Geometric Algorithm Scheduling using High Level Synthesis

Zafirul Izzat Bin Mohamad Zaidi
Hochschule Hamm-Lippstadt
Lippstadt, Germany
zafirul-izzat-bin.mohamad-zaidi@stud.hshl

***Abstract*—Scheduling is critical, particularly in architectural synthesis. Whereas a sequencing graph specifies simply the relationships between processes, a sequencing graph's scheduling specifies the specific start time of each activity. Because every set of actions associated by a sequency dependence cannot run concurrently, the start timings must meet the initial dependencies of the sequencing graph, which restrict the level of parallelism of the operations. Scheduling impacts the performance of the eventual implementation by determining its concurrency. The reader will first introduce with terminology and background of high level synthesis. After acquiring a common understanding, this article will help the reader by describing the model from scratch. Specific term that will also explain. Following that, this study will get into type of scheduling to have some knowledge on where Geometric Scheduling Algorithm located and play a role. Then, the reader may know the problem derive and how Geometric Scheduling Algorithm tackle the problem. The conclusion mark as end of the article.**

***Index Terms*—High level synthesis, Geometric Scheduling Algorithm**

I. Introduction

Nowadays, we realize that technology has evolved to the point where designing digital systems from the transistor or logic level would be incredibly difficult. Design automation on more abstract levels, where functionality and trade offs may be explicitly articulated, has become increasingly important. This resulted in the creation of CAD algorithms that could more fully search the design space and discover nearly ideal solutions. As a result, automation of the design process from concept to silicon became increasingly vital and required. Designers used a top-down process in which they described the design's goal and then allowed CAD tools construct specific physical structure to it. This approach of synthesizing systems from a design description grew more suitable for complex system design. At this point in time, using high-level synthesis (HLS), chips and electronic systems are becoming less expensive and time demanding than being totally hand created by a group of designers. Hardware may be created at a variety of abstraction layers. The most common levels of abstraction are gate level, register-transfer level (RTL), and algorithmic level [1]. The goal of HLS is to assist hardware designers build and test hardware more efficiently by providing them more control over design architecture optimization. Furthermore, it will allow the designer to specify the design at a greater degree of abstraction. To put it simply, high-level synthesis (HLS) is the process of transforming a high-abstraction-level description of a design to a register-transfer-level (RTL) description for use in standard ASIC and FPGA implementation processes [2].

In this article, I will discuss the Geometric Scheduling Algorithm in detail. Before proceeding to the main issue, the reader may discover an intriguing portion to have a thorough grasp of the Geometric Scheduling Algorithm. The first part discusses about high level synthesis in general. Next, the reader may have some knowledge about the level of modelling circuit. Every notation and graph will be thoroughly explain. After that, a few types of scheduling will be present in this paper. Furthermore, the reader will be exposed to the challenge and Geometric Scheduling Algorithm dealt with it.

II. Model

Computer-aided design tools provide an efficient way to create commercially feasible microelectronic circuits. With the help of computer aided, it can help synthesis approaches shorten the design cycle and decrease the amount of human work indirectly. Other than that, the productivity, correctness and re-targetability increase. Design quality is improved via optimization approaches to make it more user friendly. For most digital circuit designs nowadays, synthesis and optimization methods are employed. Before moving on deeper, we will now look at the circuit models and perspectives briefly as they divided into three parts which are circuit models, synthesis, and optimization [3].

A. Circuit models

The term circuit model refers to an abstraction or representation that reveals key aspects but not the accompanying details. Synthesis is the process of creating a circuit model from scratch or not detail. Models are characterized according to their levels of abstraction and viewpoints. We will look at three major abstractions here, the first one is architecture, next is logic, and last geometrical. The levels can be represented and explain as follows. A circuit executes a series of operations at the architectural level, such as data computation or transport. A digital circuit assesses a collection of logic functions at the logic level. A circuit is a collection of geometrical things at the geometrical level. HDL models and flow diagrams are examples of architecture model representations, state transition

University of applied Science Hamm-Lippstadt

diagrams and flow diagrams are examples of logic model representations [3]. “Fig. 1” are one of the example.

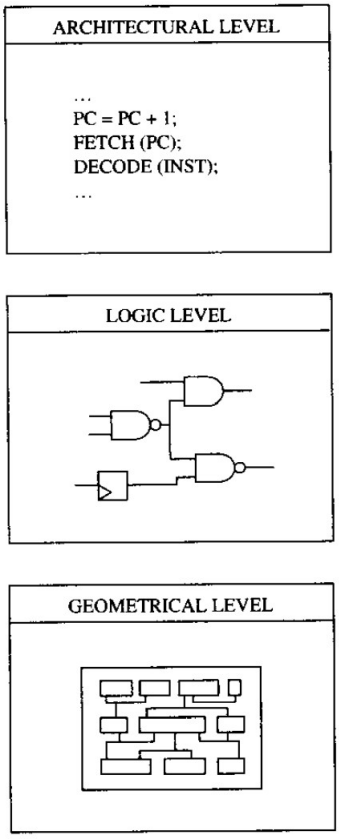

Fig. 1. Abstraction level of circuits representation.

B. Synthesis

The model categorization is related to the synthesis task taxonomy. The meaning behind synthesis may be thought of as a collection of transitions between two axial viewpoints, so we may separate the synthesis subtasks at the various modelling levels to help the readers differentiate the levels. The level of synthesis divided into three parts as follows in “Fig. 2”.

1) Architectural-level synthesis

   For the first level, it consists of generating a structural view of an architectural-level model. The structural view is a method to the study of language that concentrates or focuses on its underlying structure as opposed to the history of its development or its relationships with other languages. This will aid in the subsequent implementation of the circuit function assignment. The assignment results aid in the selection of suitable operators, which are referred to as resources. The interconnection and timing of their execution are other factors to consider. It is also known as high-level synthesis or structural synthesis since it determines the macroscopic structure of the circuit, for example, the block level structure. To eliminate uncertainty and preserve consistency, we describe it as architectural synthesis [3].

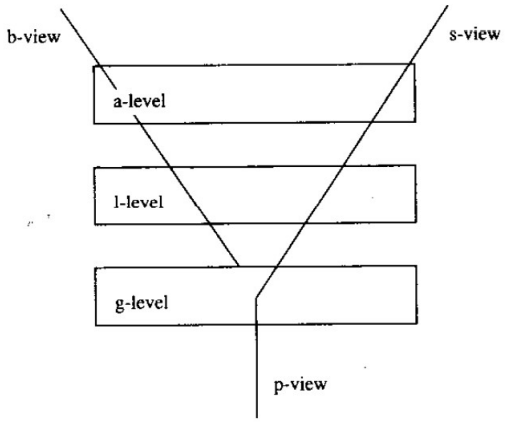

Fig. 2. Level of abstraction.

2) Logic-level synthesis

   The work of developing a structural perspective of a logic-level model lies on the second level. The manipulation of logic requirements to generate logic models as an interconnection of logic primitives is known as logic synthesis. Thus, logic synthesis defines the microscopic structure of a circuit, such as the gate level structure. The process of converting a logic model into an interconnection of instances of library cells, often known as the hack end of logic synthesis and define as library binding or technology mapping [3].

3) Geometrical-level synthesis

   We have now reached the final level, geometrical level. This synthesis essentially implies establishing a physical perspective at the geometric level. It includes the definition of all geometric patterns determining the chip’s physical layout, as well as their location. It is commonly referred to as physical design, and we will refer to it as such in the sequel. The logic synthesis duties may change depending on the nature of the circuit and the desired consequence of the designer. It can be designed in a sequential or combinational manner. Things are called sequential or logical order, if they follow a predetermined sequence, according to the meaning of sequential. A three-part process in which the stages must be completed in a certain logical order is an example of sequential process steps. Normally the first step to build a perfect circuit is by representation of diagrams as example state diagram [3]. “Fig. 3” is one of the example at logic level.

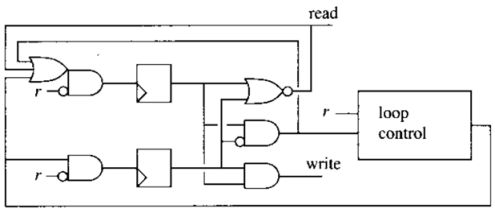

Fig. 3. Structural view at logic level.

C. Optimization

The area computation can be performed hierarchically. Usually, the fundamental components of digital circuits are logic gates and registers, whose area is known a priori. Circuit optimization also play big role. It is frequently used in combination with synthesis. The goal of optimization is to optimize both the quality and efficiency of the circuit. Without optimization, synthesis would produce non-competitive circuits, and hence its value would be limited [3].

III. Fundamental background

The next section is about fundamental background. The goal of this chapter is to give a hand to the reader by going through some basic concepts that will be use later. So, it might help for a deeper understanding specially the term used in the problems, and algorithms. The context is crucial for comprehending the information in the remainder of this article. Some notations and graphics will be explained further down [3].

A. Notation

This article will now cover a few notations that may be useful. When there is a collection of elements, it is referred to as a set. The capital letter denotes sets, whereas the lowercase letter denotes elements. Ordered sets and unordered sets are the two types of sets. For example, in an ordered set, parentheses are used to signify $[i.e., ()]$, but in an unordered set, braces are used instead of parentheses. As an example, $(i.e.,)$. Next, there is a significance behind each symbol. $\Longrightarrow$ denotes implication, $\Longleftrightarrow$ denotes co-implication, ":" denotes such that, and so on. When two sets, such as X and Y, are connected to one other by having the same property, it is termed a map or function. The function of two sets, X and Y, is denoted as $f : X \rightarrow Y$. X is a domain function in this situation, while Y is a co-domain function. There are several kinds of functions. One-to-one, many-to-one, one-to-many, and many-to-many are all possibilities.

B. Graphs

Graphs come in a variety of shapes and sizes. The reader may discover some information on undirected graphs and directed graphs in this article. Before we get to the graphs, let us just discuss about graphs in general. As an example, consider graph $G(V, E)$. (V, E) is a pair in which V is a set and E is a binary relation. V and E sets each have their own collection of elements. The elements of set V are referred to as vertices, whereas the members of set E are referred to as graph edges.

- Directed graph

  The name directed graph comes from the fact that the edges are in ordered pairs of vertices and are not random. We may claim that directed graph is a simple way. The idea behind a directed graph is head and tail. As an example (V_i, V_j). Vertex V_i is known as the tail, whereas vertex V_j is known as the edge's head.

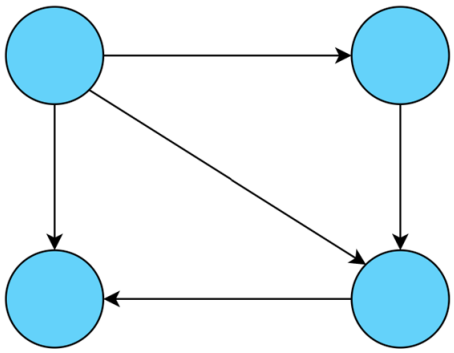

Fig. 4. Directed graph.

- Undirected graph

  The edges are in an unordered pair, which is denoted by V_i,V_j. When two vertex edges are linked, the vertex is said to be next to one another. If the edges are joined into two identical end points, the vertex is called looped. When a graph contains no loops, it is referred to be a simple graph. Otherwise, it is referred to as a multi graph.

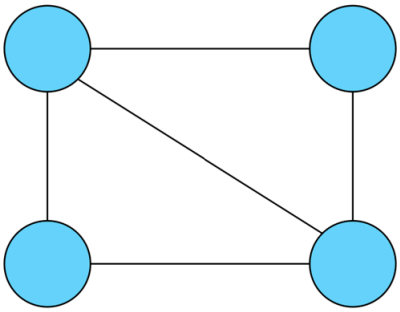

Fig. 5. Undirected graph.

IV. Scheduling algorithm

Scheduling has long been seen as a critical stage in the high-levels Synthesis process. In the literature, there is a large range of methods for efficiently executing the task of scheduling in high-level synthesis. Scheduling is a critical issue in architectural synthesis. Whereas a sequencing graph specifies

simply the relationships between processes, the scheduling of a sequencing graph specifies the exact start time of each activity. Scheduling impacts the performance of the eventual implementation by determining its concurrency. There are a lot of scheduling algorithms that already implemented nowadays. Those scheduling depends on the requirements of the circuit itself. On this paper, five main scheduling algorithms that the reader can take into a consideration. "Fig. 6" show the connection of the algorithms.

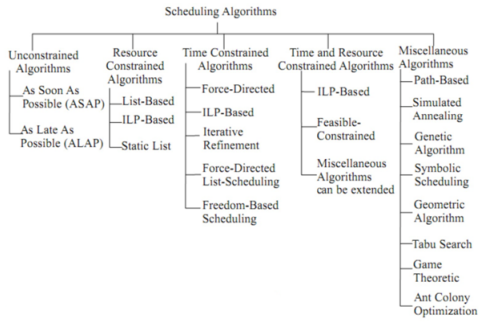

Fig. 6. Type of scheduling algorithm. [4]

A. Unconstrained algorithm

When dedicated resources are employed, unconstrained scheduling is frequently used. One scenario that leads to dedicated resources is when operations are different in kind or have a low cost when compared to steering logic, registers, wiring, and control. Aside from that, unconstrained scheduling may be utilized to determine latency constraints for restricted situations. For example, owing to a resource limitation that prevents the computation of minimal latency, the minimum latency of the schedule. The bottom bound on latency is easily found using unconstrained scheduling [3].

B. Resource constrained algorithm

Resource constraints play a big role in term of circuit area used. This is due to the fact that resource utilization dictates the circuit area of resource-dominated circuits and accounts for a major portion of the overall area for most other circuits. The solution of scheduling issues with resources constraints allows for the computation of (area/latency) trade-off points [3].

C. Time constrained algorithm

When we talk about time restrictions, we always refer to the deadlines that must be met. In general, scheduling is done based on the tasks that have been completed and reasonable assumptions about execution delays. The delay is determined after binding. It will compute the maximum route delay by considering all relevant factors and comparing it to the cycle time. If the cycle time constraint is not met, the scheduling and binding processes will be repeated, and the execution delay will become longer and longer until it stops repeating. Otherwise, when the cycle-time constraint is met by a significant margin, the iterative procedure can be stopped [3].

D. Miscellaneous algorithm

Since the main topic of this paper is about Geometric Algorithm Scheduling, the reader should know the main component of scheduling algorithm. As seen in figure Geometric Algorithm Scheduling is under Miscellaneous Algorithm. The main concept of miscellaneous Algorithm is that the combination of odd bunch of things. Based on the research, Miscellaneous Algorithm is an algorithm that combine other properties not only the algorithm itself that it might not expected to go together but still produce the expected result. To have a deeper understanding, the reader will take a look on how Geometric Scheduling Algorithm works to solve the problem arise and the meaning of combination of odd bunch things.

V. Problem

After get a knowledge on how to model a circuit, then jump into type of scheduling algorithm and now we will discuss about the problem arise and what is the approach that we take with the help of geometric scheduling. To assist the reader with the problem, arise, in this paper, the reader may find two types of constraint which are resource constraints and timing constraints. The meaning behind resource constraints is the usage of resource used based on the circuit area. In general, the appropriate way to solve this kind of constraints is to implement reasonable size that affected the area and the latency of the circuit. Next is scheduling under timing constraints. It can be the case of the operations that need to meet the deadline. So, whatever the operation, it must produce the output based on the given time. In this paper, we have two main problems involved which must be dealt with separately

The main problem is the minimization of resource with timing constraints. The total number of $Cstep$ represent timing constraints in this case. So, the Control Data Flow Graph has to be scheduled reducing number of constraints used. Basically, Control Data Flow Graph is a notation that shows all possible path during its execution [5].

- Problem A1

  The number of $Csteps\ C = max$, latency(opi), where opi is the operation in path p. In this case, the assumption is for all operation, the latency is 1. So, if the latency of all operation is 1, it can be assumed that the value would be same as the longest path. One of the solutions to this problem is that it might possible to schedule with Control Data Flow Graph and the output is the fastest [5].
- Problem A2

  The number of $Csteps$ allowed is more than the value of C in problem A1. This problem is solved depends to Problem A1. If the solution of Problem A1 produces an unexpected and unacceptable result [5].

VI. Algorithm

After knowing about the problem, the reader might look at the algorithm to have a deeper understanding about the problem. In “Fig. 7” , the conversion from Control Date Flow Graph is by breaking a loop but the condition is maintained. As a consequence, it can be converted into Directed Acyclic Graphs. It can be seen that the condition of the node [4,2] are stored in the Directed Acyclic Graph.

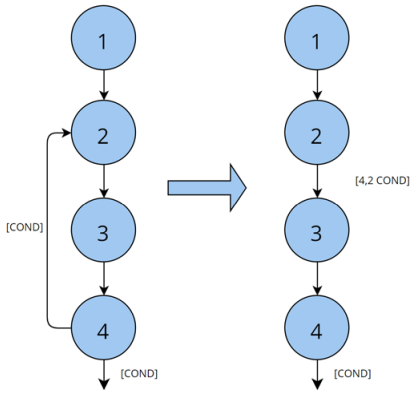

Fig. 7. CDFG converted to DAG.

Now, after the conversion into Directed Acyclic Graphs, the next step involves on how to prioritize the path. Generally, the path who has the longest path is set to be the highest priority. Due to the value of the latency which is 1, the sum of the total latency of the node provide same value of the length of the total path. The longest path will be put in a queue first since it has the highest priority. For example in “Fig. 8”, path P will be execute first. If the path, has the same length, the path with the most common nodes will be put as a higher priority compared to the path that does not consist any common or same node as the previous path that has the highest priority. So, in this case, path Q will be executed after path P is execute then followed by path R. This is because, path Q has common node with path P which are 3 and 4. The longest path criteria in essence is the incorporation of the LIST kind of priorities for the operations. The commonness criteria for priority for the path helps schedule the operations with least freedom first; the fact is made clear by. According to the commonness principle, the next priority will goes to path Q. This is because it share the same resources with the highest priority which is path P. This can be seen in “Fig. 9” . As mention in problem A1, it has been scheduled in 4 $Csteps$ which is the total number of $Cstep$ is equal to the latency of operation path P. For the problem A2, the number of $Csteps$ is more than the value of C in problem A1. As example, the reader can refer to the figure shown in “Fig. 11” that consists of 5 $Csteps$. It can be observed that path R is wrongly chosen and likely to execute first instead of path Q that has a higher priority. For additional information, the operation in both “Fig. 10” and “Fig. 11” were scheduled in the earliest available Cstep which mean for every free slot it will be maximize with the path that have same operation [5].

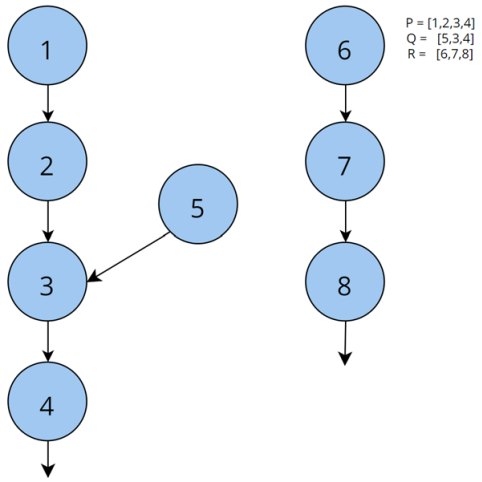

Fig. 8. Prioritizing using commonness principle.

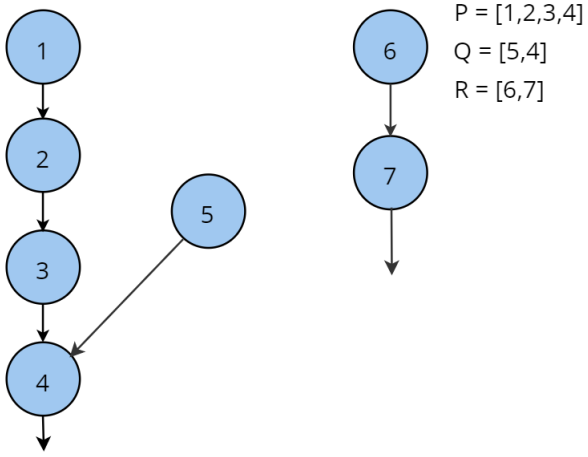

Fig. 9. Freedom criteria of CDFG.

VII. Approach

There are two approach that have been taken for both problem A1 and A2.

- Problem A1

  As mentioned before, the number of $Csteps$ is fixed and is equal to the length of the longest path. Then, based on the priority, the path will be queue accordingly starting from the highest priority and end with the lowest priority. It will be scheduled progressively. If there is no matching for the path to use any kind of operation, the Resources are added every single time to make it run and execute smoothly based on the expected output. Below is the heuristic or approach to problem solving or self-discovery that might be helpful in short term goals [5].

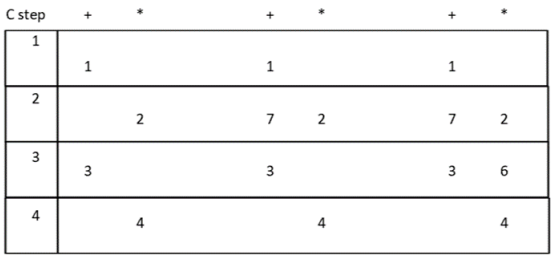

Fig. 10. Commonness principle.

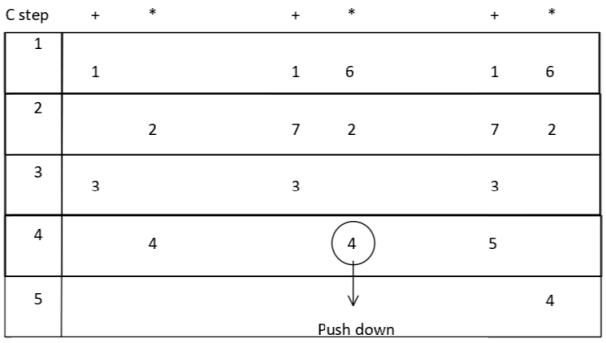

Fig. 11. Neglecting commonness principle.

- Heuristic 1 : To increase the performance, the operations are scheduled as early as possible. AFAP (as fast as possible) is use to scheduling each path [4].
- Heuristic 2 : The number of a resource will keep escalate based on the demand. This is because it can easily override any other considerations.
- Heuristic 3 : To learn the ropes the efficiency of the scheduling, it is better to avoid any mismatch in latency of operational units. So, for every operation, it will assign to the earliest possible column. As seen in "Fig. 12" operation M4(*) could be put either in space X or in y. In order to the statement above, it is better to place at X space. This will allows more freedom for the paths scheduled later. Thus the algorithm also looks ahead and does not give only a local minima.

- Problem A2

  This problem can be solved by gradually increasing the number of $Csteps$ permitted in Problem A1. The number of $Csteps$ will be increase one by one until an acceptable outcome for the number of resources required. The problem of path scheduling may be viewed as a "geometric" matching problem termed point dominance, which will be discussed in the next section.

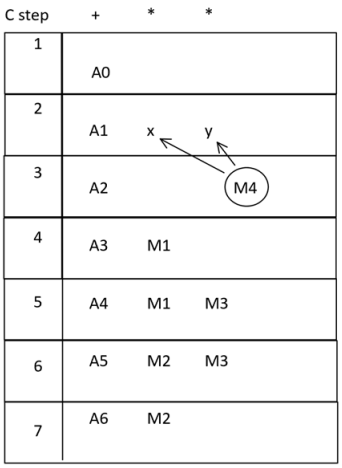

Fig. 12. Look ahead scheduling.

VIII. Geometric Mapping

We consider the problem of path scheduling. It is possible to acquire sets of OP and sets of Cs. OP sets indicate the many operations along a path, while Cs points represent the $Csteps$. Due to the dependence of the operations to be matched on the operations in the partial schedule accessible to us, we have certain limits on the matching based on the problem emerge. We can see that the number of resources remains constant, indicating that there is no drop or growth in the amount of resources. Dependencies are often expressed as a precedence relationship. Given these limits, we have a set of options for matching each operation [5].

"Fig. 13" shows how operation and $Csteps$ could be match with the help of geometric mapping. The operation OpA is mapped to $Cstep$ $C1$ or $C2$. For the next operation which is operation OpB, it is mapped to $Cstep$ $C1$, $C3$ and $C5$. Then operation OpC is mapped to $Cstep$ $C4$ and lastly operation OpD is mapped to $Cstep$ 5 based on the figure. Based on the

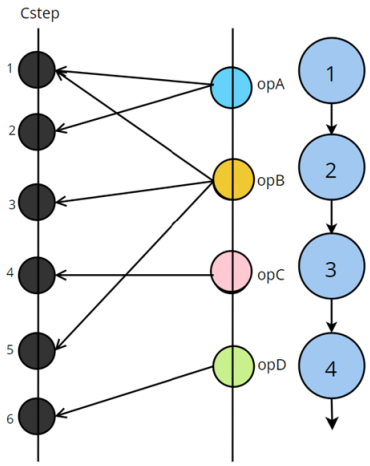

Fig. 13. Geometric mappig.

geometric mapping, We can schedule by looking at the limit of resource used but still maintains its efficiency of the output. First we have to convert from geometric to X-Y plane. Note that every point in the plane will represent a possible matching. Lets suppose there exists a possibility of operation Opi to be matched on to Cj. This will represented in the plane by a point x, y [5].

$$X := \text{operation preceding to } Opi \text{ in the path;}$$

$$Y := \text{j;}$$

Thus for "Fig. 13" we convert from geometrical mapping to obtain graph using max chain shown in "Fig. 14".

Now we define the concept of dominance in the plane for our purposes as follows. A point p is said to dominate a point q iff $X(p) > X(q)$ and $Y(p) > Y(q)$ and $p \neq q$, where $X(p)$ and $Y(p)$ respectively denote the x and y co-ordinates of point p. For example, in figure, OpA has two $Csteps$ (1,2). Both will be the starting points. For OpB, it has (1,3,5). $Cstep$ 1 is not selected since it is not follow the rule of point dominance. For opB, $Csteps$ (3,5) will be selected. Then for OpC, it has only one $Cstep$ (4). Due to the rules, $Cstep$ (5) for OpB will be not selected since it is higher that OpC. A dominance chain is described as a series of points with increasing co-ordinates, where each point dominates the previous one. The problem is now reduced to producing a max-weighted k point chain problem, where k is the number of nodes in the path.

Theorem: A maximum weighted k chain can be obtained in $O(klogk)$.

Description of the max-k-chain solution, proofs and calculation of weights on the edges are omitted for brevity

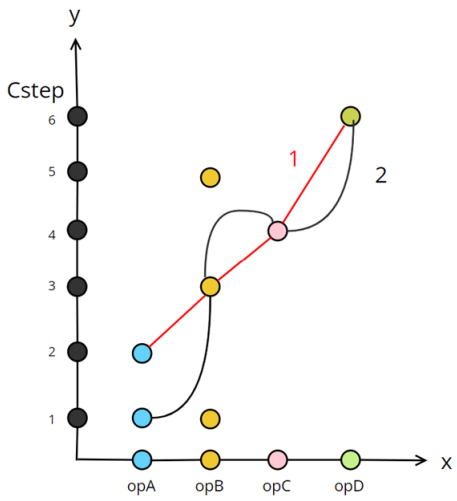

Fig. 14. Matching using max chain.

For the problem in "Fig. 13" we obtain two solution of the chain.

- $OpA \rightarrow C1/C2$
- $OpB \rightarrow C3$
- $OpC \rightarrow C4$
- $OpD \rightarrow C6$

We select the chain with maximum weight wheres it will follow the increasing of $Cstep$ based on the order of the operation [5].

IX. Conclusion

Due to the nature of the matching technique utilized, we know that the approach is time efficient. According to my study, the Geometric Scheduling Algorithm is not well-known. Based on the results of this study, I can infer that with the help of the Geometric Scheduling Algorithm, it is possible to optimize the efficiency of the circuit based on time and resource constraints. Geometric mapping also aids the designer in creating a clear visualization.

X. Acknowledgment

I am eternally thankful to my dear professor, Prof. Dr. Rettberg, Achim whose inspiration, encouragement, advice, and support from the beginning to the end helped me to build awareness and open my eyes to the importance of scheduling in real-time systems in general. I would also want to offer my appreciation, respect, consideration, and advantages to any and all persons that assisted me in any way throughout the execution of the task. I did everything in my power to gain a better knowledge and share my understanding in this paper and I hope the reader can increase their knowledge specially in this topic.

References

[1] S. Govindarajan, "Scheduling algorithms for high-level synthesis," Digital design environments, Mar. 1995.

[2] Coussy, P., Gajski, D., Meredith, M. and Takach, A., 2009. An Introduction to High-Level Synthesis. IEEE Design amp; Test of Computers, 26(4), pp.8-17.

[3] G. D. Micheli, Synthesis and optimization of digital circuits. New York, NY, State of New York: McGraw-Hill, 1994.

[4] Mohanty, S., 2008. Low-power high-level synthesis for nanoscale CMOS circuits. New York: Springer.

[5] S. Raje and M. Sarrafzadeh, "GEM: A geometric algorithm for scheduling," 1993 IEEE International Symposium on Circuits and Systems, 1993, pp. 1991-1994 vol.3, doi: 10.1109/ISCAS.1993.394143.