Efficient Generation of Timing and Power Polynomial Models from Lookup Tables for SoC Designs

Gaofeng Wang and Runip Gopisetty

Synopsys, Inc.
700 East Middlefield Road
Mountain View, CA 94043, USA

Abstract — A new scheme is presented for generating optimal timing and power models, which can speed up timing and power analyses with full accuracy in system-on-chip (SoC) designs. In this scheme, the nonlinear multi-dimensional timing and power lookup tables in semiconductor libraries are transformed into optimized (piecewise) polynomial equations in an efficient and accurate manner. The transform problem is mathematically defined as a least square problem, which is efficiently solved by a set of robust numerical algorithms. These optimized polynomial equations are then represented using the delay and power calculation language (DPCL), which can be complied into object code and used by various EDA tools.

I. Introduction

The timing and power models in gate-level libraries are commonly used by various electronic automation design (EDA) tools to analyze and optimize the design for fabrication by minimizing a cost function, which is usually a combination of the area, speed and power of the design. The reason for this gate-level abstraction is that circuit level simulation tends to be too slow for practical purposes.

The most commonly used timing models are the nonlinear timing models, in which the data is represented by multi-dimensional tables. The timing or power data table has a characterization domain which defines the independent variables that influence timing and power.

Currently, a timing table such as those in Synopsys libraries [1] typically employs a two dimensional table for combinational elements, where the independent variables are input transition time and output load. For an unbuffered sequential element, a three dimensional time table has been commonly used where the independent variables are clock transition time, data transition time and output load. A set of sampling points are selected on the multi-dimensional independent variable domain. The timing is characterized at each of the sampling points using the circuit simulator, such as HSPICE. The resulting data is then reported as a timing data table in a semiconductor library for the EDA tools to use.

This works fine for small tables where the computational complexity can be easily handled. However, the amount of data presented in a table could be very large and has been rapidly growing due to two problems. Firstly, in order to accurately model the inherent complex behaviors such as nonlinearities of the circuit element, a large amount of sampling points must be adopted to provide sufficiently fine meshes over the characterization domain so that the irregularities of the timing or power data surface can be properly addressed.

Secondly, the table size goes up quadratically as a function of the number of dimensions, which causes severe penalties in runtime and memory. For deep sub-micron (DSM) design, the number of dimensions are increasing rapidly. In particular, additional dimensions such as temperature and voltage are playing an increasingly important role in system-on-chip (SoC) designs where different regions of a chip are operating at different temperatures and voltages. Other additional independent variables could include the resistance, inductance or shared pin load in the case of unbuffered multi-outputs, etc.

A table compaction technique using dynamic programming was proposed in [2] to reduce the table size and at the same time try to maintain the data accuracy by removing possible oversampling points in the linear regions of the data surface. For the nonlinear regions, however, the number of sampling points must be kept in the optimized characterization table to maintain the data accuracy which often results in insufficient compression.

In this paper, we propose a robust technique for transforming these large multi-dimensional tables into computationally efficient polynomial equations without loss of accuracy. The resulted polynomial equations, which have considerably fewer coefficients than the sizes of original data tables, provide significant data reduction and improvement in computational efficiency.

The resulted polynomial equations are represented in a format of a language called Delay and Power Calculation Language (DPCL) [3]. The DPCL is providing a new approach to the delay and power calculation and offering a single, high-performance interface among the EDA tools [4]. These DPCL equations are subsequently compiled into native code by the DPCL compiler. Because of these equations are complied rather than interpreted by the EDA tools, a further speedup is offered during the runtime.

II. Formulations and Numerical Algorithms

A. Least Square Problem

Given a d-dimensional timing or power data table of size n

as follows:

$$\begin{matrix} \{x_1^{(1)}, x_2^{(1)}, \ldots, x_d^{(1)}, z^{(1)}\} \\ \ldots\ldots \\ \{x_1^{(n)}, x_2^{(n)}, \ldots, x_d^{(n)}, z^{(n)}\} \end{matrix} \tag{1}$$

where the timing or power data at each sampling point is denoted within a pair of parentheses, in which the first *d* components represent the values of each independent variable whereas the last component denotes the timing or power value. The objective is to find an optimal polynomial $P_k(x_1, x_2, \ldots, x_d)$ of *k* terms:

$$P_k(x_1, x_2, \ldots, x_d) = \sum_{j=1}^{k} c_j \cdot b_j(x_1, x_2, \ldots, x_d) \tag{2}$$

such as

$$\min_{\{c_j\}} \sum_{i=1}^{n} [z^{(i)} - P_k(x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})]^2 \tag{3}$$

where $b_j(x_1, x_2, \ldots, x_d) = (x_1)^{n_{j1}} \cdot (x_2)^{n_{j2}} \cdot \ldots \cdot (x_d)^{n_{jd}}$ ($j = 1, 2, \ldots, k$ and $n_{j1}, n_{j2}, \ldots, n_{jd}$ are non-negative integers) are the polynomial basis functions, and c_j ($j = 1, 2, \ldots, k$) are the unknown coefficients to be determined by solving the above least square problem.

The above minimization problem can be written in the matrix form as:

$$\min_{\xi} \|z - A\xi\|_2^2 \tag{4}$$

where *A* is an *n*-by-*k* matrix, *z* is a vector of size *n* and ξ is a vector of size *k*. The elements of the matrix and vectors are given by

$$\begin{aligned} A_{ij} &= b_j(x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}) \\ z_i &= z^{(i)} \\ \xi_j &= c_j \end{aligned} \tag{5}$$

where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$.

The case in which we are really interested is that the number of basis functions in the polynomial is much smaller than the table size, i.e., $k \ll n$. In this case, the matrix *A* is not a square matrix. Instead, it has more rows than columns. Thus, the matrix equation $A\xi = z$ is an overdetermined system.

B. QR Decomposition with Column Pivoting

Assume that $A \in \Re^{n \times k}$ has rank $r \le k \ll n$. The QR decomposition with column pivoting gives [5]

$$A\Pi = QR \tag{6}$$

where $Q \in \Re^{n \times n}$ is an orthogonal matrix, $\Pi \in \Re^{k \times k}$ is a permutation matrix, and $R \in \Re^{n \times k}$ is an upper triangular matrix as

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \tag{7}$$

with $R_{11} \in \Re^{r \times r}$ being an upper triangular matrix and $R_{12} \in \Re^{r \times (k-r)}$. By writing

$$\Pi^t \xi = \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} \quad \text{and} \quad Q^t z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{8}$$

with $\xi_1 \in \Re^r$, $\xi_2 \in \Re^{k-r}$, $z_1 \in \Re^r$, and $z_2 \in \Re^{n-r}$, one has

$$\|z - A\xi\|_2^2 = \|z_1 - R_{12}\xi_2 - R_{11}\xi_1\|_2^2 + \|z_2\|_2^2 \tag{9}$$

For any ξ_2, the vector

$$\xi_{opt} = \Pi \begin{bmatrix} R_{11}^{-1}(z_1 - R_{12}\xi_2) \\ \xi_2 \end{bmatrix} \tag{10}$$

solves the minimization problem (4). For simplicity, ξ_2 is set to zero, and thus one obtains the basic solution:

$$\xi_{opt} = \Pi \begin{bmatrix} R_{11}^{-1} z_1 \\ 0 \end{bmatrix} \tag{11}$$

The error criteria are obtained as

$$E_{mean} = \frac{\|z_2\|_2}{\|z\|_2} \tag{12}$$

$$E_{\infty} = \max_{1 \le i \le n} |(\Delta z)_i| \tag{13}$$

where $(\Delta z)_i = (z - A\xi_{opt})_i / z_i$.

C. Selection of Polynomial and Order Incremental Scheme

To achieve fast timing and power simulations, one would like to use polynomials as small as possible to estimate the timing and power data, provided that the polynomials are yielded to the desired accuracy. In order to attain the smallest optimal polynomial that fits well into the timing and power data, an order incremental scheme for selecting polynomials has been deployed.

In this scheme, the smallest polynomial, such as a constant polynomial or a linear polynomial, is used as the first trial functional form of polynomial. A set of optimal coefficients of the prescribed polynomial are determined by solving the least square problem as described in the preceding section. The error criteria are then computed and compared against the desired accuracy requirements. If the desired accuracy requirements are met, the program stops and returns the optimal polynomial. If not, a polynomial with a higher order, such

as a bilinear polynomial or a quadratic polynomial, is selected. With more basis functions and thus more degrees of freedom, the least square problem is formed and solved again. The procedure is repeated until the desired accuracy requirements are met.

D. QR Factorization

Householder transforms have been used to compute the QR decomposition. The basic idea of the Householder transforms is to zero the elements in the lower triangular portion of matrix *A* using a sequence of orthogonal matrices, called Householder matrices. One of salient features about the orthogonal transformation is that it can effectively control the amplifications of round-off errors during numerical computations.

It could happen sometimes numerically that the matrix *A* is rank deficient, i.e., $r < k$, due to either the innate property of original data or the numerical round-off errors. In this case, the conventional QR factorization breaks down and one has to resort to the QR factorization with column pivoting.

E. Basis Function Scaling and Matrix Balancing

It is a common scenery that some basis function takes much larger or smaller value than others at the grid points $\{(x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})\}$ over the independent variable domain. This is due to the fact that the basis functions in a polynomial consist of products of powers of independent variables.

For instance, the basis functions $b_1(x_1, x_2, x_3) = x_1$ and $b_2(x_1, x_2, x_3) = x_1^2 \cdot x_2 \cdot x_3$ take value 10^2 and 10^8 at point (*100*, *100*, *100*) and they differ 10^6 times in magnitude! As a consequence, there are very significant disparities among the element magnitudes in the matrix *A*, which in turn may introduce serious round-off errors into the solution. In the worst case, this could cause numerical failures during the QR decomposition.

In order to overcome this difficulty, a practical idea is to scale the basis functions or equivalently to balance the matrix *A*. This scaling procedure is equivalent to use the scaled basis functions $\{b_j(x_1, x_2, \ldots, x_d)/d_j\}$ in the place of the original basis functions by a proper set of $\{b_j(x_1, x_2, \ldots, x_d)\}$.

F. Treatment of Abrupt Changes and Adaptive Domain Decomposition

In some case where abrupt changes present in the timing or power values of a data table, the single polynomial may have a difficulty to cover the entire independent variable domain.

In this case, the decomposition of the independent variable domain is entailed to break up the domain into multiple sub-domains such that the change rate in each sub-domain is relatively uniform and can be appropriately modeled by a single polynomial. This procedure can be hierarchically implemented into multiple levels of sub-domains if necessary. That is, a sub-domain can be further decomposed into multiple second-level sub-domains if necessary. In general, an *l*-th level sub-domain will be decomposed into multiple (*l+1*)-th level sub-domains if a single optimal polynomial can not be found in this *l*-th level sub-domain. The domain decomposition stops over a sub-domain whenever a single optimal polynomial is found over the sub-domain with the desired accuracy.

Once single-piece optimal polynomials are successfully found for all the deepest level sub-domains, they are combined into piecewise polynomials level-by-level in a bottom up fashion, until the original root domain is reached. By so doing, a final piecewise polynomial is obtained over the original characterization domain, which is of course applicable to the entire characterization domain and subjected to the desired accuracy requirements.

In order to divide the domain in the desirable way, an adaptive scheme is employed to insert break-points intelligently, in accordance to the information of data change rates. In this adaptive scheme, the data change rates are first computed over the domain (or sub-domain) under the current consideration and then the lines or planes where the data undertakes the most drastic changes are identified. The domain (or sub-domain) is divided into multiple smaller sub-domains in the next level along these identified lines or planes.

III. Numerical Results

Based on the algorithms presented in the preceding sections, a software tool, called DCLPro, has been implemented. In this section, numerical examples are presented to illustrate feasibility, efficiency, and robustness of these algorithms.

In the first example, this technique is employed to search for an optimal polynomial that fits a three dimensional data table of sample size *1000*. In practice, the timing and power data tables are obtained from either measurements or accurate circuit-level simulations. However, for the purpose of illustration of the algorithmic feasibility and validation to the computer programs, the three dimensional data table is generated over a characterization domain $[0, 10] \times [0, 15] \times [0, 7.5]$ as follows:

$$\begin{aligned} x_1^{(i)} &= j_1 && (j_1 = 0, 1, 2, \ldots, 9) \\ x_2^{(i)} &= 1.5 j_2 && (j_2 = 0, 1, 2, \ldots, 9) \\ x_3^{(i)} &= 0.75 j_3 && (j_3 = 0, 1, 2, \ldots, 9) \end{aligned} \tag{14}$$

$$\begin{aligned} z^{(i)} = \; & 1 + 6x_1^{(i)} + 7x_2^{(i)} + 9x_3^{(i)} + 3x_1^{(i)}x_2^{(i)} + \\ & 4x_1^{(i)}x_3^{(i)} + 2x_2^{(i)}x_3^{(i)} + x_1^{(i)}x_2^{(i)}x_3^{(i)} \end{aligned}$$

where $i = 100 j_1 + 10 j_2 + j_3 + 1$.

The computer programs use the above data table as the input and search for an optimal polynomial. The required accuracy is specified as a relative error of 10^{-6}, that is, an optimal solution is considered to be attained if the maximum relative error between the actual values in the original data table and the estimated values from the resulted polynomial is smaller than or equal to 10^{-6}.

TABLE I OPTIMAL POLYNOMIALS OBTAINED BY THIS TECHNIQUE IN EXAMPLE 1

Iteration	Order of Polynomial	Optimal Polynomial with Given Order	E_{mean}	E_{∞}
1	1 basis	405.578	0.622	404.6
2	4 bases	$-401.469 + 62.5312x_1 + 42.4375x_2 + 70.875x_3$	0.236	402.5
3	7 bases	$103.516 - 16.7812x_1 - 8.1875x_2 - 21.375x_3 + 6.375x_1x_2 + 10.75x_1x_3 + 6.5x_2x_3$	0.051	102.5
4	8 bases	$1 + 6x_1 + 7x_2 + 9x_3 + 3x_1x_2 + 4x_1x_3 + 2x_2x_3 + x_1x_2x_3$	0	0

Table I shows the optimal polynomials obtained using this technique for each given order of polynomial during the course of optimization. At the first iteration, constant polynomials are used to model the given three dimensional data table. The closest constant polynomial to the given data is found by this technique to be $p_{1opt}(x_1, x_2, x_3) = 405.578$, which results in a mean relative error $E_{mean} = 0.622368$ and a maximum relative error $E_{\infty} = 404.578$. Since the desired accuracy, i.e., the maximum relative error below 10^{-6}, is not met by this constant polynomial, the polynomial with more terms is needed for the next minimization. At the second iteration, the polynomials with four terms (i.e., the linear polynomials) are employed to estimate the given three dimensional data table. The closest linear polynomial to the given data is attained to be

$$P_{4opt}(x_1, x_2, x_3) = -401.469 + 62.5312x_1 + 42.4375x_2 + 70.875x_3 \quad (15)$$

which results in a mean relative error $E_{mean} = 0.236349$ and a maximum relative error $E_{\infty} = 402.469$. Since the desired accuracy is still not met, the iteration goes on. Finally, at the forth iteration, the optimal polynomial with eight terms is obtained by this technique as

$$P_{8opt}(x_1, x_2, x_3) = 1 + 6x_1 + 7x_2 + 9x_3 + 3x_1x_2 + 4x_1x_3 + 2x_2x_3 + x_1x_2x_3 \quad (16)$$

which gives a mean relative error $E_{mean} = 0$ and a maximum relative error $E_{\infty} = 0$. The desired accuracy is met by this polynomial, thus the programs stop and return this polynomial as the optimal solution. In fact, the optimal polynomial with eight terms fully recovers the original polynomial by which the three dimensional data table was initially generated.

Now, an example of applying this technique to the real-world timing data is presented. In this example, an inverter, called “inv1a4”, is characterized using the accurate circuit level simulator. This inverter has an input pin *A* and an output pin *Y*. Its actual timing delay data collected from the characterization is then stored into two timing delay data tables. The two delay tables are for the timing arc from *A* to *Y*; one for a rising signal occurring at the input, whereas another delay table for a falling signal occurring at the input.

Each of these data tables is a two dimensional table and contains *400* data points. The two independent variables are input transition (IT) time and output capacitive load (CAP). There are *20* sampling points along each of the independent variables. The timing values stored in the delay tables are the measured intrinsic delays from *A* to *Y*.

This technique is applied to these two dimensional timing delay data tables. An optimal (piecewise) polynomial is attained for each of the timing delay data tables. Figures 1 and 2 show both the original timing data and the data computed using the resulted optimal (piecewise) polynomials. Good agreement between the data from the optimal (piecewise) polynomials and the original timing data is observed. Although the original timing tables are of size *400*, the polynomials obtained by this technique are relative simple. Table II shows the resulted optimal polynomial within the context of the DPCL codes. Note that one of the two polynomial equations is a piecewise polynomial as indicated in Table II.

TABLE II OPTIMAL POLYNOMIAL EQUATIONS IN DPCL

```
CALC(calc_equation_0):
 passed(real: a1, a2)
 result(real: 0.0320895+0.481278*a1+1.729*a2+1.67829*a1*a2);

CALC(calc_equation_1):
 passed(real: a1, a2)
 result(
  when(0.0061259 <= a2)
   result(real: 0.025108+0.928471*a1+1.8643*a2+2.37908*a1*a2-
1.41492*a1*a1-0.686517*a2*a2
  ),
  otherwise result(
   when(0.0024699 <= a2 && a2 <= 0.0061261)
    result(real:
0.0392948+0.472932*a1+2.07111*a2+12.6796*a1*a2-0.83541*a1*a1
   ),
   otherwise result(
    when(0.0005549 <= a2 && a2 <= 0.0024701)
     result(real: 0.0397869+0.44705*a1+2.1906*a2+14.045*a1*a2-
0.753893*a1*a1
    ),
    otherwise result(real:
     /*when(a2 <= 0.0005551)*/
     0.0401251+0.435832*a1+2.26675*a2+14.6541*a1*a2-
0.714491*a1*a1
    )
   )
  )
 );

delay(delay_model_0):
 when(SOURCE_EDGE == ‘F’ && SINK_EDGE == ‘R’)
```

```
  early(calc_equation_0(EARLY_SLEW, loadCap(TO_POINT).cap))
  late(early)
,
otherwise
/*when(SOURCE_EDGE == 'R' && SINK_EDGE == 'F')*/
  early(calc_equation_1(EARLY_SLEW, loadCap(TO_POINT).cap))
  late(early)
;
```

IV. Conclusions

A robust technique for generation of efficient and accurate timing and power polynomial models from nonlinear multi-dimensional lookup tables has been presented. In this scheme, the transformation of a nonlinear multi-dimensional timing or power lookup table into a polynomial equation has been mathematically formulated as a least square problem. The least square problem was solved using the QR decomposition with column pivoting, which is in turn implemented by Householder transforms. To circumvent the difficulty associated with the scale disparity of basis functions, a matrix balancing or basis function scaling scheme was proposed and implemented. To handle the sudden changes of timing or power data across the characterization domain, a scheme that can intelligently insert break-points over the characterization domain was implemented. The generated timing and power polynomial models in DPCL are used by EDA tools to speedup timing and power analyses.

V. References

[1] *Library Compiler Reference Manual*, Version 1998.02, Synopsys, Inc., 1998.

[2] J. F. Croix and D. F. Wong, "A fast and accurate technique to optimize characterization tables for logic synthesis," Proceedings of 1997 Design Automation Conference, pp.337-340, Anaheim Convention Center, Anaheim, CA, June 9 - 13, 1997.

[3] IEEE P1481, *Standard for Delay and Power Calculation System*, IEEE Standards Department, 1998.

[4] *Delay Calculation Standard and Deep Submicron Design*, Integrated System Design, July 1996. (also see http://www.isdmag.com/Editorial/1996/EDAFeature9607.html).

[5] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the ACM*, vol.5, pp.339-342, 1958.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 3rd Edition, 1996.

Figure 1: Recovery of the delay table from A (rising) to Y (falling) by this technique

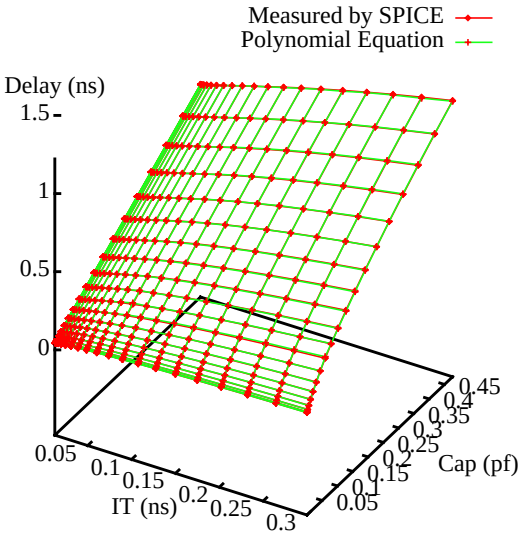

Figure 2: Recovery of the delay table from A (falling) to Y (rising) by this technique