Literature Review

Nicholas Sica

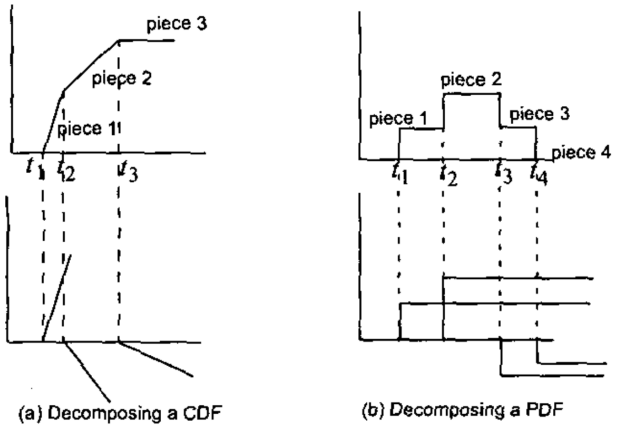

Fig. 1: Decomposing CDF and PDF functions.

Static timing analysis allows a user to analyze circuits and verify that inputs are arriving at the correct time, taking care not to wipe away the outputs too quickly. Traditionally, static timing analysis uses worst, best, and nominal cases to decide on different transistor parameters. While this has been useful in modeling imperfections in a computationally fast way, it is not accurate. Block-based static timing analysis allows for a way to incrementally analyze a circuit. This paper proposes a block-based approach that allows for uncertainty using different probability functions. The paper also proposes a novel way to deal with reconvergent fanouts, an issue that increases complexity significantly in static timing analysis.

$$A_o = max(A_i + D_{io}, A_j + D_{jo}) \tag{1}$$

$$C_o = (C_i \circledast P_{io})(C_j \circledast P_{jo}) \tag{2}$$

In the proposed methodology, arrival times are modeled as cumulative probability distribution functions (CDFs) while gate delays are modeled as probability density functions (PDFs). While any form can be used for the CDFs, they use piecewise linear functions due to ease and speed of the function type. Typically, the max and operation used in static timing analysis are given by equation 1. In this paper, convolution and other probability properties are used to determine these values giving us equation 2 for max and add. In the equations A_i is the arrival time at node i, A_j is the arrival time at node j, D_{io} is the delay between nodes i and o, D_{jo} is the delay between nodes j and o, C_i is the CDF of node i, C_j is the CDF of node j, P_{io} is the PDF of D_{io}, and P_{jo} is the PDF of P_{jo}. The new equation can be substituted in for the old equation to get it working right away. Using this new equation without the reconvergent fanout optimization would yield more accurate results, but a big performance hit since the computational complexity is increased by an exponential factor.

The piece-wise linear functions are decomposed as seen in Fig 1 before they are used in convolutions. The piece-wise linear functions can be decomposed in a different number of pieces. The greater the number of pieces, the higher the accuracy of the static timing analysis. Increasing the number of pieces also affects the time complexity of the proposed method in an exponential way.

$$\mu_x = \mu_z - \mu_y \tag{3}$$

$$\sigma_x^2 = \sigma_z^2 - \sigma_y^2 \tag{4}$$

Reconvergent nets are handled without using path tracing which is computationally expensive. Instead, the paper uses statistical subtraction to deal with reconvergent nets. The mean and variance can be computed directly from the CDF or PDF easily using equation 3 and equation 4, respectively. These can then be matched to determine the distribution in a technique called moment matching. The algorithm for dealing with all this uses a dependency list to capture all the vertices in which the current node's arrival time depends on. This algorithm pseudo-code is shown in Algorithm 1. We can then further use dependency lists in multi-input gates and expand on the pseudo-code to create another algorithm as shown in Algorithm 2. This algorithm is used to reduce the dependency list input to a single vertex

to allow for easier use of the equations discussed earlier.

Algorithm 1 Propagate dependency list

DL_o = NULL
if gate output has fanout ¿ 1 **then**
  add output node to DL_o
end if
for each input i of gate contributing to output **do**
  for each vertex v in DL_i **do**
    add v in DL_o using insertion sort
    in descending order by level
  end for
end for

Algorithm 2 Compute arrival time with reconvergent fanout

A_o = -∞
for each input i **do**
  L = NULL
  for each vertex v in DL_i **do**
    if (V covers more than one input)
      && (Y does not appear in L) **then**
      insert Y according to level in L
      mark inputs that v covers
    end if
  end for
end for
if L is empty **then**
  proceed as in independent case
else
  for each v in L **do**
    A_{ov} = $-\infty$
    for each input i that it covers **do**
      A_{ov} = max(A_{ov}, $A_i - A_v + D_{io}$)
    end for
    A_o = max(A_o, A_{ov})
  end for
end if
return A_o

The results shown in Fig 2 show an increase in accuracy with this new method. Monte Carlo simulations were used as a golden model to compare against. The results show a peak error of 0.79% for this new algorithm while a peak error of 27.2% is shown for the worst case method. The lack of numbers to show how long each run took was a bit odd and was a major point on contention in the paper. Their time complexity is worse so we should see a hit to speed except in cases where the new reconvergent fanout algorithm should help it. The lack of speed numbers makes the conversation about speed and time complexity and potential speed-ups from the reconvergent fanout algorithm significantly weaker. They talk about how using the reconvergent fanout algorithm causes a 10-30% slowdown over not using it, but those numbers are not too useful without comparison to current works. Overall this method could be useful for dealing with reconvergent nets, but there is not a good way to know for sure without concrete time numbers to compare if the accuracy increase is worth the time complexity. Changing the amount of pieces used in the piecewise linear function can also increase accuracy but at the cost of exponentially increasing time complexity. There needs to be more experimentation in the speed versus accuracy aspects of this proposed methodology.

Circuit	Monte Carlo Method 99%	Worst Case Method		Proposed Method 99%	
	Delay (ps)	Delay (ps)	Error (%)	Delay (ps)	Error (%)
C432	3588	4426	23.3%	3610	0.61%
C499	3505	4283	22.1%	3525	0.57%
C880	3344	4088	22.2%	3359	0.44%
C1908	3574	4355	21.8%	3587	0.27%
C2670	2549	3094	21.3%	2557	0.31%
C3540	5251	6500	23.7%	5280	0.55%
C6288	20704	26351	27.2%	20868	0.79%
C7552	5262	6565	24.7%	5299	0.69%

Fig. 2: Accuracy comparison of proposed method versus worst case and Monte Carlo method.