International Journal of u- and e- Service, Science and Technology
Vol. 4, No. 1, March 2011

Improving Analysis Accuracy In Integrated Circuit Design Using Cubic Bézier Curves And Patches

Eric Y. Chen
Google Inc.
ericchen@google.com

May Huang
Dept. of Electrical Engineering
International Technology University
mhuang@faculty.itu.edu

Abstract

In this paper, we introduce a method for the application of Bézier curve algorithms to lookup-table-based interpolations, particularly for the use of timing, power, and noise analysis in integrated circuit design. Bézier curves can replace conventional piecewise linear functions for accuracy enhancement. Selecting control points with physical implications and forcing curves to pass through sampled points are critical for achieving the envisioned improvements. This method can be easily applied to the interpolation of splines.

Keywords: *Lookup Table, Bézier curve, Data Interpolation.*

1. Introduction

In model-based analysis, lookup table (LUT) is a common used method for modeling characteristics. Indices on dimensions express measuring conditions, and values on a table represent sampled points measured on corresponding condition(s). For example, a table of internal energy consists of two indices: input signal slew and output capacitive load. The energy values in the table are measured with pairs of input slew and output load marked on the indices. A unit circuit can be small as an inverter or large as a processor.

Given a point on the index for a two-dimensional LUT or a pair of points on the indices for a three-dimensional LUT, many algorithms are applicable for calculating the value. Piecewise linear (PWL) functions are commonly used in design analysis. The calculation accuracy is proportional to the size of tables. With the desire of pursuing high level of accuracy without intolerable increase in table size, various methods were endeavored, such as finding critical points on contours [11] and optimizing characterization tables [4].

Growing with technologies, more effects are added into considerations. Electrical current models [1, 2] are suggested to represent time and power models for accuracy improvement. These efforts cause the number of lookup tables increased exponentially. To keep analysis efficiency, a microprocessor for PWL computation is also considered and developed [10].

A major disadvantage of the piecewise linear approach is that the continuity of first derivatives is not carried at sampled points, which limits the accuracy of interpolations. Another disadvantage is that many sample points are often required in order to describe the characteristics sufficiently, thus the size of LUTs cannot be too small. Theoretically, these disadvantages can be overcome by nonlinear fitting curves, such as splines [13]. The challenge of using splines is that the shape of nonlinear curves is not easy to control, especially to force the curves passing through sampled points.

Since a design analysis primarily relies on lookup tables, any improvement of interpolation accuracy will contribute to the reliability of the analysis.

2. Nonlinear Fitting Curves

Nonlinear fitting curves can provide smoother curves than piecewise linear functions. Among polynomial nonlinear curves, B-splines and Bézier curves are commonly used in practice, since higher-degree curves are even more difficult to control.

2.1. B-Splines

B-splines a.k.a cubic splines are defined by piecewise cubic functions, represented by a set of knots. To add an extra piece to the curve, only one knot is required. Comparing with piecewise linear functions, B-splines are smoother. These advantages enable B-splines to be favored in CAD tools for providing smooth curves. The drawback is that spline curves do not pass through any knots. In fact, knots only indirectly affect the shape of curves. Manual and interactive refinement to the knots using CAD tools is sometime necessary to obtain an ideal spline curve.

B-spline curves have been applied in circuit analysis. Vlach and Singhal have provided a method to compute the derivatives on the curves based on the knots [13]. However, the requirement of solving complicated high-dimensional equations prevents the method from replacing piecewise linear functions practically.

2.2. Bézier Curves

Bézier curves are cubic curves, represented by a set of control points. In general, this method is not so popular as B-splines for two reasons. First, its representation is less compact.

Three extra control points are required to add a piece of curve whereas splines only need one point. Second, it is less smooth than B-splines at some points on curves.

However, Bézier curves are easier to manipulate. We can control not only the curves to pass through certain points but also the derivatives on curves directly. This advantage makes it possible to design smooth curves automatically from measured data. Meanwhile, Bézier curves are evaluated in a similar way as splines, using de Casteljau's algorithm [3].

In fact, the control points of Bézier curves can be transformed into the knots of splines by duplicating some knots [8]. Therefore, the curves we generated can be automatically adopted by a system where B-splines are used as the interpolation method.

Even if there are other types of piecewise nonlinear curves, such as Lagrange polynomials, Hermite polynomials and least squares approximations [8], those curves are usually evaluated using different algorithms, and thus harder to be converted into B-splines.

3. Improving Accuracy on 2D Lookup tables

We use piecewise cubic Bézier curves to interpolate values between two sampled points. To define a piece of the curve, we only need four data: the values at the sampled points and the first derivatives at the points. The smoothness is ensured at any point on the curve.

3.1. Determining Control Points

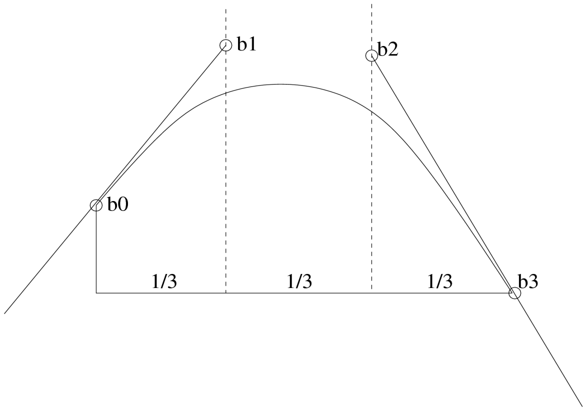


Figure 1: A cubic Bézier curve

For each piece of the Bézier curve, we define four control points. The first one and the last one will be shared with adjacent pieces. As shown in figure 1, the first control point b0 is set to the sampled point on the left, and the last control point b3 is set to the sampled point on the right. We compute the control points b1 and b2 based on the first derivatives at b0 and b3 respectively. The derivative at b0 can be represented as line l passing through b0. The control point b1 is defined at the intersection of l and the vertical line passing through the left tercile of the x-coordinates of b0 and b3. Symmetrically, we define b2 using the first derivative at b3 and the right tercile of the x-coordinates of b0 and b3.

As a well-known property of cubic Bézier curves [8], the constructed curve starts from the first control point b0 and ends at the last control point b3. The first derivative at b0 is defined by the vector b0b1 and the first derivative at b3 is defined by the vector b2b3. The constructed piecewise curves are composed by a set of Bézier curves. C0 and C1 continuity1 are preserved at any point on the curves, and C2 continuity is guaranteed everywhere except at the sampled points, where Cr continuity is defined as being r times differentiable with respect to the given parameterization is preserved [8].

As mentioned in [8], the piecewise Bézier curves can be transformed to B-splines by introducing duplicated knots that will cause the loss of C2 continuity at sampled points.

In section 5, we show the accuracy of the method.

3.2. Approximating Derivatives at Sampled Points

To avoid the inconvenience of measuring derivatives, we present a substitution to obtain the derivatives at sampled points. We approximate these derivatives with the average of the slopes of two adjacent piecewise linear functions. There are two special cases, at the first and last sampled points in a lookup table. We define b_1 of the first piece as the mid point of b_0b_2, and b_2 of the last piece as the midpoint of b_1b_3.

3.3. Evaluating Points on the Curve

We first locate the piece containing the query point. After the four control points are found, the evaluation of the curve is standard, such as using de Casteljau's algorithm [8]. Using this method, a Bézier curve is represented as $(x(t), y(t)),\ t \square (0, 1)$. Since we cannot compute y as a function of x directly, Newton's iterative method is used to determine t by $x(t)$ and then compute $y(t)$ [12].

4. Improving Accuracy In 3D Lookup Tables

There are two ways of extending the 2D lookup tables in 3D, using rectangular surface patches or using triangular surface patches. A surface of rectangular patches can be easily constructed from existing lookup tables, but it is less smooth on the boundaries of patches. A surface of triangular patches is harder to construct, but it can provide a smoother surface and requires fewer constraints on sampled points.

4.1 Rectangular Patches with Tensor Product

For existing 3D lookup tables, the sampled points are usually represented as a grid, where each point is defined as z(x, y). We compute the piecewise cubic Bézier curve along each column and row using the method introduced in section 3. By doing so, we divide the surface into a set of rectangular patches, and all the boundaries of the patches are defined as cubic Bézier curves. We evaluate the values in a rectangular patch using tensor product [8]. It guarantees C1 continuity in both x and y directions in the patch and along its boundaries and C0 continuity on the boundaries in the perpendicular direction. This method can be extended to higher dimensions.

Similar to the 2D curves, a point of a 3D surface is defined as a set of functions (x(u), y(v), z(u, v)), $u \in (0, 1)$ and $v \in (0, 1)$, instead of z(x, y). We can compute u and v using the Newton's method, and then compute z(u, v).

4.2 Triangular Patches

We introduce a method to achieve better smoothness on edges and corners using triangular patches, which can be used to process any set of sampled points without requiring their positions to be on a grid.

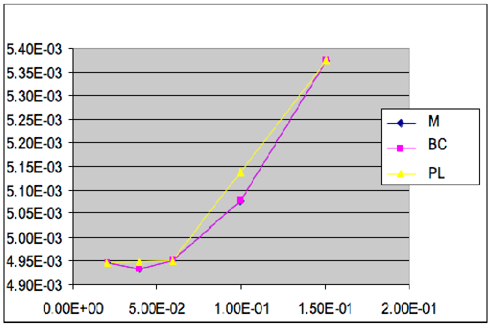
We construct a triangular subdivision of the plane using Delaunay triangulation [9], and then take each triangle in this triangulation as a patch. In the surface, triangular patches are defined as Bézier triangles. For each Bézier triangle, ten control points are required: three corner points, two tercile points on each edge, and the point at the gravity center of the triangle. The corner points have already been given as sampled points. The two control points on each edge are computed using the method mentioned in section 3.1. Required derivatives at the corners along edges can be computed from their derivatives in x and y directions. The control point at the gravity center can be flexibly determined, for example using the average of the six control points at the terciles of edges. To ensure C1 continuity on the edge in the direction perpendicular to the edge, global adjustment of the control point at the gravity center of each triangle patch is required.

The evaluation of the surface z(x,y) is more difficult than rectangular patches [5, 6, 7, 8]. In one Bézier triangle, x, y, z are evaluated independently based on the same parameter (u, v, w), where $0 \leq u \leq 1$, $0 \leq v \leq 1$, $0 \leq w \leq 1$, $u+v+w=1$. With x(u, v, w), y(u, v, w) and $u+v+w=1$, we can compute (u, v, w), and then compute z(u, v, w), correspondingly.

5. Results From Experiments

We present experiment results in 2D and 3D LUTs. With control points defined as in sections 3 and 4, accuracy improvement is illustrated in following figures with the comparison to piecewise linear interpolation. In the figures, the blue lines represent the data measured from transistor-level simulation using 90nm technology; the yellow lines represent the data calculated by piecewise linear interpolation; the purple lines represent the data from our Bézier curve interpolation.

In the figures of 2D LUTs, piecewise cubic Bézier curves are compared with piecewise linear interpolations, and in 3D rectangular Bézier patches are compared with bilinear interpolations.



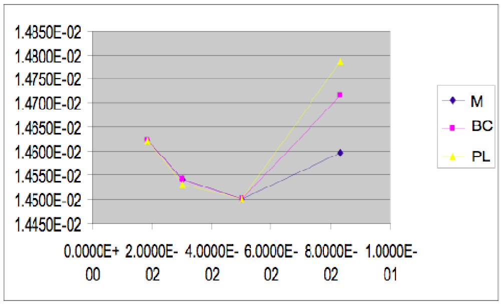
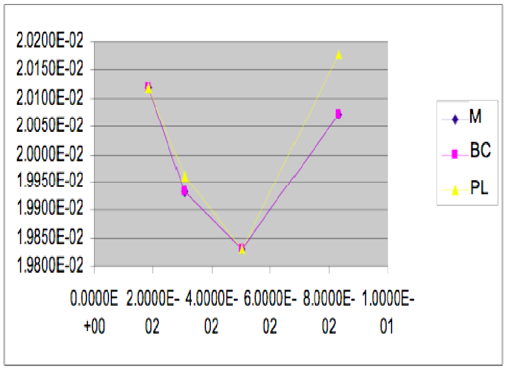


Figure 2: 2-Dimensional Table of an Inverter. The 1st, the 3rd and the 5th points are sampled points, and the other two are from calculation

Figure 3: 3-Dimensional Table of a NOR when Signal Rising. The 1st and the 3rd points are sampled points, and the other two are from calculation



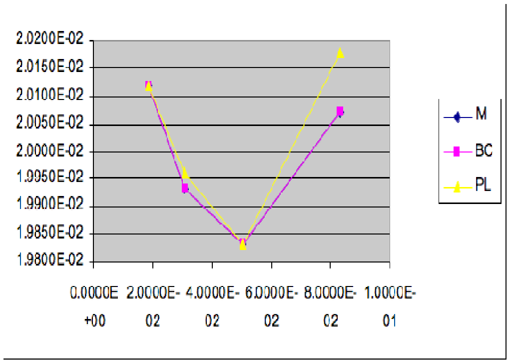


Figure 4: 3-Dimensional Table of a NOR when Signal Falling. The 1st and the 3rd points are sampled points, and the other two are from calculation

Figure 5: 3-Dimensional Table of a MUX when Signal Rising. The 1st and the 3rd points are sampled points, and the other two are from calculation

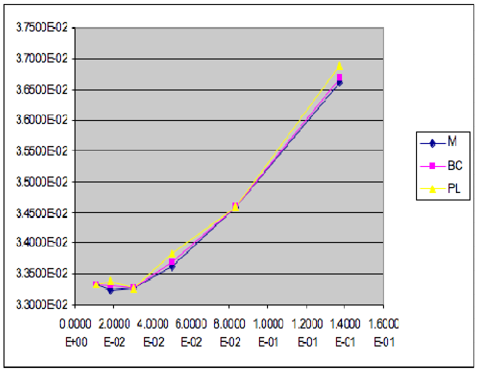


Figure 7: 3-Dimensional Table of a DFF when Signal Rising. The 1st, 3rd, and 5th points are sampled points, and the other three are from calculation

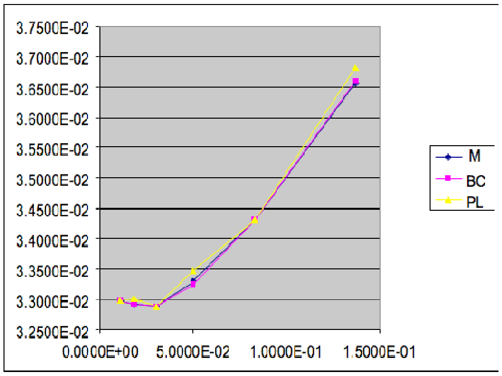


Figure 8: 3-Dimensional Table of a DFF when Signal Falling. The 1st, 3rd, and 5th points are sampled points, and the other three are from calculation

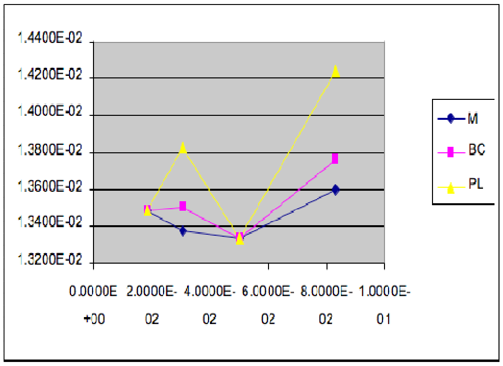


Figure 6: 3-Dimensional Table of a MUX when Signal Falling. The 1st and the 3rd points are sampled points, and the other two are from calculation

Acknowledgments

Our thanks to Xiaolan Bai and Aaron Xu for their assistance on circuit simulation.

References

[1] Tech. rep., Cadence, 1997. Datasheet of encounter library characterizer. DOI= http://www.cadence.com/products/di/library_characterizer/pages/default.aspx
[2] Tech. rep., Synopsys, 2006. CCS power technical white paper, version 3.0.
[3] Boehm, W. and Müller, A. 2000. On de Casteljau's algorithm. Computer Aided Geometric Design. Syst. 16, 7 (2000), 583–586.
[4] Croix, J., and Wong, D. 1997. A fast and accurate technique to optimize characterization tables. In Proc. of Design Automation Conference (1997), 337 – 340.
[5] De Casteljau, P. 1959. Outillages méthodes calcul. Tech. rep., A. Citroën, Paris (1959).
[6] De Casteljau, P. 1963. Courbes et surfaces à pôles. Tech. rep., A. Citroën, Paris (1963).
[7] Farin, G. 1980. Bézier polynomials over triangles and the construction of piecewise Cr polynomials. Tech. rep., Brunel University, Uxbridge, England (1980).
[8] Farin, G. 2002. Curves and Surfaces for CAGD, A Practical Guide, fifth ed. Academic Press (2002).
[9] O'Rourke, J. 1998. Computational Geometry in C, second ed. (1998).
[10] Rodríguez, A., Jiménez, V., Lifschitz, O., Julián, P., and Agamennoni, O. 2009. Implementation of an application specific microporcessor for pwl computations using synopsys. Synopsys Articles (2009).
[11] Salman, E., Dasdan, A., Taraporevala, F., Küçükçakar, K., and Friedman, E. 2007. Exploiting setup-hold-time interdependence in static timing analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Syst. 26, 6 (2007)
[12] Van Loan, C. F. 2000. Introduction to Scientific Computing, second ed. (2000).
[13] Vlach, J., and Singhal, K. 1983. Computer Methods for Circuit Analysis and Design (1983).

Authors

Dr. Eric Y. Chen is a software engineer at Google Inc. This work is partially related to his research in the Ph.D program in School of Computer Science, University of Waterloo, Canada.

Dr. May Huang is a professor and chair of the electrical engineering department at International Technological University, USA. She also serves as a professor at School of Software and Microelectronics, Peking University (PKU), China. She leads a joint research team with members from ITU and PKU. She was a senior, a staff and a principal design engineer of VLSI Technology, Inc., Hitachi Semiconductor America and Virtual Silicon Technology, etc. She participated as a balloter on VITAL, Verilog and Analog Extensions of VHDL to IEEE standard.