Generating Scalable Polynomial Models: Key to low power high performance designs

Girishankar G., Shitanshu Tiwari
Texas Instruments India
girishankar@ti.com, shitanshu@ti.com

Abstract

As the need for power reduction techniques based on on-chip dynamic voltage scaling is on the rise, a design flow that can take full advantage of the performance/power tradeoffs is required. In this paper, we present a Scalable Polynomial Model (SPM) based approach to precisely estimate the power and performance achievable through voltage scaling techniques. The main challenge here is generating accurate Scalable Polynomial Models for Timing and Power and coming up with the right usage methodology. Novel techniques have been employed to generate piecewise polynomials for any given set of data without loss of accuracy. A new approach for incrementing the polynomial orders for reduction in run time is presented along with results. SPM libraries were generated for an entire library and validated at the cell level and design level. A complete usage model for designs using dynamic voltage scaling is also presented.

1. Introduction

It has become extremely difficult to meet power, performance requirements across the entire design space of Process (P), Voltage (V) and Temperature (T). While designs have been using discrete voltages to 'gear-shift', current day products are moving towards integrating adaptive loops to eliminate the traditional 'corners' from the design space.

Designers are pursuing concepts of adaptive voltage control wherein the supply voltage is determined by performance of silicon within the current environment (process, temperature etc.). The method gauges the performance of silicon and ensures the design can maintain target performance while guaranteeing the minimum required voltage for a given performance. This reduces excess performance for stronger silicon and results in lower active and leakage power. In a way the concept also results in an increase in the quoted frequency of operation of the device, as some DC supply accuracy margin is eliminated, thus increasing the Vmin for STA.

In order to calculate the voltage at which the required performance will be met while we are in the design stage, we require delay and power numbers for a continuous range of voltage and temperatures. Modeling timing data using polynomials has been attempted in [1] but SPICE correlation is an issue with such approaches in 90nm node and beyond. The widely used method of representing actual SPICE data is in the form of look-up tables. Techniques to optimize these look-up tables have been discussed in [2]. But the voltage, temperature dependency is not taken care in the above works. Scalable Polynomial Models are best suited for this since the delay and power numbers are a function of the voltage and temperatures. These models were first introduced in [2]. The paper also describes the syntax for specifying piece-wise polynomials for Timing and Power with a few examples. Replacing the traditional look-up table based Non Linear Models (NLMs) with Scalable Polynomial Model (SPM) results in quick and easy interpolation on any of the input parameters like voltage, temperature. A single SPM library would replace multiple NLM libraries.

However, the main challenge with Scalable Polynomial Models is to generate polynomials that can accurately model timing and power numbers. There have been attempts at finding a single equation that would model all the data ([3]), but given the irregularities in data, one equation is not sufficient. Though [4] deals with approaches to find the polynomial coefficients, the need for a completely automated method of generating piece-wise polynomials for any given type of data: delay, constraints or power without any user interference still exists.

In section 2, we describe our approach to this problem wherein we have come up with a novel way of generating piece-wise polynomials that can model *n* dimensional data. In section 3, we present a set of validations done on SPM libraries along with accuracy and runtime results. Section 4 deals with the flow aspect, where we describe how these SPM libraries are used in PTV determination and power/performance tradeoff analysis.

2. Polynomial Model Generation

2.1. Polynomial model

Proceedings of the 19th International Conference on VLSI Design (VLSID'06)
1063-9667/06 $20.00 © 2006 IEEE

To model the n dimensional lookup table data, a multivariate polynomial model is required. Let us consider a two variable polynomial with orders 2 and 1, as an example:

$$f(x,y) = (c_{10} + c_{11}x + c_{12}x^2)(c_{20} + c_{21}y) \qquad (1)$$

The same equation (1) can also be represented as:

$$f(x,y) = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5x^5y \qquad (2)$$

Here x and y could correspond to output capacitance (Load) and input transition time (Slew). If there are m_1 different loads and m_2 different slews for which we have the corresponding delay numbers, then there are m_1m_2 combinations of datasets in the 2D table, that are to be modelled with polynomial equations. When the orders of the polynomial model are fixed, the problem reduces to one of multivariate polynomial curve fitting. We have implemented a Singular Value Decomposition (SVD) based Least Square Error minimization algorithm to determine the polynomial coefficients. The details of Least Squares method and SVD computation can be found in [5], [6] and [7]. SVD works on Householder matrices, a triangularization technique described in [8]. While the orders are fixed to (2,1) in the above example, in reality the optimal order needs to be determined.

Instead of providing an R-square or Sigma value as an accuracy target for our curve-fitting tool, we use maximum allowable percentage difference or absolute difference as the criteria. In our case, we have used 3% as the percentage error limit. Given this stringent accuracy limit, a single polynomial is rarely adequate to model the complete lookup table over the entire range of input parameters. This is because the functions that we are trying to model are in reality not polynomial functions or do not follow the same trend throughout the range. For example, leakage power varies exponential with respect to temperature.

This implies that the entire data domain needs to be broken down into sub-domains, each modelled by a polynomial. Hence the two main challenges are 1) coming up with the sub-domains that can be modelled using one polynomial each and 2) determining the order of each of the variables for each of the piece-wise polynomials.

We have come up with a recursive approach to tackle the problem of sub-domain determination, and a new method of incrementing orders in each iteration, to determine the right polynomial models. These are explained in the following sections.

2.2. Domain Partitioning

Partitioning the n dimensional table based on rate of change of data is discussed in [2]. The main disadvantage of such techniques is the fact that partitioning this way does not guarantee the existence of a polynomial for each of the sub-domains and if we did not find an accurate model, we are left with no choice but to manually split the table and follow a trial and error method to arrive at piece-wise polynomials. We also observed that with the extremely stringent accuracy criteria, more than a few tens of polynomials are required to accurately model data across the whole of Voltage, Temperature, Load and Slew space. Our approach is to integrate partitioning and checking the existence of polynomial into one.

Consider an n-dimensional look up table with input variables $x_1, x_2, \ldots, x_n$. If there are m_i unique data values for variable x_i, we define a set X_i as :

$$X_i = \{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(m_i)}\} \qquad (3)$$

where the data at each sampling point is denoted by a set of braces. For example, if dimension 1 corresponds to Voltage, then each of $x_i^{(1)}, x_i^{(2)}, x_i^{(3)} \ldots$ in (3) could be 0.8, 0.9, 1.1,..; and m_i denotes the number of such unique voltages. All possible combinations of these X_i's will give the inputs of the n-dimensional look-up table, and the circuits are characterized at these points to get the timing and power numbers corresponding to each combination.

We define a set D, that contains the set of all X_i's as its elements.

$$D = \{X_1, X_2, \ldots, X_n\} \qquad (4)$$

Since the combinations of elements in D in (4) correspond to the complete domain of inputs, we call this set as the 'domain' set.

Our idea is to partition this domain into sub-domains and find polynomials modeling each of these, and if we are unable to do so, recursively further partition these sub-domains till we get piecewise polynomials for the entire domain.

This technique is explained below:

We define the initial sub-domain D_s as

$$D_s = \bigcup_{i=1}^{n} X_{s_i} \qquad (5)$$

where $X_{s_i} = \begin{cases} \{x_i^{(m_i-1)}, x_i^{(m_i)}\} & \text{for } i=1 \\ X_i & \text{for } i \neq 1 \end{cases}$

It is clear from (5) that we have partitioned the domain to have only 2 data points along dimension 1 and all data points in rest of the dimensions. We call dimension 1 as the 'split dimension'. The quality of results does not depend on this choice of initial split dimension, since we have the ability to dynamically switch dimensions, as we shall see later in this section.

We check if it is possible to determine a polynomial for this sub-domain D_s. This is done using a combination of LSE algorithm and order-incremental scheme described in the next section. If we find a polynomial, then we expand this sub-domain along the 'split dimension' and again check if an accurate polynomial exists. X_{s_1} is now given by the following equation :

$$X_{s_1} = \left\{ x_1^{(m_1-2)}, x_1^{(m_1-1)}, x_1^{(m_1)} \right\} \quad (6)$$

The expansion continues as long as we are able to find one polynomial that can model the entire sub-domain D_s. If no more data points can be modeled, we remove the sub-domain D_s, that has been successfully modelled so far, from the domain D. We then continue with the rest of the unresolved domain in a similar manner.

There could arise a situation when the initial sub-domain does not get modelled with a single polynomial. This leaves us with no scope for expanding the sub domain. At this point, we change the 'split dimension' (to say dimension 2) and go into recursion, with the D_s now becoming the complete domain D.

This recursive approach guarantees that, in the worst case, the domain gets partitioned to such an extent that there are only 2 data points along each of the dimensions $x_1, x_2, \ldots, x_n$.

$$D_{min} = \bigcup_{i=1}^{n} \{x_i^{(k_i)}, x_i^{(k_i+1)}\} \ , \ 1 \le k_i \le m_i \quad (7)$$

The smallest domain set is given by (7), that has just 2 adjacent data points along each dimension.

A polynomial function of order 1 along each dimension will always be able to model D_{min} with sufficient accuracy, as it is just a generalization of the fact that a straight line can always be found passing through any 2 points in a plane.

The complete algorithm is shown in Figure 1.

The FITPIECEWISEPOLY function takes the domain D and the initial split dimension as input. A sub-domain D_s is derived from D as in (5) and fed as input to the 'CURVEFITLSE' function. This function returns polynomial coefficients and a *return_state* of SUCCESS or FAILURE depending on whether the accuracy criteria are met or not. If *return_state* is FAILURE, the domain is partitioned further. If we are unable to do so, the *split_dimension* is changed, and we call the function FITPIECEWISEPOLY recursively. If the return state is SUCCESS, we keep expanding the sub-domain D_s, till we cover the entire domain D.

```
FITPIECEWISEPOLY (D, split_dimension)
1  #Inputs : complete dataset D and the split dimension
2  derive D_s ⊂ D  #as given in (5)
3  while D ≠ NULL-SET
4    polynomial coefficients, return_state =
CURVEFITLSE(D_s)
5    if return_state is FAILURE
6      if |X_{s_split_dimension}| == 2
7        # Unable to partition further along
split_dimension
8        tmp ← split_dimension
9        split_dimension ← split_dimension+1
10       FITPIECEWISEPOLY (D_s, split_dimension)
11       D ← D - D_s
12       split_dimension ← tmp
13       derive D_s ⊂ D
14     else
15       # remove the already modeled data points and
continue
16       D ← D – prevD_s
17       derive D_s ⊂ D
18   else
19     # return state was SUCCESS
20     prevD_s ← D_s
21     expand D_s along split_dimension
22   endwhile
23   return polynomial coefficients
```

Figure 1: Algorithm illustrating the recursion technique used.

Note that even though we have not explicitly indicated, corresponding to each input data combination there exists an actual timing or power number that is used to determine coefficients of the polynomials in the CURVEFITLSE routine.

The CURVEFITLSE function is responsible for solving the Least Squares problem, incrementing the orders and checking for over-fitting. Order incremental method is described in section 2.3 and over-fit reduction techniques in section 2.4.

2.3 Optimal Order determination

In the technique explained in section 2.2, in order to determine whether an accurate polynomial can be found for the given data range or not, we need to determine the orders (maximum powers) of each of the variables.

One way is to increment the orders of input variables one at a time and evaluate all possible combinations, evaluating the LSE algorithm each time, until the desired accuracy criteria is met. This method has a computational complexity of $O(m^n)$ where n is the number of dimensions and m is the maximum possible order for each of the dimensions.

We have adopted a more intelligent approach based on the fact that for a given data set the accuracy of the polynomial will not improve if the order of a particular dimension is decreased (other orders kept constant). Consider a two variable (x,y) case as an example. Assume that in reality orders (2,2) yield the best polynomial. The most straightforward approach is to start from (0,0) and start evaluating all possible combinations (i.e. (0,1), (1,0), (1,1), (2,0), (0,2),…) till we hit orders (2,2). Our idea is to make use of the results at each stage to arrive at the next order to be evaluated. We start with (0,0), (0,1) and if we find (0,1) to be more accurate than (0,0), we would not evaluate (1,0) in the next iteration. We would rather evaluate (1,1). This way we converge to the actual orders much faster. We have found this to give around 10X to 15X reduction in run-times on an average.

As this could potentially lead to over-fitting, few techniques have been incorporated to ensure we do not over-fit. These are discussed in section 2.4.

2.4 Improving Generalization

One of the problems that occur during curve-fitting is over-fitting. Two methods have been used to prevent over-fitting. One is by generating a second data-set on the fly from the available data set through linear interpolation. The error with respect to the second data set is also continuously monitored. Once the error crosses a certain bound the order incrementing is stopped. The other method involves modifying the cost-function of the LSE algorithm. The improvement in accuracy is monitored and as long as there is no significant improvement in accuracy, the orders are not increased.

Figure 2 shows an inverter delay curve across output load and input slew. It is obvious that there is over-fitting along the Load axis. Figure 3 shows the same delay graph after we employed the over-fitting prevention methods that were explained in this section.

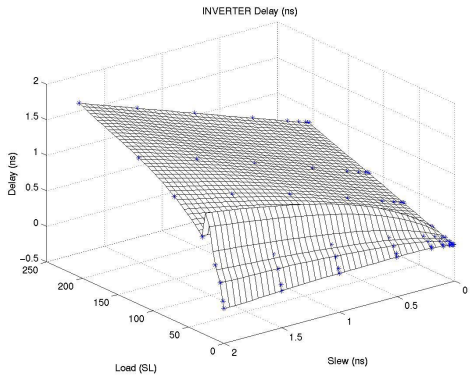

Figure 2 : Delay Graph showing over-fit along Load axis

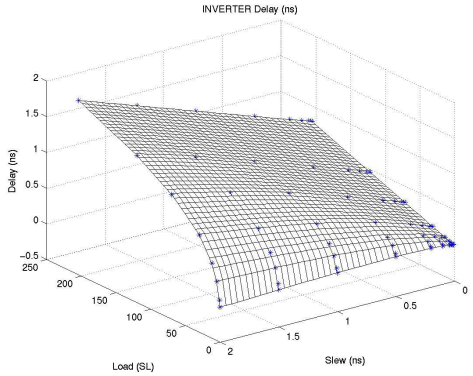

Figure 3: Delay graph after using over-fit reduction techniques

3. Validation

A combination of all techniques described above guarantees a solution in the form of piece-wise polynomials for any set of n dimensional data. The algorithm was used to generate SPM libraries containing delay, constraints (setup and hold), active power and leakage power data in the form of piece-wise polynomials. A rigorous validation was done on this data to check accuracy and tool runtimes.

3.1 Correlation at the Cell level

We found the optimal number of V, T points required to model the data accurately (i.e. <3% error) through a set of experiments on an inverter and a few basic cells in the library. Based on this, all cells were characterized at 7 voltages and 4 temperatures covering the entire V, T range for each process.

In order to check the accuracy, the cells were additionally characterized at a large number of intermediate voltage, temperature, load and slew combinations (36,000 data samples in all) that were not used for generating the polynomials. This validation set was compared with the

corresponding data obtained by interpolation of polynomial models. The results show a very good correlation between the two data sets. The error was found to be within 3% to 4% as compared to NLM data.

The error distribution graph in Figure 4 shows the percentage difference in rise delay of an inverter. At the few points where the percentage difference is more than 5%, the difference in the absolute values of the delay is less than 3ps or 4ps.

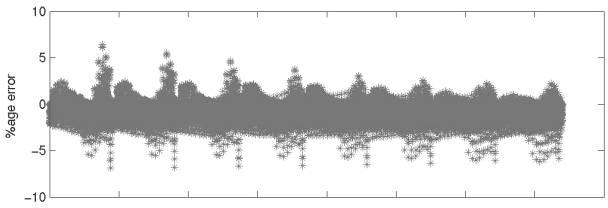

Figure 4: Graph showing the Percentage Error along Y axis for each of the 36,000 data samples taken along X axis.

All the cells in the library were found to give similarly good correlation. We chose to represent the inverter graph as an example as it is most sensitive, among other library cells, to V, T or load/slew changes.

3.2 Validation using PrimeTime®

Three designs were used to evaluate SPM .libs. The STA was performed using PrimeTime.

Additional NLM libraries were generated at intermediate voltage points. These were read into PrimeTime and the delays of paths measured. The SPM libraries were then read in and the delays were compared. The analysis showed a good correlation between two data, indicating that the SPM .libs can be used for determining the delay/power values at different supply voltages and temperatures without loss of accuracy. Another noteworthy observation regarding the usage of SPM .libs is that while the initial time taken to load the SPM libraries is high (around 3 times more compared to NLM .libs), the operating conditions can be switched on the fly once it is loaded. This would enable us to very quickly iterate and determine the minimum voltage required to meet the performance targets.

The bar graph in Figure 5 shows the percentage error on a critical path (consisting of a wide variety of cells) of a 160K instance design.

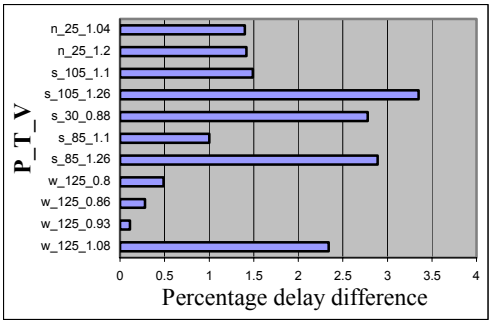

Figure 5: Percentage error in the critical path delay across different PTV combinations.

The SPM checkout, done on other designs, also shows results very similar to the one shown above.

3.3 Validation for Power

Scalable Polynomial Power Models (SPPM) were generated in a similar manner and were checked for accuracy, both at cell level and using PrimePower® and PowerCompiler® at the design level.

A 64K instance design was used as a testcase and the power numbers were calculated using SPPM and NLPM (Non Linear Power Models) and compared. Table 1 below shows the errors report. For each PTV(in the first column), the percentage of cells that have an error less than 1% is shown in the second column, and those with error between 1% and 3% is shown in the column under 1~3% and so on. We find that at most of the PTVs, the SPPM power numbers for bulk of the cell instances show extremely good correlation with actual power numbers.

Table 1: Table showing the percentage distribution of errors

	Error Percentage ranges					
PTV	0~1%	1~3%	3~5%	5~10%	10~15%	15~20%
N_25_1.05	97.30	3.60	0	0	0	0%
S_-40_1.16	92.79	6.31	0	0.90	0	0.90%
S_30_0.88	95.50	4.50	0	0	0	0%
W_-40_1.08	93.69	6.31	0	0	0	0%
W_125_1.08	87.39	8.11	4.50	0.90	0	0%
W_30_0.88	97.30	2.70	0	0	0.90	0%

4. SPM in the flow

The SPM libraries can be used with minimal changes in the existing STA flow to take full advantage in terms of accurate power/performance tradeoff analysis.

4.1 Operating PTV determination

As the main advantage of SPM libraries is interpolation across different operating conditions, they can be used in an iterative manner to determine the supply voltage for a given frequency of operation.

Once the libraries are read in and the constraints applied, we check for the maximum frequency of operation. If it is higher than the target frequency, we reduce the supply voltage (i.e. change operating_condition) on the fly.

In the NLM based flow, we need to generate NLM libraries for the newly set operating_condition, read in the libraries again and continue this process until we find the minimum voltage of operation.

4.2 Optimal Design flow

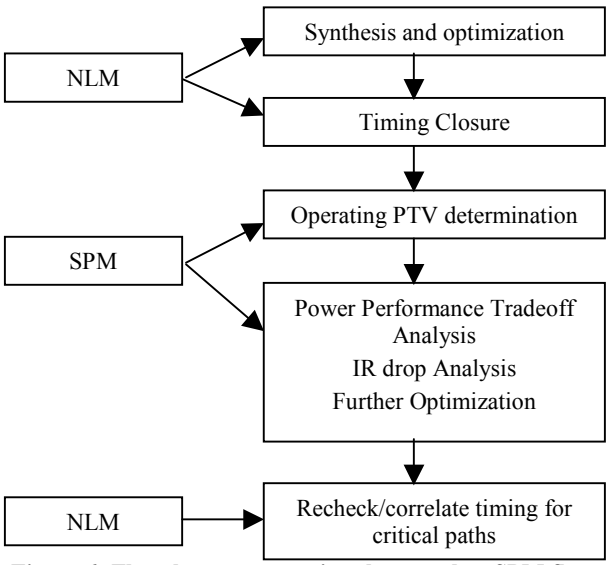

Figure 6: Flowchart representing the complete SPM flow.

As can be seen from the run time reports in sections 3.1, 3.2 and 3.3, tools using SPM libraries show high runtimes compared to NLM libraries because of the increase in library size and also high memory usage in order to store and evaluate polynomials. To overcome this disadvantage, we use a combination of NLM and SPM libraries for different phases in our design flow. Figure 6 illustrates the complete design flow.

5. Conclusion

A complete usage model for Scalable Polynomial Models is presented, in the context of dynamic scaling of supply voltage to meet power, performance goals. The entire process of generating Scalable Polynomial Models has been automated by employing several techniques including a new domain-partitioning technique and the results indicate a very good correlation with the conventional NLM data at the cell level and chip level. In order to keep the design cycle time minimal, a combination of NLM and SPM libraries are used in the design flow. Future work includes determining the minimal and optimal set of PTV points that would yield the best accuracy before generating SPM libraries. Techniques to speed up the flow also need to be looked at.

6. References

[1] Young-Hyun Jun, Ki Jun and Song-Bi Park, *An Accurate and Efficient Delay Timing Modeling for MOS Logic Circuits Using Polynomial Approximation,* IEEE Trans. Computer-Aided Design, vol. 8, pp. 1027 1032, Sept. 1989.

[2] J.F.Croix, D.F.Wong, *A Fast And Accurate Technique to Optimize Characterization Tables For Logic Synthesis,* IEEE Proc. 34th Design Automation Conference, pp.337-340, June 1997.

[3] Arvind Krishnamurthy and Djahida Smiti, Scalable Polynomial Delay and Power Model, *A Technical white paper on a new delay and power model for DSM technology, Synopsys.* May. 2001.

[4] Wen-Tsong Shiue and Weetit Wanalertlak, *An Advanced Cell Polynomial Base Modeling for Logic Synthesis,* Proceedings, IEEE International SOC conference 2003.

[5] Wang, Gaofeng and Gopisetty, Runip, *Efficient generation of Timing and Power Polynomial models from Lookup Tables for SOC Designs,* 1999 12th IEEE International ASIC/SOC conference, Washington, DC, Sep. 15-18,1999.

[6] Klema, V and Laub, A, *The singular value decomposition: Its computation and some applications,* IEEE Transactions on Automatic Control, Volume:25, Issue:2, Apr. 1980, Pages: 164-176.

[7] Enders A. Robinson, *Least Squares Regression Analysis in Term of Linear Algebra,* Goose Pond Press, 1981

[8] W.H.Press, S.A.Teukolsky, W.T.Vetterling, and B.P.Flannery, *Numerical Recipes in C,* second edition, Cambridge University Press.

[9] A.S.Householder, *Unitary triangularization of a non-Symmetric Matrix,* Journal of the ACM, vol.5, pp339-342, 1958.