Fast Delay Estimation in Segmented Channel FPGAs

Marko Chew
Jung-Cheun Lien
Actel Corporation
Sunnyvale, California 94086

Abstract
Two existing methods are used in computing interconnection delays for FPGAs. The first method, based on SPICE-like circuit simulation, provides very accurate results but requires intensive computations. The second method, based on (some variation of) a bounding box perimeter wire length estimate for the routing, is extremely fast. However, its inaccuracy is high (up to 50%) for FPGAs. Neither methods can be incorporated into optimization applications requiring reasonable accurate delays and a large number (100,000 or more) of delay estimates.

In this paper, we propose a third method, applicable to FPGAs with segmented channels to quickly estimate net delays accurately (within 10% of SPICE). This method allows place and route (and other applications) to take advantage of the more accurate delays during iterative operations.

1 Introduction
Existing fast timing estimation methods based on simple wire length estimates work well for gate arrays. This is because the delays scale almost linearly with wire length to a first order approximation. This simple wire length estimate approach is not applicable to FPGAs because of two factors. The first factor is that interconnect delays cannot be accurately represented by wire length alone; it must also include the number of switching elements. The second factor is the interconnect delays dominates rather than the intrinsic gate delays; this is due to the large values of capacitances and resistances contributed by the interconnect switching elements.

The implication is that a given net can have widely varying delays depending on the FPGA routing selected. For example, in a net with one driver and four input pins, the delay variations can span a factor of 4 depending on the FPGA routing. The quality of result from an optimization application is a strong function of how accurately the objective is being monitored. If the optimization objective involves interconnect delays, delays estimate inaccuracies of 50% are unacceptable.

To accurately estimate the FPGA interconnect delays, the interconnection network must be modeled as an RC network driven by a nonlinear driver. Although there are efficient methods[1, 2, 3, 4] to evaluate RC tree delays, all these methods suffer inaccuracies because of the assumption that the RC tree is driven by an ideal current/voltage source. More importantly, the additional computation required is unacceptable for some applications where upwards of 100,000-1,000,000 delay estimates are required.

We have developed a methodology which addresses the speed and accuracy issues related to FPGA interconnect delay estimates. Our methodology exploits certain characteristics unique to segmented channel FPGA architectures requiring so called one switch routing[5]. This class of FPGA architectures restrict the number of anti-fuses between the driver and any of its input pins. These restrictions result in a limited number of topologically unique routing. In a fully segmented architecture[6, 7] which allows multiple switch routing, the number of
unique routing topologies is several orders of magnitude larger. In this situation, our methodology no longer applies.

Our methodology is composed of two stages. In the first stage, we classify the allowable topologies into a relatively small number of categories. In the second stage, we build a simplified but accurate timing delay model for each category. This timing model is valid for all the routing topologies belonging to that category. This is because the timing model uses as input parameters the routing’s circuit parameters.

The procedure for estimating the timing delays for a given routing requires a computationally inexpensive two-step operation. The first step consists of classification step which determines the routing's topological category. This is then followed by the evaluation step which evaluates the timing model with the specifics of the routing.

In this paper we outline our classification methodology and our approach for creating the timing model for each category of routing topology. We then present preliminary results about the accuracy of our technique.

2 Problem Definition

The details of our approach requires some background information about a segmented channel FPGA architecture. Specifically, we discuss characteristics unique to one-switch segmented channel FPGAs.

In the segmented channel FPGA architecture, the routing tracks in the horizontal and vertical channels are divided into segments[8]. Two or more routing segments can be electrically connected by blowing the appropriate anti-fuses. An anti-fuse is a two terminal device whose two terminals are normally disconnected but can be connected by blowing the anti-fuse. The term anti-fuse and switch will be used interchangeably in this paper.

Figure 1 shows a typical segmented channel FPGA with its rows of logic modules and segmented routing channels. Some of the vertical segments are inputs to the logic modules while other vertical segments are uncommitted routing resources. In the Actel architecture horizontal segments are used just for routing. Switching elements exist at all segment intersections. An intersection is defined to be a crossing of a vertical and hori-
horizontal segments or to be an abutment of two horizontal segments or two vertical segments. In the figure, locations of the unconnected switches are denoted by unfilled circles while closed switches or blown anti-fuses are denoted by an X.

A routing requires the connecting of a module output to the following routing segments:

- Vertical segment.
- Horizontal segment.
- Vertical segment at a module input.

The highlighted lines in Figure 1 illustrates the segment connections necessary to create an electrical path between the driver and an input pin.

Performance considerations limits the number of anti-fuses between the driver and any of its inputs. The one switch routing[5] restriction is directly attributable to the limit on number of anti-fuses. In one-switch routing only, two horizontal segments can be connected with an anti-fuse. In the Actel architecture, a maximum of four anti-fuses can exist between driver and any module input pin.

At minimum, two anti-fuses are required to get from a vertical segment to a module input. As a result the main Steiner trunk can only have a maximum of 2 anti-fuses. This results in the single or double trunk Steiner tree as being the only allowable routing topology. This is the main reason why the number of unique allowable routing topology is much smaller than in the general case.

3 Approach

In our methodology we first classify the routing topologies into categories and then build timing models for each category. To aid in the classification procedure, we partition the set of allowable routing topologies into two groups. Each topology in a group is then further classified into their final categories based on the number of horizontal segments.

The key to the final decomposition step is to consider the number of module input on a given horizontal segment as one of the timing model parameters. The number of unique topology is reduced to manageable number by this step.

In the first group, the module output connects to only one vertical segment which, in turn, is connected to one or more horizontal segments. We call this configuration a single vertical segment. Figure 2 is an example of this typical case along with equivalent RC circuit in Figure 3. Variations in this configuration are further partitioned into their final category depending on the number of horizontal segments. For example, one horizontal segment configuration is considered to be a different category from that of a three horizontal segments. The number of unique topological categories for a group is a function of number of horizontal routing channels.

The second partition contains those routing topologies with the module output connecting to two vertical segments. We call this configuration a double vertical segment. Figure 4 shows this configuration and Figure 5 illustrates the equivalent RC circuit. The topologies in this group are further decomposed depending on the number of horizontal segments.

Once the categories of unique routing topologies have been determined, the equivalent circuit for each category is built. We model the interconnection as a lumped RC network driven by a nonlinear driver. Our modeling is rather coarse since the resistances and capacitances for a given routing segment are lumped into one node. This means module inputs connected to the same horizontal segment are indistinguishable; however, we have found that such a modeling approximation incurs errors of less
than 5% from the distributed RC modeling. The lumped resistances are extracted as a function of the number of blown anti-fuses. The capacitances are calculated as a function of segment length and location dependent overlap parasitic capacitances.

Our timing model building approach borrows from techniques in the field of experimental design[9, 10, 11] to explore the dynamics of complex systems. In this approach a simpler model is constructed to mimic the behavior of a more complex system. The input variables to the timing model are:

- resistance values
- capacitance values
- number of input pins on each horizontal segment.

Our timing model is based on a first-order linear polynomial with two-way interaction terms[10]. In vector/matrix form the polynomial structure is:

\[
delay(\vec{z}) = k + \vec{b}^T \vec{z} + \vec{z}^T \vec{A} \vec{z}
\]

where:
- \(k\) is a scalar coefficient.
- \(\vec{z}\) are RC network’s resistances, capacitances, and loads on horizontal segments.
- \(\vec{b}\) is a vector of coefficients for the main factors.
- \(\vec{A}\) is upper triangular matrix of coefficients with zeros along its diagonal.

An example of the timing model for the class of circuit topology contained in Figure 6 is shown below. The timing model is valid for the three circuits shown since the number of pins on a horizontal segment is one of the timing model parameters. In this example, the vector \(\vec{z}\) contains \([r_1 \ c_1 \ r_2 \ c_2 \ p]^T\) where \(p\) is the number of inputs on the horizontal segment.

\[
delay([r_1 \ c_1 \ r_2 \ c_2 \ p]^T) =
\]
naming conventions to label the various interconnection configurations:

- \( S<\text{number}> \) : single vertical segment case with \(<\text{number}>\) horizontal segments.
- \( S<\text{numberA}>L<\text{numberB}> \) : input delay on horizontal segment attached to the shorter vertical segment and \(<\text{numberA}>\) horizontal segments on the short vertical segment and \(<\text{numberB}>\) horizontal segments on the long vertical segment.
- \( L<\text{numberA}>S<\text{numberB}> \) : input delay on horizontal segment attached to the long vertical segment and \(<\text{numberA}>\) horizontal segments on long vertical segment and \(<\text{numberB}>\) horizontal segments on the short vertical segment.

For example, \( L2S1 \) is the timing model for an input on a horizontal segment attached to the long vertical segment in a RC topology composed of:

1. Two horizontal segments attached to a long vertical segment.
2. One horizontal segment attached to the short vertical segment.

\( S4 \) is the timing model for a single vertical segment topology with four horizontal segments.

Table 1 contains statistics about our timing model accuracy for some representative interconnection configurations. The first half of the table contains result from the single vertical segment case. The second half of the table contains results from the double vertical segment case. Our timing model results are being compared to the results from a SPICE-like simulator using a non-linear driver to drive the lumped RC interconnection network.

In the table we have included the maximum percentages that our timing model deviated from the timing simulator results. In addition we have included the \( R^2 \) statistics from the linear regression curve fitting[12]. The \( R^2 \) statistics is a measure of how well the model fits the observed data. The value of \( R^2 \) is constrained to the interval \( 0.0 \leq R^2 \leq 1.0 \).

---

**Figure 6: Timing Model Circuit Examples**

\[
\begin{align*}
&k + b_1 r_1 + b_2 c_1 + b_3 r_2 + b_4 c_2 + b_5 p + \\
&a_{12} r_1 c_1 + a_{13} r_1 r_2 + a_{14} r_1 c_2 + a_{15} r_1 p + \\
&a_{23} c_1 r_2 + a_{24} c_1 c_2 + a_{25} c_1 p + \\
&a_{34} r_2 c_2 + a_{35} r_2 p + \\
&a_{45} c_2 p.
\end{align*}
\]

We use a standard least-square fitting algorithm to estimate the coefficients in the vector \( \hat{b} \) and matrix \( \hat{A} \) given a set of sample points. It is important to remember that the RC network is being driven by a nonlinear independent source. The sample points for a given routing topology are generated by running SPICE-like timing simulator for various values of the model parameters.

4 Results

We have identified and developed timing models for about thirty distinct categories of routing topologies. In the case of two vertical segments being driven by module output, a distinction is made between the shorter and the longer segments. We use the following
A low value of $R^2$ indicates a bad timing model while a value approaching 1.0 indicates an almost perfect fit.

The maximum negative error is the percentage that our timing models underestimated the simulation results. Likewise, the maximum positive error is the percentage that our timing models over-estimated the simulation results. Except for one or two outliers, most of the relative percentage errors are well within 10%.

<table>
<thead>
<tr>
<th>Config.</th>
<th>Max Neg Error %</th>
<th>Max Pos Error %</th>
<th>$R^2$ Stat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>-0.688</td>
<td>0.942</td>
<td>0.999756</td>
</tr>
<tr>
<td>S2</td>
<td>-2.136</td>
<td>2.026</td>
<td>0.999392</td>
</tr>
<tr>
<td>S3</td>
<td>-2.277</td>
<td>4.831</td>
<td>0.999082</td>
</tr>
<tr>
<td>S4</td>
<td>-2.142</td>
<td>4.679</td>
<td>0.998569</td>
</tr>
<tr>
<td>S5</td>
<td>-6.365</td>
<td>7.209</td>
<td>0.995508</td>
</tr>
<tr>
<td>S1L1</td>
<td>-2.537</td>
<td>5.100</td>
<td>0.999370</td>
</tr>
<tr>
<td>S1L2</td>
<td>-2.407</td>
<td>4.756</td>
<td>0.999417</td>
</tr>
<tr>
<td>S1L3</td>
<td>-2.484</td>
<td>5.190</td>
<td>0.999615</td>
</tr>
<tr>
<td>S1L4</td>
<td>-4.485</td>
<td>3.058</td>
<td>0.999683</td>
</tr>
<tr>
<td>S2L1</td>
<td>-2.289</td>
<td>7.430</td>
<td>0.999187</td>
</tr>
<tr>
<td>S2L2</td>
<td>-2.978</td>
<td>8.160</td>
<td>0.999234</td>
</tr>
<tr>
<td>S3L1</td>
<td>-2.884</td>
<td>10.660</td>
<td>0.999080</td>
</tr>
<tr>
<td>S4L1</td>
<td>-5.267</td>
<td>12.295</td>
<td>0.998802</td>
</tr>
<tr>
<td>L1S1</td>
<td>-2.389</td>
<td>2.339</td>
<td>0.999343</td>
</tr>
<tr>
<td>L1S2</td>
<td>-3.190</td>
<td>3.477</td>
<td>0.999202</td>
</tr>
<tr>
<td>L1S3</td>
<td>-3.355</td>
<td>4.486</td>
<td>0.999391</td>
</tr>
<tr>
<td>L2S1</td>
<td>-2.446</td>
<td>5.533</td>
<td>0.999103</td>
</tr>
<tr>
<td>L2S2</td>
<td>-3.347</td>
<td>5.593</td>
<td>0.999275</td>
</tr>
<tr>
<td>L2S3</td>
<td>-4.082</td>
<td>7.593</td>
<td>0.999239</td>
</tr>
<tr>
<td>L3S1</td>
<td>-2.517</td>
<td>9.640</td>
<td>0.999224</td>
</tr>
<tr>
<td>L4S1</td>
<td>-6.489</td>
<td>6.290</td>
<td>0.999411</td>
</tr>
</tbody>
</table>

Table 1: Timing Model Accuracy

5 Conclusions

We have developed a computationally efficient and very accurate method for estimating the timing delays in a segmented channel FPGA. Such a fast delay estimation technique is necessary in optimization applications where upwards of 100,000 fairly accurate delay estimates are required. Our method takes advantage of the fact that only a relatively small number of routing topology are allowable in one-switch routing segmented channel FPGA architectures. Our results show very good agreement with the SPICE results.

6 Acknowledgments

We would like to thank Runip Gopisetty, Sinan Kaptanoglu and William Shu for the many fruitful discussions relating to this work and to Yinan Shen and Meng-Jin Lin for their helpful criticisms in writing this paper.

References


