Algorithm for Minimization of Clock Skew in VLSI Systems

Gagan P
SRN: PES1UG20CS144
Section:C

Shreyas S
SRN: PES1UG20CS408
Section:G

Abstract

In 21st-Century VLSI design, clocking plays crucial roles for both performance and timing convergence. A skew in the clock signal to components can interrupt the pacing of components and may cause them to not function as intended due to synchronization issues. In this project we present a basic algorithm that can route the clock signal to different components distributed across a 2-D space that represents anything between a PCB to an integrated circuit die. It aims to effectively use the properties of centroids of clusters to distribute the signal across various nodes that are input into the algorithm. These nodes are clustered based on the euclidean distance between each other, using a modified hierarchical clustering algorithm. The modification is to exclude outliers nodes that can be added to a cluster due to its relative proximity to said cluster. This algorithm was tested on a set of nodes and was observed to return satisfactory results.

1 Introduction

The operation of most digital circuits is synchronized by a periodic signal known as a "clock" that dictates the sequence and pacing of the devices on the circuit. This clock is distributed from a single source to all the memory elements of the circuit, which for example could be registers or flip-flops. In a circuit using edge-triggered registers, when the clock edge or tick arrives at a register, the register transfers the register input to the register output, and these new output values flow through combination logic to provide the values at register inputs for the next clock tick.

Clock skew (sometimes called timing skew) is a phenomenon in synchronous digital circuit systems in which the same sourced clock signal arrives at different components at different times due to gate or, in more advanced semiconductor technology, wire signal propagation delay. The instantaneous difference between the readings of any two clocks is called their skew.

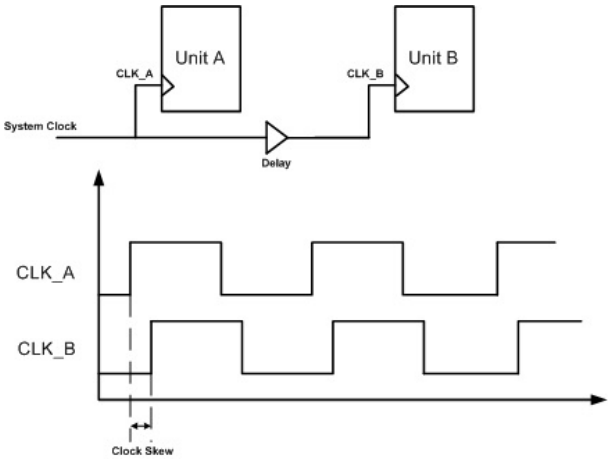

Figure 1: Clock skew

Clock skew can be caused by many different things, such as wire-interconnect length, temperature variations, variation in intermediate devices, capacitive coupling, material imperfections, and differences in input capacitance on the clock inputs of devices using the clock. As the clock rate of a circuit increases, timing becomes more critical and less variation can be tolerated if the circuit is to function properly.

There are two types of clock skew: negative skew and positive skew. Positive skew occurs when the receiving register receives the clock tick later than the transmitting register. Negative skew is the opposite: the transmitting register gets the clock tick later than the receiving register. Zero clock skew refers to the arrival of the clock tick simultaneously at transmitting and receiving register.

2 Algorithm

This project uses a Hierarchical clustering algorithm with a single agglomeration criteria, the Euclidean Distance between the nodes.

The Minkowski distance of order p (where p is an integer) between two points

$$X = (x_1, x_2, ..., x_n) \text{ and } Y = (y_1, y_2, ..., y_n)$$

is defined as

$$D(X, Y) = (\sum_{n=1}^{n} (|(x_i - y_i)|^p)^{1/p})$$

The Euclidean distance is a specification of the Minkowski Distance metric, with the value of the Minkowski parameter $p = 2$. When in the context of points in 2-D space, it gives the physical distance between the points.

The Euclidean distance between every point is calculated and then the closest values are clustered using agglomerative hierarchical clustering algorithm, with a limit to the intra-cluster distance at a fixed maximum. Once the clusters are generated, their centroids are picked as the distribution node and it is connected to every element in the cluster. This centroid is then connected to the centroid of the entire set, which is assumed to be the source of the clock signal.

3 System Architecture And Workflow

The workflow of the algorithm is such that, it takes the input as a text file in a format that contains the spatial information of the nodes (representations of registers) as coordinates in a 2-D plane.

This information is then processed by the algorithm, which in this case was written using the C++ language. The algorithm explained in the previous section is made to run on the dataset. The output obtained is then stored into two CSV(comma separated value) files for further analysis/ visualization. The format of the first CSV is: node-ID, x , y , h, where node-ID represents the assigned number to the node, x and y are the coordinates of the node , and h stands for highlight which is a Boolean value which determines if the node is to be highlighted or not. This is used to determine if the node is an edge node or a distribution node.
The second CSV file indicates the features of the edge connecting these nodes.

These CSV's are then processed by a Python script where libraries *networkx* and *grave* were used to represent the clock tree with the necessary features extracted from the CSV for visualization.

4 Results

In order to evaluate the algorithm, it was run over a dataset consisting of 7 nodes whose coordinates were randomly allocated. The results obtained after execution are as follows:

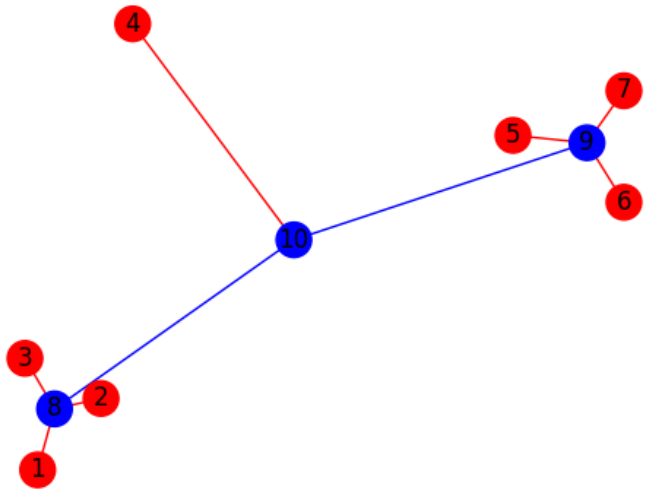

Figure 2: Example clock tree

In the image, the red represents component nodes that are input into the C++ program through a text file, blue represents signal distribution nodes and the edges are just paths that are to be routed into the PCB/silicon die layer.

5 Future Work

The algorithm presented draws inspiration from hierarchical clustering algorithms. While hierarchical clustering algorithms are an effective way to analyze and address the given problem, the use of better clustering algorithms to base the algorithm upon can be done. Moreover, since the algorithm uses the spatial features of the nodes, better and more efficient methods to further exploit these features can be used to optimise the algorithm. Since the dataset fed into the algorithm treats it as an ideal scenario, further constraints on the allocation of the deployment like existence of data paths along the PCB, the prevention of the crossing of paths can be considered.

6 Conclusion

The algorithm presented was executed on a randomly generated small sample dataset containing 7 nodes in space with the spacial coordinates given. Upon execution, the algorithm was able to give out the good path to all the nodes in such a way that the calculated clock skew was as low. Considering the sample was randomly generated, it is reasonable to assume that this can be scaled for handling larger datasets consisting of more varied and complex scenarios.