_{\ddot{p}_2}$ using $\ddot{p}_0$, $\ddot{p}_1$, $\ddot{p}_2$ power second order derivatives values (section 3.2.1).

Care must be taken with Verilog/Liberty time units. The implemented framework used Verilog *timescale*[2] matched to Liberty time unit. If Verilog timescale cannot be matched to Liberty time unit, a scale factor must be applied to model delay annotations.

```
    always @ (A) begin
       if      (Yx===1) `fdpe_mdl(Ts_Y_A_r, T_Y_A_r,(eY_A_r+eY_load))
       else if (Yx===0) `fdpe_mdl(Ts_Y_A_f, T_Y_A_f,eY_A_r)
    end
```

Figure 12: Inverter power model trigger

```
always @ (A) begin
    if (B === 1'b1) begin
      if      (Yx===1) `fdpe_mdl(T_start_Y_A_r, T_Y_A_r, (eY_A_r+eY_load))
      else if (Yx===0) `fdpe_mdl(T_start_Y_A_f, T_Y_A_f, eY_A_r  )
    end
end
always @ (B) begin
  if (A === 1'b1) begin
    if      (Yx===1) `fdpe_mdl(T_start_Y_B_r, T_Y_B_r, (eY_B_r+eY_load) )
    else if (Yx===0) `fdpe_mdl(T_start_Y_B_f, T_Y_B_f, eY_B_r)
  end
end
```

Figure 13: NAND power model trigger

[2]Verilog compiler directive used to specify the time unit and precision of the delays.

```
`define fdpe_mdl2(d_,t_,dde_) \
     begin \
        #(d_);   $fdpe_update_pwr(dde_);\
        #(t_/2); $fdpe_update_pwr(-2*dde_);\
        #(t_/2); $fdpe_update_pwr(dde_);\
     end

`define fdpe_mdl(dly_,tau_,e_) `fdpe_mdl2(dly_,tau_, (4*e_/(tau_**2)) )
```

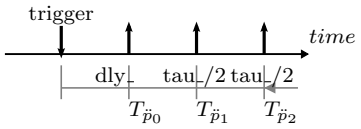

Figure 14: FDPE power model

```
// transition measurement threshold scale
// TR = vh-vl
// Ex: 10%-90%  transition -> TR = 0.9 - 0.1 = 0.8
`define TR  0.8

// FDPE delay offset
`define XX 3

// VDD used for Liberty characterization
`define VDD  1.8

// transistor VT
`define VT 0.55
```

Figure 15: FDPE model defines

4.2 VPI functions

4.2.1 Initialization

The FDPE model parameters represented by Verilog variables (section 4.1) are resolved by a VPI function at simulation initial time. The VPI initialization function reads the circuit in simulator database, and for each cell fan-out capacitance, transition times, and propagation delays are computed and annotated to power models according to the procedure in figure 16. The parameters are computed based on the Liberty data, which are retrieved by an open library parser.

The parameters are calculated by interpolation of look-up tables recorded in Liberty file. The interpolation procedure is detailed as follows.

LUT interpolation one-dimension case NLDM/NLPM are recorded in Liberty files as tables of discrete points $y[k]$, where each $y[k]$ corresponds to an $x[k]$. Therefore, for any x within $\langle x[1], x[k]\rangle$, y can be approximated by (17) which is a linear approximation, as illustrated in figure 17. Notice that this approximation is similar to the first two elements of a Taylor series approximation for a function $f(x)$ stated in (18).

$$y = y[k] + \frac{y[k+1] - y[k]}{x[k+1] - x[k]} \times (x - x[k]) \tag{17}$$

$$f(x) \approxeq f(x_0) + f_x(x_0) \times (x - x_0) \tag{18}$$

```
1  begin
2      Open Liberty library;
3      foreach cell instance of circuit do
4          foreach output pin of cell instance do  compute total fanout capacitance and
           annotate to power model ;
5      repeat
6          foreach cell instance of circuit do
7              if  τ_out has not been annotated and all τ_in have been annotated then
8                  foreach output pin of cell instance do
9                      Compute τ_out and annotate to model;
10                     Propagate τ_out for all fanout pins;
11     until all output pins transitions τ_out have been annotated;
12     foreach cell instance of circuit do compute propagation delays and annotate to model ;
13     foreach cell instance of circuit do
14         foreach pin of cell instance do  compute pin energy and annotate to model ;
       // Initialize variables for power calculation
15     t_{n-1} ← 0 ;
16     p_{n-1} ← 0 ;
17     ṗ_{n-1} ← 0 ;
```

Figure 16: Initialization procedure

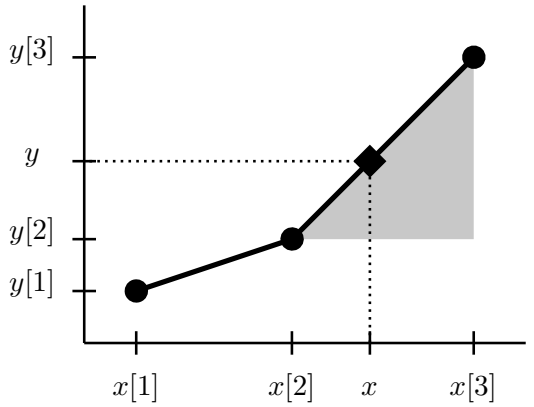

Figure 17: One-dimension Liberty table representation

LUT interpolation two-dimension case In this case, NLDM/NLPM are recorded in Liberty files as tables of discrete points $y[k_1, k_2]$, where each $y[k_1, k_2]$ corresponds to a $(x_1[k_1], x_2[k_2])$ pair. Therefore, for any (x_1, x_2) pair within $(\langle x_1[1], x_1[k_1]\rangle, \langle x_2[1], x_2[k_2]\rangle)$ range, y can be approximated by (19) which is a two dimension approximation similar to (17). Notice that this approximation is similar to the first two elements of a Taylor series approximation for a function $f(x_1, x_2)$ stated in (20).

	$x_2[1]$	$x_2[2]$	$\dots$	$x_2[j]$
$x_1[1]$	$y[1,1]$	$y[1,2]$	$\dots$	$y[1,j]$
$x_1[2]$	$y[2,1]$	$y[2,2]$	$\dots$	$y[2,j]$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_1[i]$	$y[i,1]$	$y[i,2]$	$\dots$	$y[i,j]$

Figure 18: Two-dimension Liberty table representation

$$y = y[k_1, k_2] + \frac{y[k_1+1, k_2] - y[k_1, k_2]}{x_1[k_1+1] - x_1[k_1]} \times (x_1 - x[k_1]) + \frac{y[k_1, k_2+1] - y[k_1, k_2]}{x_2[k_2+1] - x_2[k_2]} \times (x_2 - x[k_2]) \quad (19)$$

$$f(x_1, x_2) \cong f(x_{1_0}, x_{2_0}) + f_{x_1}(x_{1_0}, x_{2_0}) \times (x_1 - x_{1_0}) + f_{x_2}(x_{1_0}, x_{2_0}) \times (x_2 - x_{2_0}) \quad (20)$$

4.2.2 Power update

Whenever the power waveform has to be updated by `$fdpe_power_update` system task (figure 14), the power second order derivative has to be double integrated. The figure 19 shows the procedure for power update.

Input: Power second order derivative $\ddot{p}$, simulation time t

1 **begin**
2 $t \leftarrow$ Actual simulation time;
3 $\Delta t \leftarrow t - t_{n-1}$; `// time since last update`
4 $\dot{p}_n \leftarrow \dot{p}_{n-1} + \ddot{p}$; `// second order derivative integration`
5 $p_n \leftarrow p_{n-1} + \dot{p}_{n-1} \times \Delta t$; `// first order derivative integration`
6 Update power waveform with p_n value;
`// prepare variables for next function call`
7 $t_{n-1} \leftarrow t$;
8 $p_{n-1} \leftarrow p_n$;
9 $\dot{p}_{n-1} \leftarrow \dot{p}_n$;

Figure 19: Power update procedure

5 Usage and known limitations

5.1 Workflow

The required workflow for FDPE (Fig. 20) does not differ from a standard Verilog gate-level simulation. The simulator takes as inputs:

- Gate level circuit definition: a Verilog netlist and annotated delays (SDF format);
- Modified Verilog library, which contains cell behavioral models and stated modifications for power estimation (section 4.1);
- VPI Library, which contains the functions described in section 4.2 . The VPI library does not need to be modified or recompiled for different circuits or Liberty libraries.

The power waveform is recorded in a Verilog variable and can be viewed and/or analyzed during the simulation or when the simulation has finished.

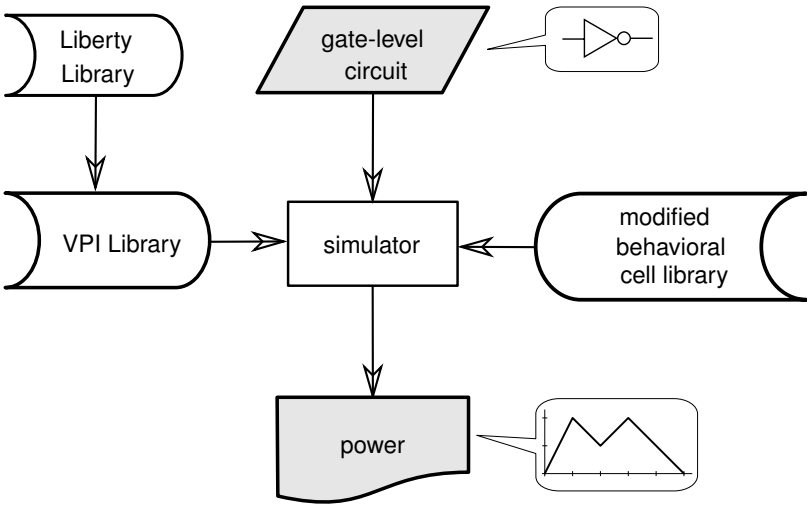

Figure 20: FDPE Workflow

5.2 Testbench requirements

From the user point of view, FDPE expects a Verilog real variable in order to record the power waveform and a call to initialization procedure (Figure 16) after the testbench signals have been initialized. Figure 21 shows a Verilog example with FDPE requirements. The initialization function arguments are the power waveform variable and transition time to be annotated at circuit inputs.

```
module testbench();

  reg  reset, in_data;
  wire output_data;

  real pwr_var;          // power waveform variable

  parameter tau = 1.27 ; // default transition value in ns
  parameter cap = 0.01 ; // default circuit output capacitance in pF

  dut_inst dut_module (
                        .in   (in_data)    ,
                        .reset(reset)      ,
                        .out  (output_data)
                      );
  initial begin
    in_data = 1'b0;          //testbench/DUT signl initialization
    reset   = 1'b0;          //testbench/DUT signl initialization
    $fdpe_init("liberty_file.lib",
               "library_name",
               pwr_var,
               tau,
               cap ); //FDPE initialization

    //More code here for DUT stimuli....

  end
endmodule
```

Figure 21: Testbench example

5.3 Model time offset adjustment

Depending on timing characteristics of used cells, the waveform offset (section 3.2.1) may need to be adjusted, in order to avoid negative delays during simulation.

Notice that the waveform time offset cannot be very large because there is a known limitation regarding the waveform time offset (see section 5.4.2).

5.4 Known Limitations

5.4.1 Combinational loops

FDPE does not support combinational loops (e.g. Figure 22). Transition times of pins in a combinational loop path cannot be computed in the initialization procedure (section 4.2.1) because cell input pins transition time must be annotated prior to output transition time, and output transition time depends on the input pins transition time.

An alternative to overcome this limitation is to implement a combinational loop detector, break the loop at a cell input pin, and annotate a default transition time at the broken connection. Another alternative is to implement a combinational loop detector and ask the user where to break the loop, and let the user choose the missing transition time value.

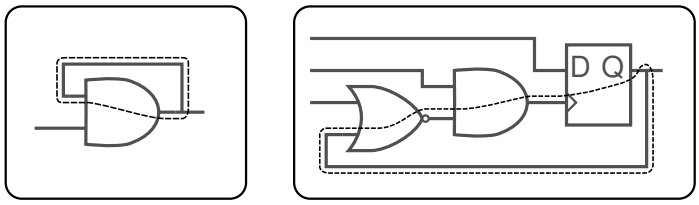

Figure 22: Combinational loop examples

5.4.2 Signal toggling vs offset time

All events in power model are delayed by a fixed offset (section 3.2.1). If any power model event, triggered by an input transition is scheduled for a time later than the next signal toggle, the power model trigger may miss a power event. Figure 23 shows an example of this limitation.

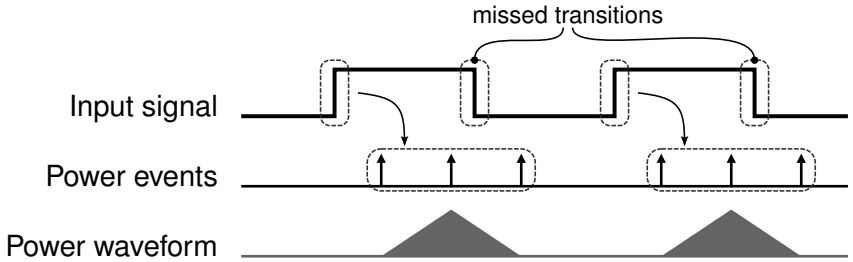

Figure 23: Signal toggle frequency vs. offset time limitation.

6 Conclusions

This Technical Report presented the FDPE framework implementation and its usage guidelines. This framework is publicly available for download and open for contributions under the *GNU GPL* license at *Sourceforge* community.

FDPE source code, some circuit examples, instructions to run FDPE [3], and any additional documentation produced after the publication of this report can be found at:

- http://fdpe.sourceforge.net

Finally, the authors understand that there is room for improvements in the present FDPE implementation. The current envisioned improvements may be implemented in a future version and are listed below:

- Dynamic model refinement regarding to triangular model for better accuracy and higher Spice correlation;
- Leakage power inclusion in the power model implementation, in order to increase FDPE accuracy and to enable its use for newer technology nodes;
- A tool to fully automate the Verilog cells modifications.

[3] Instructions are given in README and Make files

References

[1] M. Rewieński, "A perspective on fast-spice simulation technology," in *Simulation and Verification of Electronic and Biological Systems*, pp. 23–42, Springer, 2011.

[2] D. Vidal and M. Côrtes, "Fast and Accurate Solution for Power Estimation and DPA Countermeasure Design," in *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2014 24th International Workshop on*, pp. 1–7, Sept 2014.

[3] Liberty, "User guide and reference manual suite." `http://www.opensourceliberty.org`. Retrieved: 2015-01-10.

[4] IEEE, "Verilog hardware description language," *IEEE Std 1364-2001*, pp. 0–856, 2001.

[5] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer, 2009.

[6] D. K. Tala, "Verilog tutorial." `http://www.asic-world.com/verilog/veritut.html`. Retrieved: 2015-01-10.

[7] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. ADDISON WESLEY Publishing Company Incorporated, 2011.

[8] A. Bogliolo, L. Benini, G. De Micheli, and B. Ricco, "Gate-level power and current simulation of cmos integrated circuits," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 473–488, 1997.

[9] M.-S. Lee, C.-H. Lin, C.-N. J. Liu, and S.-C. Lin, "Quick supply current waveform estimation at gate level using existed cell library information," in *Proc. of the 18th ACM Great Lakes Symp. on VLSI*, GLSVLSI '08, (New York, NY, USA), pp. 135–138, ACM, 2008.

[10] M. A. Cirit, "Powerteam™: There is more to verilog beyond behavioral simulation," 2004.