Mathematical Modeling of Clock Binary Tree

By

Viren R Gajjar

09MEC018

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2011

Mathematical Modeling of Clock Binary Tree

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

In

Electronics & Communication Engineering

(VLSI Design)

By

Viren R Gajjar

(09MEC018)

Under the Guidance of

Mr. Pawan Dwivedi

Intel Technology India Pvt. Ltd.

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2011

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Viren R Gajjar

Certificate

This is to certify that the Major Project entitled **"Mathematical Modeling of Clock Binary Tree"** submitted by **Viren R Gajjar**(09MEC018), towards the partial fulfillment of the requirements for the degree of **Master of Technology** in the field of **VLSI Design** awared by **Institute of Technology, Nirma University, Ahmedabad** is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

External Guide

(Mr. Pawan Dwivedi)
Intel Technology India Pvt. Ltd.

HOD

(Prof. A. S. Ranade)
Professor, EC

Manager

(Lakshmi Venkatachalam)
Intel Technology India Pvt. Ltd.

Director

Dr. K. Kotecha
Director, IT, NU

Internal Guide

(Dr. N. M. Devashrayee)
PG Coordinator (VLSI Design)
Nirma University

Acknowledgements

I would like to express my greatest thanks and appreciation towards all those with whom I have worked at **Institute of Technology, Nirma University**, for their cooperation and continuous evaluation.

I express my gratitude and great respect to all the people with whom I have worked and interacted during the Internship at **Intel Technologies Pvt. Ltd., Bangalore, India** and who have directly or indirectly helped me to accomplish the project goal. I would like to thank each one of them for their valuable guidance. I would like to especially thank my project guide, **Mr. Pawan Dwivedi, Intel Technologies Corporation**. His continuous evaluation, suggestions and feedback inspired me to put the best effort in my work. Under his expert guidance, my work was in constant phase of improvement. I thank him for his availability and support, in spite of his busy work schedule. I would also like to thank **Mr. Lakshmi Venkatachalam, Intel Technologies Corporation** to gave me opportunity to work in clock domain and gave chance to work in Clock Binary Tree design and development.

I would like to express my sincere thanks to **Dr. N. M. Devashrayee, PG Coordinator VLSI Design department**, for his valuable guidance and informative suggestions for the betterment of the project throughout this major project phase. I would also like to thank **Prof. N. P. Gajjar, Nirma University**, for his sincere support and guidance. Finally would like to thank the entire **Electronics & Communication Department** for providing me the platform to work efficiently and fruitfully, at a prestigious corporation such as Intel Technology, India.

- Viren R Gajjar (09MEC018)
M.Tech (VLSI Design)
Nirma University, Ahmedabad, Gujarat

Abstract

All the Digital circuits run on clock signal that also makes those circuits synchronous by nature. In a synchronous digital system, clock signal is used to define a time reference for the movement of data within that system. The **clock distribution network** (or **clock tree**, when this network forms a tree) distributes the clock signals from a common point to all the elements that need it. Since this function is vital to the operation of a synchronous system, much attention has been given to the characteristics of these clock signals and the electrical networks used in their distribution. Clock signals are often regarded as simple control signals. however, these signals have some very special characteristics and attributes. As clock signal is the only versatile signal thus routing this signal to all the block without **skew** is tough task. This signal should also have higher driving strength so that it can drive number of modules. Therefore from the origin of clock, there should be a mechanism to increase driving strength with zero skew to reach the RCB (Regional Clock Buffer). Currently for Global clock distribution a structure called **clock binary tree** with input connected to input and output connected to output of other gates, are used to have minimum skew to complete clock binary tree. This thesis is a compilation of algorithm developed to design clock binary tree and experiments done on different tree structures, which are used or can be used in global clock distribution for clock binary tree design.

Contents

List of Figures

Chapter 1

Introduction

Today's high performance systems need low skew clock generation and distribution. Clock skew is defined as the difference in time between simultaneous clock transitions within a system. The skew has become the major part of constraints that form the upper boundary for the system clock frequency.

Optimized clock distribution are central to high performance of synchronous systems. If clock distribution networks are not properly constructed, they may degrade system-level performance. This compilation of thesis defines the skew in a system, describing its effect on performance. Recommendation and supporting analysis are given for designing near optimal clock networks. This thesis also illustrates that clock network design can be a surprisingly complex task involving many tradeoffs.

Clock networks must be designed to minimize skew or the differences in delay throughout a clocking network. It is ideal that every component, such as sequential elements, i.e. flip-flops and latches that need clocking, should receive the edge of the clock at the same time within each clock period. Fully synchronous designs require this methodology. To ensure that the network operates as closely to the ideal as possible, the skew must be minimized along the entire clocking network. This ensures that all sequential elements see a common clock edge. For any design, with more than 100

flops and latches, it is strongly suggested that some form of clock tree structure for clocking required. Buffer delays and wiring delays are the two most significant factors contributing to skew. The clock topology can significantly contribute to skew. As will be discussed, a good clock network must balance factors such as skew, risetime, power and clock tree delays.

The goal of this experiment is to find out all the possible combination of clock tree and simulate them. These simulation data can give proof that which design is better in what sense. The prior concern is to get lesser skew in each stage of particular clock binary tree.

1.1 List of Chapters

1.1.1 ASIC Design Flow

Study and understanding of whole IC manufacturing process will help to understand the flow used in clock binary tree designing. This chapter explains the necessary steps of IC manufacturing before actual synthesis of clock tree. Group of steps are followed in predefined patter to manufacture every IC. This section has detailed steps of Back-end design. Refer **Chapter 2** for more information.

1.1.2 Related Terminology

This chapter is compilation of different terminologies used in experiments conducted to design clock binary tree. It has detailed description of binary tree, types of binary tree and clock binary tree. Refer **Chapter 3** for more information.

1.1.3 Problem Definition

This chapter defines the problem statement for thesis. Refer **Chapter 4** for more information.

1.1.4 Algorithm and Implementation for Horizontal Tree

This chapter describes solution to the problem definition. It has detailed description of algorithm which is developed to find possible valid binary tree and its implementation for Horizontal clock binary tree. This chapter depicts inputs taken from user, implementation of algorithm, operation of algorithm and its results in pictorial form. Refer **Chapter 5** for more information.

1.1.5 Algorithm and Implementation for Vertical Tree

This chapter describes approach and implementation used in designing vertical clock binary tree. working and functionality of algorithm is described in this chapter. refer **Chapter6** for more information.

1.1.6 Experiment On Routing of Horizontal Clock Binary Tree

Experiment on synthesized Clock Tree structure is documented in this chapter. Routing of nets for clock binary tree is explained in this part. To achieve minimal skew experiments on routing is exercised for all clock tree structures, for more information refer **Chapter7**.

1.1.7 Experiment on Transition and Power optimization

Experiment on Transition and Power optimization is depicted in **Chapter 8**. This chapter is compilation of experiment done to achieve good transition at the output of each stage by changing driving strengths of clock buffers.

1.1.8 Conclusion and Future work

Outcome of experiments and algorithms are concluded here in this chapter. Refer **Chapter 9** for more information.

Chapter 2

ASIC Design Flow

As we know, there are many steps to follow to manufacture IC TO(tapeout). There are several attributes associated with IC fabrication process. As Example Specs designing, Architecture designing, Front end designing, Backend designing and at last physical production of silicon. In this case many horizons to work together for a single chip. This Chapter describes mainly a back-end design part of chip designing.

2.1 Backend Design (Structural design)

2.1.1 Synthesis

RTL (HDL - design) is the input to backend design flow and synthesis steps. Going down in the abstraction level of ACIS design flow can be called Synthesis, which is the process of converting HDL design to the netlist. Where, netlist can be defined as connection of cells (basic gates), sometimes those cells are complex cells. These cell information is been defined in technology library files.

2.1.2 Floor plan

Floor planning itself is a large topic, deals with block placement, IO planning, power grid and top level clocks. Block placement is typically a placement of predefined

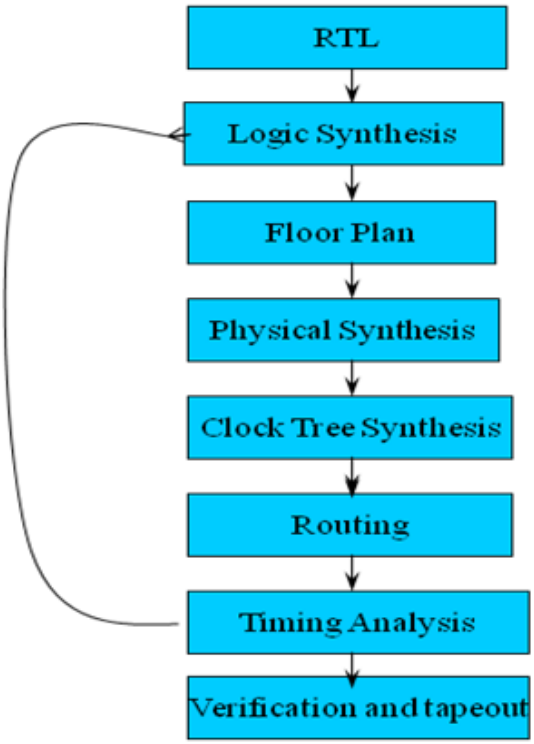


Figure 2.1: Back end design flow

blocks called BLACK-BOX. IO pins are planned at the time of floor planning. Clock binary tree is derived in parallel. Power ring and power grids and drop of points of clock are designed on this stage too. In this process chip is divided into several partitions to reduce execution time of the synthesis process.

2.1.3 Physical Synthesis

Physical synthesis is the logic optimization process in back-end flow. Library files are input of the synthesis process. Standard cells like and, or, not, latch, etc. are placed on the partition, placement of cells is the process, where logical code is converted to particular cells as per its functionality. In case, if RTL has if-else block then by doing synthesis the block will turn out to be 2x1 Mux after synthesis. Now 2x1 Mux is defined in library file and synthesis process uses it to give netlist as final output. Netlist is .v file, which contains connections of individual cells and cell types.

2.1.4 CTS (Clock Tree Synthesis)

CTS mean clock tree synthesis, during floor plan clock binary trees are defined and those are pushed top down the hierarchy. This can be divided in two parts global CTS and local CTS. Where global is something which is already told, and local is within partitions drop of points are defined priory. Clock distribution from RCBs to local partition cells are also the part of CTS.

2.1.5 Routing

Metal layers should be defined priory to routing. Generally vertical routing and horizontal routing are kept in different layers to increase simplicity in routing. This process is iterative process where it finds the smallest path between two cells and then places the wire. It takes care of constraints to use while placing wires.

2.1.6 Timing Analysis

Timing analysis can be carried out after routing of clock and when CTS is over. This is basically calculating parasitics and delays of each and every wires which are routed. Timing analysis is basically delay calculation of all possible paths for functionality based on physical parameters (like parasitic), constraints and technology limitation to see if operating frequency, which was targeted is met or not.

2.1.7 Layout Verification

Layout verification is the verification of physical aspects like spacing between two metal lines, some technological limitation like pinch-off of the shape for the device which we will be developing on silicon. This is the place where all DRC rules checking happens. The layout verification also develops some new algorithms to overcome problems like via density, Antenna effect reduction etc. hence layout verification is the process of checking of layout matching the netlist. This process also maintains

spacing rules.

2.1.8 Tapeout

Is the process of giving final database to mask shop. Mask shop separates all the fabrication layers and those are used in fabrication laboratory.

2.1.9 Other steps of flow

During the entire back end flow there are some intermediate steps to follow, those are listed as below.

1) FV - known as Formal verification where RTL (FUB) and netlist are being compared and verified

2) ECO - Engineering Change Over : where late changes that are made directly to netlist (without re-running synthesis), and compared to updated RTL to make sure that both of them matches.

Chapter 3

Related Terminology

This chapter describes used terminology used in this experiments. It also has detailed description of binary tree, types of binary tree and clock binary tree.

3.1 Binary Tree

Binary tree is a tree structure in which each node has at most two child nodes, usually distinguished as "left" and "right". Child nodes can again have two child nodes respectively. Thus here the possibility for each node can be like having two nodes as child or having 0. As shown in figure 3.1, stages are denoted on the right side of the tree. Number written inside circle are basically number of nodes. Node 1 is the origin of entire tree thus, it is known as root of the tree.

3.1.1 Principle of Binary tree

Principle to generate Binary Tree is "Each buffer should face same amount of load on both the side". By following this principle only M'ry tree is possible, where M is equal to 2k (even number). To remove interstage skew M'ry should be migrated to binary. For example 4'ry tree has equal load from root node but distance from root node for all stage2 cells are not same. By this basic understanding there can be

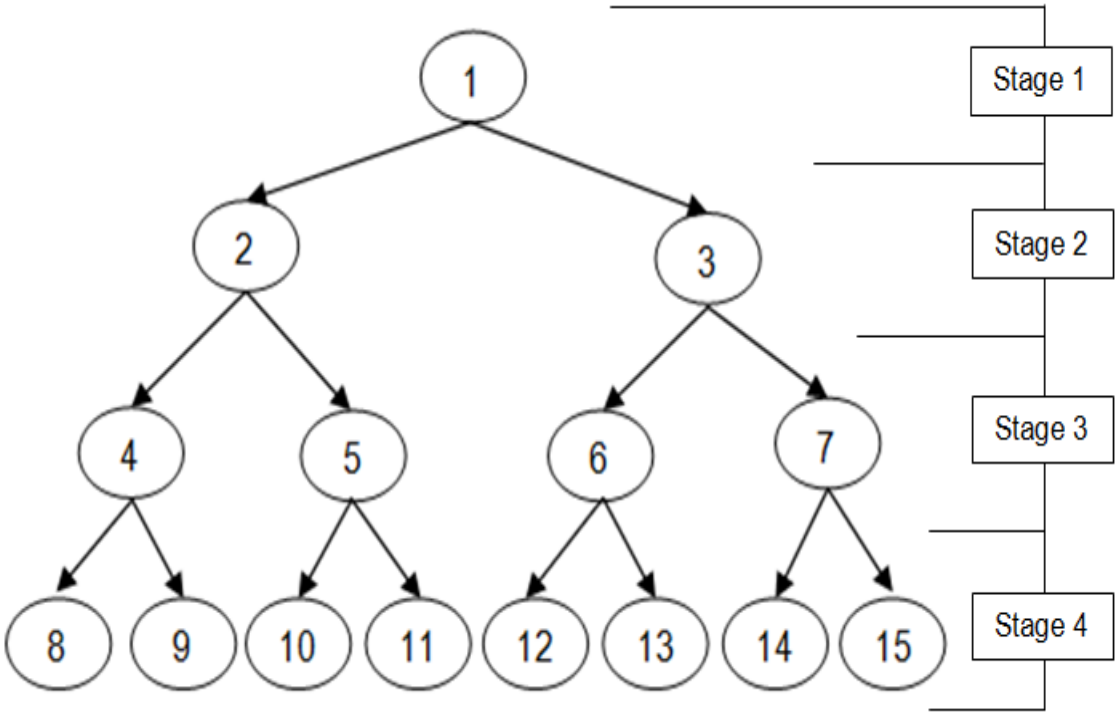


Figure 3.1: Binary tree

two types of binary tree structure construction possible (1) Linear structure. and (2) $2^{(stage-1)}$ structure. Refer 3.1.2 Types of Binary tree for more details.

3.1.2 Types and Mathematical equation of Binary tree

This section contains structures/types of binary tree which can be use in clock binary tree design where principle of Binary tree is maintained. This section also has mathematical equations of both the tree structures.

Linearly increasing structure

As shown in figure 3.2, Text in circles are basically stage and number of node where stage is the prefix and number of node is suffix. Here the basic concept of this tree structure is that as number of nodes increase the stages also increase, that is the reason why the name is give linear structure. As shown in figure, binary tree principle

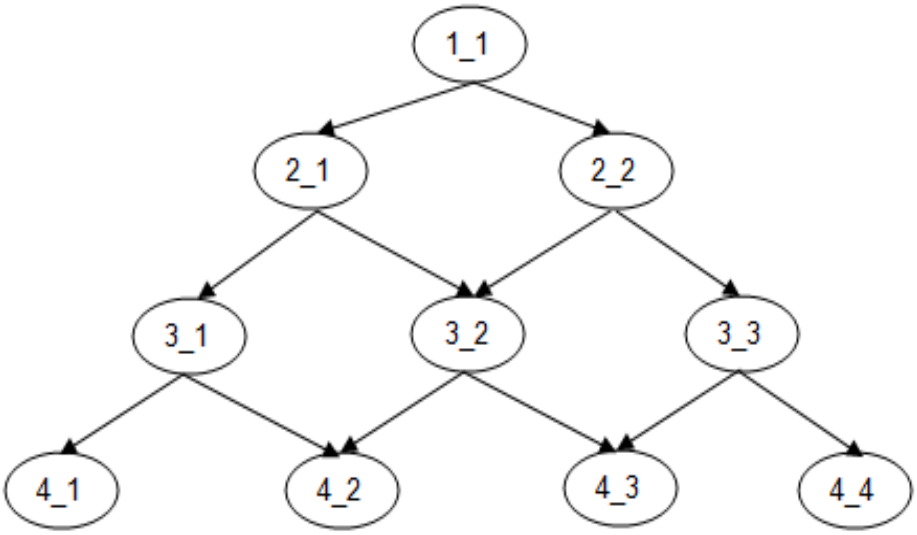


Figure 3.2: Linearly increasing structure of tree

also follows. For example, moving from stage2 to stage3 we can see that node2-1 facing node3-1 and node3-2 as load and node2-2 faces 3-2 and 3-3 load in other words node3-2 is common for 2-1 and 2-2 node. Current from each node flows equally provided all the nodes within same stage are identical in nature.

Mathematical equation:

By referring to above structure we can derive that No. of nodes in each stage are same as what stage they are into.

No. of Cells = Stage ——————————————————————————- Eq1

But, we want to derive basic formula of this structure with respect to previous stage. Hence, For **stage = 1**

No. of Cells in stage = 1 , else

No. of Cells in stage = No. of Cells in (stage - 1) + 1 ——————————— Eq2

$2^{(stage-1)}$ structure

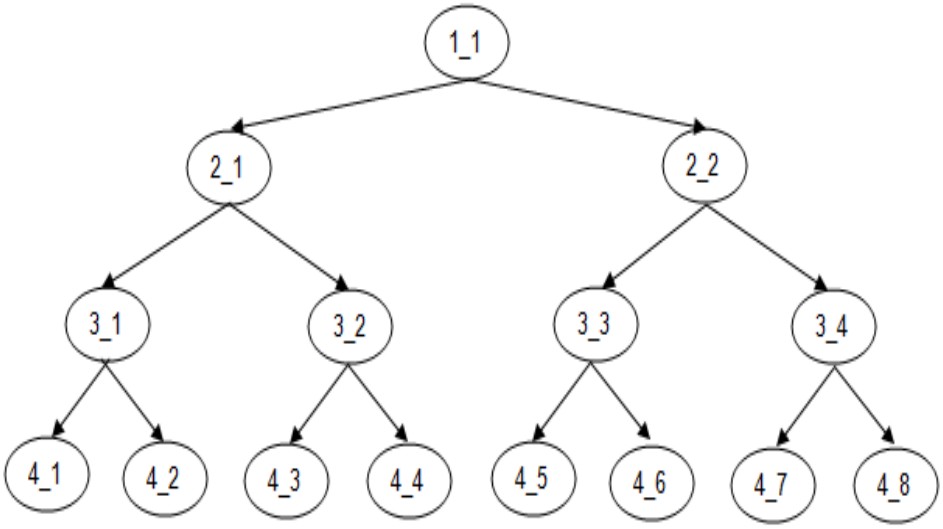


Figure 3.3: $2^{(stage-1)}$ structure of tree

As shown in figure 3.3 the text in circles are basically stage and number of node, where stage is the prefix and number of node is suffix. Here the basic concept of this tree structure is number of nodes are same as twice of stage hence tree structure becomes as shown in figure and we can clearly observe that binary tree principle also follows. We can also observe that each node is splitting by two to generate new nodes from the original node. Current from each node flows equally provided all the nodes within same stage are identical in nature.

Mathematical equation:

By referring above structure we can derive that No. of nodes in each stage are double then its stage they are into.

No. of Cells = 2 * Stage ——————————————— Eq3

But we want to derive basic formula of this structure with respect to previous stage. Hence, For **stage = 1**

No. of Cells in stage = 1 , else

No. of Cells in stage = 2 * (No. of Cells in (stage - 1)) ——————— Eq4

3.2 Clock Binary Tree

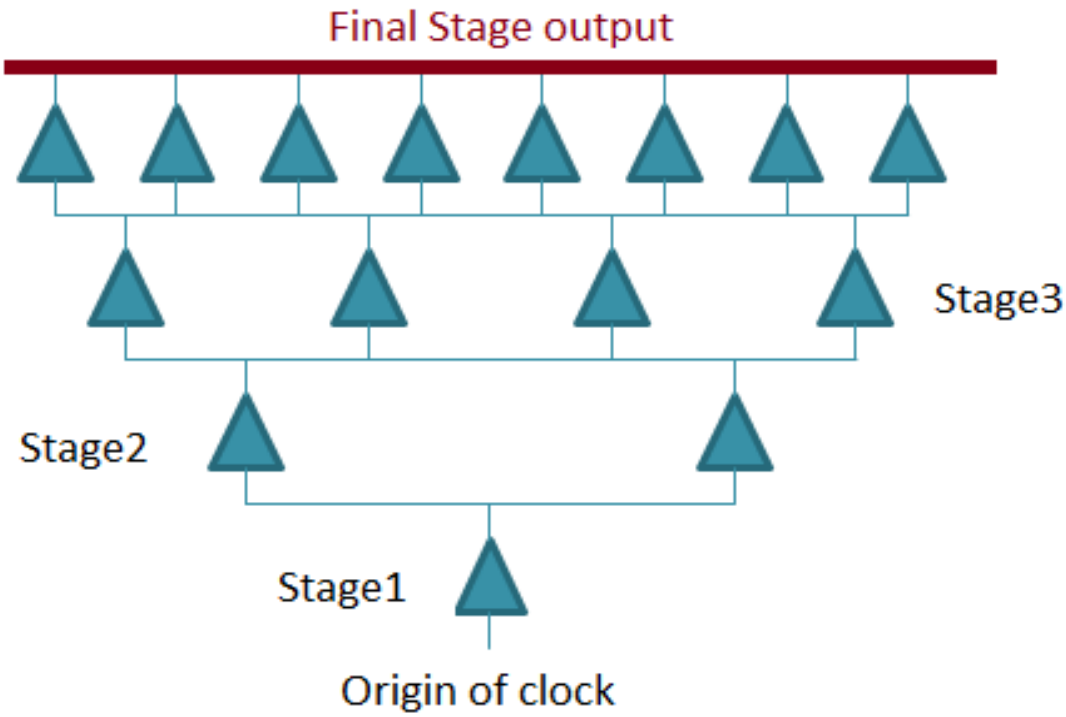


Figure 3.4: Clock Binary tree

As shown in figure 3.4 the top most horizontal line is knows as Clock binary tree of clock network. This structure is basically known as global clock network where all buffer increase driving strength of the clock signal and lessen the skew. As shown in figure inputs and outputs of buffers are connected to inputs and outputs of other cells

in the same stage, it is basically to decrease skew at the output of all buffers. Here also it becomes mandatory to have identical buffer within the stage. This structure is popular because it has advantage of lesser skew across the clock binary tree. If we measure from left corner to right corner the skew between those end points becomes very low with using this structure. If number of clock origins are different then that many clock binary tree structures are required. Local distribution of clock is generated by taking taping from clock binary tree itself and these tapings are such a way that the end to end poins having skew difference near to zero.

Chapter 4

Problem Definition

4.1 Generate possible combination of tree structure by taking stage as input

First problem definition is to generate number of tree structures by taking number of stage required as input from user. These all generated structures should also have information regarding number of buffers in each stage.

4.2 Generate Coordinates of each cells/buffers for all tree structure

There can be two possibilities

1) **Horizontal clock tree** structure

2) **Vertical clock tree** structure

Algorithm should find coordinates with respect to options given by user. Once number of buffers in each stage are found based on vertical/horizontal structure entered by user. Algorithm should find geometrical coordinates of physical placement of buffers by taking length of clock binary tree and coordinates of origin as input from user. Algorithm should be able to find X and y location of all the buffer cells for all

the tree structure and provided this X and Y coordinates should follow binary tree principle, This also means that dimensions derived by this algorithm should have perfect symmetry.

4.3 Synthesis all clock tree structures

All the coordinates of all the cells/buffers in structure should go to synthesis flow. This synthesis flow is essentially comprises of placement of cells, selection of cells in terms of driving capability, metal layer connection and routing. This part takes main turnaround time as this process is bit slow.

4.4 Simulate all possible combination for best results

Once all the structures are gone through synthesis process and all Saved_cels(designs of different Structure after synthesis) are saved in then simulation data extraction could be done. The designs is simulated for respective parameters.

- Stage skew
- Insertion delay
- Transition
- Power Trade off

4.5 Optimization of Clock Binary Tree Structures based on Skew,Transition and Power

Based on simulation results one can be analyze, which structure needs what kind of optimization. Some experiments have been exercised to get optimum skew numbers,

optimum transitions and optimum power numbers. Refer **Chapter 7** for detailed description of experiment on routing of Horizontal Clock Binary tree to reduce skew numbers, these concepts also can be applied for vertical clock binary tree.

Experiment on Transition and Power optimization is depicted in **Chapter 8**. This chapter is compilation of experiment done to achieve good transition at the output of each stage.

Chapter 5

Algorithm and Implementation for Horizontal Tree

This section is basically a solution to problem definition which is given in **Chapter 4**. There can be many ways to derive all the structures of binary tree. One of the ways is implemented and documented in these sections. This algorithm primarily takes care of **Eq2** and **Eq4** to find possible tree structures. This algorithm tries all the combination where both the equations are being applied on all the stages and comes up with number of valid possibilities. This section also contains details description of working of algorithm, primary inputs to algorithm and results and out comes of algorithm. This algorithm is a perl script where it interacts with user buy using standard input-output. To understand its primary input and working refer the next sections.

5.1 Input from User

As shown in figure 5.1 primary inputs of algorithm are listed below.

- **length of Clock binary tree** (Distance between end cells)

  Clock binary tree length is measured in micron, it is basically width of clock

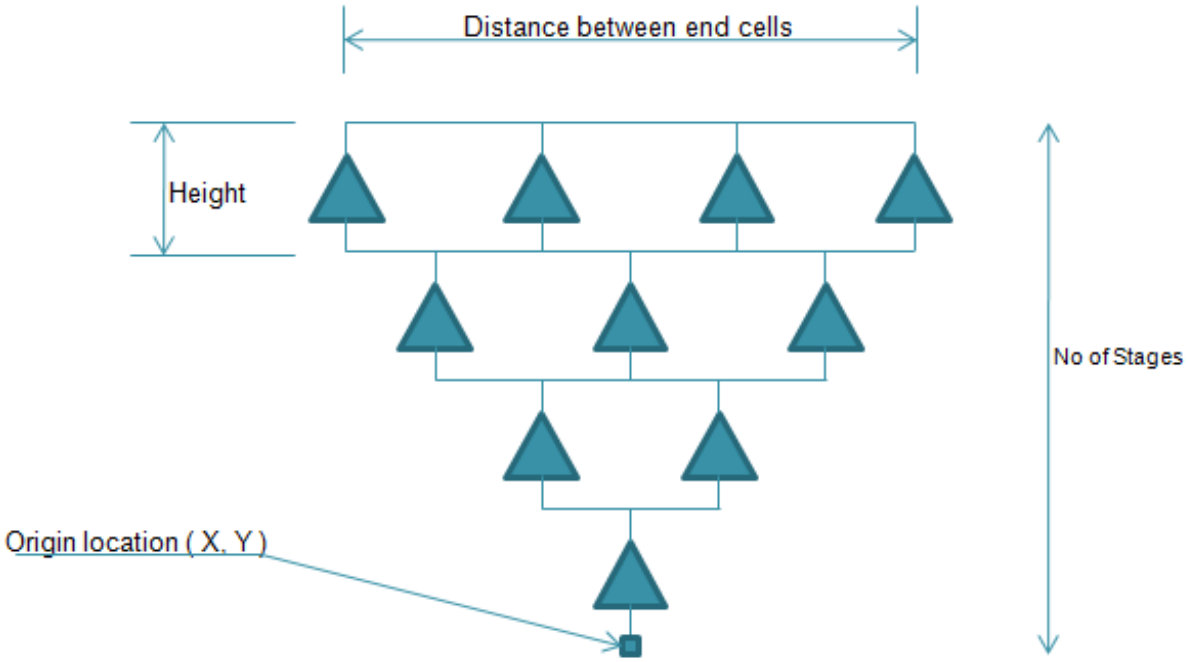


Figure 5.1: Input from User

tree if Tree structure is horizontal.

- Height

  It is the Y distance that user wants between of two stages.

- No. of stages

  No. of stage, required to find out No. of combinations of valid Clock binary tree.

- Coordinates of origin point

  This are X and Y location of origin taken from user. This dimensions are added as offset letter on, once coordinates of buffers are found by algorithm.

5.2 Finding total possible tree structures

Eq2 and **Eq4** are used to generated total number of possibilities of clock tree. Figure 5.2 has two type of trees. One is completely constructed by linear and the other by

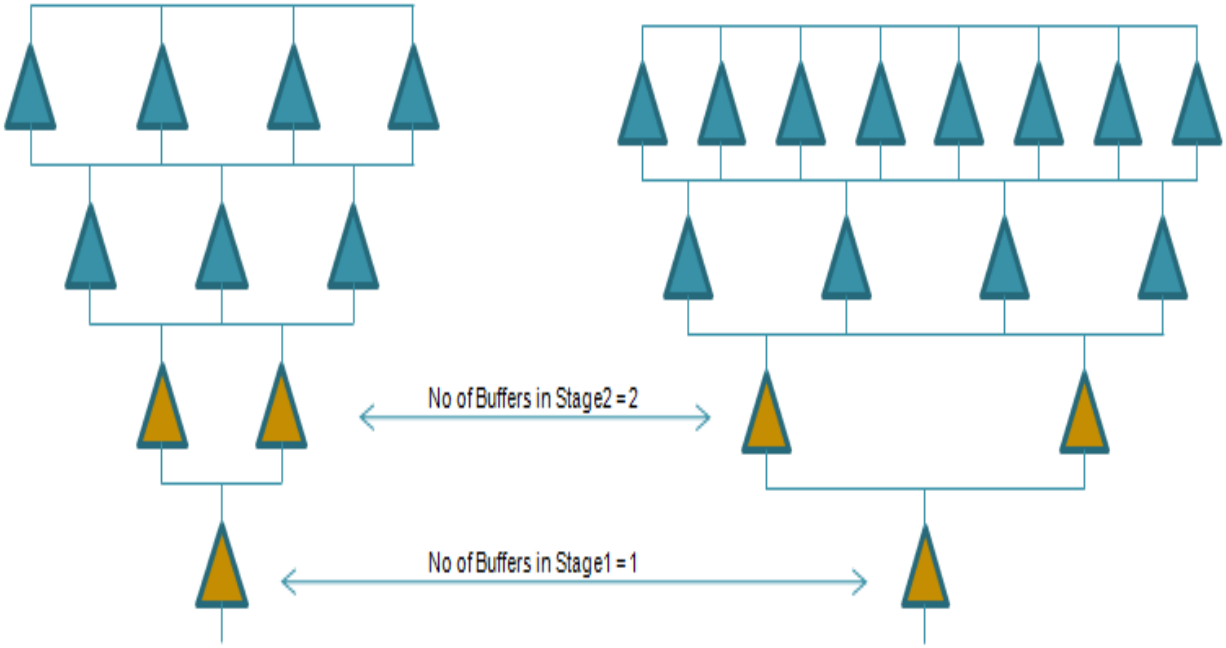


Figure 5.2: Comparison of tree structures

$2^{(stage-1)}$. By comparing both the structures of tree it can be said that linear and $2^{(stage-1)}$ has one buffer in stage1 and two buffers in stage2 in all the conditions. Irrespective of equation being used to derive no of buffers the stage 1 and stage 2. for both the combinations (linear or $2^{(stage-1)}$) always has same number of buffers. Hence one should not try applying combinations on the same.

The next task is to find equation for valid possible combinations of clock binary tree generated by taking stage as input from user. It is clear, that to find out possible combinations it is necessary to find No. of cells/buffers present in each stage. Once those are found, based on number of buffers present in particular stage No. of cells/buffers of the next stage could be derived. Refer 5.3 for more information on how algorithm works and finds cells/buffers for particular stage and how algorithm generates all the combination of possible tree structures.

5.3 Equation for possible structures

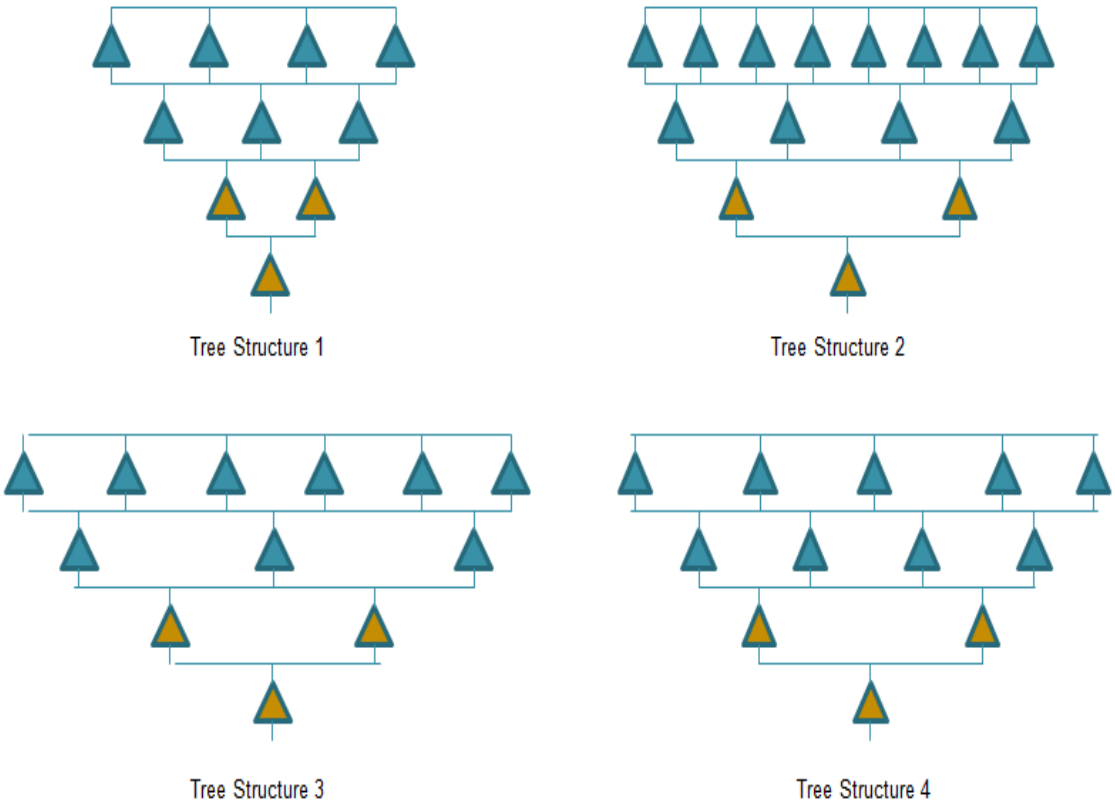


Figure 5.3: Valid Structures for stage = 4

To derive number of all possible combination of tree structure directly depends on how many stages the structure has as input from user. The generating combination is depended on two equations. As it is known that stage1 will always have one buffer and stage2 will always have two buffers. In other way for any combination it should be one and two for respective stages. E.g. where number of stage = 4

Now stage2 will always have two buffers and from stage2 to stage3 there can be two combinations, generated using Eq2 and Eq4. For example Eq2 (linear) is used to find number of cells for stage3 then the output will be 3 for stage3. and Eq4 ($2^{(stage-1)}$) is used instead of Eq2 then the output will be 4 cells in stage3. Same

case applied for every stages, where the actual computation is done based on number of cells present in previous stage. Hence, going further total combinations available become 4 at last by using both the equations and its combinations. As shown in figure 5.3 there can be 4 valid possible tree structures generated when stage=4. So mathematical equation can be derived by this experiment.

Mathematical equation:
As we have seen in experiment the mathematical equation for number of possible tree structures can be

$$\textbf{No. of Possible Tree structure} = \mathbf{2}^{(Stage-2)} \quad \text{Eq5}$$

Hence Stage$\bar{5}$ then no of combinations will become 8 and it goes on respectively.

5.4 Finding X and Y location

In the process first task is to find number of possible tree structures based on outcome the next task is to find coordinates for cell placement for each tree structure, But prime requirement is to maintain symmetric nature of tree.

5.4.1 Finding X location

Finding X location for all the cells present in particular tree combination can be described by flow chart shown in figure 5.4. The precess of deriving X positions for all the cells of particular tree structure could be repeated for every tree structure. This is implemented in script by looping current set of logic. Length of clock binary tree distance is taken from user as primary input and numberof cells in each stage is derived by algorithm.

Now dividing total distance of last stage as user has entered by (No. of cells - 1) gives the required distance between each cell for all the stages, say that as **Dist_x**.

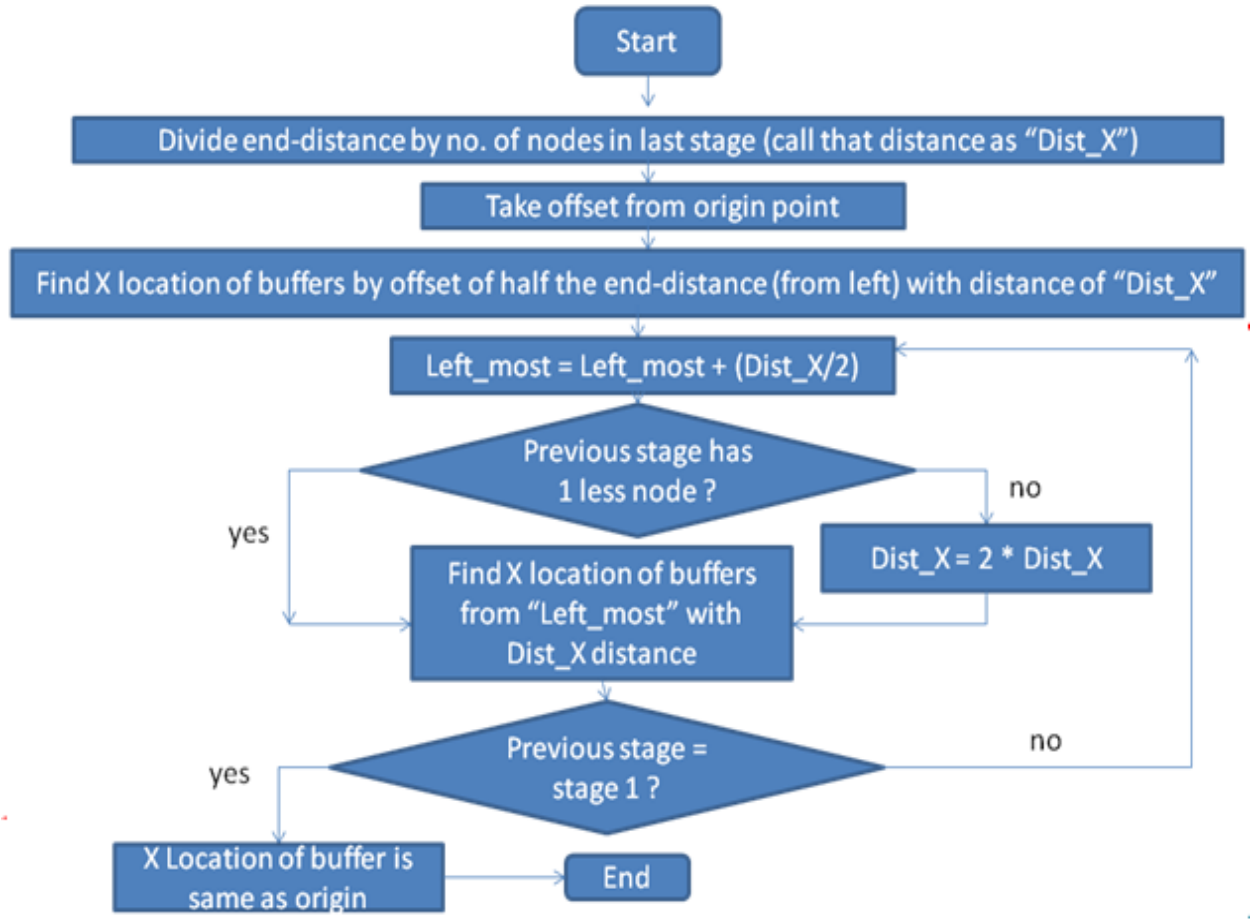


Figure 5.4: Flowchart to find X Co-ordinates for all cells

co-ordinates of origin point could be useful to add the offset, basically offset of half the clock binary tree length and subtract from the origin (x) co-ordinates. That is how left most position of clock binary tree is derived, say that as **Left_most**. Algorithm will find X-location of all the last stage cells and will add Dist_x/2 to Left_most and there after it checks the combination of previous cell weather it is a linear case or $2^{(stage-1)}$ case. If it is linear then Dist_x remains same and algorithms finds all the x locations for all the cells in current stage. If the combination of previous stage is $2^{(stage-1)}$ then algorithm doubles the Dist_x and finds distances of all the cells in current stage. This thing goes on till previous stage is not root. once stage is equal to 1 algorithm directly assigns X-location value as X-location of origin. When these

iterations are over for one design, algorithm takes another structure and does. This iteration this happens until all the combinations are finished. Now next task is to find out Y location for all the cells of all the tree structures.

5.4.2 Finding Y location

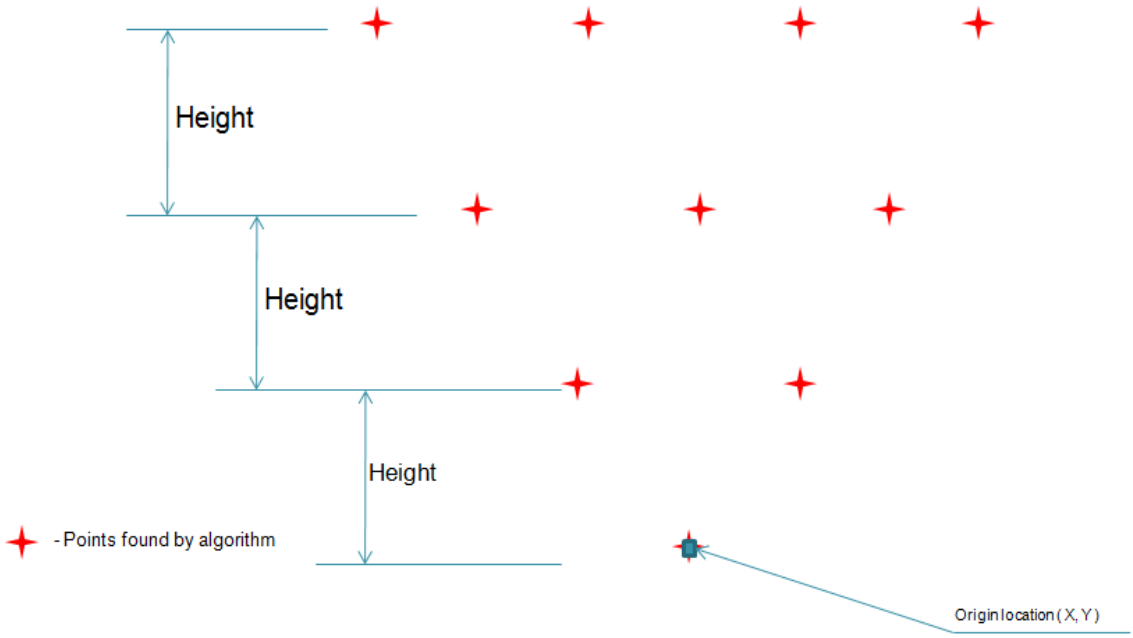


Figure 5.5: Finding Y Co-ordinates

Finding Y location for all cells become easier as Height is predefined input, from user. One more parameter is coordinates of origin also a userdefined Input. Algorithm takes Height as offset and keeps on adding to the Y location of origin. Algorithm takes care of stage and assigns Y location to all the cells of particular stage that is how it becomes in parallel line as shown in figure 5.5. Finding X location and Y locations are independent of each other and that makes them stand alone. Thus, we can find Y location prior to X or Y location prior to X will always gives the same result however algorithm does not have privilege to run them in whatever order it wants. as shown in figure cross points are basically the cell location found by algorithm which are stored in Arrays.

5.4.3 Modification in Location by considering Cell height and width

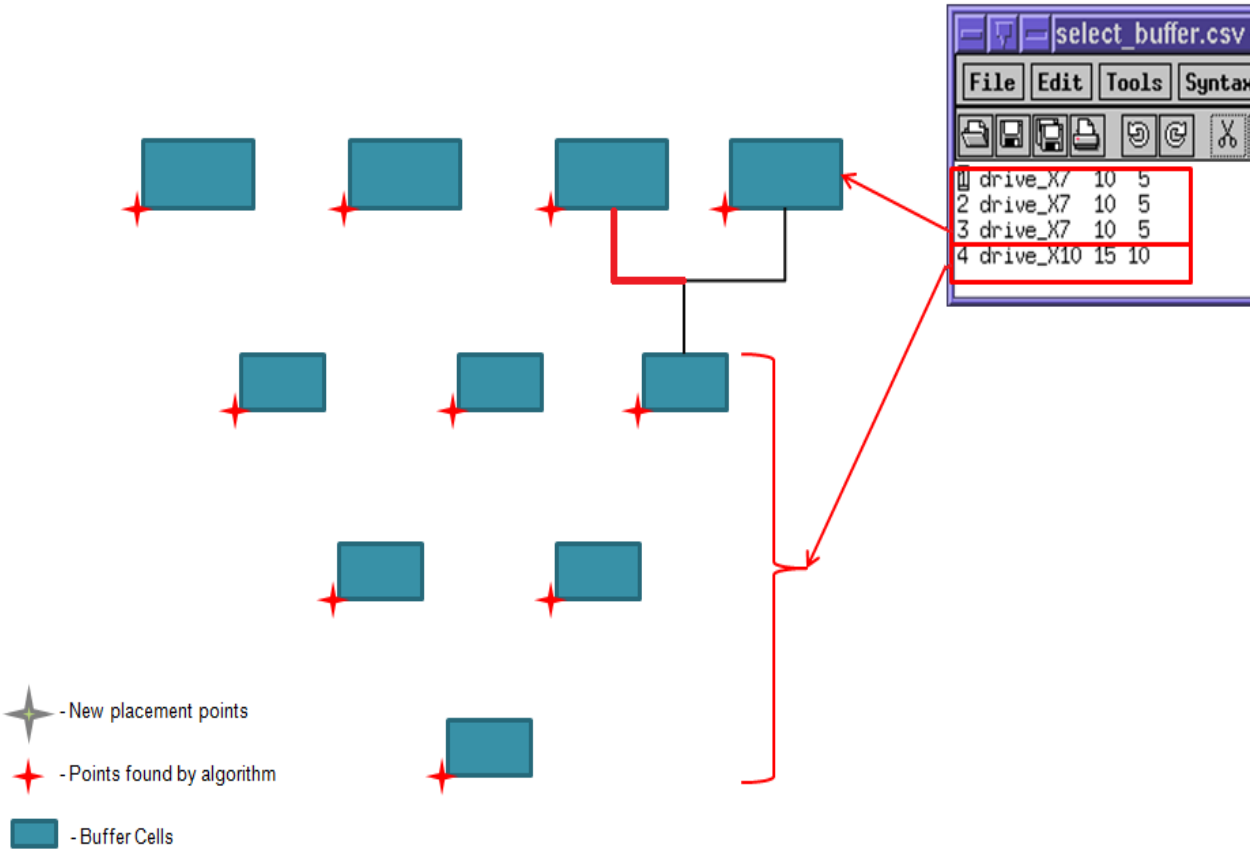


Figure 5.6: Modification in location of cells

As described in previous topic the coordinates found by algorithm are denoted as cross sign. Now placement of cells on derived location is exercised with help of TCL scripts. TCL scripts run on backend tools. Cell positions are given to tools to place cell, those placement of cells look as show in figure 5.6. Basically backend tool considers those location as left-lower corner of cells and places accordingly.

If cells for all the stages are of equal height and width then this algorithm will work properly, but we know all the stages can have different types of cells across the different stages but it should be identical within a stage. Figure 5.6 is basically a example of

such case where width and height of cell is different. Because of difference in width of cells there can be problem in symmetric nature of tree as shown in figure. for two upper right cells, where thin-line and thick-line are of different length. Thus, load cannot be equal for source cell and that lead to skew. Therefore design will give worst results, therefore algorithm should be aware about it. Select_buffer.csv file is taken as input by algorithm were the data inside that file is number of stage, cell name used for particular stage, width of that cell for particular stage and height of that cell for particular stage. Algorithm already has details of Height and Width of each cell used for specific stage thus the solution to this problem can be seen as show in figure 5.7. Where the solution is basically shifting cell by half of the width and height and derive new point of placement after this. This solution will essentially maintain symmetry structure and hence not lead to skew.

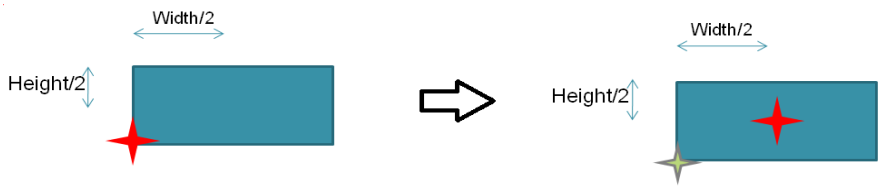


Figure 5.7: Shifting cell by width/2 and hight/2

5.5 QC of X and Y location

Though algorithm finds proper locations for each cells but Quality checking of output is must. Therefore algorithm itself does its quality checking by measuring difference of X position for all the cell positions. As show in figure 5.8 and figure 5.10 buffer cells are given name as "buffer_x_y" where x is stage and y is no of node in particular stage.

Figure 5.9 is the QC result of figure 5.8 where linear equation is used. Figure 5.11 is QC result of figure 5.10. When the structure for particular stage is found to be linear then algorithm does its X-location measurement as show in figure 5.8. For an

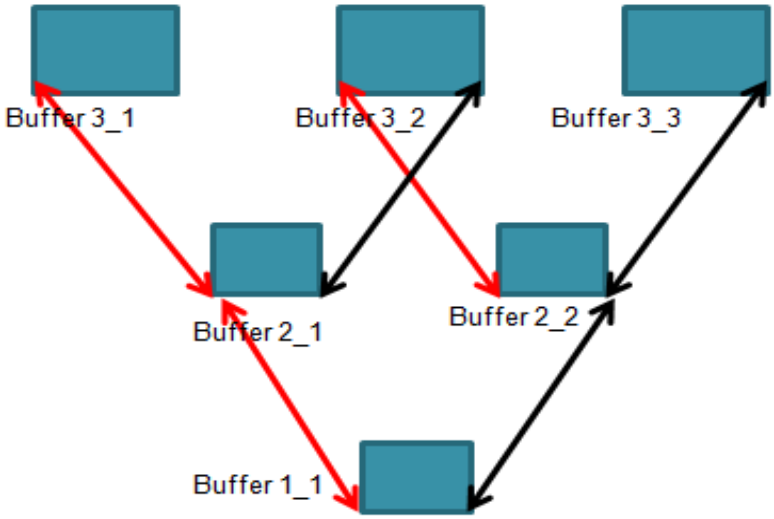


Figure 5.8: Distance between cells for linear structures

Distance_bet	Distance	Distance_bet	Distance	Type
dis_2_1-1_1	33.3333	NA	NA	Linear
dis_2_2-1_1	33.3333	NA	NA	Linear
Distance_bet	Distance	Distance_bet	Distance	Type
dis_3_1-2_1	33.3333	NA	NA	Linear
dis_3_2-2_2	33.3333	dis_3_2-2_1	33.3333	Linear
dis_3_3-2_2	33.3333	NA	NA	Linear

Figure 5.9: Distance between cells for linear tree structure

example distance between buffer3_1 and buffer2_1 should be equal to distance between buffer3_2 to buffer2_1 then only symmetry can be confirmed. In figure 5.9 dis_3_1-2_1 is 33.33 micron and dis_3_2-2_1 is again 33.33. Thus, it is confirmed that both buffers (buffer3_1 and buffer3_2) are place equidistance from buffer2_1. For others cells table has same value hence it can be said that structure is having proper symmetry.

When the structure for particular stage is found to be $2^{(stage-1)}$ then Algorithm does its X-location measurement as show in figure 5.10. For an example distance between buffer3_1 and buffer2_1 should be equal to distance between buffer3_2 to buffer2_1,

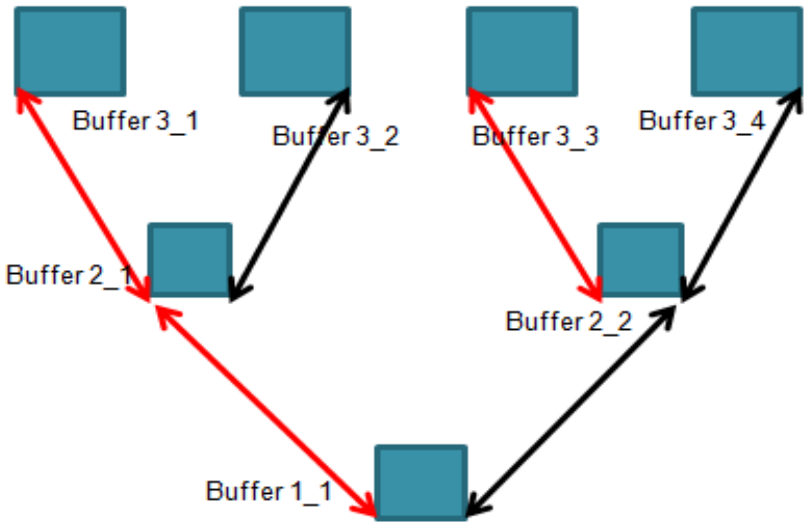


Figure 5.10: Distance between cells for $2^{(stage-1)}$ structure

Distance_bet	Distance	Distance_bet	Distance	Type
dis_2_1-1_1	50	NA	NA	Linear
dis_2_2-1_1	50	NA	NA	Linear
Distance_bet	Distance	Distance_bet	Distance	Type
dis_3_1-2_1	25	NA	NA	2^n
dis_3_2-2_1	25	NA	NA	2^n
dis_3_3-2_2	25	NA	NA	2^n
dis_3_4-2_2	25	NA	NA	2^n

Figure 5.11: Distance between cells for $2^{(stage-1)}$ tree structure

distance between buffer3_3 and buffer2_2 and buffer3_4 and buffer2_2 should be equal then only symmetry can be confirmed.

In figure 5.11 dis_3_1-2_1 is 25 micron and dis_3_2-2_1 is again 25. Thus, it is confirmed that both buffers (buffer3_1 and buffer3_2) are place equidistance from buffer2_1. Same as this in figure 5.11, dis_3_3-2_2 is 25 micron and dis_3_4-2_2 is again 25. Thus, it is confirmed that both buffers (buffer3_3 and buffer3_4) are place equidistance from buffer2_2. Algorithm works well for both the cases. When both the approaches (linear

and $2^{(stage-1)}$) are separate or with mixed approach. Algorithm does QC check after finding new location of cells by shifting them by cell width/2 and cell height/2 thus it can be proved that output of algorithm is trustworthy to integrate in actual design flow.

5.6 Results of Algorithm

Algorithm generates PNG file for giving pictorial view of all combinations of trees. Graphical display can help to debug the symmetry properly in case there is misbehavior in output of simulation. These results do not contain simulation and integration part of all the tree structure.

5.6.1 PNG file of all clock tree structure

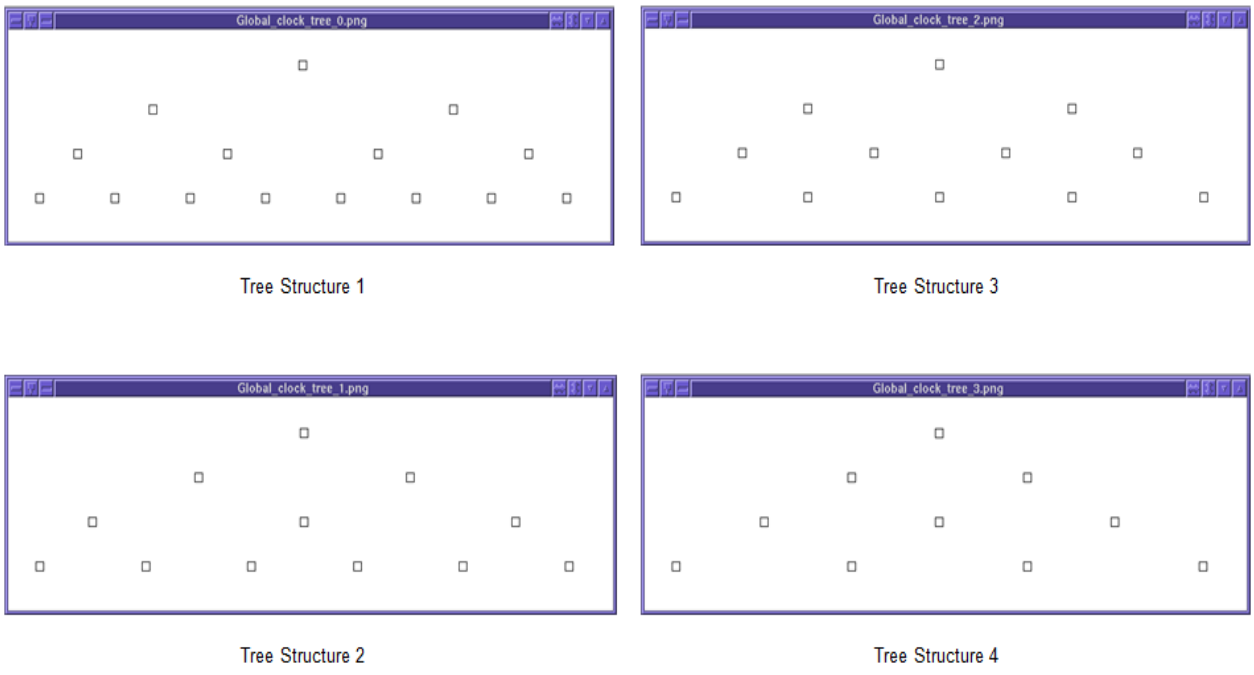
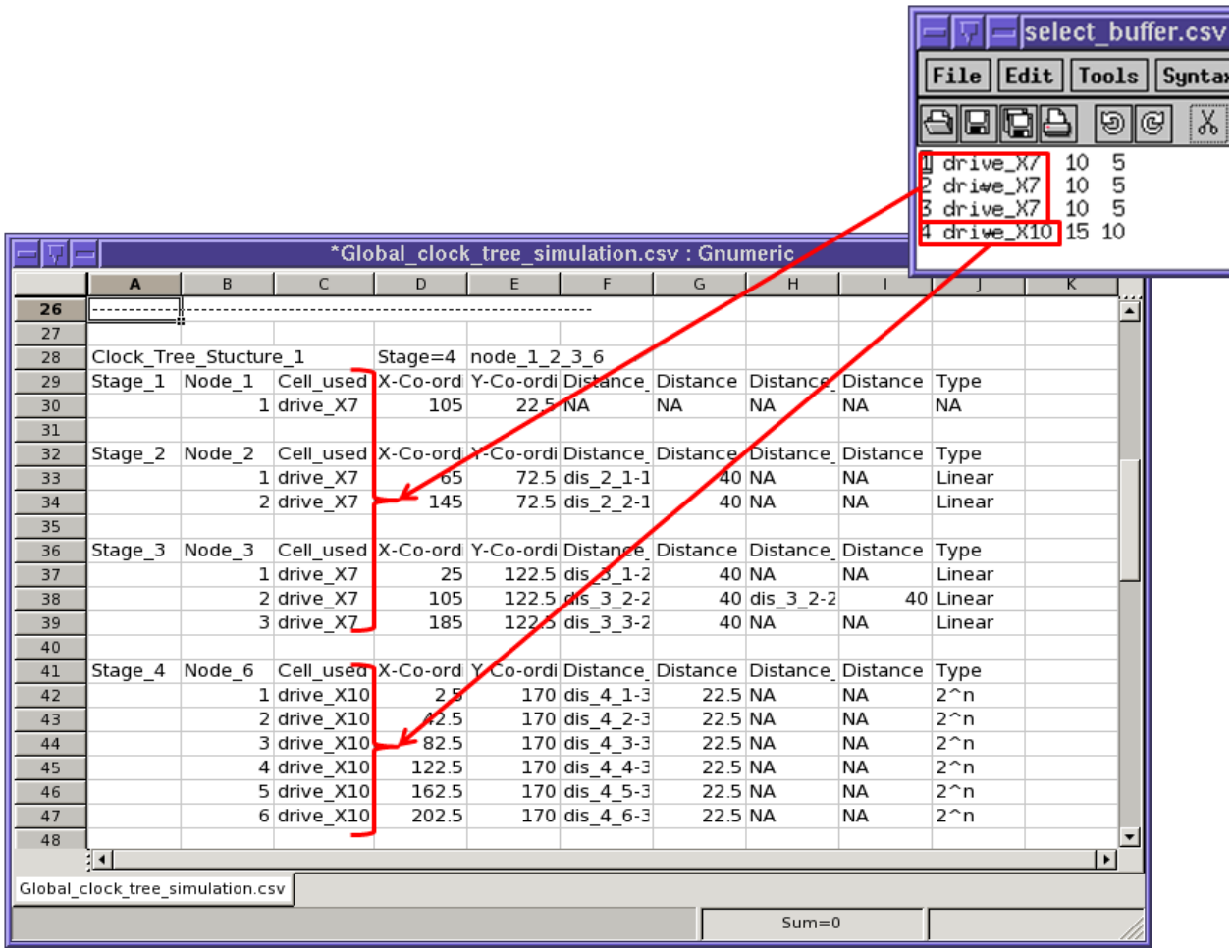


Figure 5.12: PNG output of algorithm when stage = 4

Figure 5.12 is the output of algorithm when input is given as stage = 4. Here we can clearly see different combinations of structures are generated. For example Structure

1 is generated by $2^{(stage-1)}$ equation for all the stages, Structure 4 is generated by linear equation for all the stages and Structure 2 and structure 3 are mixed approach. Algorithm generates different PNG files for all possible combinations, if stage input is 5 then total 8 PNG files will be generated by name "Global_clock_tree_X" where X is number of combinations that algorithm generates.

5.6.2 CSV file of all clock tree structure



	A	B	C	D	E	F	G	H	I	J	K
26	----------		----------								
27											
28	Clock_Tree_Stucture_1			Stage=4	node_1_2_3_6						
29	Stage_1	Node_1	Cell_used	X-Co-ord	Y-Co-ordi	Distance_	Distance	Distance_	Distance	Type	
30		1	drive_X7	105	22.5	NA	NA	NA	NA	NA	
31											
32	Stage_2	Node_2	Cell_used	X-Co-ord	Y-Co-ordi	Distance_	Distance	Distance_	Distance	Type	
33		1	drive_X7	65	72.5	dis_2_1-1	40	NA	NA	Linear	
34		2	drive_X7	145	72.5	dis_2_2-1	40	NA	NA	Linear	
35											
36	Stage_3	Node_3	Cell_used	X-Co-ord	Y-Co-ordi	Distance_	Distance	Distance_	Distance	Type	
37		1	drive_X7	25	122.5	dis_3_1-2	40	NA	NA	Linear	
38		2	drive_X7	105	122.5	dis_3_2-2	40	dis_3_2-2	40	Linear	
39		3	drive_X7	185	122.5	dis_3_3-2	40	NA	NA	Linear	
40											
41	Stage_4	Node_6	Cell_used	X-Co-ord	Y-Co-ordi	Distance_	Distance	Distance_	Distance	Type	
42		1	drive_X10	2.5	170	dis_4_1-3	22.5	NA	NA	2^n	
43		2	drive_X10	42.5	170	dis_4_2-3	22.5	NA	NA	2^n	
44		3	drive_X10	82.5	170	dis_4_3-3	22.5	NA	NA	2^n	
45		4	drive_X10	122.5	170	dis_4_4-3	22.5	NA	NA	2^n	
46		5	drive_X10	162.5	170	dis_4_5-3	22.5	NA	NA	2^n	
47		6	drive_X10	202.5	170	dis_4_6-3	22.5	NA	NA	2^n	
48											

Figure 5.13: CSV file output of algorithm when stage = 4

To track/store results of X and Y co-ordinates of all the cells with their respective Clock tree structures algorithm stores its database into CSV file. Figure 5.13 is the example how things are stored in CSV file. This file stores X and Y location by

separating Clock tree Structure as shown in figure 5.13. The database are of Clock Tree Structure 1 where the configuration is 1 - 2 - 3 - 6 for respective stages. Column A is describing that data is of which stage, Column B describes how many nodes/cells are there in stage and Column C gives name of cells which is used of particular stage this information of name comes from Select_buffer.csv file. CSV file has X and Y location of all the cells. This CSV file also includes QC of X and Y location also.

Chapter 6

Algorithm and Implementation for Vertical Tree

6.1 Generation of Vertical clock binary tree

By modifying existing output of Horizontal tree algorithm vertical clock binary tree could be derived. Vertical clock binary tree is achieved by swapping x and y coordinates with each other as illustrated in figure 6.1. Provided origin of vertical clock binary tree is defined properly.

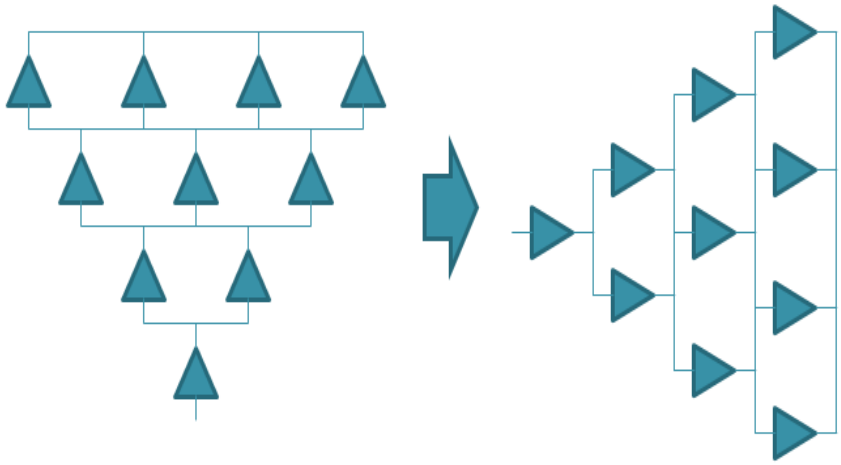


Figure 6.1: Interchanging X and Y coordinates

6.2 Routing problem with cell placement

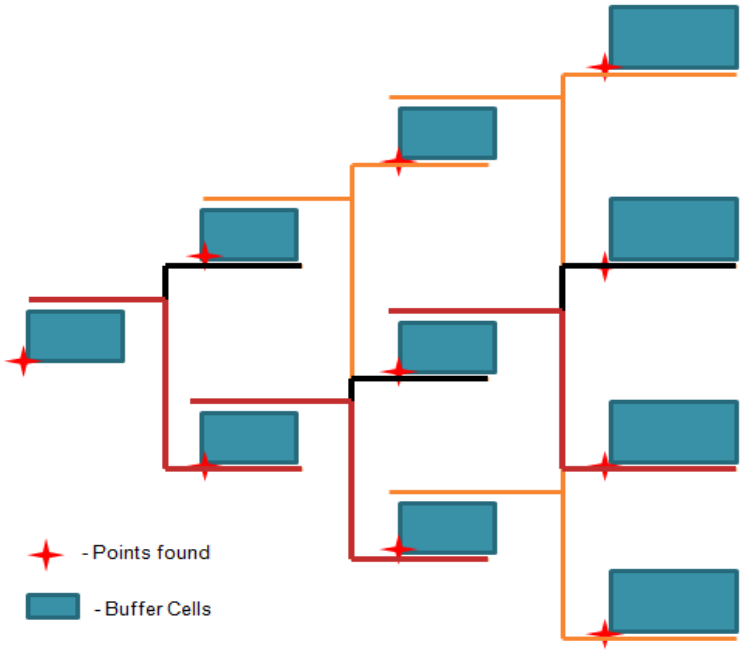


Figure 6.2: Routing problem with cell placement

Coordinates are derived by swapping the coordinates between x and y are shown in figure 6.2 by "+" signs, thereby placement of cells are as shown in figure. Cells are always placed such that their left lower corner edge placed on that coordinates found by algorithm. Considering all the cells are placed on swaped coordinates, if connections are made from output to input of next stage cells then routing breaks symmetry of its own this case is depicted in figure 6.2. As routing is not symmetry results will have more skews.

6.3 Solution to routing problem

Solution of routing problem is to rearrange cell placement to get symmetry routing between of cells. There can be two way of rearranging cell positions one is keeping last stage cells at their position and push respective cells down such that symmetry

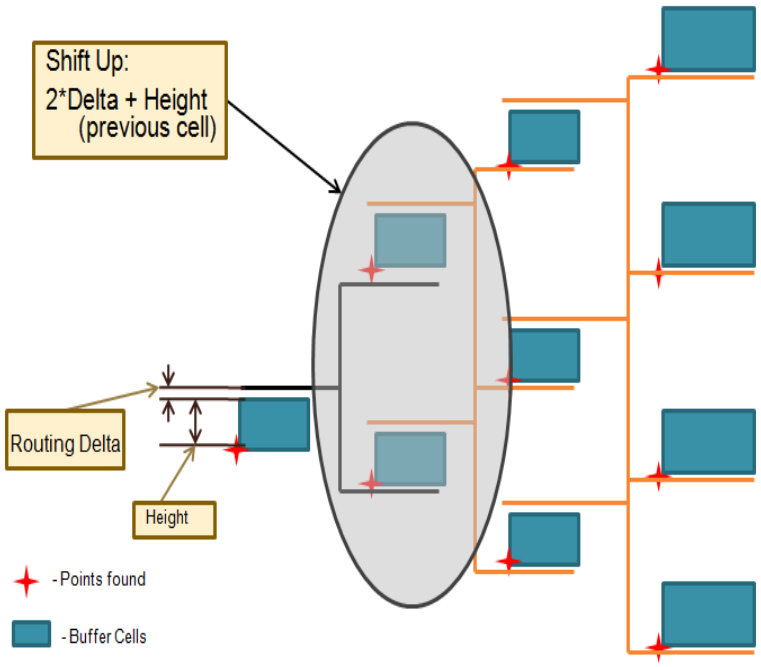


Figure 6.3: Solution to routing problem

can be achieved and the other possibility is put first cell of tree (stage 1) at its own position and shift up respective cells. As shown in figure 6.3 the concept of keeping stage 1 cell at is position and shifting other cells upwards respectively. The distance between cell and routing wire is taken as routing delta, usually that remains same for all the cells, for generating symmetry structure delta value is taken to calculate shift for all the cells. As shown in figure stage 2 cells are shifted up wards by 2 times delta and + height of previous cell, here the previous cell means first stage cell. But only shifting of 2nd stage will not help whole tree cells should shift with 2nd stage cells and respectively shifting entire stage cells with shift of position of existing stage. This process becomes the iterative process, which needs to go till the last stage is operated, by going to the last stage tree structure becomes as shown in figure 6.4. This figure has the last stage shift shown in it, this technique will also help when cells have different height e.g. first 3 stage has lesser height compared to last stage as shown in figure. but, the shift also has the parameter of previous cell height so it would give symmetry in terms of cells with different height. While different width of

cells will not become bottle neck for symmetry of tree, because Height value given by user while running this algorithm will be maintained between of stage (this distance is basically distance between centroid of the cell of stages).

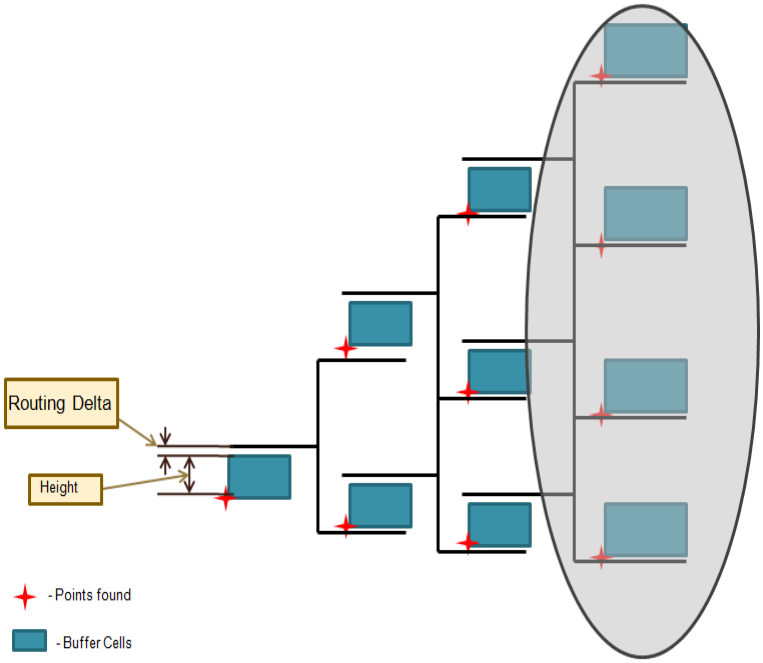


Figure 6.4: Solution to routing problem after all position shifting

Chapter 7

Experiment On Routing of Horizontal Clock Binary Tree

7.1 Routing Example of horizontal clock binary tree

While routing the metal connections the scenario is as demonstrated in figure 7.1 , power grids are fixed prior to synthesis cycle thus those power lines becomes obstacle for clock wire to route within that areas. One such example is demonstrated in below figure.

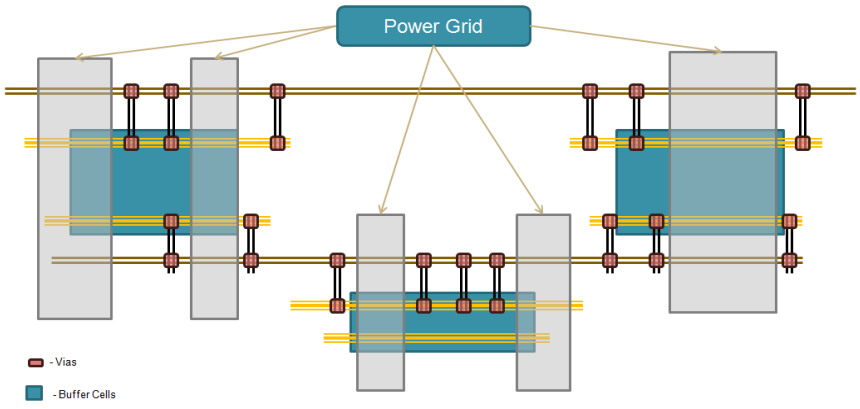


Figure 7.1: Routing Example of horizontal clock binary tree

As shown in figure vertical connection to the horizontal are blocking due to power grid. To drive more current to next level of buffer we need more vertical connections this can be seen that there is a tradeoff between power grid and vertical wire connections.

7.2 Description of Uncertainty window

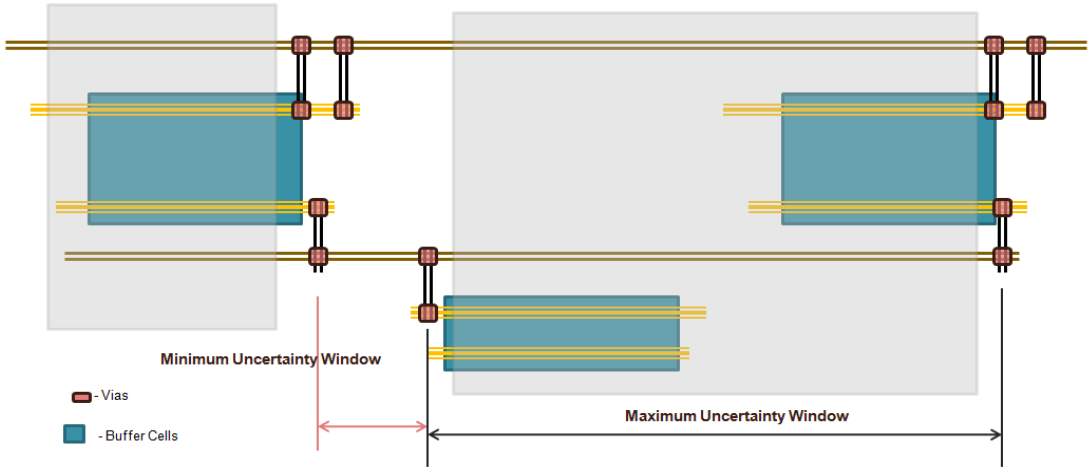


Figure 7.2: Uncertainty window in horizontal clock binary tree

During step of wire routing clock binary tree, there could be possibility that power grid covers cell completely. This situation is rare but, still it degrades the result. When power grip overlaps the clock cell, one of these example is illustrated in figure 7.2. As shown in figure, driver (bottom) cell is almost cover by power grid from right hand side and same is the case for other two, they are covered from left hand side. Now possible routing for connecting driver to load cells are also illustrated in figure, it also states that the distance from driver cell to left load cell is comparatively lesser than the distance between right load cell to driver cell. This minimum distance can be defined as Minimum Uncertainty window and the maximum can be defined as maximum uncertainty window. These windows can be defined by equations shown below.

Minimum Uncertainty Window = Routing distance between cells - Extra metal

Maximum Uncertainty Window = Width of Stage (N) cells + Width of Stage (N-1) cell + Routing distance between cells + Extra metal

7.3 Experiment to minimize uncertainty window

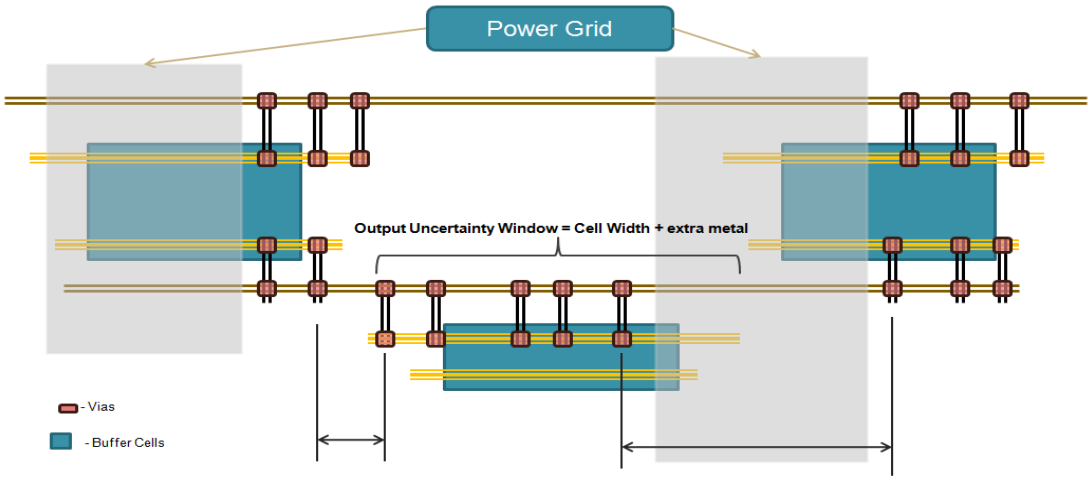


Figure 7.3: Horizontal Tree routing before experiment

Figure 7.3 shows Horizontal clock binary tree before beginning experiment for uncertainty window optimization. Here the power grid is not covering entire cell. This uncertainty won't be equal to maximum or minimum as defined in previous section. Driver cell still has lesser length to travel to load cell placed towards left hand side compared to load cell placed towards right hand side. As demonstrated in figure left hand side uncertainty window would be equal to minimum uncertainty window. To optimize this uncertainty we can reduce horizontal wire length routed (connected) on the cell, which is define as extra metal. This experiment will certainly optimize the uncertainty windows on both the side as described in figure 7.4. After reducing metal wire length the uncertainty window increases on the left hand side and uncertainty window remains same as previous one on right hand side of the driver cell this would give more symmetry to the routing of the clock binary tree structure.

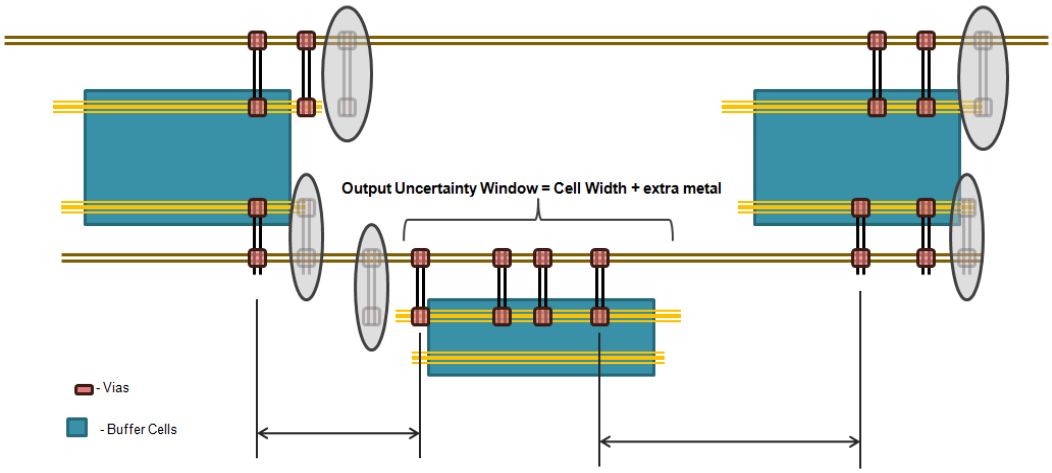


Figure 7.4: Shorten wire-length on Horizontal Tree routing

Though this experiment gives more symmetry structure but it's not worth to use this concept as no of vertical connections are reduce as shown in figure 7.4. This would reduce current flowing from the drivers and hence rise time of cell degrades. Though we get some skew benefits out of this experiment. We would not use this experiment as a solution to reduce uncertainty window and skew.

Chapter 8

Experiment on Transition and Power optimization

8.1 Concept of Transition improvement in Clock binary tree

All the cells have curve on input Vs. Output transition. Output transitions are always good compared to input transition for all the clock cells. One such example is depicted in figure 8.1. Here clock signal is applied to the input of clock buffer having poor transition as shown in figure, on the contrary clock buffers improves transition of the same signal based on input vs. output curves. These curves are predefined in libraries used in design.

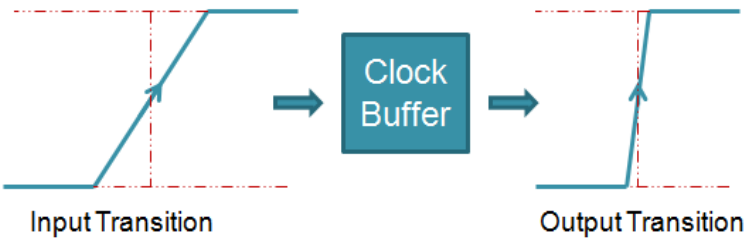


Figure 8.1: Input Output Transition comparison

One such example of clock buffer with capacitive load is shown in figure 8.2. As load increases output transition degrades compared to figure 8.1. Output transition of clock buffer shown in figure 8.2 is poor. Hence driving capability of clock buffer connected with capacitive load is lower to make output transition more powerful, the size of buffer (driving strength of buffer) must be higher.

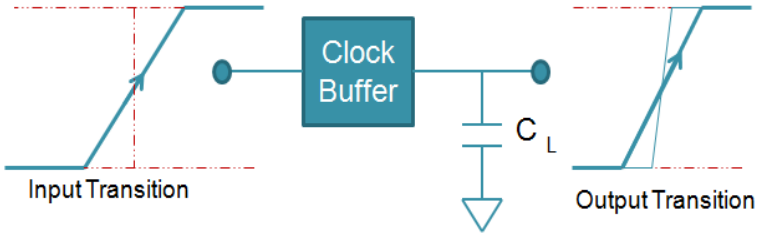


Figure 8.2: Input Output Transition with load connected

As Depicted in figure 8.3(b) lower size of buffer gives poor transition to the poor input transition given to input of clock buffer. To improve such cases switching clock buffer driving strength to upper level will help. As given in figure 8.3(a) improves output transition of a buffer. Hence it can be said that transition numbers are dependent on driving strength of clock buffer.

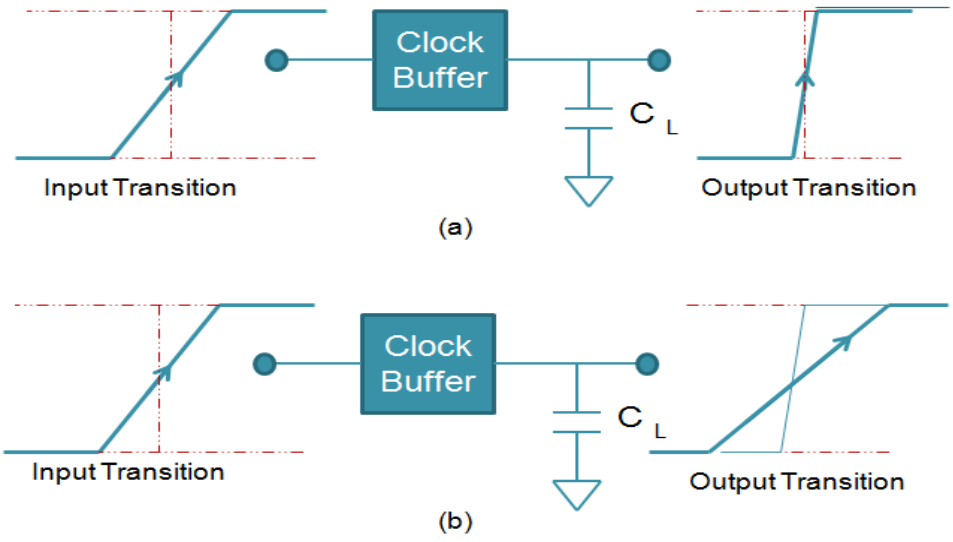


Figure 8.3: Input Output Transition of clock buffer of (a)High Drive Strength buffer (b)Low Drive Strength buffer

Clock binary tree is a meshed structure of cells where number of cells connected are dependent upon the structure used to design that particular clock tree. Going from first stage to the last stage, number of cells/buffers used in tree increases, number of buffers are dependent on which structure is used. Hence, input capacitance of the cell comes into account and net capacitance of the interconnection between stages also needs to be calculated. Such one case is depicted in figure 8.4. Where input Transitions and Output Transitions are shown across the interconnection of cells. For understanding point of view this three cells topology is used in figure since it follows binary tree concept. Input transition is bit slant and first clock buffer makes it steep due to path load and input capacitance of cell load. Hence, input transition of previous stage becomes poor. This concept goes on till the last stage where RCBs are connected to clock buffer cells as load.

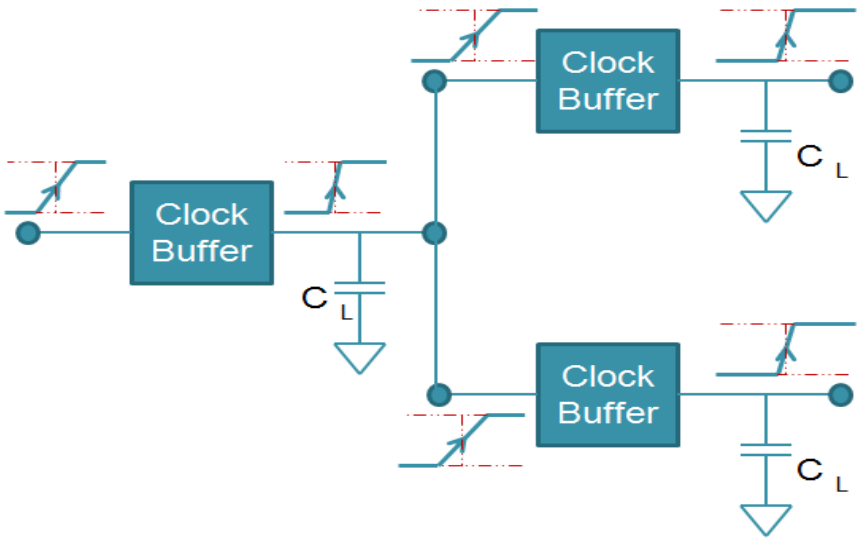


Figure 8.4: Input Output Transition of buffers connected to buffers

Capacitance load offered to clock buffers are as shown in figure 8.5. Where net capacitance is denoted by $\mathbf{C}_{net}$ and input pin capacitance of clock buffer is denoted by $\mathbf{C}_{i/p}$. To model the driving strength of tree structure all the capacitance values are extracted to calculate actual capacitance called as total capacitance value, this is denoted by $\mathbf{C}_L$ in figure. Calculation of total capacitance is as follow.

(1) $\mathbf{C}_{net}$ is extracted from the design, for all the stages which are different.

(2) $\mathbf{C}_{i/p}$ is predefined in the library files based on what structure of tree is used in particular stage number of cells are being multiplied with input pin cap value to find out total $\mathbf{C}_{i/p}$. In figure number of cells connected to buffer is 2 hence $\mathbf{C}_{i/p}$ is multiplied by **2**. These two capacitance values now added together to get total load capacitance $\mathbf{C}_L$.

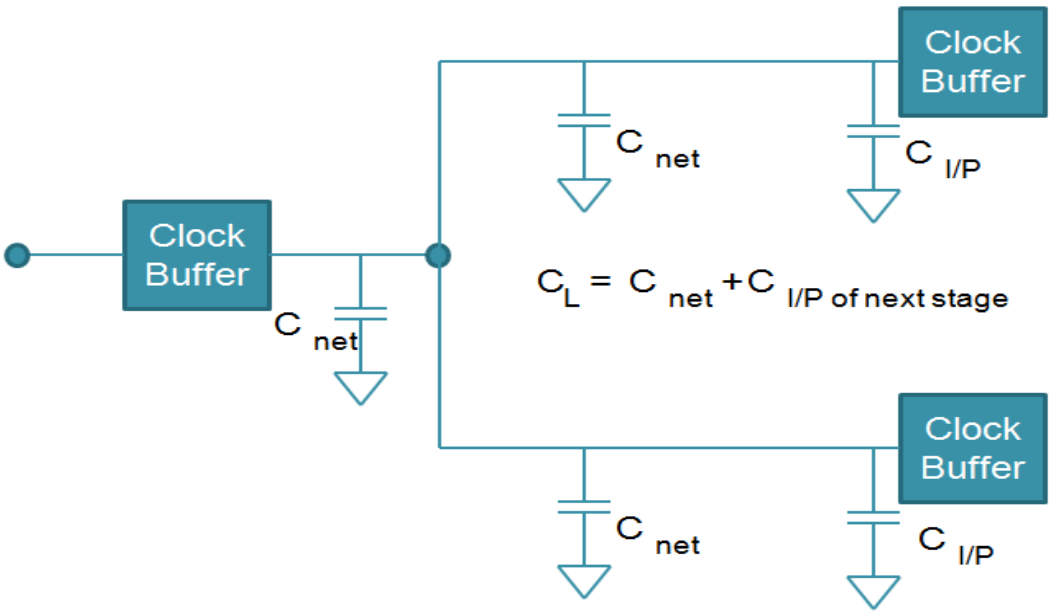


Figure 8.5: Capacitance measurement of clock binary tree

For particular stage

$$\mathbf{C}_L = \mathbf{Total}(\mathbf{C}_{net}) + (\mathbf{No.\ of\ Cells\ in\ next\ stage})^*(\mathbf{C}_{i/p}) \quad \text{——————} \quad \text{Eq6}$$

Eq6 is used to calculate total load capacitance offered to existing stage cell/cells. Every stage cells has limit of capacitance that it can drive without affecting quality of signal. This value is called **Cell-MAX-CAP** value, Max_cap value is again multiplied by number of cells present in specific stage to get total Max_cap value that can be driven by particular stage. To decide the driving strength of that particular stage

these values are important. Theoretically Max_cap value should be greater than Total(C_L) for specific stage but practically there should be some optimistical guard band to mitigate practical differences.

8.2 Concept of Power for Clock binary tree

Special cells are used in clock binary tree, with range of driving strengths. Hence, selecting a proper one is crucial factor. As described in **Section 8.1** to improve transition switching from lower driving capability to higher is necessary if output transitions are poor. But, over using driving capability unnecessarily increases capacitance value offered as load, hence increase in dynamic power. By experiment on selecting different buffers it is found that the trade-off between selecting buffer size and power requirement. Optimally selecting buffer size is always helping to improve power numbers and Transitions of clock binary tree.

Chapter 9

Conclusion

These Horizontal and Vertical Clock binary Tree generator algorithms brought automated process for finding all possibilities/Stuctures of valid possible clock Binary tree. Prior to this algorithm development manual heuristics were used to find out actual structure and cell location to get expected output, this algorithm could explore such configuration of cell trees which was never simulated. In nutshell designing algorithm could help the actual process by finding X and Y location and finding all the possibilities of binary tree structures automatically.

To optimize skew numbers on various clock binary tree is experimented by exercising different experiments, from these experiment the flow of process to get optimum skew in specific clock binary tree structure is achieved. Transition numbers of each clock binary tree is extracted from design based on capacitive load offered to each stage.

Based on comparison of results $2^{(stage-1)}$ tree structure is proven to be more efficient structure for giving good transition number at each stage and skew numbers. This Tree structure also has the advantage of removing metal between buffers, could help to improve skew numbers.

Chapter 10

Acronyms

Acronyms	Descriptions
RCB	Regional Clock Buffer
FF	Flip Flop
FUB	Functional Unit Block
HDL	Hardware Descriptive Language
ASIC	Application specific integrated circuit
RTL	Register transfer language
IP	Intellectual Property
GLS	Gate Level simulation
SV	System validation
MAS	Micro architectural specification
FP	Floor plan
CTS	Clock tree synthesis
FV	Formal verification
TO	Tapeout
POP	Product overview proposal
ECO	Engineering change order
TCL	Tool command language