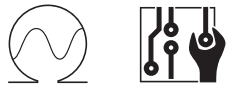
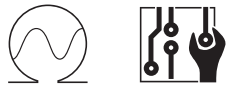


Technische Universität München
Department of Electrical Engineering and Information Technology
Institute for Electronic Design Automation

Timing with flexible flip flop model using piecewise linearization

Master Thesis

Armin Sadighi



Technische Universität München
Department of Electrical Engineering and Information Technology
Institute for Electronic Design Automation

Timing with flexible flip flop model using piecewise linearization

Master Thesis

Armin Sadighi

Supervisor : Dipl.-Ing. Bing Li
Supervising Professor : Prof. Dr.-Ing. Ulf Schlichtmann

Abstract

The continuous downscaling of semiconductor technology according to Moore's law, has made high speed VLSI circuits possible. Determining the clock frequency for these circuits is a crucial part of circuit design. Static Timing Analysis (STA) is a conventional method used for this purpose that is widely spread and used. This conventional method is not suitable for nanometer designs anymore. In timing signoff for SOCs, even a timing violation of few picoseconds will cause degradation of the design quality and also increase the turnaround time. On the other hand the timing analysis, might mistakenly cause an over optimistic result on a critical path, leading to propagation of timing delays on the path causing errors. These problems, caused by STA, are due to over pessimism or over optimism induced by the independent consideration of setup/hold time with clock-to-Q delay of flip-flops.

The goal of this thesis is to propose a new method for timing analysis based on piecewise linearization that considers the interdependency between setup/hold times and the clock-to-Q delay of flip flops and hence reducing the pessimism and optimism of conventional STA method leading to more accurate timing analysis for circuit design. First the interdependency between the setup/hold time and clock-to-Q delay of flip flops is captured by modeling their relation with a piecewise linear function. Then a mathematical method will be introduced and used to convert this piecewise linear function to a linear function. The timing constraint framework for the design is constructed and finally a linear programming solver is used to optimize the clock period.

The experimental results obtained by implementing the method, also prove the advantage of the proposed methodology over the conventional STA tools. The results show improvements in clock period, compared to conventional methods by up tp 12%.

Contents

List of Figures

List of Tables

1. Introduction

Nowadays the advancement of digital technology and devices has changed peoples lives. Modern computers and electronic devices have found their way in peoples jobs, cars and daily routines. Multimedia systems with high performances let us present digital contents and at the same time interact and navigate with them. Mobile phones which were merely devices used to communicate to one another, have changed to smart phones that are replacing a lot of other things like calenders, organizers, clocks and many more, because of these advancements. Cars are using more and more electronic and embedded systems that control critical parts like the breaking system. Todays gaming consuls have more processing power than the supercomputers of the previous decades. Also the memory required to store all the information of a company is now comparable to the memory of smart watches that fit on the human wrist.
These are just a few examples to give a feeling about the change that the progress in digital design and more specifically, CMOS design has brought us. This unbelievable progress is because of miniaturizing the transistors and also improvements in the process that leads to its manufacturing. Most fields in engineering and science, have a tradeoff between price, power and performance, while in this field, as the transistors become smaller spectacular things happen. With the increase of the number of transistors on the chip the functionally improves, at the same time the structure size decreases, and reduces the parasitic transistor capacitance which leads to less power dissipation. Also the smaller the transistors become they get faster. This fact has not only changed electronics, but a lot of change in society itself. More's law indicates that "the number of transistors in an integrated circuit doubles approximately every two years". Based on Moore's law more and more transistors are fitting on the same area of a chip and this continuous downscaling of the transistor brings us theoretically twice the speed in the same amount of time. So there is also an outline for the rate of miniaturization that has been hold true for over forty years.
One of the main concerns that makes realizing this result very hard is timing. The design teams of digital circuits could work for several months on the architecture of a chip and iterate in the design flow of the chip to reach the timing goals specified for the chip. For nanometer designs this task becomes more and more complicated and sophisticated. The designers sometime have to remove a few picoseconds of timing error, which can be extremely hard, time consuming and challenging. Some of the solutions that are used are gate width or channel length sizing and buffer addition, which causes the design quality to decrease. Another effect that these methods could have is the increase in chip area size. Also the time to market and turnaround time of the design will increase, which are all undesired effects that should be avoided in the design process of modern chips.
For ensuring that the chip can work under the required timing constraints, signoff with static timing analysis (STA) is necessary. This method has some imposed pessimism that ensures the correct working of the chip. In the new designs, with decreased transistors sizes, and hence higher clock frequencies that can be around multiple gigahertz, the pessimism used in

conventional methods is not suitable and acceptable any longer. New timing methods are hence required to obtain better results and prevent the aforementioned problems.
The present work discusses the problem of conventional timing methods and the root of this problem in the second chapter. After the complete picture of the problem is given, in the third chapter a new method for solving this problem is proposed and explained. The effectiveness of this solution will be put under test in the fourth chapter, where the experimental results of the method are given. Summary and conclusion in fifth chapter will conclude the discussion about the topic with some ideas and suggestions to further improve this work.

1.1. Preliminaries

Before going deeper in the topic, some background has to be covered for better understanding of the subjects. For this reason a brief look will be taken at flip flop timing components and the working of static timing analysis (STA).

1.1.1. Flip flop timing components

There are some timing components defined for each flip flop, as can be seen in figure 1.1. These set of parameters are setup skew, hold skew, and clock-to-Q delay. The first two parameters impose restrictions on the input signal of the flip flop and are important for the correct working of flip flops. If they are not in a specific region the flip flop will go in metastable state and malfunction. The third parameter, namely the clock-to-Q delay is defined for the output of the flip flop. Unlike the input parameters it does not imply any condition on the flip flop but is important for the timing analysis because it is the time that the signal will be available for the next stages of logic in the circuit.

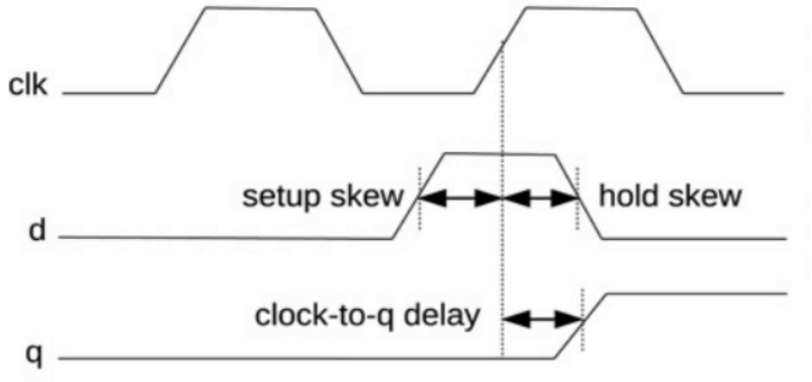


Figure 1.1.: Timing parameters of a flip flop

These parameters are formally defined as follows:

- Setup skew: The time the data input became stable before the clock reaches the flip flop.
- Hold skew: The time that the data at input is constant after the clock has reached the flip flop.
- Clock-to-Q delay: The time required until the output is stable and ready after the clock signal arrived the flip flop.

Normally instead of setup skew and hold skew that are the actual physical and measurable values and can vary case by case, constant values are defined as a constraint such that if these constraints are not held there will be an error and all the checks are done on these values. These values are called setup time and hold time and are defined as follows:

- Setup time: The minimum amount of time that the input data must be ready before the clock signal reaches the flip flop.
- Hold time: The minimum amount of that the data input must remain unchanged after the clock reached the flip flop.

These values can be observed graphically in figure 1.2, where the C signal is the clock signal inputted to the flip flop, D is the data input and Q is the output of the flip flop.

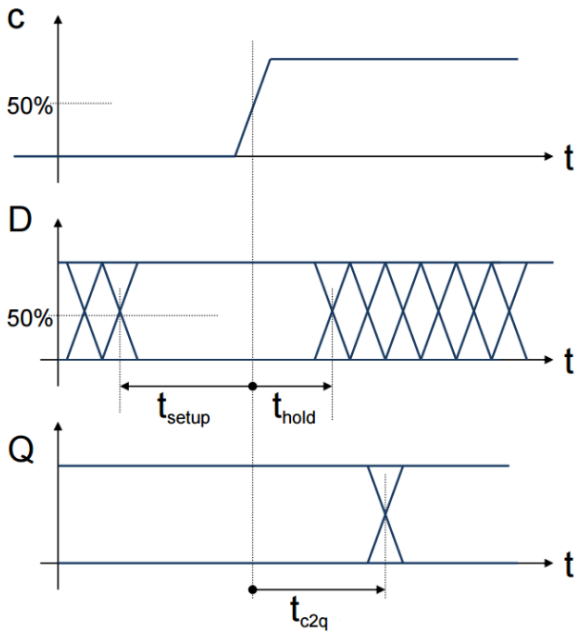


Figure 1.2.: Waveforms representing setup time, hold time, and clock-to-Q delay

1.1.2. Statistic timing analysis (STA)

There are many different techniques to verify the timing of a digital design. One of the most used and accepted ones is called Static Timing Analysis (STA). This method does not depend on the input values of the design, meaning the input test vectors for the design, that 2^n of them exists with n inputs for the chip, do not all need to be tested and checked, and this is the reason this method is called static.
The final goal of STA is to validate if the design can work under certain conditions with a specific speed, meaning that at the required clock frequency no timing violations will happen in the circuit. This method, in its basic form, is a function that gets the netlist of the circuit as input, the cell library of the used technology and also a proposed clock period. The output of this function is also a true or false answer. If the answer is true the meaning is that the inputted proposed clock period is suitable for the design. On the other hand if the answer is false the proposed clock period will not work for the chip and has to be changed. The STA returns a timing log with detailed information about the timing analysis. The basic functionality of the STA is represented in figure 1.3.

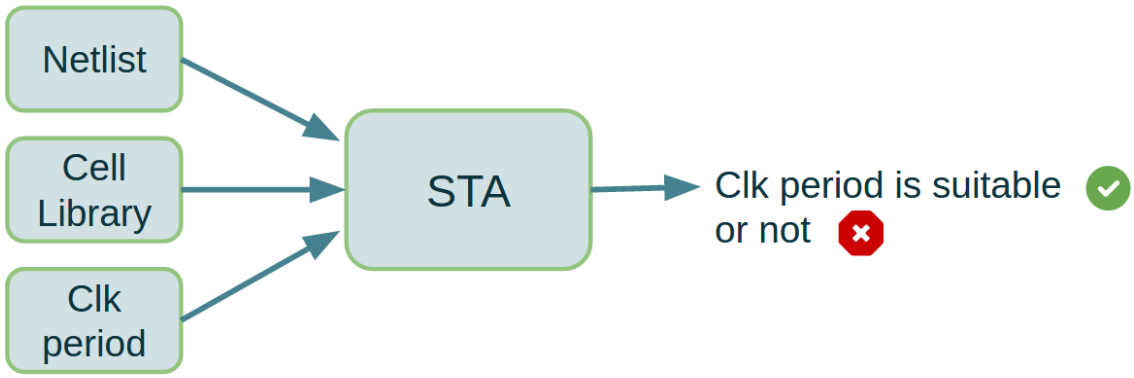


Figure 1.3.: The functionality of a basic STA tool

In STA analysis the method has to make sure that the timing constraints are held true. For this the analysis has to check that the input of the flip flop, data input, is ready and stable by a margin of setup time before the clock signal arrives at the flip flop. Also the value of the input should not change on the flip flop for hold time after the arrival of the clock. This constraint guarantees that the data can be captured correctly and without any error by the flip flop(Neil H. E. Weste 2011). After the data has been successfully captured and stored in the flip flop, it takes clock-to-Q time delay to propagate the value out of the flip flop, after the clock signal was received.
The STA gets it's delay model data about the combinatorial and sequential components in the cell library input. For combinatorial gates this data is normally stored in a two dimensional array called look up tables. The indexes of this look up table are the slew rate of the input to that specific gate and also the load of that gate. So if the STA has the information about

the input slew and the load of a certain gate it can calculate the propagation delay. On the other hand for the sequential elements of the circuit, this differs a little bit. For a flip flop, which is an important sequential element of circuits, the setup and hold time constraints for the input of the flip flop are obtained by, because there are clock as well as data input to the flip flops, the clock slew and also the data input slew to the flip flop as indexes. The clock-to-Q delay as well as the output slew are specified as the output of the look up table with clock slew and the load seen at the output as indexes (The open source liberty library modeling format specification 2016).
The required analysis of the STA happens by propagating the arrival times of the data for all the flip flops, by starting at the primary inputs or the output of the previous stage flip flops and finishing at primary outputs or the flip flop data inputs in the next stage of the design. The question that remains is that where does the STA get the information about the load of the elements as well as the other required parameters. The answer is that after the placement and routing of the components in a circuit, using a standard cell library that contains the required information, the load of each gate is determined. Also the tools used for clock distribution, together with the capacitance of the clock pins, provide the clock slew information very accurately for the clock network used in the design. Now the information required for the start of the STA process is provided. With the calculation of output slews and also the clock-to-Q delay of all the flip flops the propagation of the times that the data arrives at the flip flops, the arrival times, starts. STA has to check the minimum and maximum combinatorial paths in terms of propagation delay to ensure that no setup or hold time violations happen. For the setup time violation, a time called the required time is defined as the latest time that the data has to be ready and steady in the input of the flip flop. Also the data will be available at that point after it propagated from the flip flop and has passed through the maximum delay path of the combinatorial circuit. This time is called the available time of the data. The time between the available time and required time is a window that the data has to be ready and stable in this window otherwise there will be a setup constraint violation. This concepts is illustrated in figure 1.4. On the other hand for the hold constraint violation the minimum delay path has to be checked so that it does not change the flip flop input sooner than the hold time is over.

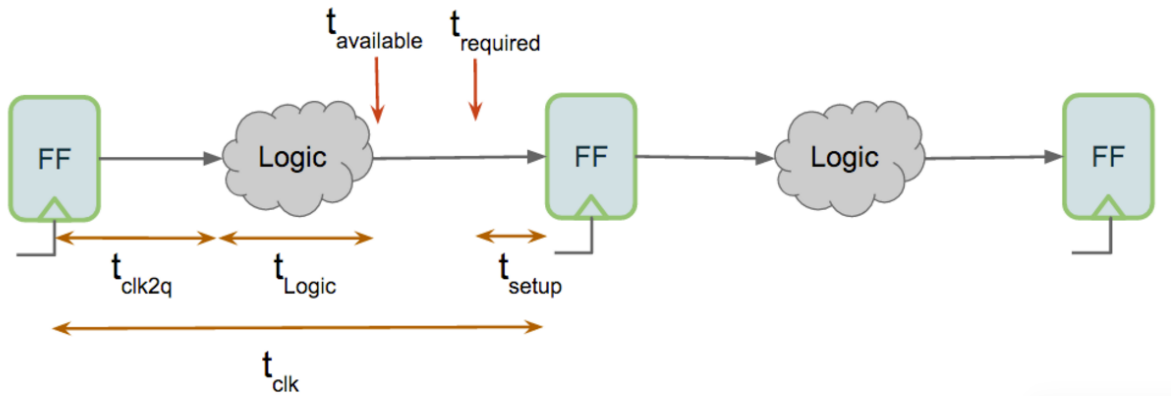


Figure 1.4.: The graphical representation of STA calculations

For the hold time the constraint can be mathematically written as:

$$t_{clk2q} + \min t_{logic} \geq t_{hold}$$

Also for the setup time constraint the formulation will be written as:

$$t_{available} = t_{clk2q} + t_{logic}$$
$$t_{required} = t_{clk} - t_{setup}$$

and the constraint will be checked as $t_{available} < t_{required}$ such that the signal is in the window. The difference between these two values is also called the slack time, meaning that $t_{slack} = t_{required} - t_{available}$, and the slack needs to be positive. An STA will output the slacks for the stages in its timing analysis report.

2. Problem formulation

In the last chapter the importance of timing analysis for circuits and the fact that conventional methods are not suitable anymore were discussed. This chapter a deeper look will be taken into the problem and the reason for the need for a new timing analysis method will be investigate. Also some previous endeavors to solve this problem and their effectiveness will be discussed.
To analyze a specific circuit the STA tools are using the information gathered from the cell libraries. Hence it is important that the information residing in these libraries are accurate enough. Basically the correctness and accuracy of the information provided by the cell library is what guarantees the correctness and accuracy of the STA tools(Patel 1990)(Cirit 1991). For a flip flop the important information residing in the cell library are the setup/hold time constraints and inaccuracy in these values causes the result of the STA to be either over optimistic or over pessimistic. In case of an over optimistic result, the fabricated chip will not work and the circuit fails, leading to monetary loss and costumer dissatisfaction, whereas in a case of over pessimism , the circuit will not reach the possible frequency. This degrades the circuit speed and its whole capacity will not be used. The second case is still better than a malfunction chip but it is also not acceptable for gigahertz frequency domains. The main source of this inaccuracy in cell libraries is due to the interdependent characterization of timing constraints that are actually dependent to each other. The most important characterization that is discussed in this work is the relation between setup/hold skew and clock-to-Q delay. These values are characterized independently in practice, however it has been shown in research that these values are not independent and there can be introduced more than just one pair of setup time and hold time and this was first introduced in (Lang 2004). How this characterization works in traditional ways is that in cell library characterization, setup and hold times are captured independently.
The clock-to-Q delay is calculated using a constant input data and the consideration of infinite setup and hold times. After that for calculating the setup time is, the relation between setup skew and clock-to-Q delay at a specific and constant hold skew must be examined, which is called "counterpart skew" (Neil H. E. Weste 2011). The same process is done for the hold time calculation with a fixed and constant setup skew. For this process the plot clock-to-Q delay based on setup/hold skew is depicted. The calculated clock-to-Q delay will be called the nominal delay on the graph and setup/hold time will be the setup/hold skew that represent the nominal clock-to-Q delay that has been 10% degraded on the graph. Figure 2.1 has been produced for the flip flop in 90 nm technology in (E. Salman & Friedman 2007) and is presented here in the same way. In this image the nominal clock-to-Q value that had been calculated independently with the chosen input is marked. The point where this value is increased by 10% is marked as well. On this point for setup/hold skews the corresponding value is chosen as setup/hold time.

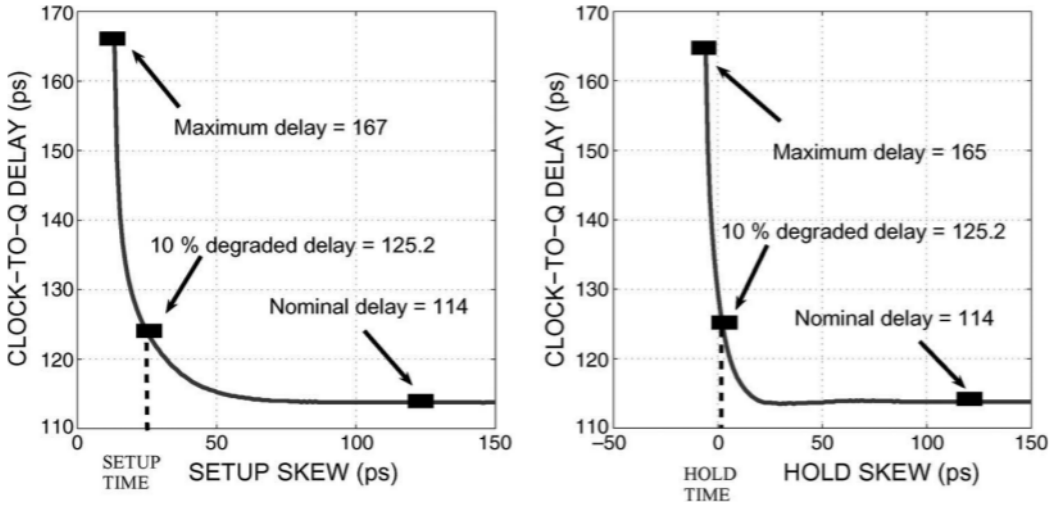


Figure 2.1.: Independent setup time and hold time characterization based on nominal clock-to-Q delay

In 2.1 according to (Stojanovic & Oklobdzija 1999), the plot can be divided in three sub domains. The "stable" region, the "metastable region" and the "failure" region. The part of the graph that the clock-to-Q delay seems to be independent of the value of setup/hold skew, is the stable region. As can be seen on the graph, with the reduction of setup/hold skew, the clock-to-Q value decreases with an exponential rate (Princeton 1992). When the skew is extremely small, then flip flop will not be able to store the data correctly and hence the data is wrong. This part of the graph is the failure region, in which it is sure that the flip flop will fail. There is also the third region which lies between these two regions and it is not completely clear what will happen in this region. This region is called metastable region. What can happen in this region is a so called race condition in the output of the flip flop. When this condition happens the output will change between logic zero and one and never get stable making a lot of problems in the circuit.
The important factor in choosing the setup and hold times of the flip flop is the fact that they should not be in the failure region. The reason for this, as mentioned above, is that the flip flop is unable to latch the data input and will fail to operate. So what is normally done is that the setup and hold times are chosen at the point that the transition from stable region to the metastable region happens. There are some different methods to chose this point which are discussed with details in (Stojanovic & Oklobdzija 1999). The most common method is the 10% degradation method discussed above.
The setup and hold times are also not independent, they are dependent on the counterpart skew. In other words, the setup time depends on the hold skew which is used to obtain the clock-to-Q delay information and vise versa the hold time depends directly on the setup skew that was chosen. The change this counterpart skew makes is that when for example for the hold time characterization, the setup skew is considered fixed and constant, the smaller the chosen setup skew becomes the larger the value of clock-to-Q delays and hence the value for the hold time changes. On thee other hand when the setup time is characterized, then the

decision for the hold skew changes the value that will be calculated and used for the setup time. And this part, and the dependence of clock-to-Q to both of the skews, is the part that in conventional methods is ignored and hence makes it unreliable and inconvenient for the new technologies.
The described characterization issue causes two big problems. The first problem is that the independent characterization of the parameters causes one of the cases of over pessimism or over optimism. When the chosen counterpart skew is larger than necessary then the setup hold times are optimistic. The optimism is because the assumption that in the manufactured circuit the responding setup or hold skew is always as large as or even larger than the considered counterpart skew. However when this condition does not happen in the actual circuit then the chip will fail. This failure happens without any kind of violation in the STA and hence the designer believes that the design should work flawlessly. On the other hand if the counterpart skews are smaller than necessary, then the setup hold times are pessimistic. The pessimistic case will lead to false and unnecessary violations during the STA and hence the designers believe that the chip will not work and have to change the design all the time, even though the circuit might have worked and performance and time is lost as a result of this.
The second problem of the independent characterization is the fact that, using the interdependence between the parameters can actually be exploited to make the results of the STA and other timing analysis methods better. So the conventional method is not really a good compromise as well and has to be changed so that a better timing analysis can be obtained.
In the rest of this chapter the previous works that are done in order to improve the timing analysis and their strengths and weaknesses will be discussed.

2.1. Previous works

After the introduction of this problem in (Lang 2004) there have been some works done in order to exploit the interdependency between the timing parameters. The work in (Rao & Howick 2003)the first problem that was discussed, namely the problem that there can be more than one suitable pair for the setup and hold times, was solved. The work presented a pair of interdependent setup and hold time. Their method was to use a characterization that consisted of only two steps The issue of this method was that it only introduced one pair and hence it could not improve the STA process itself and the slacks in STA did not change. Srivastava and Roychowdhury (Srivastava & Roychowdhury 2007)(Srivastava & Roychowdhury 2008) introduce a methodology that can accurately characterize the setup and hold times based on mathematical concepts. They use the Euler-Newton curve tracing that achieves 26 times faster speeds in generating the curves than simulation with spice methods does. In the work of (E. Salman & Friedman 2007), a method is proposed to solve the second problem, namely exploiting this method in the process of STA analysis. This method works on the knowledge that there is a nominal delay for clock-to-Q delay, but there are more than one solution for choosing the setup and hold times based on the 10% degradation. So instead of choosing one pair, distinct pairs are chosen and they are all valid pairs since they fulfill the requirement that they have the same degradation in the nominal clock-to-Q delay. Based on the obtained pairs a constant degradation curve can be obtained. In this graph all the pairs are connected and the graph is built. In (E. Salman & Friedman 2007) the pair that has the minimum sum of setup time and hold

time is called MSHP. Also two other user defined pairs are specified on the graph, called ESP for effective setup pair and EHP for effective hold pair, that determine the end points of the graph that can be used. Because this graph in convex some further simplifications can has been done in (E. Salman & Friedman 2007) and with a piecewise linearization method these points have been connected. The strength of this method is that it can be easily used in the STA tools by extending those tools. The reason behind this is that the only thing changing is the choice for the setup and hold times that has to be iterated on the lines obtained by the graph. In figure 2.2, that has been taken from (E. Salman & Friedman 2007), the constant degradation graph and the chosen points and also the piecewise linear approximation, can be observed.

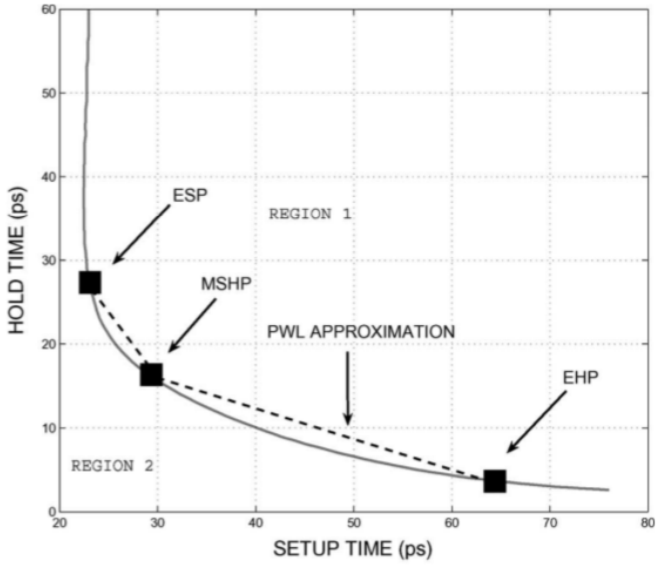


Figure 2.2.: Clock-to-queue delay function based on varying setup and hold skews at 10% degradation

In the above picture it can be seen also two regions are declared, namely as region 1 and region 2. These sections are created by the clock-to-Q curve that separates the space in 2 parts. The part that lies below the curve, region 2, is the optimistic region, meaning that a pair in this section might work in some cases but most probably it will fail if in a real circuit the setup skew and hold skew are equal to the chosen point. So the STA tools try to avoid this region and find points that they can be more confident to work. The second region, which lies above the curve, is the pessimistic section. As the name implies any point here has a setup and hold skew that not only will work but easily satisfy the timing constraint. So the art of choosing good timing parameters is to be in a place that the circuit works without any problem as fast as possible. The piecewise linear lines that are chosen in figure 2.2 are adding pairs to exploit this fact in the STA tool. In a different approach to solve this problems and characterize the circuit timing analysis better Chen et al. (N. Chen 2011) introduce a new model for flip flops. In this work they model the clock-to-Q of the flip flops as analytical functions. The input

parameters of the function consists of not only the setup and hold skews but also the load capacitance of the flip flop, and the clock skew. A new iterative timing analysis is proposed and the function is used to employ this analysis. This proposed approach is a non linear and also considers the complete circuit as a interconnected and interdependent entity. The working mechanism for this method is that a initial clock-to-Q value is associated to each flip flop and then the iterative algorithm starts to compute the values for this flip flops based on the analytical formula. Like a conventional STA, the work of (N. Chen 2011) also outputs the possibility of a given clock period can be used in the circuit or not. At the end the minimum clock period that can be used for the design is found with binary search. This work is a new approach with good results and the strength is that not only does it consider the interdependency between the timing parameters but also all the variables that may affect the clock-to-Q delay, like the load. Also this method solves another big issue of conventional STA methods, which is the problem that each stage of the circuit is considered isolated from all the other parts. In this work all the circuit is an interconnected and interdependent system of flip flops and the timing is much closer to the real world. A drawback of the method is the non linear function chosen, this function does not guarantee the best fit for the actual curve that is happening in reality and is a general formula that the points will be interpolated based on it. Also the iterative approach might based on the initial conditions and also the arrange of the flip flops in the hierarchy, not result to the best global answer. What this means is that based on the order of flip flops and the calculation of the formula for the circuit network, different answers may be obtained and the answer that is calculated from the algorithm does not guarantee to be the order that produces the best timing overall.

Because of the existing problems in the proposed method in (N. Chen 2011), Kahng and Lee in (Andrew B. Kahng 2014) introduced a new method. They suggested to use a sequential linear programming solver, to optimize the global timing in the circuit more effectively. For this a simplified linear model of the data was assumed and used in the linear programming (LP) solver. This method solved the problems introduced in the previous works, but there are still some problems. The first problem of this method is that it can not easily be extended and used with the conventional STA methods which are existing. The other problem is that the optimization method can take unbounded time to converge to a solution in very large designs. What can also be seen as a problem in this case is that the method uses two linear programming problems to solve the problem. The three way relation between setup skew, hold skew, and clock-to-Q delay is broken to three 2 dimensional relations. The graph for clock-to-Q versus setup skew and hold skew, which are two of the relations, and also the data for the relation of setup and hold skew. These graphs are then considered as linear functions to be used in the solver. So not only is the method solving two linear programming problems, which is time and computation consuming, but also the simplification, makes a lot of accuracy in the data to get lost.

Jiang et al. in their recent work (Y. M. Yang 2015) discuss yet another problem in the nanometer design. This is the issue with the "criticality dependent" paths. In these connected path the optimism resulted from the STA, can propagate the delay in a path, and cause the flip flops in the path due to the propagation of delay to fail. In this work the main concerns are simplicity and extendability of the current STA tools. For this to be done a triangle approach is introduced that linearizes a region of the relation graph of clock-to-Q delay versus setup skew and uses the linear function with an estimated error in the STA process. The STA will also be extended to search on this linear function for the best answer. The strength of this algorithm

is the speed of the process, an answer will be reached in constant iterations and as claimed in (Y. M. Yang 2015) in 10 iterations. Although the method is very simple and effective, it has some important drawbacks. One of the drawbacks is that the three way relation between setup skew, hold skew, and clock-to-Q delay is not exploited and only the setup skew is considered. To solve some of the problems mentioned, the present work is going to introduce a new method to tackle this problem. The problems that mostly motivates the work of this thesis are the weakness in the real data modeling. All of the aforementioned works somehow oversimplify the data or consider only some part of the actual relations in the three way relation between the setup/hold skew and clock-to-Q delay. Another issue that appeared in (Andrew B. Kahng 2014) is the use of two linear programming solutions, the current work, due to the modeling technique, will use only one linear solver iteration to solve the complete problem which is an important improvement in computation time.

3. Flexible flip flip model

In this chapter the proposed method, timing with flexible flip flop using piecewise linearization, will be introduced. The general idea will be described first and based on this idea, the steps and challenges required to get to the desired outcome will be discussed. After the method has been proposed, at the end of this chapter, the used algorithm is analyzed and the strengths and weaknesses of the method in general and the algorithm are discussed.

3.1. General idea and timing framework

Based on the previous works discussed in the last chapter, the general idea of a new method that is proposed seems clear. The first objective is to model the circuit as an independent and interconnected network of flip flops to be close to reality. Also to keep the model simple and fast, so it can be used properly , it is desirable that the model is linear. The main purpose is exploit the three way relation between setup skew, hold skew, and clock-to-Q delay, and is also a central part of the proposed method. To solve the model and obtained the final results, instead of iterative methods, to obtain a better global solution to the problem, it is solved using a linear programming solver.
Any digital circuit consists of sequential elements and combinatorial elements. The circuit can be simplified and shown as a network of flip flops with some combinatorial clouds between them. The result of each combinatorial parts enters the next stage flip flops as can be seen in figure 3.1. The clock frequency of the circuit is restricted by these combinatorial parts.

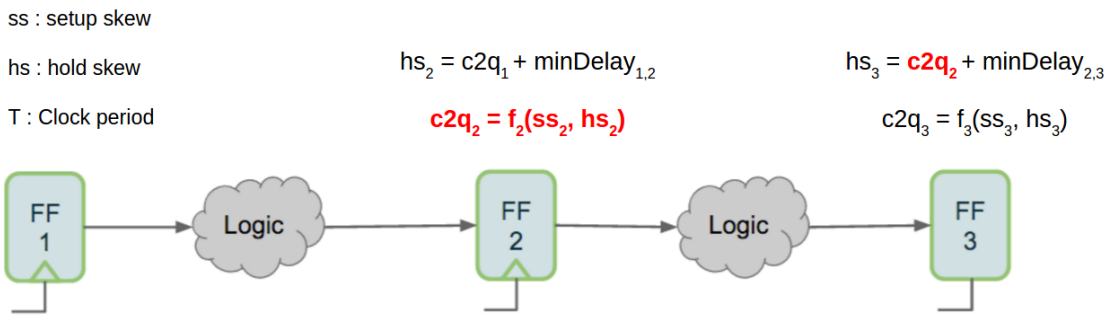


Figure 3.1.: Propagation of clcock-to-queue delay in flip flop network

For every flip flop in 3.1, the equation of its respective hold skew, setup skew, and clock-to-Q

delay can be written. The fact that the timing parameters depend on each other is shown here by writing the clock-to-Q delay as a function, denoted as f, of the input parameters that are the skews for setup and hold parameters. With writing these equations it becomes clear that the stages are interdependent and the parameters of previous stages are actually influencing the next stages. With this the model has reached its first goal, creating a close to reality model that shows the circuits interdependence and interconnectivity. So if all the equations for all the flip flops are written, the complete circuit will be represented by all the equations. The only remaining problem is to solve this equation systems to minimize the clock period, and the answer will be the minimum clock period that can be used for this circuit. Ideally all of the equations are inputted to the linear solver and the program will do the rest. Unfortunately it is not that easy in reality, the problem is that the linear programming solvers are functions and their input is also restricted. The input to a linear programming solver can only be linear functions and constraints and only then the solver can work. The relation between the three timing parameters, namely the setup skew, hold skew, and clock-to-Q delay is a non linear relation.
For this method to be able to work, the clock-to-Q delay surface has to be linearized. When this function has been transformed to a linear function of setup and hold skews, only then it can be used as the input of the linear solver. In the remaining of this chapter, first the actual clock-to-Q delay surface generation will be discussed. This function then has to be linearized and this will be the main focus of this chapter.

3.2. Clock-to-queue delay surface

The graph for clock-to-Q-delay with respect to setup skew and hold skew, has to be generated for later use. This function can be drawn by sweeping its parameters on certain ranges. These parameters are setup skew and hold skew in this specific example. The used flip flop is taken from a 45 nm standard cell library and with the help of spice simulation the parameters will be swept. For the purpose of sweeping the different parameters, a python function is written that runs the spice file, records the output results, changes the appropriate parameters inside the spice file, and runs the file again. The circuit for the written spice script looks as depicted in figure 3.2. In this setting the input waveforms should be chosen carefully, such that the results become reliable. The VDD input is the main source of power that makes the flip flop work. For this input a power supply of 1.1 volts has been chosen.
The VCLOCK signal is the clock input of the flip flop. For this input the PULSE command in spice is used. The general PULSE input in spice is **Vname N1 N2 PULSE(V1 V2 TD Tr Tf PW Period)**. Vname is the signal name, N1 and N2 are the nodes that this signal is applied between them. V1 and V2 represent the peaks of the pulse waveform. TD is the delay time until the waveform should begin, for example if its set to 5 ns then the actual pulse signal begins after 5ns. Tr and Tf represent the rise and fall times of the pulse.

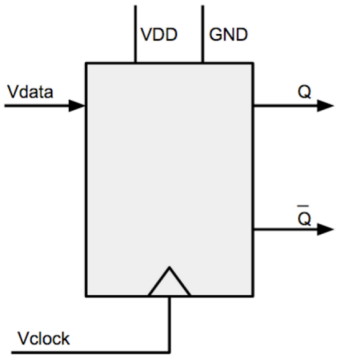


Figure 3.2.: Device under test in the spice simulation

These values are set to 0 if the pulse would be an ideal signal. In reality it is impossible to create an ideal waveform, so to also simulate the effect of these values a value should be chosen for them. PW is the time that the signal is at its highest value, V2, and Period represent the period time for the signal. For the purpose of this project, this signal is applied to the clock of the flip flop with rise and fall times of 110 ps. The signal starts at 0 volts and the peak is 1.1 volts for the case that the power supply is 1.1, and 0.7 volts if the power supply is set to this value. The PW of this signal is 5 ns and the period is 10 ns.
The most important input in this setting is the data input to the flip flop. For this input the spice PWL, piecewise linear. input is chosen because of the flexibility it can provide. The general form of this command in spice is **Vname N1 N2 PWL(T1 V1 T2 V2 T3 V3 ...)**. It can be seen in this command that a T, V patten with identical numbers are followed. This pairs are the basic block for this command and they specify the voltage value of the signal in that specific time. What happens is that the spice simulator creates straight lines between this point tuples and the signal is created. This part is also really important because the setup skew and hold skew are hidden in this command. The input for our purpose is PWL (0s 0v (10000 - setup skew - rise time)p 0v (10000 - setup skew)p 1.1v (10000 + hold skew)p 1.1v (10000 + hold skew + fall time)p 0v). This input will cause a signal that goes from 0 to the max voltage, the voltage of the power supply, setup skew time before the clock, stays high untill hold skew time after the clock and then returns to 0. The python function that controls the spice simulation basically changes these values and reruns the file again. All the other lines in the spice simulation script remains constant. When the clock signal reaches the flip flop in 10000 ps then the input is latched and the output of the flip flop will be captured. The correct function of the flip flop has to be ensured, this is the reason why the input signal returns back to 0 after hold skew, such that if the hold skew is not appropriate for the flip flop the output will not remain constant, the flip flop enters the metastable region and a racing condition will be seen at the output. The waveforms inputted to the system are presented in figure 3.3.

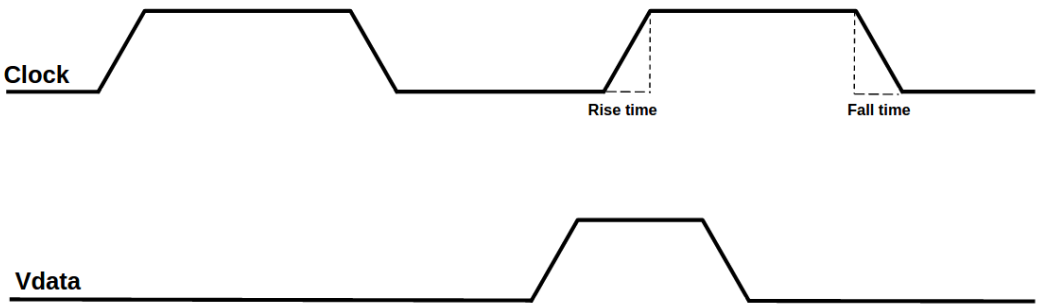


Figure 3.3.: Inputed waveform to obtain the dependent clock-to-Q delay

The function wrapper also has another functionality, namely each time the spice file is run the output is saved in an excel file. The simulation is done for setup and hold skews starting from 8 ps to 100 ps in a 5 ps step increase. After the simulation is completely done the outcome is as presented in figure 3.4.

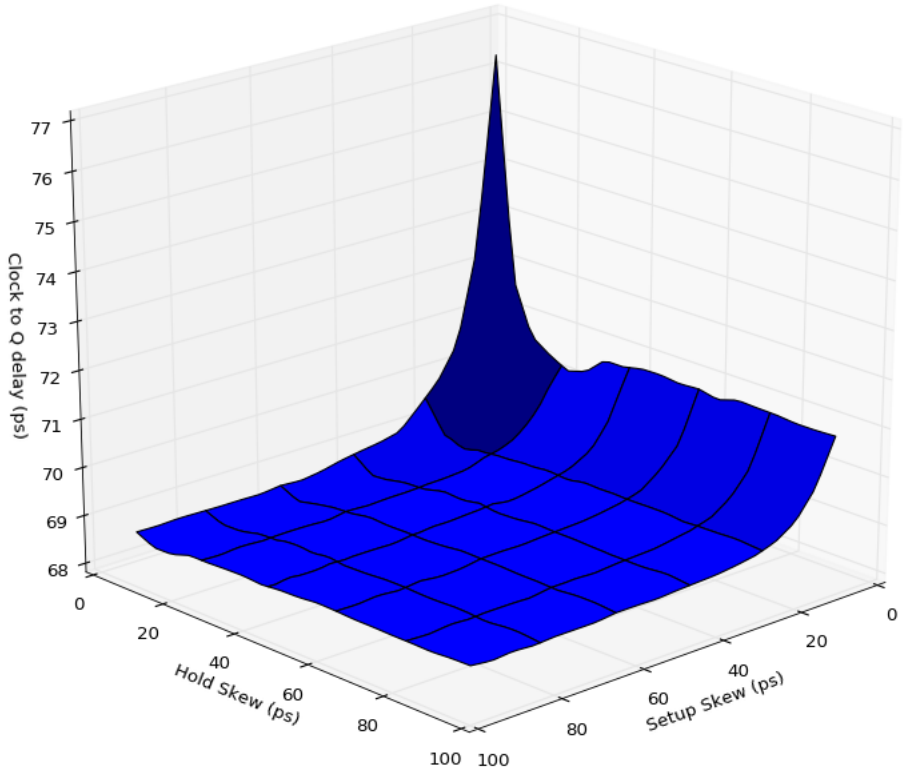


Figure 3.4.: The clock-to-Q delay surface based on the setup skew and hold skew

It can be seen from the graph that the clock-to-Q delay increases when the setup skew and hold skew, either independently or simultaneously, decrease. When the setup and hold skews are larger it seems from the graph that the clock-to-Q delay becomes independent of the input values, namely setup skew and hold skew, but this is not true for the complete domain of the surface.
One important point to consider though, is that this graph is drawn by a limited countable number of points. In the used simulation, with steps of 5 ps from 8 ps to 100 ps, there are 361 points available. But the graph is a continuous graph drawn by guessing the position of all the other points based on regression techniques for drawing a function. To make the graph more accurate, and hence also increase the quality of the used model, more spice simulation should be done. What can be inferred here is that the quality of the surface, which translates directly to the quality of the model, is determined by the simulation. Spice simulation is also a costly and time consuming process, so there can be seen a trade off and it should be carefully thought through in real applications, how much accuracy is required to obtain good models.

3.3. Piecewise linearization of data

After obtaining the three dimensional surface of the clock-to-Q delay based on setup skew and hold skew, it is necessary to linearize the data. The first step in this process would be to piecewise linearize the function. Making a function piecewise linear basically means to divide the function in arbitrary sub domains and in each of this sub domains replace the actual function with a linear approximation of the same function. In two dimensional functions a linear function is a straight line, so the problem is to replace the original function with straight lines in the specific sub domains that are chosen for the function. This line has to represent the data as close as possible to the original function. The clock-to-Q surface is a three dimensional function, so the rule of two dimensional functions has to be expanded to three dimensions. For this, the fact has to be considered that in three dimensions a linear functions becomes a plane instead of a line, changing the problem to replace the surface with an appropriate plane in the chosen sub domains of the three dimensional space.
When approximating a three dimensional surface, it should be considered that the goal is to be as close as possible to the original surface. For reaching this goal the chosen planes should be as close as possible to the the values that the original function has. One way to achieve this is to increase and approach the number of sub domains to infinity, so that the surface can be modeled by infinite number of small planes that build almost the exact surface. This leads us to a extremely close piecewise linear model of the surface, but on the other hand infinite number of linear functions are created that each of them has a formula that has to be stored for the sub domains which is not possible in practice.
The other side of this spectrum would be to consider the complete surface as one sub domain, which is the whole domain of the surface, and approximate the complete function with just one plane. This time there is only one linear equation to store, but there is a huge loss of information. This loss of information is caused due the fact that the surface has ups and downs and other three dimensional features that can not be captured by a linear plane and the approximation would be the furthest approximation too the original surface as possible.
For using the piecewise linear model in the proposed timing model, this effect can be directly

observed. If the number of sub domains and hence the number of planes are low, the clock-to-Q delay surface is modeled inaccurately, which causes the timing result to be inaccurate. On the other hand if the number of sub domains and planes are a huge amount, the result will be more accurate but the latency of the program running to determine the clock frequency would be too high. So basically a trade off between speed and accuracy is happening here that has to be resolved.
So it is important to keep a good balance between the number of planes and the accuracy of the planes. To resolve this problem, the proposed solution is using two variables in the programming called limit value and guard value. The limit value, basically plays the role of the slope constraint. The limit value is responsible to create new sub domains and make more rectangles. How this parameter works is that if the difference between the values of the function in two consecutive x or y values, with the other parameter remaining constant, changes more than the limit value then this difference is not tolerable and a new plane has to be created to capture this jump or rapid change in the data with better details. On the other hand as long as the function grows or decreases smoothly, which is when the difference is smaller than the limit value, then the details can be captured and shown in the same rectangle and the boundaries of the quadrilateral plane will be pushed, until the difference in data becomes large again.
Sometimes there are anomaly in the data or a sudden peak and inconsistency in the data. The best action in these situations would be to ignore this sudden bump as it doesn't represent the whole flow of the function. The guard value is introduced to overcome this problem. The guard value is an integer number that tells the algorithm how many times the limit value constraint can be ignored. As an example if the guard value is 1 and the limit value is 0.5 then if the difference in data is 0.6, a new rectangle should be created, but here because the guard value is 1 it means that the algorithm will ignore this constraint once and the rectangles boundaries will be pushed until the limit value is reached again.
The algorithm for piecewise linearizing the data works by moving in different directions in the x and y axes. In each direction the algorithm moves with the respective limit and guard values until it reaches the break point in all the left, right, up, and downward movements. So the first quadrilateral plane is made. This process is repeated for all the directions over and over again until the end of the function domain. With this the whole space of the original surface is divided into the sub domains and in each one of them a quadrilateral plane is made. The planes should always stay above the original function and never under estimate the value. For this reason the process of the algorithm starts at the minimum of the function and the planes will start to be created from this point on.
One advantage of the proposed algorithm is that it can be used on any arbitrary inputted function. So if a random function is created and given to the method, it will piecewise linearize the function. In a complete random function however it is not always guaranteed that the planes will be above the original function.Figure 3.6shows a randomly created graph that has been piecewise linearized with the proposed algorithm.

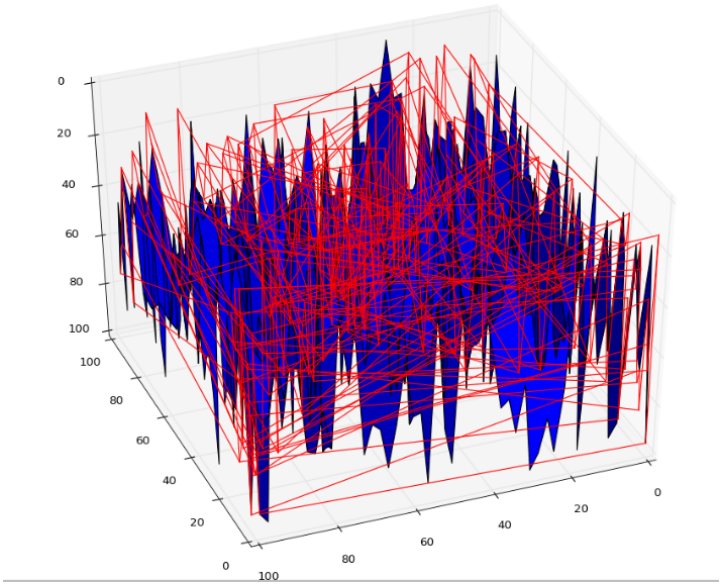


Figure 3.5.: A random function piecewise linearized by proposed method viewed from beside

The quadrilateral planes that piecewise linearized this function, span the whole domain of the random function, this can be observed better from above and is represented in figure 3.6.

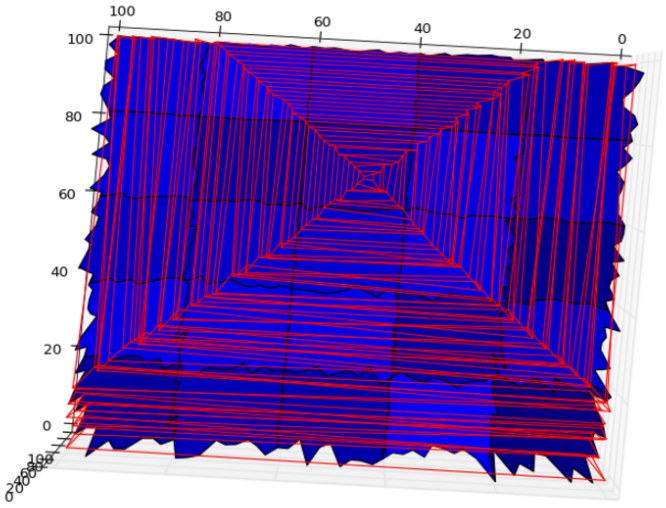


Figure 3.6.: Random function piecewise linearized from top angle

The purpose of this algorithm is to piecewise linearize the real data obtained in previous section. After running the algorithm on the real data, figure 3.7 is obtained. The red rectangles represent the piecewise linear model that will be used. Basically what this means is that all the blue curve will be ignored from this point forward. With this method the clock-to-Q surface

has been piecewise linearized.

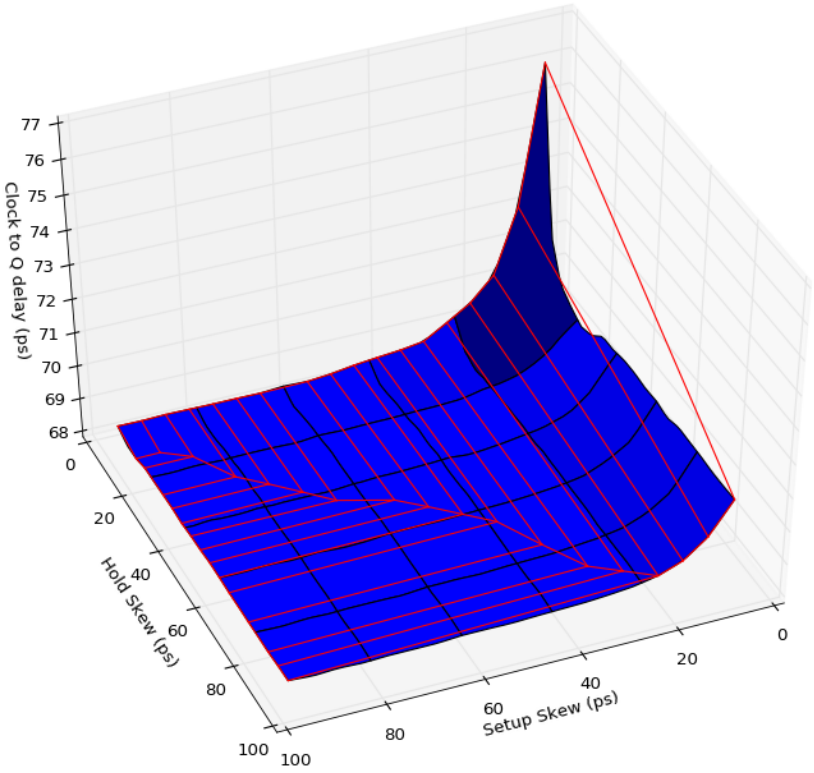


Figure 3.7.: The piecewise linearized clock-to-Q surface

Now it is time to use this piecewise linear data in the framework and solve the optimization problem. But the problem of not being able to use this data still exist. The linear solver program, Gurobi in this project, only gets linear functions as input (Gur 2016). A piecewise linear function also being linear in all the defined sub domains, with the approximate planes, is considered non linear as a whole function mathematically. This brings us to the next step required for changing the acquired data for the flip flops to a usable model for the timing framework and solve the complete problem. The next step would be to convert a piecewise linear function to a linear function mathematically. For doing so the concept will be built and discussed completely for a two dimensional function, after that this concept will be generalized to three dimensions and on the real data.

3.4. Conversion of piecewise linear to linear function in 2 dimensions

In this section the concept of linearizing a piecewise linear function in two dimensions is discussed. But first it is necessary to note that a piecewise linear function can be built for any

arbitrary two dimensional function $f(x)$ and represent that function in the defined sub domains. Each sub domain has a starting and ending point, these points will be called breaking points. A piecewise linear function with n sub domains will hence have $n+1$ breaking points. In figure 3.8 it can be seen that an arbitrary function has been piecewise linearized. The linear segments of each sub domain is represented with the dashed lines.

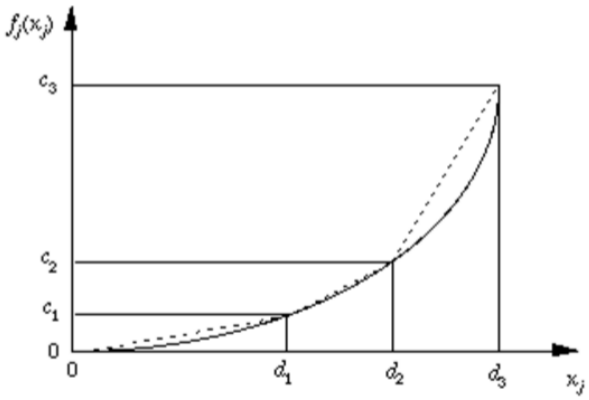


Figure 3.8.: A piecewise linearized arbitrary function in two dimensions

The general concept of converting a piecewise linear function to a linear function in two dimensions will be illustrated with the help of figure 3.8. Consider an arbitrary value for the input x_j, between the two break points d_1 and d_2. This value for the input can be written as $x_j = \lambda d_1 + (1-\lambda)d_2$. In this equation $0 \leq \lambda \leq 1$ and with changing the value of λ from 0 all the way to 1, all the possible values will be created for x_j from d_1 to d_2. Because the function $f(x)$ is also a line segment in the same sub domain, the function itself can also be written as $f(x) = f(\lambda d_1 + (1-\lambda)d_2)$ and this will become $f(x) = \lambda f(d_1) + (1-\lambda)f(d_2)$. In this equation the values of $f(d_1)$ and $f(d_2)$ are the values of the function in the two segment breakpoints, and constant, hence the function is now a linear function of λ. To be able to generalize this idea further, the variable λ will be called λ_1, and for $1-\lambda$ a new variable called λ_2 is introduced. By replacing this two new variables in the formulas above the result is that: $x_j = \lambda_1 d_1 + \lambda_2 d_2$ and also $f(x) = \lambda_1 f(d_1) + \lambda_2 f(d_2)$. It can also be deduced directly from $1-\lambda = \lambda_2$ that the constraint $\lambda_1 + \lambda_2 = 1$ holds true.
This idea, that was shown for two consecutive break points, is the basic idea for linearizing a piecewise linear function. What has been done up to here, made the piecewise linear function linear in the sub domain d_1 to d_2. For creating a single linear function the method used is to generalize this idea to the whole domain of the function, meaning all the break points, and write the input variable and the function based on these values. To generalize the idea the input variable can be written as $x_j = \lambda_1 d_1 + \lambda_2 d_2 + \lambda_3 d_3 + ... + \lambda_{n+1} d_{n+1}$. In this equation n is the number of segments and hence there will be $n+1$ break points. Also the constraint $\lambda_1 + \lambda_2 + \lambda_3 + ... + \lambda_{n+1} = 1$ holds true. Putting this input value in the function results to $f(x_j) = f(\lambda_1 d_1 + \lambda_2 d_2 + \lambda_3 d_3 + ... + \lambda_{n+1} d_{n+1})$ and so $f(x_j) = \lambda_1 f(d_1) + \lambda_2 f(d_2) + \lambda_3 f(d_3) + ... + \lambda_{n+1} f(d_{n+1})$. All the $f(d_k)$ for arbitrary value of k between

1 and $n+1$ are constant values and the value of the function in all the breaking points. For this equations to hold true there is an important constraint that has to hold true and then and only then this equation is correct. This constraint is that at most two consecutive λs can be nonzero. What this constraints is ensuring is that the point should be reside exactly in one of the segments. For example if λ_1 and λ_2 are nonzero and have a positive value it means that the point is between the break points d_1 and d_2 and hence all other values of λs should be zero. Also $\lambda_1 + \lambda_2 = 1$ should hold as mentioned.
The expression "at most two consecutive λs can be nonzero" is not a mathematical expression and is not suitable for a formulation that can be solved by a linear solver. Fortunately this expression can also be turned into a mathematical expression. To do this boolean variables have to be introduced. The boolean variable y_k is used here, where k is between 1 and n, the number of line segments. So the number of y variables is one less than the number of λ variables. Each variable y_k can have either the value 0 or 1, based on the definition of boolean variable, and this value states if the variable is in segment k or not and so has to control the value of λ_k and λ_{k+1}.
Also as the specific point can only be in one segment the sum of all the ys should also result to 1. These constraints have to be added to the function definition and then the augmented function and all the constraints make the function a linear function on the variables λ and y and their linear constraints. So in the following the complete formulas for linearizing a piecewise linear function can be seen:

$$\begin{cases} x = \lambda_1 d_1 + \lambda_2 d_2 + \lambda_3 d_3 + ... + \lambda_{n+1} d_{n+1} \\ f(x) = \lambda_1 f(d_1) + \lambda_2 f(d_2) + \lambda_3 f(d_3) + ... + \lambda_{n+1} f(d_{n+1}) \\ \lambda_1 \leq Y_1 \\ \lambda_2 \leq Y_1 + Y_2 \\ . \\ . \\ . \\ \lambda_{2n-1} \leq Y_{2n-2} + Y_{2n-1} \\ \lambda_{2n} \leq Y_{2n-1} \\ \sum_{i=1}^{2n-1} Y_i = 1 \\ Y_k = 0 or 1 \forall k \\ \sum_{i=1}^{2n} \lambda_i = 1 \\ \lambda_k \geq 0 \forall k \end{cases} \tag{3.1}$$

Now the function is a purely linear function and can be used in a linear solver. For better understanding this method a numerical example is introduced here. Consider the following piecewise linear function:

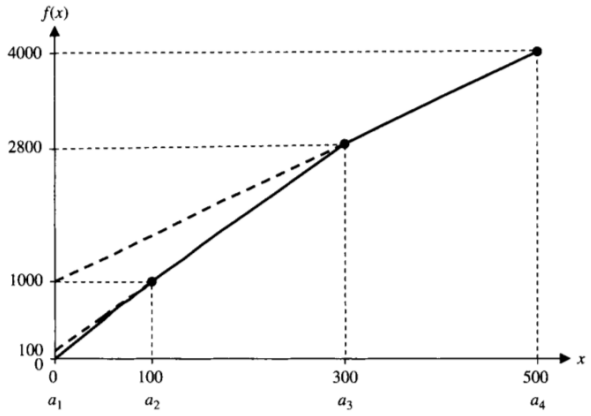


Figure 3.9.: Piecewise linear function example in two dimensions

Based on the depicted function in figure 3.9, it can be seen that this function has 3 sub domains. The first sub domain is the interval from 0 to 100, the second from 100 to 300, and the final sub domain is when x lies between the 300 to 500 interval. The extension of each line, crosses the vertical axis in a point, this point is called the intercept of the line. This intercepts represents the constant term in the function formula. Also the slope of each of the lines will become the coefficient of the line formulas. For the first line the intercept is 0 and the slope is 10, in the second sub domain the lines slope is 9 and the intercept is 100, and for the final line segment the slope is 6 and the intercept is 1000. So based on these values the equation for the drawn function can be written as follows:

$$f(x) = \begin{cases} 10x & \text{if } 0 \leq x \leq 100 \\ 9x + 100 & \text{if } 100 \leq x \leq 300 \\ 6x + 1000 & \text{if } 300 \leq x \leq 500 \end{cases}$$

Also there are four breaking points in this function, which are denoted with a_1 to a_4 in the picture, which have values of $a_1 = 0$, $a_2 = 100$, $a_3 = 300$, and $a_4 = 500$.
To make this function completely linear, the formula and constraints calculated in formula (3.1) will be used. By putting the all the required values in the equations the linearized function will be written as follows:

$$
\begin{gathered}
x = 0\lambda_1 + 100\lambda_2 + 300\lambda_3 + 500\lambda_4 \\
f(x) = \lambda_1 f(0) + \lambda_2 f(100) + \lambda_3 f(300) + \lambda_4 f(500) \\
\lambda_1 \leq Y_1 \\
\lambda_2 \leq Y_1 + Y_2 \\
\lambda_3 \leq Y_2 + Y_3 \\
\lambda_4 \leq Y_3 \\
\sum_{i=1}^{3} Y_i = 1 \\
Y_k = 0 or 1 \forall k \\
\sum_{i=1}^{4} \lambda_i = 1 \\
\lambda_k \geq 0 \forall k
\end{gathered}
$$

To show that these two functions are actually the same also a numerical value will be tested. Consider the point $x = 150$. This point is in the second segment of the piecewise linear function and hence the second formula of the function is used to calculate the value. By putting the value in the equation we obtain $f(150) = 9 \times 150 + 100 = 1450$. On the other hand to find the values for Y and λs, the knowledge is used that the point is in the second segment and hence $Y_2 = 1$. From the constraints it is concluded that $Y_1 = 0$ and $Y_3 = 0$. For the λ the calculated values are $\lambda_1 = \lambda 4 = 0$, $\lambda_2 = 0.75$, and $\lambda_3 = 0.25$. By placing these values in the purely linear equation obtained for the function the value of $f(x)$ will be $f(x) = 1000 \times 0.75 + 300 \times 0.25 = 1450$ which is exact same value.

3.5. Conversion of piecewise linear to linear function in 3 dimensions

The concept explained in previous section has to be extended for three dimensions. In the context of the problem, a three dimensional function is available, which is piecewise linearized with the method of using limit and guard values. In the following section the explanation of how this piecewise linear function is turned into a three dimensional purely linear function is discussed exhaustively. After the function has been piecewise linearized by the means of quadrilateral planes, the next step is to convert it to a linear function. This step is necessary because for using a model as an input to a linear solver, the function must be purely linear and although all the segments are linear it is still considered non-linear.

For reaching this goal, first consider one of the planes that constitutes the whole piecewise linear function. This plane is built between the x-axis points a_k and a_{k+1} and the y-axis points

b_k and b_{k+1} for an arbitrary value k.

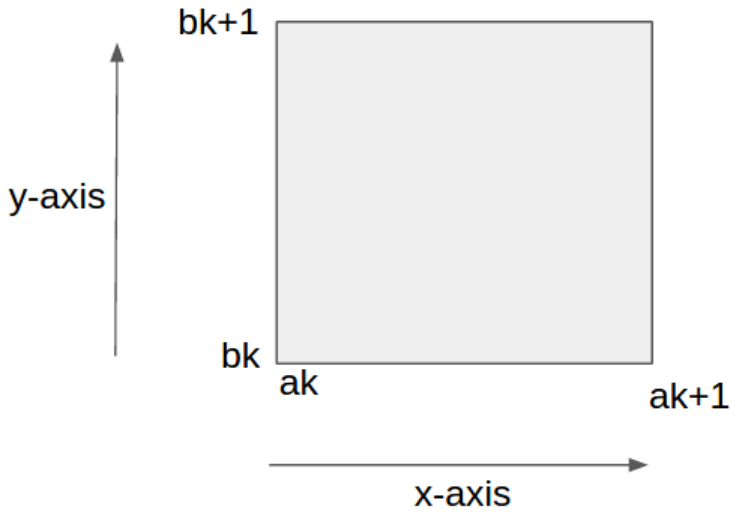


Figure 3.10.: Arbitrary rectangle

Between the consecutive break points a_k and a_{k+1} on the x-axis and b_k and b_{k+1} on the y-axis, x and y can be described as follows:

$$x = \lambda_k a_k + (1 - \lambda_k) a_{k+1} \tag{3.2}$$

$$y = \gamma_k b_k + (1 - \gamma_k) b_{k+1} \tag{3.3}$$

Where $0 \leq \lambda_k \leq 1$ and $0 \leq \gamma_k \leq 1$. For this specific rectangle the following equation can be written and proven.

$$f(x,y) = \lambda_k f(a_k, b_k) + (1-\lambda_k) f(a_{k+1}, b_k) + \gamma_k f(a_k, b_k) + (1-\gamma_k) f(a_k, b_{k+1}) - f(a_k, b_k) \tag{3.4}$$

Any three dimensional function $f(x, y)$ can be written in the general form of:

$$f(x,y) = k_1 x + k_2 y + c_0 \tag{3.5}$$

So equation (3.4) can be proven by putting the equations (3.2) and (3.3) in equation (3.5) as

follows:

$$\begin{aligned}
f(x,y) &= \lambda_k(k_1a_k + k_2b_k + c_0) + (1-\lambda_k)(k_1a_{k+1} + k_2b_k + c_0) \\
&\quad + \gamma_k(k_1a_k + k_2b_k + c_0) + (1-\gamma_k)(k_1a_k + k_2b_{k+1} + c_0) \\
&= \lambda_k k_1a_k + \lambda_k k_2b_k + \lambda_k c_0 + k_1a_{k+1} + k_2b_k + c_0 - \lambda_k k_1a_{k+1} - \lambda_k k_2b_k - \lambda_k c_0 \\
&\quad + \gamma_k k_1a_k + \gamma_k k_2b_k + \gamma_k c_0 + k_1a_k + k_2b_{k+1} + c_0 - \gamma_k k_1a_k - \gamma_k k_2b_{k+1} - \gamma_k c_0 \\
&= \lambda_k k_1a_k + k_1a_{k+1} - \lambda_k k_1a_{k+1} + k_2b_k + \gamma_k k_2b_k + k_2b_{k+1} - \gamma_k k_2b_{k+1} + k_1a_k + c_0
\end{aligned}$$

From equation (3.5) it can also be deduced that $f(a_k, b_k) = k_1a_k + k_2b_k + c_0$ and hence equation (3.4) is obtained.

In equation (3.4) the last term namely $f(a_k, b_k)$, is dependent on the specific rectangle considered. This fact is a dependency that makes it hard to generalize the formula for more than one rectangle. As the formula has to be generalized over all the rectangles that constitute the complete function, hence generalizing this formula is of utter importance. So a method has to be deployed to either omit the constant value completely or generate a term that is constant and independent of the chosen rectangle, meaning for all the arbitrary break points a_k and b_k. The idea is to divide the function to 2 separate x and y direction functions, such the the part of the function that builds the x-axis value has a constant value for y input and the part that constitutes the y-axis value has a constant value for x input. The constant value considered here for the x and y value is the smallest break point in the respective directions denoted as $\min a$ and $\min b$. The modified version of (3.4) can be written as:

$$f(x,y) = \lambda_k f(a_k, \min b) + (1-\lambda_k) f(a_{k+1}, \min b) + \gamma_k f(\min a, b_k) + (1-\gamma_k) f(\min a, b_{k+1}) - f(\min a, \min b) \tag{3.6}$$

To prove (3.6) again by using (3.2) and (3.3) and inserting them in equation (3.5) the result will be:

$$\begin{aligned}
f(x,y) &= \lambda_k(k_1a_k + k_2 \min b + c_0) + (1-\lambda_k)(k_1a_{k+1} + k_2 \min b + c_0) \\
&\quad + \gamma_k(k_1 \min a + k_2b_k + c_0) + (1-\gamma_k)(k_1 \min a + k_2b_{k+1} + c_0) \\
&= \lambda_k k_1a_k + \lambda_k k_2 \min b + \lambda_k c_0 + k_1a_{k+1} + k_2 \min b + c_0 - \lambda_k k_1a_{k+1} - \lambda_k k_2 \min b - \lambda_k c_0 \\
&\quad + \gamma_k k_1 \min a + \gamma_k k_2b_k + \gamma_k c_0 + k_1 \min a + k_2b_{k+1} + c_0 - \gamma_k k_1 \min a - \gamma_k k_2b_{k+1} - \gamma_k c_0 \\
&= \lambda_k k_1a_k + k_1a_{k+1} - \lambda_k k_1a_{k+1} + k_2 \min b + \gamma_k k_2b_k + k_2b_{k+1} - \gamma_k k_2b_{k+1} + k_1 \min a + c_0
\end{aligned}$$

And from equation (3.5) it results that $f(\min a, \min b) = k_1 \min a + k_2 \min b + c_0$. Hence the above proof will result to (3.6).
With this method the goal of making a constant term which is independent of the rectangle, and hence independent of the break points, is fulfilled. To generalize this method on the complete function, consider that the function is linearized by n rectangles. Any rectangle creates four break points, two on the x-axis and also two on the y-axis, therefore n rectangles will make $2n$ break points in each axis. The breaking points are named $a_1, a_2, ..., a_{2n}$ for the x-axis and $b_1, b_2, ..., b_{2n}$ for the y-axis. Any x and y value for the function can be represented as a linear function of the break points throughout their corresponding axes. The following equations will

satisfy this fact by generalizing (3.2) and (3.3).

$$x = \lambda_1 a_1 + \lambda_2 a_2 + ... + \lambda_{2n} a_{2n} \tag{3.7}$$

$$y = \gamma_1 a_1 + \gamma_2 a_2 + ... + \gamma_{2n} a_{2n} \tag{3.8}$$

The function $f(x, y)$ can now be written as a generalized form as follows:

$$\begin{aligned} f(x,y) = {} & \lambda_1 f(a_1, \min b) + \lambda_2 f(a_2, \min b) + ... + \lambda_{2n} f(a_{2n}, \min b) && (3.9)\\ & + \gamma_1 f(\min a, b_1) + \gamma_2 f(\min a, b_2) + ... + \gamma_{2n} f(\min a, b_{2n}) && (3.10)\\ & - f(\min a, \min b) && (3.11) \end{aligned}$$

The condition for this equation to hold true is that only two of the consecutive λs and γs are nonzero and also the sum of all λs and the sum of all γs are equal to 1. This insures that exactly one rectangle is chosen, so all the other λ and γs are zero and are omitted. Based on equation (3.6) this equation is true for any arbitrary rectangle and so the function has been converted to a linear function. But the sentence "only two of the consecutive λs and γs are nonzero" is not a mathematical constraint. For turning this sentence into a mathematical expression two additional boolean arrays are introduced and used. These arrays are Y and Z and they are used to ensure that only two adjacent λs and two adjacent γs are nonzero at the same time. For n rectangles the size of Y and Z arrays is $2n - 1$. Each element of Y is responsible for controlling two consecutive λ values and each element of Z is also responsible for two adjacent γ values. For example if $Y_1 = 0$ then λ_1 and λ_2 would become zero whereas if $Y_1 \neq 0$ then all the λ_ks for $k \neq 1$ and 2 will be zero while $0 \leq \lambda_1 \leq 1$ and $0 \leq \lambda_2 \leq 1$ and $\lambda_1 + \lambda_2 = 1$. To fulfill this purpose we add the following constraints to the function:

$$\begin{gathered} \lambda_1 \leq Y_1 \\ \lambda_2 \leq Y_1 + Y_2 \\ . \\ . \\ . \\ \lambda_{2n-1} \leq Y_{2n-2} + Y_{2n-1} \\ \lambda_{2n} \leq Y_{2n-1} \end{gathered}$$

$$\sum_{i=1}^{2n-1} Y_i = 1$$
$$Y_k = 0 or 1 \forall k$$
$$\sum_{i=1}^{2n} \lambda_i = 1$$
$$\lambda_k \geq 0 \forall k$$
$$\gamma_1 \leq Z_1$$
$$\gamma_2 \leq Z_1 + Z_2$$
$$.$$
$$.$$
$$.$$
$$\gamma_{2n-1} \leq Z_{2n-2} + Z_{2n-1}$$
$$\gamma_{2n} \leq Z_{2n-1}$$

$$\sum_{i=1}^{2n-1} Z_i = 1$$
$$Z_k = 0 or 1 \forall k$$
$$\sum_{i=1}^{2n} \gamma_i = 1$$
$$\gamma_k \geq 0 \forall k$$

To use this technique inside a linear solver, all the occurrences of x and y and also the function $f(x, y)$ in the original function, which was an arbitrary function, should be replaced with the λ, γ, Y, and Z variables. After that, all the above constraints are added to the problem, so the augmented problem is the same as the original one. Note that this model now contains only the new variables and is a linear function of these new variables. After the solution to this equal problem has been found, the solution of the original problem will be found by plotting all the values in the equations (3.7), (3.8), and (3.9)

3.6. Complete timing framework

After the clock-to-Q surface has been completely linearized, the timing framework for calculating the maximum frequency of the circuit, the minimum clock period, can be completed. This

framework will be constructed and discussed in more depth in this section. When the equations for the flip flop network was written, the clock-to-Q delay as a nonlinear function could not be inputted in the linear solver. This problem has been solved in the previous sections with the introduced techniques. The framework, hence can become complete here.
For each flip flop some equations will be constructed. Based on figure 3.11, the equations will be for the arrival time of the data, which here means the latest arrival time of the input, so the maximum delay in the combinatorial path will determine the arrival time. Also the clock-to-Q delay of the previous stage has impact on the arrival time, so the arrival time will become the sum of the maximum combinatorial path and the previous stage clock-to-Q delay. The setup skew of a flip flop is the time that the data input gets ready before the clock signal comes to the flip flop, so the setup skew is the clock period minus the arrival time. The hold skew is dependent to the minimum delay of the combinatorial path. The sooner the data reaches the flip flop, the shorter the hold skew would be. Finally the clock-to-Q delay function that has been linearized.

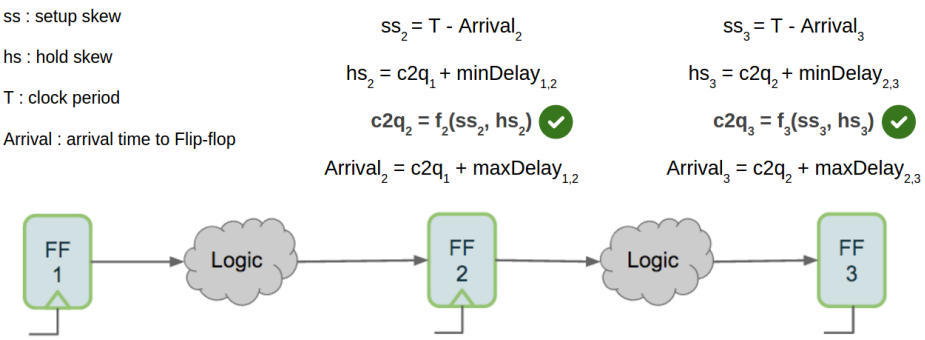


Figure 3.11.: The complete framework for a network of flip flops

In the programming environment all of these variables will be defined for each flip flop in a programming loop iterating through all the flip flops. The important point is that all the fan-ins to the flip flop should also be considered and the maximum or minimum value, based on the parameter, should be chosen for the flip flop. At the end the complete equation for all the flip flops remains, which is based on the linearization parameters discussed in previous section. The equations and the built constraints are then inputted to the linear programming solver.

3.7. Algorithm analysis

In this section, the steps required to solve the problem and their respective algorithms analysis will be discussed in more depth. Also the used algorithm has some strengths and weak points that are highlighted in the present section.
The process of finding the highest clock frequency for the circuit starts with the piecewise linearization of the clock-to-Q delay surface based on setup skew and hold skew. The algo-

rithm used here requires the data of the graph, which is stored in a .csv file. This data is read and stored by the program. If the data number of graph points read into the program are denoted with n the algorithm has to iterate over all of them in order to read all of the data, so the computational complexity will be $O(n)$. For using the algorithm that breaks the graph into quadrilateral planes, the data has to be sorted based on the axes. The sorting method used here is a merge sort with complexity of $O(n \log n)$. After the data is sorted for the axes separately, the movement on this data begins. In order to complete the piecewise linearization all of the data has to be visited once, so here again the the algorithm has a linear time complexity. When having multiple complexities in an algorithm, the complexity that has the fastest growing rate will be dominant as the complexity for the entire algorithm. In the case of the used algorithm, the growth of $O(n \log n)$ is bigger and so it will be the complexity for the algorithm that piecewise linearizes the data. Notice that the checks with the limit and guard values to decide where to break the planes and make new quadrilateral planes are done in constant time, thus do not affect the complexity of the algorithm. The data for the clock-to-Q delay depends on the time steps that are chosen for the spice circuit simulation. The smaller the time intervals are chosen the more data, and also more accuracy in the graph, will be obtained so the runtime of the algorithm increases.
The framework also reads in the circuit information. The information consists of all the nodes connections and their timing information, each path in one line. If the number of lines in the file is considered m then reading the file requires $O(m)$ complexity which is a linear relation to the input. The required information of the number of flip flops and each flip flop name is also extracted from this file in the programming framework, which also goes through the complete data and therefore has the same complexity as the read. The next part in complexity calculation is to calculate the number of variables that has to be defined. This part depends on the number of flip flops and also the number of quadrilateral planes created by the first part of the algorithm. The whole concept is that each flip flop is defined by all the planes created, so if k flip flops are in the circuit and the algorithm has created l planes, then a loop with $k \times l$ iteration is required to define all the variables and instantiate them. Accordingly this calculation requires also $O(kl)$ complexity. The parameter l is strongly dependent to the choice of guard and limit values, so it will vary case by case. The more accuracy required the larger l becomes and so the time complexity, the delay of calculation, increases.
The algorithms complexity is finally defined as the maximum complexity of all the parts, so it can be written as $\max(O(n \log n), O(m), O(kl))$. So the larger part of the algorithm makes the final computation complexity, if the input data is extremely large, but the desired accuracy and also the number of flip flops are low, the complexity will be dependent to the input. Other cases are that the number of quadrilateral planes are small, the number of flip flops are not that high but in the circuit the connection of flip flops are really high, hence m will be a big number, the algorithm is decided by the lines of the circuit input file. In the final case, which seems to be the most probable scenario, the complete algorithm depends on the number of flip flops and the number of defined planes.
The surface of clock-to-Q delay in reality for the current technologies is convex, but the piecewise linearization of the clock-to-Q delay is a general algorithm, which does not make any assumptions about the convexity. This is an advantage for the algorithm, and one of the strengths of the proposed method. The advantage here is that, the present work can be used with any random three dimensional function. If in the future the underlying technology for flip flops changes and the surface becomes non convex as a result, then the flexible flop flop method

proposed will still be useful in the timing analysis of that technology. Another advantage of the method is that all the quadrilateral planes, created in the piecewise linearization problem, are above the actual data in the surface. This creates some buffer for error, by inducing some pessimism. This buffer can ensure that the calculated clock period at the end of the method, can work in reality and also is robust so that it can tolerate some degree of error.
On the other hand there are also some limitations and unsolved challenges that should be clarified. The linear solver used to complete the calculation, Gurobi linear solver, can handle a limited number of variables. The flexible flip flop model, though effective and accurate, uses a lot of variables. Each plane created in the process of linearization creates 4 variables, also for each flip flop some parameters like setup skew, hold skew, arrival time, and clock-to-Q delay, has to be stored. Hence if there are n planes and m flip flops a total number of approximately $4mn + 4m$ variables will be defined. So the linear solver will not work for big circuits that have a lot of flip flops. Also if the accuracy requirement is extremely high and the number of planes becomes larger, there will be a limit on this number other wise the linear solver will fail. Another important issue that is not solved in the present work is the hold time constraint. The minimum combinatorial delay of a path plus the clock-to-Q-delay of the previous flip flop should be larger than the hold time of the next stage flip flop. In the context of exploiting the setup/hold skew versus the clock-to-Q delay, this constraint should also be considered, but because the effect of this constraint is really small in real circuits, it is neglected in the present work.

4. Experimental results

For showing the effectiveness of the proposed method, the results obtained from experiments have to be compared to those of conventional STA method. This chapter discusses the conditions and test results of the proposed methodology in the present work.
The framework of the method is programed with the python programming language. In order to use the linear solver, which is Gurobi in the presented work, it provides a python interface that is embedded in the code. Also to make comparisons, the STA method has been implemented. The STA implemented for the comparison uses the 10% degradation criterion for characterization of the setup and hold times. The test is conducted on 8 circuits with different flip flop counts and complexities. The timing information is obtained from a 45 nm technology library for the flip flops as well as all the combinatorial elements of the circuit. The combinatorial path delays between the flip flop network is provided as a text input file to the program. This file includes all the starting and end nodes and the minimum and maximum delay between the corresponding flip flops. Also the clock-to-Q delay surface based on setup skew and hold skew, which is obtained from spice simulation, is read by the program from a .csv file. The results are illustrated in table 4.1.

Table 4.1.: Experimental results

Circuit name	Number of flip flops	T_{STA}	Limit value	Guard value	T_{FPWL}	min (r)
mem_ctrl_syn	1065	102.27	2	0	91.544	0.888
				1	91.737	
				2	91.737	
			1	0	90.887	
				1	90.884	
				2	91.732	
			0.5	0	90.884	
				1	90.886	
				2	91.204	
ac97_ctrl_syn	2199	96.31	2	0	86.255	0.889
				1	86.446	
				2	86.446	
			1	0	85.595	
				1	85.595	
				2	84.44	
			0.5	0	85.591	
				1	85.594	
				2	85.915	

PCIbridge32_syn	3321	100.18	2	0	90.47	0.896
				1	90.668	
				2	90.715	
			1	0	90.12	
				1	90.34	
				2	90.68	
			0.5	0	89.807	
				1	89.915	
				2	90.57	
s9234	125	106.72	2	0	96.45	0.897
				1	96.64	
				2	96.65	
			1	0	95.79	
				1	95.794	
				2	96.63	
			0.5	0	95.754	
				1	95.754	
				2	96.07	
s13207	426	107.19	2	0	96.07	0.89
				1	96.02	
				2	96.026	
			1	0	95.18	
				1	95.34	
				2	96.026	
			0.5	0	95.06	
				1	95.17	
				2	95.49	
s15850	442	108.32	2	0	96.31	0.882
				1	96.501	
				2	96.501	
			1	0	95.656	
				1	95.814	
				2	96.511	
			0.5	0	95.64	
				1	95.631	
				2	95.951	

s38584	1233	107.32	2	0	95.108	0.879
				1	95.293	
				2	95.295	
			1	0	94.448	
				1	94.448	
				2	95.293	
			0.5	0	94.427	
				1	94.427	
				2	94.747	
usbfunct_syn	1746	100.34	2	0	90.348	0.892
				1	90.537	
				2	90.537	
			1	0	89.64	
				1	89.692	
				2	90.537	
			0.5	0	89.574	
				1	89.65	
				2	89.979	

The first three columns of table 4.1 contain information about the used circuits names, the number of flip flops that build the sequential part of the circuit, and the calculated clock period based on the conventional STA methods, denoted as T_{STA}. The next two columns represent the parameters in the algorithm. And finally the respected clock period, written as T_{FPWL}, based on these parameters, is introduced in the next column. The last column of the table 4.1 presents the minimum ratio of clock period obtained by the proposed flexible flip flop model to the conventional clock period. Mathematically this can be written as $\min(r) = \min(T_{FPWL})/T_{STA}$. The value of this column represents the maximum improvement of the algorithm on the specific circuit.

Based on the information in table 4.1, the maximum improvement obtained by the method in "s38584" circuit is 12.1% with respect to the STA method. The average improvement percentage for all the test circuits is also approximately 11%.

It can also be observed that in some cases the change in guard value does not have a big impact on the obtained clock period. The reason behind this is that the data for the clock-to-Q surface is completely convex and there are not outlier data or bumps in the graph. So the guard value just makes the created planes bigger by a small number of sample points. This leads to little or no change in the outcome of the algorithm. Another observation is that the improvement in clock period converges to a minimum. For some circuits this happens faster than the others, with a higher limit value parameter, than the other circuits. For instance in "mem_ctrl_syn" the clock period with limit values of 1 and 0.5 are almost the same, but for "PCIbridge32_syn" the impact of this change is more observable.

Another important aspect of the algorithm is to consider the trade off between accuracy and speed. The limit value parameter in the algorithm is responsible for the accuracy of the method. Figure 4.1 depicts the relation between the limit value and the time it takes, in second, to obtain the result for "s13207" circuit. It can be observed that the general trend in

the graph is, the higher the accuracy the more time is required.

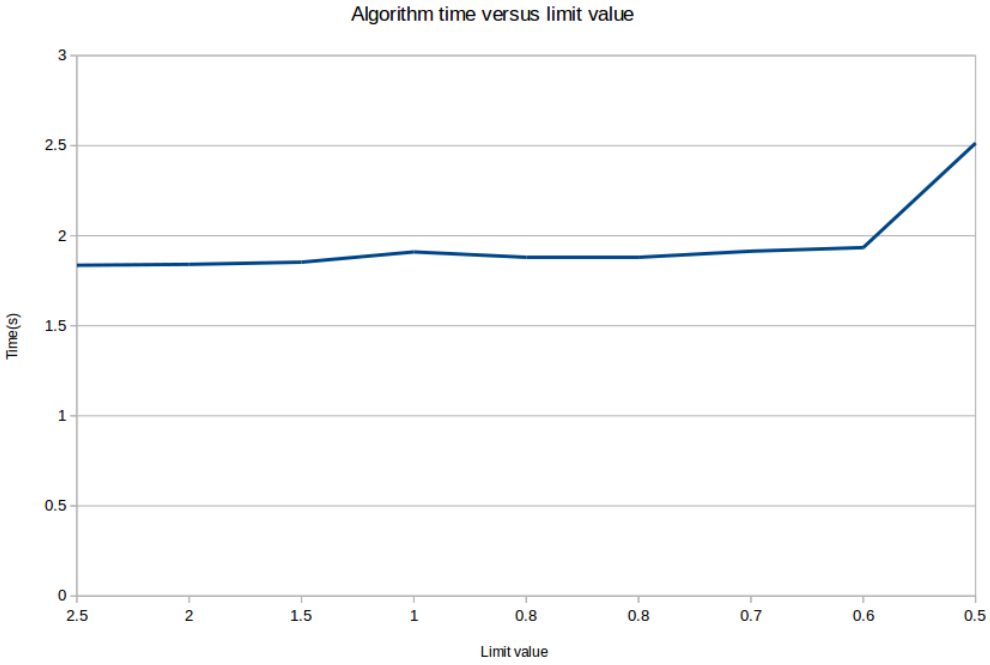


Figure 4.1.: Modeling time versus algorithms limit value

5. Summary and conclusion

In the present work, the trend toward smaller and faster circuits based on Moore's law and the importance of determining more accurate and exact clock frequency was introduced. Based on this importance the wide used and conventional methods for finding the clock period were investigated and the sources of their incompetency were discussed. One of the problems of these methods was the assumption of independent timing characterizations, between setup/hold skew and clock-to-Q delay to be precise. This problem leads to over pessimism or over optimism in calculation of clock period to be used, that can lead to the circuit degradation or complete failure and malfunction and hence have to be addressed properly. So the endeavors and works done for solving this problem were introduced, which also had their strengths and weaknesses. Based on the previous work and with the motivation to improve their weak points, a new flexible flip flop model based on piecewise linearization for timing analysis was introduced. In this model the complete circuit is considered as an interdependent and interconnected system. The timing framework for solving the clock period optimization problem was built. The required clock-to-Q delay surface information was also gathered with spice simulation. The initial obstacle to solve the problem by using a linear solver, which is the nonlinearity of the clock-to-Q delay surface, was mentioned. This problem led to the idea to find a way to linearize the required data for solving the optimization problem.
First the clock-to-Q delay surface was approximated with a piecewise linear model. For reaching this goal, the methodology to model the original function with quadrilateral planes was introduced, the challenges and important accuracy-speed compromise were discussed, and the surface was completely modeled. This model has the advantage that it not only works for this specific clock-to-Q surface but for any random three dimensional surface. This has the advantage that if in the next generations of flip flop or possible other technologies, the interdependency of the setup/hold skew and clock-to-Q delay can be modeled as well. Also the week point of this modeling was introduced which is the fact that the quadrilateral planes are always above the actual surface. This is unreliable for the hold skew constraint characterization.
After acquiring this model for the surface, to use this data in the proposed timing framework, still a piecewise linear function had to be converted to a purely linear model. The presented work discusses this conversion in depth. For better understanding this concepts is discussed in the most basic form in two dimensions and builds upon this basis. The generalization and use of this method in three dimensions is explained, the mathematical formulation to prove the concept are also completely given, while the challenges that appear in this process are solved.With the linearization the proposed timing framework became complete and the method is able to find the optimized clock period for any given circuit.
After all the required steps toward the optimization goal were taken, the framework was programmed in python. The experimental results obtained from the framework showed improvement for the circuits under test. The amount of improvement was different for the circuits based on their internal structure and the complexity of their paths. The maximum improve-

ment obtained was around 12%.
Although this method has advantages over the previous works represented in the present thesis, there need to be some improvements and completions in order to use it in practice. The next section discusses some of the ideas that might be helpful to improve the idea and hopefully use it in near future as an industrial tool.

5.1. Future work

As in all scientific and engineering work, this work is done based on the foundation of previous works and the continuous improvement of ideas and implementations. In this part some ideas for improving the presented work and also some points to continue this work is suggested.
In the present work, a single clock-to-Q delay relation to setup skew and hold skew is used. Meaning that the timing information used in a flip flop is concluded from the three way relation under a specific supply voltage, temperature and rise/fall time for the signals. To improve the data collection a suggested work to expand the thesis would be to find a method to combine all the situations and make an average graph that considers more than one condition for the supply voltage, temperature and also the rise and fall times. This will make the timing analysis result of the method more reliable in real conditions and can also give some insights in the extreme corners that the chip might operate in.
One of the limitations of the proposed algorithm is the lack of consideration for hold time constraint. To expand the present work and be able to use it industrially this condition can be added to the work. This work requires to introduce a algorithm for piecewise linearization of the clock-to-Q delay surface such that all the created planes are under the actual graph to ensure that the circuit operates correctly after manufacturing. The fact that these planes are under the surface, makes the current algorithm inefficient and also the breaking points might differ drastically with the planes from the setup time constraint.
Another idea to improve the work, is to find better mathematical ways to reduce the number of variables required to model the data. The linear solver has an upper limit for the number of defined variables and will not work after that limit is passed, hence it's important for real size circuits to be able to be modeled, that the number of variables decreases. Although the proposed algorithm can work with any random function, one way to reduce the number of parameters is to use the fact that the current technology has convex surface. With the consideration of this mathematical fact there might be a way to reduce the number of parameters. This is also a trade off that can be investigated. Finally another possible direction of continuing the ongoing work is to combine other areas to solve this problem. One of the ideas here is to use machine learning techniques to model acquired data in various situations.

Bibliography

Andrew B. Kahng, Hyein Lee (2014): Timing margin recovery with flexible flip-flop timing model, Proc. Int'l Symp. on Quality Electronic Design S. 496–503.

Cirit, M. A. (1991): Characterizing a vlsi standard cell library, IEEE Custom Integrated Circuits Conf.

E. Salman, A. Dastan, F. Taraporevala K. Kucukcakar & Friedman, E. G. (2007): Exploiting setup-hold-time interdependence in static timing analysis, IEEE Trans. on CAD S. 1114–1125.

Gur (2016): GUROBI OPTIMIZER QUICK START GUIDE, 6.5 Aufl.

Lang, A., Bergler S. (2004): Method and apparatus for circuit verification of meeting setup and hold time requirements (verfahren und vorrichtung zum ueberpruefen einer schaltung auf einhaltung von setup- und holdzeiten).

N. Chen, B. Li, U. Schlichtmann (2011): Iterative timing analysis based on nonlinear and interdependent flipflop modelling, IET Circuits, Devices & Systems .

Neil H. E. Weste, D. M. Harris (2011): CMOS VLSI Design A Circuits and Systems Perspective, Pearson Education, Inc., publishing as Addison-Wesley.

The open source liberty library modeling format specification (2016). `http://www.opensourceliberty.org/`

Patel, D. (1990): Charms: Characterization and modeling systems for accurate delay prediction of asic designs, IEEE Custom Integrated Circuits Conf.

Princeton, M. Shoji (1992): Theory of CMOS Digital Circuits and Circuit Failures, NJ: Princeton Univ. Press.

Rao, G. & Howick, K. (2003): Apparatus for optimized constraint charac- terization with degraded options and associated methods.

Srivastava, S. & Roychowdhury, J. (2007): Interdependent latch setup/hold time characterization via euler-newton curve tracing on state- transition equations, Proc. ACM/IEEE DAC.

Srivastava, S. & Roychowdhury, J. (2008): Independent and interdependent latch setup/hold time characterization via newton-raphson solution and euler curve tracking of state-transition equations, EEE Trans. on CAD S. 817–830.

Stojanovic, V. & Oklobdzija, V. G. (1999): Comparative analysis of master- slave latches and flip-flops for high-performance and low-power systems, IEEE J. Solid-State Circuits, vol. 34, no. 4 .

Y. M. Yang, K. H. Tam, I. Hui-Ru Jiang (2015): Criticality-dependency-aware timing characterization and analysis, DAC .