19L720 – PROJECT WORK I

COLLISION DETECTION FOR MACHINE TOOL AUTOMATION

SHEENA S – 20L141

TARUNISREE A V – 20L149

NATHEESH KUMAR B –21L410

SENTHILNATHAN B – 21L417

Project Report (Phase 1) submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF ENGINEERING

Branch: ELECTRONICS AND COMMUNICATION ENGINEERING

of Anna University

OCTOBER 2023

ELECTRONICS AND COMMUNICATION ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

CONTENTS

ACKNOWLEDGEMENT

We sincerely owe our gratitude to **Dr. K. Prakasan**, Principal, PSG College of Technology, Coimbatore, for providing us with the best teachers and facilities. It is our privilege to place on record our sincere thanks to **Dr. V. Krishnaveni**, Professor and Head of the Department, Department of ECE, PSG College of Technology, Coimbatore, for her support and guidance.

We would like to place our everlasting gratitude to our Programme Coordinator **Dr. K. V. Anusuya**, Associate Professor (CAS), Department of ECE for her excellent advice, guidance and continued assistance throughout the course of this project.

We would like to express our sincere gratitude to our project guide and tutor, **Dr. S. Hema Chitra**, Associate Professor (CAS), Department of ECE, for her constant motivation, direction and guidance through each step of the project.

We kindly acknowledge the timely help and valuable suggestions of our Professors, PG scholars and non-teaching staff at PSG College of Technology, Coimbatore. Our sincere and heartfelt thanks to our classmates for their moral support and encouragement to complete this work.

ABSTRACT

Collision detection of two objects plays an essential role for the machine tool automation. Although the collision detection of two objects has been studied in applications like virtual reality, the collision detection for the machine tool requires high precision to avoid overcut damage to the high-cost machine tools. The current collision detection for machine tools is under the soft-ware based computer numerical control (CNC), the low computation capability of which refrains the CNC based approach from real-time collision detection. To enhance the collision detection for machine tool automation, the Separating Axis Theorem (SAT) based algorithm is designed and implemented using Verilog language.

The bounded objects are represented by meshed triangles using the STL format and hence the separating axis theorem (SAT) based detection algorithm is considered. Furthermore, by considering high precision required by machine tool applications, the SAT algorithm includes collision detection of either non-coplanar or coplanar triangles. The Collision Detection Algorithm is developed using Verilog HDL and then implemented using FPGA board. For two objects represented by meshed triangles, this algorithm can provide collision detection in real-time as compared to the conventional CNC based approach for collision detection of two objects.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Collision detection of two objects plays an essential role for the machine tool automation. Any unintended physical contact or interference between the moving components of the machine or its tooling system is the collision. Five-axis CNC machine tools have been more popular in machining area because of their ability to machine parts with complex geometries efficiently and achieve higher dimensional accuracy. five-axis CNC machines are widely used in machining of sculptured surfaces such as turbine blades, impeller, aerospace parts, molds, and dies. These collisions may occur between the tool and the workpiece, fixtures, between the workpiece and machine components, or between moving machine components. The current collision detection for machine tools is under the soft-ware based computer numerical control (CNC), the low computation capability of which refrains the CNC based approach from real-time collision detection is less effective. To overcome this problem and to enhance the collision detection for machine tool automation, SAT algorithm is considered.

1.1 MACHINE TOOL AUTOMATION

India a leading manufacturer of machine tools, ranked 17th in production and 12th in the consumption of machine tools in the world. Machine tool automation refers to the integration of various technologies and systems to automate the operation of machine tools used in manufacturing and machining processes. Machine tools are devices that are capable of cutting, shaping, and forming materials like metal, wood, plastic, and more with a high degree of precision. These tools are essential in a wide range of industries, including automotive, aerospace, electronics, and general manufacturing. Hence, Machine tool automation is very important and crucial to enhance the value and the efficiency of machine tools.

1.2 CNC MACHINES

Computer Numerical Control (CNC) machines have revolutionized the manufacturing industry by automating and enhancing the precision of various machining processes. CNC machines are a critical component of modern manufacturing, enabling the efficient production of complex and precise parts for a wide range of industries, including aerospace, automotive, medical, and more. These machines can perform tasks such as cutting, drilling, milling, turning, and additive manufacturing (3D printing) with a high degree of accuracy and repeatability. It operates based on a set of instructions known as G-code or CNC code. These codes contain specific commands that dictate the tool's movements, speeds, and tool changes. The CNC machine's computer control unit interprets these codes and translates them into precise movements of the tool relative to the workpiece. Thus, CNC machines are essential tools in modern manufacturing, offering unparalleled precision, efficiency, and versatility. CNC machines offer exceptional accuracy and repeatability.

Efficiency: They reduce material waste and minimize manual labor.

Flexibility: Easily switch between different machining tasks by changing the CNC code.

Automation: Can run unattended, reducing labor costs and increasing productivity.

1.2.1 TYPES OF CNC MACHINES

CNC machines encompass a wide range of types, each tailored to specific manufacturing tasks and industries. One common categorization divide, CNC machines into milling, turning, and specialized machines. Milling CNC machines are characterized by their ability to remove material from a workpiece using rotating cutting tools. These machines can be further categorized based on the number of axes they possess.

- **3-AXIS CNC MACHINE**

3-axis CNC machine is a fundamental type of computer numerical control machine commonly used in various industries for machining operations. It is characterized by its ability to move the cutting tool along three primary axes: X, Y, and Z.

X-Axis: The X-axis represents the horizontal movement of the cutting tool. It typically moves left and right along the workpiece's horizontal plane.

Y-Axis: The Y-axis represents the vertical movement of the cutting tool. It typically moves forward and backward along the workpiece's vertical plane.

Z-Axis: The Z-axis represents the depth or axial movement of the cutting tool. It moves up and down, allowing the tool to penetrate or retract from the workpiece.

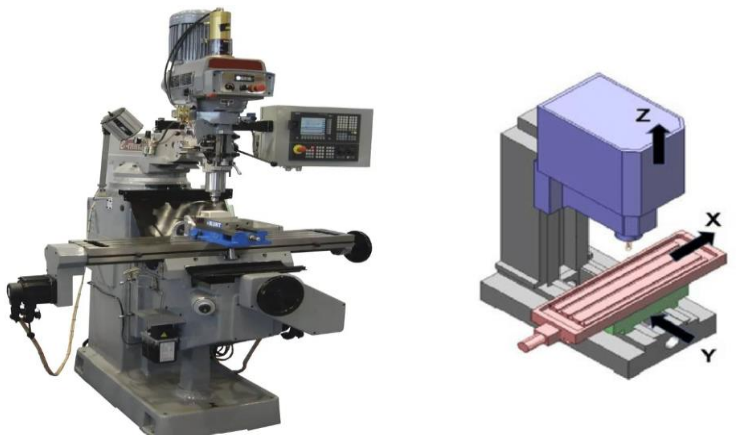


Fig 1.1 3-axis CNC Machine

In a 3-axis CNC machine, the workpiece remains stationary, and the cutting tool moves to different positions to perform various machining operations. The orientation of the workpiece is critical, as the tool can access different areas of the workpiece by moving along the X, Y, and Z axes.

Simplicity: 3-axis CNC machines are relatively straightforward to program and operate compared to more complex multi-axis machines. They are often more affordable and require less maintenance than higher-axis machines. Suitable for a wide range of materials, from metals to plastics and wood. 3-axis machines can achieve high levels of accuracy and repeatability when properly set up and maintained. Some areas of a workpiece may be challenging to access with only three axes, leading to restrictions in certain geometries. Machining complex parts may require multiple setups and tool changes, increasing production time. They are less suitable for machining highly complex or contoured parts that require more axes of movement.

- **4-AXIS CNC MACHINE**

A 4-axis CNC machine is a type of computer numerical control machine equipped with four axes of motion, which enables it to perform more complex machining operations compared to 3-axis machines. In addition to the three primary axes (X, Y, and Z), which control movement in three-dimensional space, a 4-axis CNC machine has an additional rotational axis, often referred to as the A-axis. Here's a detailed explanation of a 4-axis CNC machine.

X-Axis: The X-axis represents the horizontal movement of the cutting tool. It typically moves left and right along the workpiece's horizontal plane.

Y-Axis: The Y-axis represents the vertical movement of the cutting tool. It typically moves forward and backward along the workpiece's vertical plane.

Z-Axis: The Z-axis represents the depth or axial movement of the cutting tool. It moves up and down, allowing the tool to penetrate or retract from the workpiece. These three axes provide the same capabilities as those of a 3-axis CNC machine.

A-Axis: The A-axis is the fourth axis and provides rotational movement. It allows the workpiece or cutting tool to rotate about a specific axis, typically perpendicular to the X-axis. The A-axis rotation can be continuous or indexed, depending on the machine's design and capabilities.

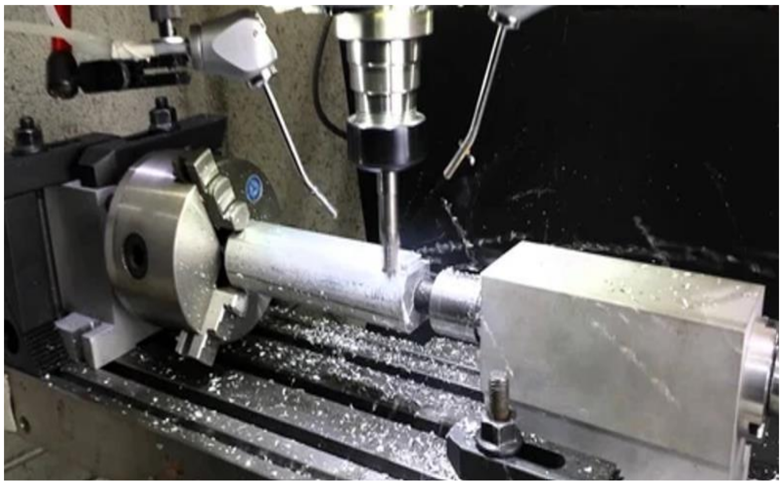


Fig 1.2 4-axis CNC Machine

They are ideal for tasks that require cutting, shaping, or engraving complex contours or features on a workpiece. The A-axis allows for angled cuts and facilitates the machining of complex components, reducing the need for multiple setups. Complex geometries that would require multiple setups on a 3-axis machine can often be completed in a single setup on a 4-axis machine, reducing production time. The ability to machine parts from various angles enhances precision and surface finish quality.

- **5-AXIS CNC MACHINE**

A 5-axis CNC machine is an advanced computer numerical control machine equipped with five axes of motion, offering even greater versatility and precision in machining operations compared to 3-axis or 4-axis machines. In addition to the three primary linear axes (X, Y, and Z) that control movement in three-dimensional space, a 5-axis CNC machine incorporates two rotational axes (typically B and C axes). Here's a detailed explanation of a 5-axis CNC machine.

X-Axis: The X-axis represents the horizontal movement of the cutting tool. It typically moves left and right along the workpiece's horizontal plane.

Y-Axis: The Y-axis represents the vertical movement of the cutting tool. It typically moves forward and backward along the workpiece's vertical plane.

Z-Axis: The Z-axis represents the depth or axial movement of the cutting tool. It moves up and down, allowing the tool to penetrate or retract from the workpiece. These three axes provide the same capabilities as those of a 3-axis CNC machine.

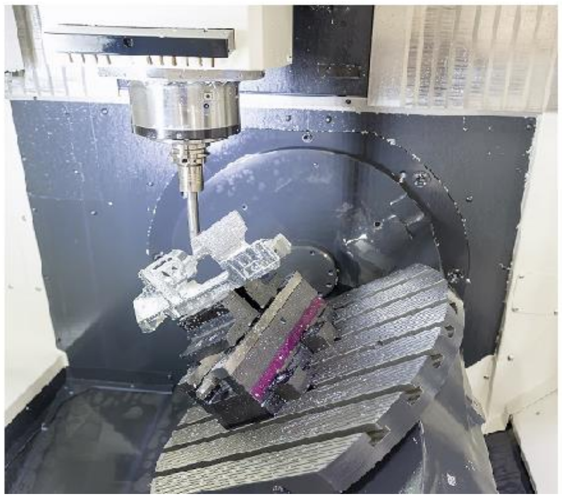
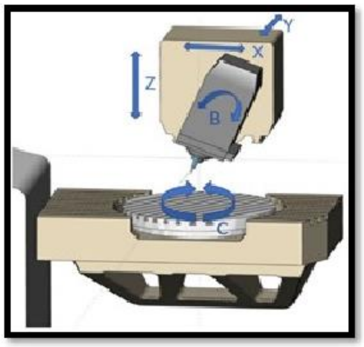



Fig 1.3 5-axis CNC Machine

The B and C axes are the fourth and fifth axes, respectively, and provide rotational movement to the machine. The B-axis typically rotates around an axis parallel to the Y-axis, while the C-axis rotates around an axis parallel to the Z-axis. These rotational axes enable the cutting tool to tilt and swivel, allowing it to approach the workpiece from various angles and orientations.

5-axis CNC machines excel at tasks that demand intricate and complex machining, including multi-sided machining, contouring, and simultaneous machining of complex surfaces. The ability to machine from multiple angles and orientations reduces the need for repositioning the workpiece and multiple setups, resulting in greater efficiency and precision. Complexity: Operating and programming a 5-axis CNC machine can be more complex than machines with fewer axes. 5-axis machines are typically more expensive than their 3-axis or 4-axis counterparts. Due to their complexity, 5-axis machines may require more maintenance.

1.2.2 TOOLS USED

- **DRILL BITS**

CNC drills are specialized cutting tools designed for drilling holes in various materials with precision and efficiency. These drill bits come in a variety of types and sizes, allowing manufacturers to choose the most suitable option for their specific machining needs. Here are some common types of drill bits used in CNC machines.

Twist Drill Bits: Twist drill bits are the most common and versatile type of drill bits used in CNC machining. They have a spiral design that helps to remove material as they penetrate the workpiece. Twist drill bits are available in a wide range of sizes and materials, making them suitable for drilling holes in metals, plastics, and wood.

Center Drill Bits: Center drill bits, also known as spotting drill bits, are used to create a starter hole or pilot hole for accurate drilling. They have a conical shape with a short, rigid body and a 60-degree angle at the tip. Center drill bits are often used in CNC machines to ensure that subsequent drilling operations are precisely centered.

Indexable Drill Bits: Indexable drill bits consist of a replaceable cutting insert (usually carbide) mounted on a drill body. These bits are known for their cost-effectiveness and efficiency, as the inserts can be easily replaced when they become dull. Indexable drill bits are commonly used in high-production CNC environments.

Countersink Drill Bits: Countersink drill bits are used to create countersunk holes that accommodate screws or fasteners with flush or recessed heads. They are essential for creating clean and aesthetically pleasing holes in CNC-machined parts.

- **END MILLS**

End mills are versatile cutting tools with a cylindrical shape and cutting edges on both the end face and the sides. They are used for various milling operations, such as contouring, slotting, and face milling. Variations include ball end mills (for sculpting and 3D profiling) and flat end mills (for flat surface milling).

Ball End Mills: Ball end mills have a rounded end with a ball-shaped cutting edge. They are ideal for 3D contouring, sculpting, and finishing operations, as they produce a smooth surface finish and reduce tool marks.

Tapered End Mills: Tapered end mills have a conical shape and are used for creating angled or tapered features. They are often used in CNC operations that require tapered walls or surfaces.

- **TURNING TOOLS (LATHE TOOLS)**

Turning tools are used in CNC lathes and turning centers for cylindrical part machining. They include various toolholders and inserts for facing, turning, grooving, and threading operations.

Threading Tools: Threading tools are used to create external threads on cylindrical workpieces. These tools can be single-point tools for single-point threading or insert-based tools for multiple-point threading.

Knurling Tools: Knurling tools are used to create decorative or functional knurled patterns on workpiece surfaces for improved grip or aesthetics.

Face Grooving Tools: Face grooving tools are specialized tools used for creating grooves on the face of a workpiece, often for retaining rings or other applications.

1.3 COLLISION DETECTION

Collision detection is crucial in CNC (Computer Numerical Control) machines for several important reasons:

Safety: The most fundamental reason for collision detection is safety. CNC machines often involve high-speed cutting tools and heavy workpieces. Detecting collisions can prevent accidents that might result in injury to operators and damage to the machine.

Tool and Workpiece Protection: Collision detection helps safeguard expensive cutting tools and workpieces. A collision can damage the tool, the workpiece, or both, leading to costly replacements and rework.

Machine Protection: CNC machines are significant investments, and collisions can cause severe damage to the machine's mechanical components, such as the spindle, guide rails, or ball screws. Detecting and preventing collisions helps prolong the machine's lifespan.

Accuracy and Quality: CNC machines are used for precision machining. Collisions can disrupt the accuracy of the machining process, leading to suboptimal part quality. By detecting and avoiding collisions, you ensure the final products meet design specifications.

Efficiency: Collisions often lead to machine downtime for repairs and adjustments. Collision detection can minimize downtime, increasing the machine's overall efficiency and productivity.

Cost Savings: By avoiding collisions, the costs associated with tool replacement, rework, and machine repairs can be reduced. This leads to cost savings in the long run.

Complex Machining: In multi-axis CNC machining, where the tool moves in multiple directions simultaneously, collision detection is essential to ensure that the tool's path doesn't intersect with the workpiece or other machine components.

Real-time Monitoring: Advanced CNC systems incorporate real-time collision detection that can dynamically adjust toolpaths to avoid collisions during the machining process, further enhancing efficiency and safety.

Thus, collision detection in CNC machines is vital for safety, protecting equipment, maintaining accuracy, ensuring efficiency, and reducing costs. It plays a central role in the smooth and productive operation of CNC machining processes.

1.3.1 FLOW CHART OF COLLISION DETECTION

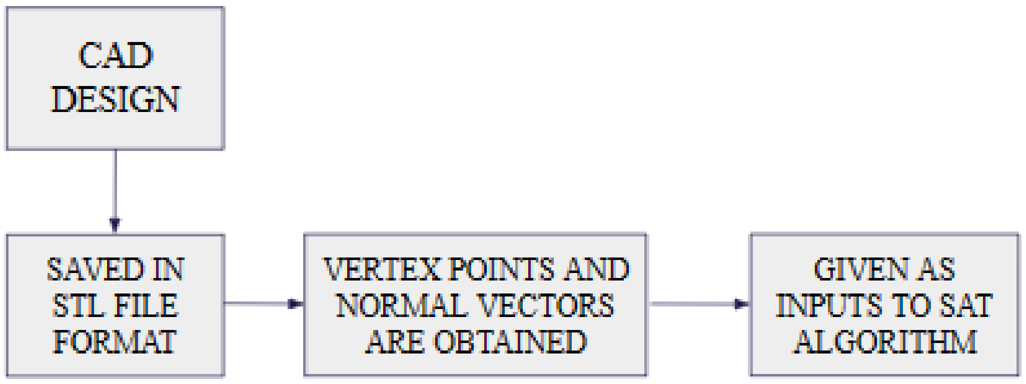

CAD DESIGN
SAVED IN STL FILE FORMAT
VERTEX POINTS AND NORMAL VECTORS ARE OBTAINED
GIVEN AS INPUTS TO SAT ALGORITHM

Fig 1.4 Flow Chart of Collision Detection

The CAD software is used to design the required model and the design file is saved in STL (Stereolithography) file format and fed to the CNC machine. Also, this file format is utilized to get the coordinates of the vertices of the triangles which are given as inputs to the SAT (Separating Axis Theorem) based algorithm. The algorithm is developed and used for collision detection between two objects that are represented as meshed triangles.

1.4 NEED FOR THE PROJECT

Collisions often lead to tool and work piece damage. Collisions between machine components or with workpieces can cause machinery to stop working. Identifying and preventing collisions in real-time minimizes downtime, ensuring that manufacturing processes run smoothly and efficiently. So, collision detection in machine tool automation is a crucial safety and efficiency measure. It protects expensive machinery and delicate workpieces, reduces downtime, preserves product quality, enhances operator safety, and ultimately contributes to cost savings and improved manufacturing performance.

Machine tool automation requires the collision detection between the object and the cutter to enhance the efficiency of the machine tools. The current collision detection algorithm for machine tools is the software-based computer numerical control (CNC) approach, the low computation capability of which refrains the CNC based approach from real-time collision detection less effective. To overcome this problem, relatively computationally faster algorithm has to be developed and utilized.

Determining the collision at a faster rate helps in prevention of the damage of the object to be designed and thereby helps in improving the accuracy at mass productions. And hence, a collision detection algorithm with higher computational speed and more precision should be implemented.

1.5 OBJECTIVES OF THE PROJECT

- To prevent collisions between machine tool and workpiece by developing and implementing a system that can accurately detect collisions between objects.
- To detect the unintended collisions between objects and machineries in CNC machines for ensuring machine tool automation.
- To develop the computationally faster collision detection algorithm using Verilog HDL and to implement the same in FPGA board.
- To analyze the movement of machine tools, workpieces, and other relevant objects to prevent collisions and ensure safe and efficient operation.
- To ensure the proper working of the machine tools and to prevent damages to the tool parts of the CNC machines and to prevent the damages to the workpieces as well.
- To reduce the number of sample pieces wasted during mass productions and to reduce the cost losses involved.

Fig 1.5 Sculpturing Machine

1.6 ORGANIZATION OF THE REPORT

The report is structured as follows. In Chapter 2, we discuss the literature review that outlines the various approaches used in collision detection of objects for machine tool automation. The suggested algorithm for collision detection is explained in Chapter 3. The system requirements for the hardware implementation of the algorithm as well as the implementation details are discussed in Chapter 4. The simulation results and the results from hardware implantation of the algorithm are covered in Chapter 5. The outcome of the algorithm and the applications of the work are reported in Chapter 6. Finally, the references that we have referred to are all given in Chapter 7.

CHAPTER 2

LITERATURE SURVEY

[1] In this paper, the algorithm discussed is applicable to all general polygonal models. It pre-computes a hierarchical representation of models using tight-fitting oriented bounding box trees (OBBTrees). At runtime, the algorithm traverses two such trees and tests for overlaps between oriented bounding boxes based on a separating axis theorem, which takes less than 200 operations in practice. It has been implemented and we compare its performance with other hierarchical data structures. In particular, it can robustly and accurately detect all the contacts between large complex geometries composed of hundreds of thousands of polygons at interactive rates. The algorithm is general-purpose and makes no assumptions about the input model. We have presented new algorithms for efficient construction of tight-fitting OBBTrees and overlap detection between two OBBs based on a new separating axis theorem. We have compared its performance with other hierarchies of spheres and AABBs and find it asymptotically faster for close proximity situations. The algorithm has been implemented and is able to detect all contacts between complex geometries (composed of a few hundred thousand polygons) at interactive rates.

[2] In this paper, a new algorithm based on the sweep plane approach for global collision detection for five-axis NC machining is presented. This algorithm takes into account not only collisions between the tool and workpiece, but also collisions between the other parts of the CNC machine, especially the change of the workpiece geometry is included in the detection process. The workpiece and machine bodies are firstly approximated by an octree of bounding spheres. Collision detection is conducted between these spheres. If there is any interference between these bounding spheres, their subspheres are further tested. A new algorithm for the collision detection for five-axis NC machining based on the sweep plane approach has been presented. In this approach, the collision is firstly detected by using the bounding sphere algorithm; and the colliding spheres are then further checked with the sweep plane algorithm. During the detection process, the change of the workpiece geometry is taken into account. The algorithm computes the slices of the updated workpiece and checks the interferences of these slices with slices of other machine parts. The collisions between the tool and workpiece or between other parts of the machine can be detected. The algorithm has no restriction on the shape of any bodies. The proposed algorithm was implemented using Visual C++ and OpenGL. With the use of the proposed sweep plane approach, the accuracy of the collision detection can be adjusted to meet the required accuracy by changing the slice distance. The integration of the sweep plane with the bounding sphere algorithm considerably reduces the time complexity of the collision detection algorithm, from an exponential function of the number of octree levels (m) to a quadratic function with respect to the number of slices within the colliding spheres of the octree. Since the time complexity of the algorithm largely depends on the trade-off between the number of levels of octree and the number of slices within the colliding spheres.

[3] Among all process disturbances, collisions cause the highest repair costs and the longest downtime periods. In some situations, the collision cannot be predicted by the numerical control unit. Hence, the machine tool has to be stopped as fast as possible whenever load limits on the tool are exceeded in order to avoid consequential damage. A new approach for faster reaction is the detection of the first contact as the beginning of a collision. Considering this approach it is of particular importance to make a distinction between a collision contact and the contact caused by the beginning of a manufacturing process. The following paper discusses sensor principles which allow the fast detection of a contact between a milling tool and the work piece. Furthermore, the analyses of different contact situations are described in detail. Essential information about the actual machining process, such as rough machining or finishing, are taken into consideration as well as external sensor signals. The described approach provides a new method for process monitoring in machine tools. Splitting the application into two parts – contact detection and signal processing – allows a faster reaction on collisions depending on the overall situation. The described proceeding is divided into several modules, two of them are explained. The contact module senses the first contact between two components of a machine. Thus, the decision logic will be activated much faster than conventional sensor applications would do. Although a contact does not immediately signify a collision, in some situations a contact is not allowed and therefore the emergency stop will be triggered.

[4] The incremental subdivision is introduced as a new adaptive subdivision method for triangle meshes in this paper. While regular (global) subdivisions produce a smooth surface from a given polygon mesh by refining all of its faces, adaptive subdivision produces a surface by refining only some selected areas of the mesh. Consequently, the selected area becomes fine and high resolution while the rest of mesh is coarse. Incremental subdivision produces a surface whose subdivided area is identical to when the entire mesh is subdivided regularly. In addition, as a good effect, the resolution of the produced surface gradually increases from coarse to fine. The incremental subdivision method expands the specified area to create a buffer region that is subdivided along with it. This method is efficient and easy to implement. We apply the incremental method to Loop and Butterfly subdivision schemes, and we compare it with other adaptive subdivision methods. We discuss some applications of incremental subdivision. However, this algorithm is not efficient and it is complicated for implementation. We introduced incremental adaptive subdivision for triangle meshes. It produces surfaces that have proper connectivity and geometry with a gradual change in subdivision depth between coarse and fine areas. Based on our comparison, incremental adaptive subdivision is more efficient than other methods while it is still simple to implement and can be effectively used in both modeling and rendering.

[5] In this paper, interactive environments for dynamically deforming objects play an important role in surgery simulation and entertainment technology. These environments require fast deformable models and very efficient collision handling techniques. While collision detection for rigid bodies is well-investigated, collision detection for deformable objects brings in additional challenging problems.

This paper focuses on these aspects and summarizes recent research in the area of deformable collision detection. Various approaches based on bounding volume hierarchies, distance fields, and spatial partitioning are discussed. Further, image-space techniques and stochastic methods are considered. Applications in cloth modeling and surgical simulation are presented. In these methods, the basic bounding volume and the strategy for generating and updating the hierarchy have to be chosen very carefully in order to handle frequent update requests in simulation environments with deformable objects. Stochastic methods are a promising approach to time-critical applications, since they allow for balancing performance and accuracy. This is not possible due to several reasons. First, the approaches require different input data, e. g., some approaches only work with closed surface meshes. Second, the approaches provide different collision information. They detect interfering surfaces or estimate penetration depth information. Some approaches are optimized for two-body collision tests, other methods, such as spatial subdivision, can handle n-body environments.

[6] Five-axis CNC machine tools are more and more popular in machining area, because of their ability to machine parts with complex geometries efficiently as well as achieve higher dimensional accuracy. Since two additional rotational axes are introduced in five-axis machines, there are difficult geometric problems that need to be solved in order to take full advantages of five-axis machining, and the most complex problems are collision detection and avoidance. These include the surface properties analysis based method, convex hull based method, C-space based method, accessibility based method, bounding volume and space partition method, distance calculation (vector) based method, rolling ball method, radial projection method, graphic-assisted method, and sweep plane approach. This paper aims at providing a state of the art review on algorithms for collision detection and avoidance in five-axis NC machining. In addition, a comparison of algorithms for collision detection and avoidance is considered. Collision detection and avoidance is one of the important problems in area of five-axis NC machining. A review of the algorithms for collision detection and avoidance for five-axis NC machining has been carried out in this work. The proposed solutions can be classified as the surface properties analysis based method, convex hull based method, C-space based method, accessibility based method, bounding volume and space partition method, and distance calculation based method, rolling ball method, radial projection method, graphic assisted method, and sweep plane approach. Most of the algorithms only detect the collision between the tool and workpiece, possible collisions between the machine and part, the machine and tool or among moving machine components as well as complete machine modeling have not been considered yet. Therefore, many challenges still need to be further addressed to improve the quality and efficiency of the tool path generation and tool orientation in five-axis NC machining. In addition, most of the algorithms for the local and global collision detection and avoidance are computationally expensive, hence time complexity of the algorithms is also a major issue for further improvement.

[6] In this paper, a new collision avoidance strategy and its integration with the collision detection for five-axis NC machining that improves upon the earlier collision detection algorithm is discussed. The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC

machining. The proposed algorithm automatically detects and corrects the collision based on the biggest collision boxes. The collision detection algorithm is based on the bounding volume and the sweep plane approach. The collision is firstly detected by using the bounding sphere algorithm, and the colliding spheres are then further checked with the sweep plane algorithm. The change of the workpiece geometry is included in the detection process. After the collision detection, the collision data are stored. Only the biggest collision boxes (the boxes with the biggest edge in the X, Y, or Z direction) for each type of collisions are stored. The collision avoidance algorithm corrects the biggest collision based on heuristic strategy. With this strategy, when the biggest collision is corrected, most of other collisions will disappear automatically. Therefore, the time complexity of the collision avoidance strategy is considerably reduced. The algorithm has been implemented in Visual C++ and OpenGL, demonstrated for five-axis machine with two rotary axes and can be customized to apply for any five-axis CNC machines. The proposed algorithm automatically detect and correct for the collisions. The collision detection algorithm is based on the bounding volume and the sweep plane approach. After the collision is detected, the biggest collision boxes for each type of collisions are stored and the others are discarded. The collision correction based on the heuristic strategy where the biggest collision boxes are corrected first. This newly proposed strategy reduces the time complexity of the collision avoidance algorithm since most of the other collisions automatically disappear when the biggest collision is corrected.

[7] The performance of virtual machine tools and virtual robot arms relies on the use of efficient and precise collision detection methods. This study proposes an octree-based collision detection method, called the ‘shared triangles extended octrees’, applied on virtual machine tools and virtual robot arms. The proposed scheme combines the high computational efficiency of the octree test and the high numerical accuracy of an analytical surface boundaries intersection test. In the proposed algorithm, the overlapping voxels between neighboring geometries in the virtual mechanisms are identified using octrees, and the intersections of the triangles within these voxels are then checked. It works efficiently, with rapid generation of deformed geometry by shared triangles without using decomposition. The proposed collision detection scheme is implemented on virtual machine tools and the virtual robot arms in a virtual manufacturing cell. The virtual machine tools and virtual robot arms are composed of component trees, which describe the kinematic relation between components. The necessity to check for collisions between each component is indicated by a Boolean matrix. The pairs of components selected by the Boolean matrix are checked using the proposed shared triangles extended octree method. The proposed collision detection method is an efficient tool for verifying the manipulation of CNC machine tools and robot arms and is of great help to CAD/CAM engineers and manufacturing engineers and operators of machine tools and robot arms.

[8] In this paper, the design of application specific integrated circuit (ASIC) to enhance the collision detection for machine tool automation. The bounded objects are represented by meshed triangles and so the separating axis theorem (SAT) based detection algorithm is considered. Furthermore, by considering high precision required by machine tool applications, the proposed algorithm includes the

collision detection of either non-coplanar or coplanar triangles. Following the collision detection algorithm, the hardware architecture with parallel processing to provide higher throughput rate over the architecture is reported in this paper. This algorithm can also be utilized in the film animation, computer games, robots, etc. To overcome the various drawbacks of the soft-ware based CNC approach, the design of application specific integrated circuit (ASIC) for collision detection of two objects is studied in this paper. In general, the SAT is applied to the collision detection of two convex sets. But here, the results related to the collision detection of two triangles in the 3D or 2D space are alone presented.

CHAPTER 3

SAT BASED COLLISION DETECTION

The Separating Axis Theorem (SAT) is a fundamental algorithm utilized in collision detection within computer graphics and physics simulations. At its core, SAT operates on the premise that two convex shapes do not intersect if and only if there exists an axis along which the orthogonal projections of the shapes are disjoint. The algorithm begins by representing the geometric shapes involved as convex polygons, ensuring that complex shapes are broken down into simpler, convex components if necessary. Next, a set of normalized axes, typically perpendicular to the edges of the polygons, is computed to serve as potential separation axes. The vertices of each shape are projected onto these axes, generating one-dimensional intervals along these directions. Subsequently, for each axis, the algorithm determines whether the intervals overlap. If there is no overlap along any axis, the shapes are not colliding. However, if an overlap is detected on every axis, the algorithm identifies the axis associated with the minimum overlap, signifying the direction of the smallest penetration vector. To confirm the collision and determine the precise overlapping region, a more detailed two-dimensional check is performed along this axis. In case of a confirmed collision, appropriate response mechanisms can be employed, such as separating the shapes or adjusting their velocities. SAT's efficiency and accuracy make it a widely adopted approach for collision detection, particularly for convex shapes. Nonetheless, for non-convex shapes, additional strategies may be necessary, such as shape decomposition or alternative collision detection techniques.

3.1 DESIGN

SAT is applied to the collision detection of two convex sets.3D triangles are said to be collision-free if and only if there exists a separating plane between the two triangles. A vector that is orthogonal to the separating plane is the associated separating axis.

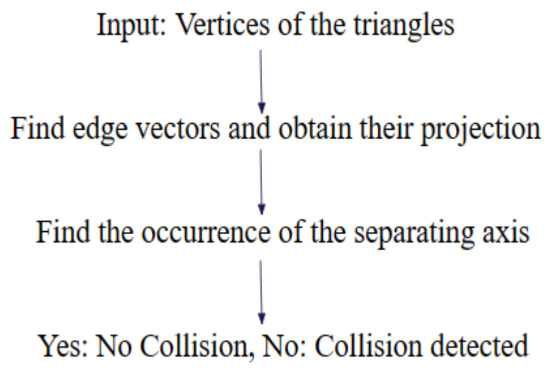

Input: Vertices of the triangles
Find edge vectors and obtain their projection
Find the occurrence of the separating axis
Yes: No Collision, No: Collision detected

Fig 3.1 SAT Algorithm Flow Diagram

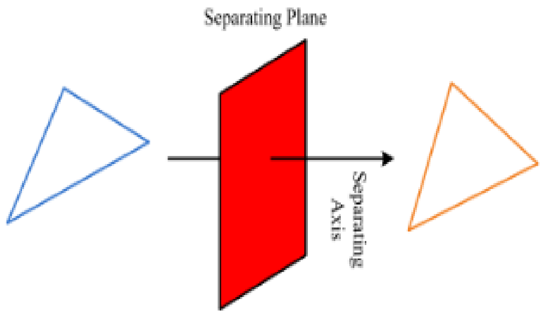


Fig 3.2 Separating Plane and Separating axis

3.1.1 PSEUDOCODE OF SAT ALGORITHM

Input : OA1, OA2, OA3, NA, OB1, OB2, OB3, NB
Output : Collision Occurs

```
compute edge vectors: $\overrightarrow{A1A2}$, $\overrightarrow{A2A3}$, $\overrightarrow{A3A1}$, $\overrightarrow{B1B2}$, $\overrightarrow{B2B3}$, $\overrightarrow{B3B1}$
Collision Occurs = 1

for k = 1 to 3 do
    Compute tk = $\overrightarrow{OAk}$ .$\overrightarrow{NA}$   and sk= $\overrightarrow{OBk}$ .$\overrightarrow{NA}$
    Compute tk = $\overrightarrow{OAk}$ .$\overrightarrow{NB}$   and vk= $\overrightarrow{OBk}$ . $\overrightarrow{NB}$
end

if ((mink tk > maxk sk ) or (maxk tk > mink sk )) or
   ((mink uk > maxk vk ) or (maxk uk > mink vk )) then
      Collision Occurs = 0
      return
else
  end

if ((t1 == v1) and (t2 == v2) and (t3 == v3)) then
 % Coplanar triangles
 for k = 1 to 6 do
     Assign $\overrightarrow{C}$ to be the k-th possible separating axis for coplanar triangles
       Compute the necessary inner product terms
```

```
                if (inequality holds) then
                    Collision Occurs = 0
                    return
                else
                  end
end
else
  % Non-Coplanar triangles
  for k = 1 to 9 do
      Assign $\vec{C}$ to be the k-th possible separating axis for non-coplanar triangles
        Compute the necessary inner product terms
          if (inequality holds) then
              Collision Occurs = 0
              return
          else
            end
   end
end
```

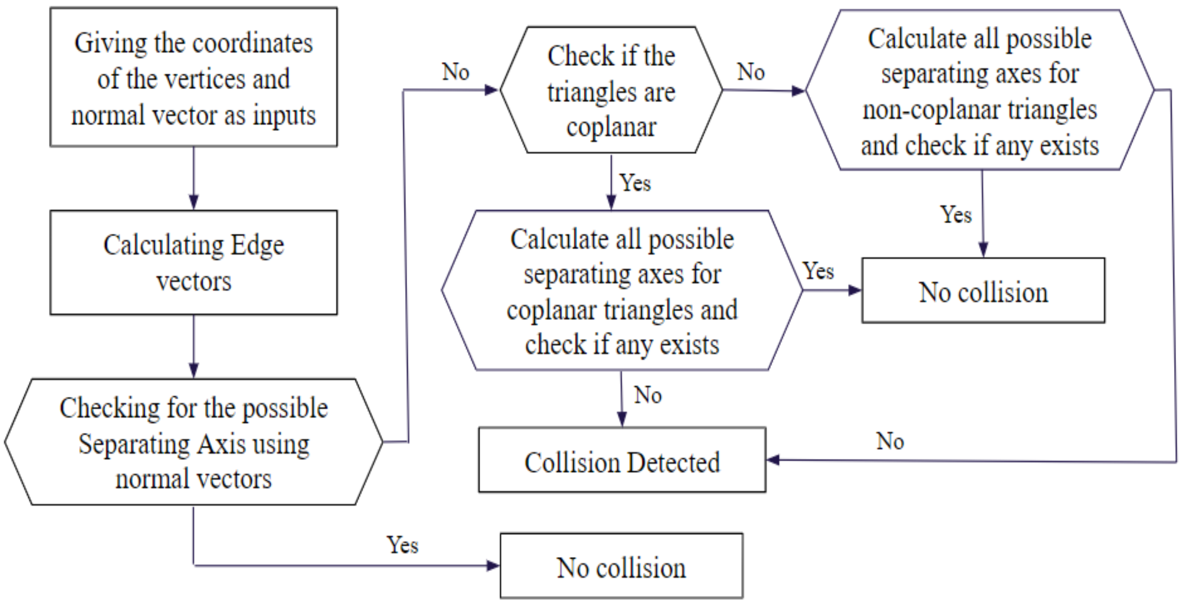

Giving the coordinates of the vertices and normal vector as inputs
Calculating Edge vectors
Checking for the possible Separating Axis using normal vectors
Yes
No collision
No
Check if the triangles are coplanar
No
Calculate all possible separating axes for non-coplanar triangles and check if any exists
Yes
Calculate all possible separating axes for coplanar triangles and check if any exists
Yes
No collision
Yes
No
Collision Detected
No

Fig 3.3 Flow Chart of SAT Algorithm

$$\min\left\{\overrightarrow{OA_1}\cdot\vec{c},\overrightarrow{OA_2}\cdot\vec{c},\overrightarrow{OA_3}\cdot\vec{c}\right\} > \max\left\{\overrightarrow{OB_1}\cdot\vec{c},\overrightarrow{OB_2}\cdot\vec{c},\overrightarrow{OB_3}\cdot\vec{c}\right\} \quad \text{or} \quad \max\left\{\overrightarrow{OA_1}\cdot\vec{c},\overrightarrow{OA_2}\cdot\vec{c},\overrightarrow{OA_3}\cdot\vec{c}\right\} > \min\left\{\overrightarrow{OB_1}\cdot\vec{c},\overrightarrow{OB_2}\cdot\vec{c},\overrightarrow{OB_3}\cdot\vec{c}\right\}$$

Fig 3.4 Inequality conditions to be checked

3.1.2 STEPS FOLLOWED IN SAT ALGORITHM

Step 1 :

The algorithm is to test each and every potential for any separating axis. When a separating axis is found, the indicator Collision Occurs = 0 is returned; otherwise, the indicator variable Collision Occurs = 1 is returned.

Step 2 :

Computing the projections (inner products) of triangles A and B onto the normal vectors NA and NB, respectively.

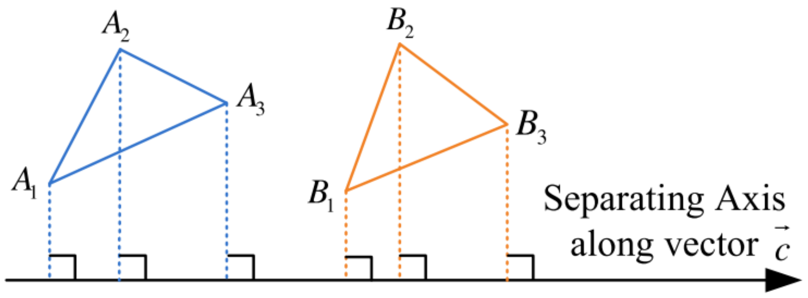


Fig 3.5 Projections of triangles A and B on the separating axis

Step 3 :

To test whether NA or NB is a separating axis. (General condition)

Step 4 :

The computed values tk and vk, ∀k, are tested for their equality to check whether the two triangles are coplanar. If the triangles are found to be coplanar, check if any possible separating axis exists and accordingly clear or set the output variable.

Table 3.1 Potential separating axes for two coplanar triangles A and B

Possible Separating Axes			
1	$\overrightarrow{N_A} \times \overrightarrow{A_1A_2}$	4	$\overrightarrow{N_B} \times \overrightarrow{B_1B_2}$
2	$\overrightarrow{N_A} \times \overrightarrow{A_2A_3}$	5	$\overrightarrow{N_B} \times \overrightarrow{B_2B_3}$
3	$\overrightarrow{N_A} \times \overrightarrow{A_3A_1}$	6	$\overrightarrow{N_B} \times \overrightarrow{B_3B_1}$

Step 5 :

If the triangles are found to be non-coplanar, check if any possible separating axis exists and accordingly clear or set the output variable.

Table 3.2 Potential separating axes for two non-coplanar triangles A and B

Possible Separating Axes					
1	$\overrightarrow{A_1A_2} \times \overrightarrow{B_1B_2}$	4	$\overrightarrow{A_2A_3} \times \overrightarrow{B_1B_2}$	7	$\overrightarrow{A_3A_1} \times \overrightarrow{B_1B_2}$
2	$\overrightarrow{A_1A_2} \times \overrightarrow{B_2B_3}$	5	$\overrightarrow{A_2A_3} \times \overrightarrow{B_2B_3}$	8	$\overrightarrow{A_3A_1} \times \overrightarrow{B_2B_3}$
3	$\overrightarrow{A_1A_2} \times \overrightarrow{B_3B_1}$	6	$\overrightarrow{A_2A_3} \times \overrightarrow{B_3B_1}$	9	$\overrightarrow{A_3A_1} \times \overrightarrow{B_3B_1}$

3.2 EXISTING WORK

3.2.1 HARDWARE ARCHITECTURE

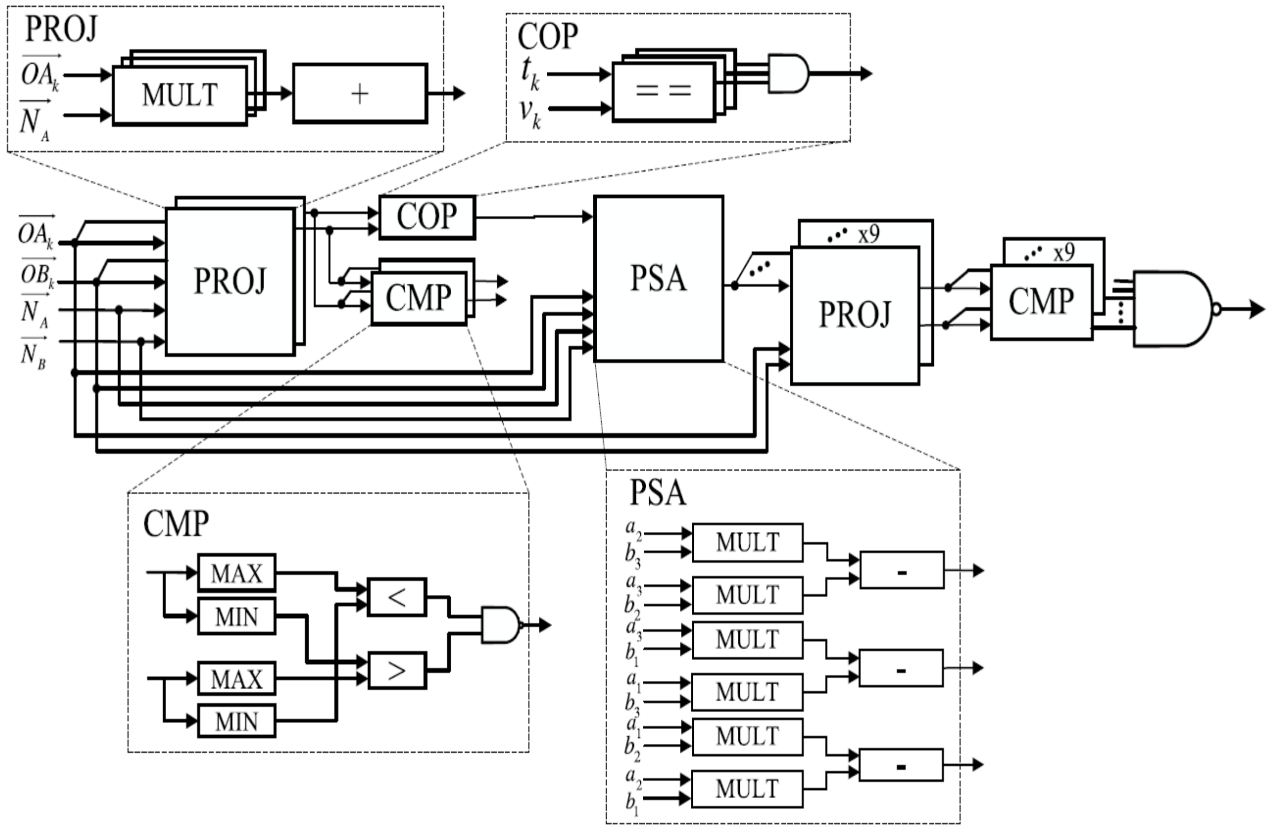


Fig 3.6 Hardware Architecture

- PROJ: Finding Projection of the Vertices
- CMP: Comparator to compare the minimum and maximum points
- COP: Finding whether the triangles are coplanar or not.
- PSA: Based on the coplanarity of the triangles, finding the possible separating axis.

The architecture is designed for the worst computational complexity case, which requires 188 RMULs to determine whether two non-coplanar triangles collide or not. For practical consideration, the information of meshed triangles. Each triangle from the meshed triangles for representing an object is characterized by the coordinates of the three vertices and the associated normal vector. These coordinates and associated normal vectors for two triangles are inputs to our hardware architecture. Each PROJ module is for computing the inner product of two vectors. Each CMP module that follows a PROJ module performs the comparisons to yield indication about whether a separating axis is found. Two PROJs on the left-hand side are associated with the operations of the above mentioned pseudocode. The COP module produces an indicator to indicate whether the two input triangles are coplanar or not. The PSA module produces the remaining 9 or 6 potential separating axes for the non-coplanar or coplanar triangles, respectively. Parallel 9 PROJs are associated with the inner product operations of the pseudocode. When coplanar triangles are detected, only six of the 9 PROJs are activated to compute the inner products.

3.3 PROPOSED WORK

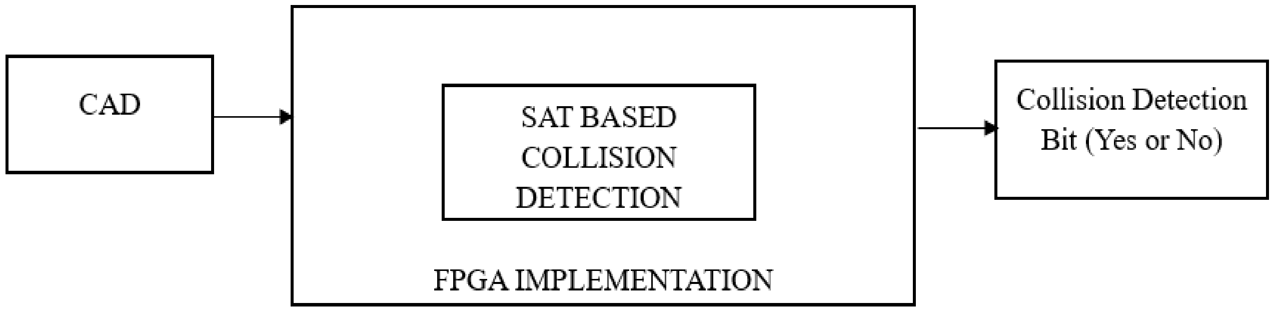

CAD
SAT BASED COLLISION DETECTION
FPGA IMPLEMENTATION
Collision Detection Bit (Yes or No)

Fig 3.7 Block Diagram of Proposed Work

FPGA realization of collision detection of two objects for the machine tool automation is considered. For the 3D objects represented by meshed triangles in the STL file format and SAT based algorithm is developed to process it multiple times for collision detection of multiple sets of two triangles. To achieve the required 1ms for real-time processing of collision detection of two objects, the FPGA implementation is done to determine the separating axes from the multiple potential separating axes. As machine tool automation is essential for the future precision machinery, the hardware design of collision detection of two objects is therefore valuable.

The collision detection of two objects requires large amount of computational work. For example, if either of the two objects is represented by 200 polygons, the computational burden for collision detection of two objects is equivalent to the burden for 40,000 times of collision detection of two polygons. To realize the collision detection of two objects for the SMT, the software-based computer numerical control (CNC) approach has been developed. Due to its low computing capability, the soft-ware based approach makes the SMT unable to achieve real-time collision detection. To overcome the above-mentioned drawbacks of the soft-ware based approach, the SAT algorithm is considered.

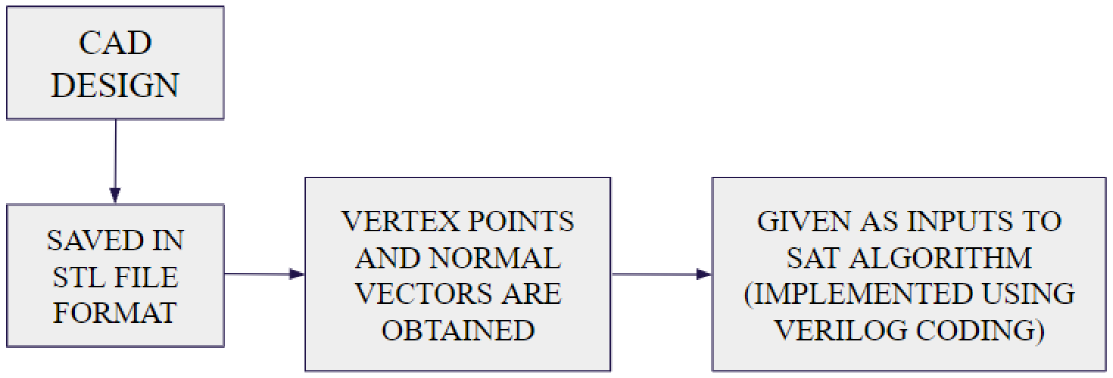

CAD DESIGN
SAVED IN STL FILE FORMAT
VERTEX POINTS AND NORMAL VECTORS ARE OBTAINED
GIVEN AS INPUTS TO SAT ALGORITHM (IMPLEMENTED USING VERILOG CODING)

Fig 3.8 Flow Diagram of proposed work

The problem of collision detection of two objects is converted to the problem of collision detection of two triangles from the two objects. Thus, by following the SAT based algorithm, we develop the algorithm to be implemented in FPGA board for collision detection of two objects. Due to the parallel structure, the designed architecture performs collision detection of two objects in less than 1ms and meets the requirement for the automation of Machine Tools.

3.3.1 MANUAL COLLISION DETECTION

In CNC (Computer Numerical Control) machines, manual collision detection is a crucial safety feature designed to prevent potential damage to the machine, workpiece, or tooling during the machining process. Collision detection involves the use of sensors, feedback mechanisms, and specialized software to monitor and analyze the machine's movements and positions in real-time. These systems are equipped with sensors placed strategically on the machine, which continuously monitor the tool's position and detect any deviation from the programmed toolpath. If a potential collision is detected, the machine's control system promptly halts the operation, preventing any damage that could occur. Manual collision detection systems often allow operators to override the automated safety measures if necessary, providing a human intervention option when a collision is suspected but not confirmed. This combination of automated monitoring and human oversight ensures a safer machining environment and minimizes the risk of costly accidents and equipment damage.

3.3.2 AUTOMATED COLLISION DETECTION

Automated collision detection in CNC machines using ASIC (Application-Specific Integrated Circuit) and FPGA (Field-Programmable Gate Array) implementation is a cutting-edge advancement in manufacturing technology. ASICs and FPGAs are custom-designed electronic components that are optimized for specific tasks offering high processing speeds and efficient parallel processing capabilities. When integrated into CNC systems, they facilitate real-time collision detection by rapidly processing data from various sensors and feedback mechanisms. These integrated circuits can analyze the machine's position, tool movements, and workpiece location in real-time, comparing them to the programmed toolpath.

If a potential collision is detected, the ASIC or FPGA swiftly triggers a safety protocol, halting the machine to prevent damage. This automated collision detection system significantly enhances manufacturing safety and efficiency by enabling faster and more precise identification of collision risks, minimizing downtime, reducing the likelihood of accidents, and ultimately improving the overall productivity of CNC machining operations.

3.3.3 ADVANTAGES OF PROPOSED WORK

Automation in CNC (Computer Numerical Control) machine tools offers a plethora of advantages that revolutionize modern manufacturing processes. Firstly, automation enhances precision and accuracy by eliminating the potential for human errors. CNC machines equipped with automation can consistently produce complex and intricate components with high levels of precision, meeting tight tolerance requirements.

Secondly, automation significantly boosts productivity and efficiency. Machines can operate continuously, running multiple shifts without the need for manual intervention, leading to increased production rates and quicker turnaround times for projects. Automated CNC machines can also optimize tool changes, material handling, and other processes, streamlining operations and minimizing idle time.

Moreover, automation enhances cost-effectiveness by reducing labor costs and minimizing material waste. The ability to run unmanned or with minimal supervision means that labor hours can be allocated to more strategic tasks. Additionally, automation optimizes material usage by efficiently nesting parts within raw materials, resulting in reduced scrap and increased material utilization. Furthermore, automation in CNC machines facilitates improved flexibility and adaptability.

Rapid reprogramming and setup changes allow for quick adaptation to new product designs or modifications, making it easier to respond to market demands and shifts in production requirements. In terms of safety, automation enhances the overall workplace safety by minimizing direct human interaction with the machine during the production process. This can lead to a decrease in workplace accidents and injuries, ensuring a safer working environment.

Lastly, automation enables comprehensive data collection and analysis. Advanced CNC machines can generate and collect vast amounts of data regarding operations, performance, and tool health. This data can be analyzed to optimize processes, predict maintenance needs, and drive continuous improvement in manufacturing operations. Overall, automation in CNC machine tools plays a pivotal role in transforming the manufacturing landscape, unlocking a range of benefits that are fundamental to the modern industrial era.

CHAPTER 4

IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS

Implementing FPGA (Field-Programmable Gate Array) technology for CNC (Computer Numerical Control) machine tool automation requires careful consideration of system requirements to ensure optimal performance and functionality.

4.1.1 NEXYS 4 DDR FPGA BOARD

- 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops.
- 4,860 Kbits of fast block RAM.
- Six clock management tiles, each with phase-locked loop (PLL).
- 240 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)
- 16 user switches
- USB-UART Bridge
- 12-bit VGA output
- 3-axis accelerometer
- 128MiB DDR2
- Pmod for XADC signals
- 16 user LEDs
- Two tri-color LEDs
- PWM audio output
- Temperature sensor
- Serial Flash
- Digilent USB-JTAG port for FPGA programming and communication
- Two 4-digit 7-segment displays
- Micro SD card connector

- PDM microphone
- 10/100 Ethernet PHY
- Four Pmod ports
- USB HID Host for mice, keyboards and memory sticks

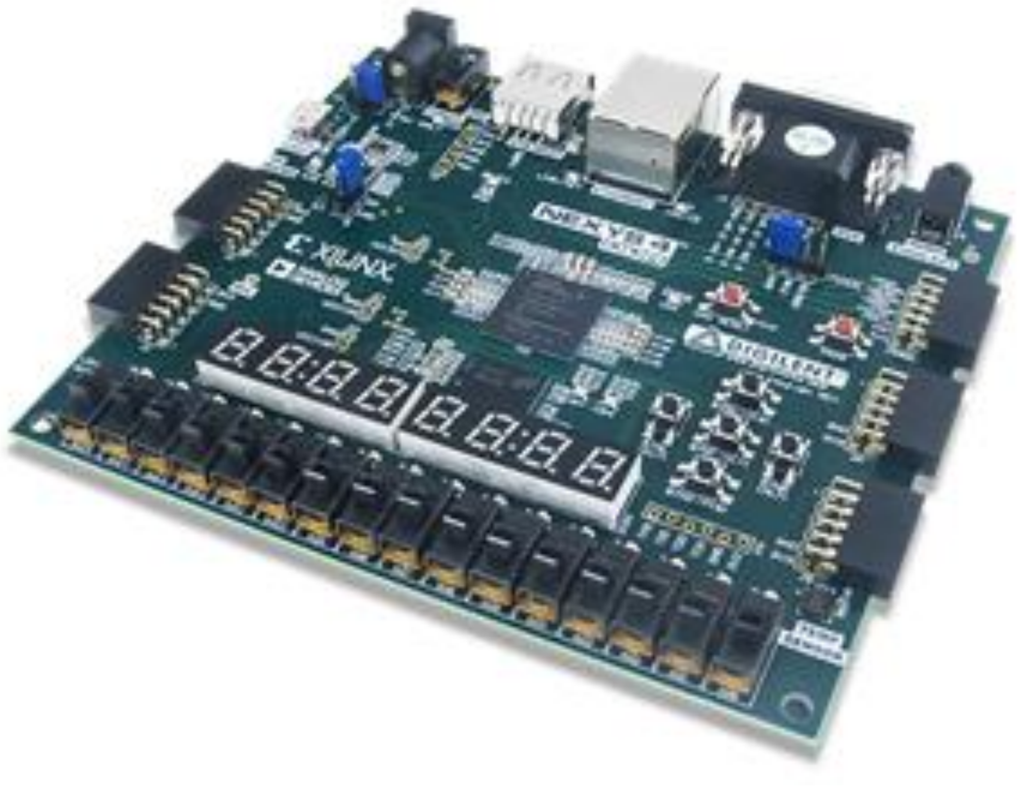


Fig 4.1 Nexys 4 DDR Board

4.1.2 XILINX VIVADO

Fig 4.2 Logo of Xilinx Vivado Software

Xilinx Vivado is one of the powerful and versatile software tools designed for FPGA (Field Programmable Gate Array) based implementation, especially in the realm of machine tool automation.

Vivado is a comprehensive development environment provided by Xilinx, a leader in FPGA (Field-Programmable Gate Array) technology. It offers a rich set of features and capabilities that facilitate the design, development, and optimization of complex digital circuits and systems, making it an ideal choice for automating machine tools.One of the key strengths of Vivado lies in its ability to support high-level hardware description languages (HDLs) such as VHDL and Verilog. These languages enable engineers to describe the behavior and structure of the FPGA-based automation system at an abstract level, allowing for efficient modeling, simulation, and verification of the design. Vivado's advanced synthesis and optimization algorithms further refine the HDL code, ensuring the resulting FPGA design meets the desired performance, power, and area specifications.

Moreover, Vivado offers a robust set of tools for system integration, including IP (Intellectual Property) integration, allowing developers to easily incorporate pre-designed modules and functions into their IC designs. This significantly accelerates the development process and improves productivity. Additionally, Vivado provides an array of debugging and analysis features, enabling engineers to thoroughly analyze the FPGA design's functionality, performance, and timing, ultimately leading to a more reliable and efficient automation system.

In the context of machine tool automation, Vivado supports seamless integration with various hardware components and interfaces, ensuring compatibility with the diverse array of sensors, actuators, and communication protocols prevalent in CNC (Computer Numerical Control) machines. This facilitates the creation of FPGA designs that can precisely control the movements, operations, and safety features of CNC machines, resulting in improved accuracy, efficiency, and safety within the manufacturing environment.

Overall, Xilinx Vivado stands as an invaluable tool for engineers working on FPGA-based implementations in the realm of machine tool automation. Its versatility, robust features, and intuitive interface empower developers to craft sophisticated and efficient FPGA designs that drive innovation and advancements in the field of CNC machine automation.

4.1.3 STL FILE FORMAT

An STL file is a 3D model saved in the Stereolithography (STL) file format developed by 3D Systems. It contains plain text or binary data that describes a set of triangular facets, which comprise a model. STL files are quite common, and they can be opened in many CAD and 3D modeling programs. STL files are also known as Standard Triangle Language files. These files are generated for the specified design and fed to the machine.

An STL file represents a 3D object as a collection of triangular facets. Each facet is defined by a normal vector and three vertices that define the shape and position of the triangle. Each facet has a unit normal vector, which points outward from the surface of the object. This helps in determining the orientation of the triangle. Three vertices with their 3D coordinates (x, y, z) define each triangle. These vertices are connected to form the facets.

STL files are typically plain text (ASCII) or binary. Binary STL files are more common due to their smaller file size and are the focus of this description. Binary STL files consist of a 80-byte header that is generally ignored by most software, followed by a 4-byte unsigned integer representing the number of facets (triangles) in the file. After this, each facet is described by 12 bytes for the normal vector and 36 bytes for the 3 vertices (12 bytes each). There are no delimiters or separators; the data is simply packed together. ASCII STL files represent the same data but are human-readable. They include textual descriptions of the facets, making them more accessible for manual inspection but comparatively larger in file size.

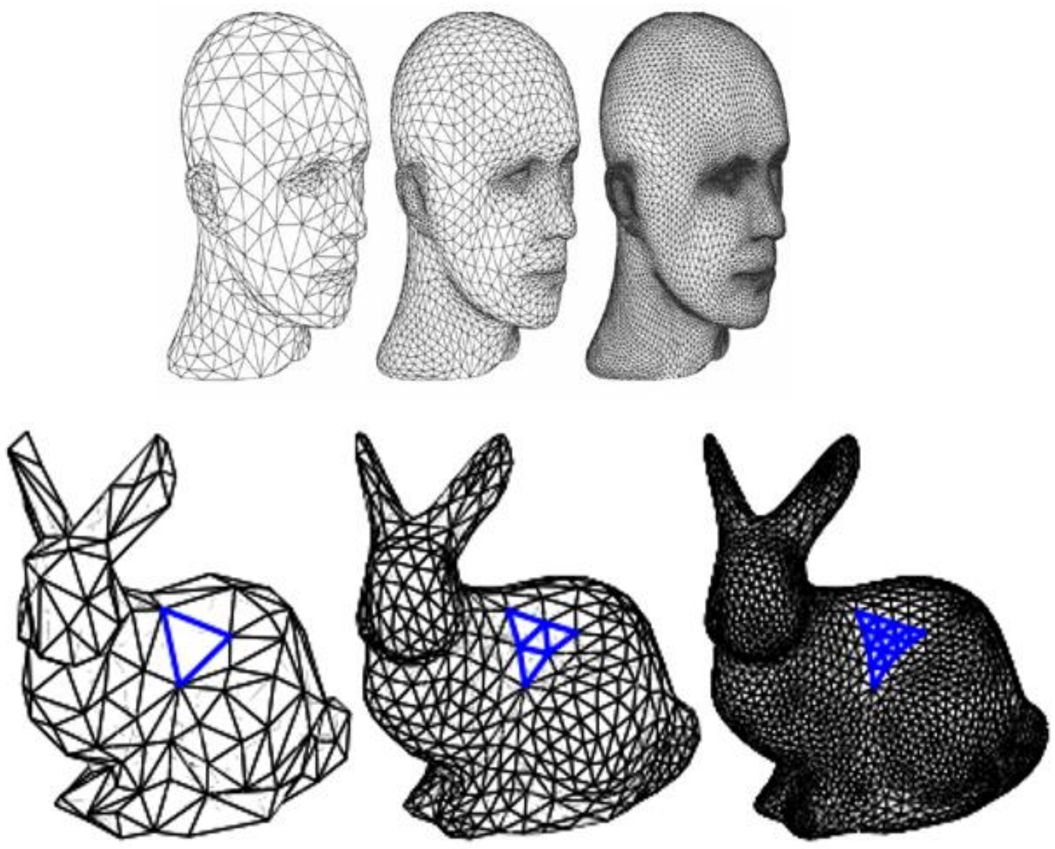


Fig 4.3 Triangulation of Images

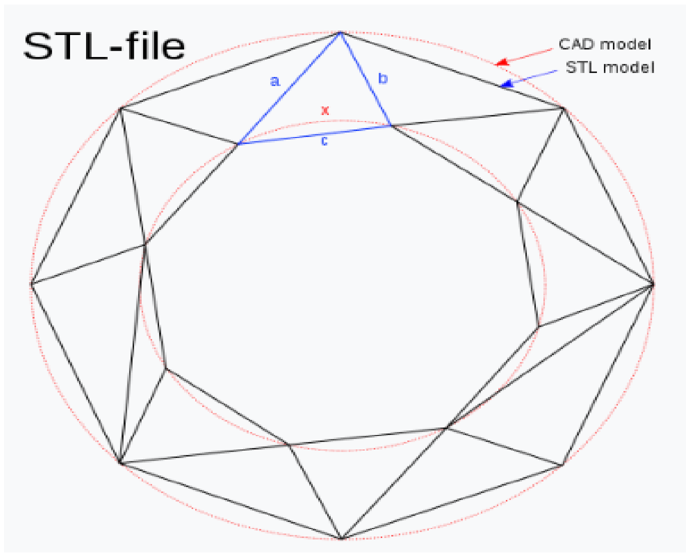


Fig 4.4 STL file format of a Torus Structure

4.2 IMPLEMENTATION DETAILS

FPGA (Field-Programmable Gate Array) implementation in machine tool automation for CNC (Computer Numerical Control) machines represents a cutting-edge approach to revolutionizing modern manufacturing. FPGA technology offers a flexible and reconfigurable hardware platform that can be tailored to specific automation requirements. By utilizing FPGA, critical aspects of CNC machine tool automation, such as motor control, trajectory planning, speed regulation, and real-time feedback processing, can be efficiently addressed. FPGA's parallel processing capabilities are leveraged to precisely control the motors, ensuring accurate tool positioning and movement along programmed paths. Moreover, FPGA enables the implementation of sophisticated algorithms for path planning, optimizing tool movements to enhance efficiency and minimize machining time. Real-time feedback from sensors and encoders is processed swiftly, allowing for error correction and maintaining precise toolpath adherence. FPGA also plays a pivotal role in collision detection and prevention, rapidly analyzing sensor data to detect potential collisions and triggering immediate responses to mitigate risks. This level of automation enhances productivity, precision, and safety in CNC machining, paving the way for more advanced and efficient manufacturing processes.

4.2.1 IMPLEMENTATION IN XILINX VIVADO 2018.2

Xilinx Vivado 2018.2 is a pivotal version of the Vivado Design Suite, a comprehensive development environment for FPGA (Field-Programmable Gate Array) design and implementation. Released in 2018, this iteration introduced significant enhancements and features that bolstered FPGA design workflows. Vivado 2018.2 built upon Xilinx's reputation for innovation, enabling engineers and designers to create highly efficient, reliable, and scalable FPGA-based solutions.

DESIGN WINDOW

```
module CollisionDetector(
  input signed [15:0] OA1_x, OA1_y, OA1_z,
  input signed [15:0] OA2_x, OA2_y, OA2_z,
  input signed [15:0] OA3_x, OA3_y, OA3_z,
  input signed [15:0] NA_x, NA_y, NA_z,
  input signed [15:0] OB1_x, OB1_y, OB1_z,
  input signed [15:0] OB2_x, OB2_y, OB2_z,
  input signed [15:0] OB3_x, OB3_y, OB3_z,
  input signed [15:0] NB_x, NB_y, NB_z,
  output reg CollisionOccurs
);

reg signed [15:0] A1A2_x, A1A2_y, A1A2_z;
reg signed [15:0] A2A3_x, A2A3_y, A2A3_z;
reg signed [15:0] A3A1_x, A3A1_y, A3A1_z;
reg signed [15:0] B1B2_x, B1B2_y, B1B2_z;
reg signed [15:0] B2B3_x, B2B3_y, B2B3_z;
reg signed [15:0] B3B1_x, B3B1_y, B3B1_z;

reg signed [31:0] t[1:3], s[1:3];
reg signed [31:0] u[1:3], v[1:3];
reg signed [31:0] min_t, max_t, min_s, max_s;
reg signed [31:0] min_u, max_u, min_v, max_v;

reg signed [15:0] cx[1:6], cy[1:6], cz[1:6];
reg signed [15:0] ncx[1:9], ncy[1:9], ncz[1:9];
integer i;

always @(*)
 begin
  // Step 1: Compute edge vectors
  A1A2_x = OA2_x - OA1_x;
  A1A2_y = OA2_y - OA1_y;
  A1A2_z = OA2_z - OA1_z;
  A2A3_x = OA3_x - OA2_x;
  A2A3_y = OA3_y - OA2_y;
  A2A3_z = OA3_z - OA2_z;
```

```
A3A1_x = OA1_x - OA3_x;
A3A1_y = OA1_y - OA3_y;
A3A1_z = OA1_z - OA3_z;
B1B2_x = OB2_x - OB1_x;
B1B2_y = OB2_y - OB1_y;
B1B2_z = OB2_z - OB1_z;
B2B3_x = OB3_x - OB2_x;
B2B3_y = OB3_y - OB2_y;
B2B3_z = OB3_z - OB2_z;
B3B1_x = OB1_x - OB3_x;
B3B1_y = OB1_y - OB3_y;
B3B1_z = OB1_z - OB3_z;

// Step 2: Initialize CollisionOccurs
CollisionOccurs = 1;

// Step 3: Compute tk and sk
t[1] = OA1_x * NA_x + OA1_y * NA_y + OA1_z * NA_z;
s[1] = OB1_x * NA_x + OB1_y * NA_y + OB1_z * NA_z;
t[2] = OA2_x * NA_x + OA2_y * NA_y + OA2_z * NA_z;
s[2] = OB2_x * NA_x + OB2_y * NA_y + OB2_z * NA_z;
t[3] = OA3_x * NA_x + OA3_y * NA_y + OA3_z * NA_z;
s[3] = OB3_x * NA_x + OB3_y * NA_y + OB3_z * NA_z;

// Step 4: Compute uk and vk
u[1] = OA1_x * NB_x + OA1_y * NB_y + OA1_z * NB_z;
v[1] = OB1_x * NB_x + OB1_y * NB_y + OB1_z * NB_z;
u[2] = OA2_x * NB_x + OA2_y * NB_y + OA2_z * NB_z;
v[2] = OB2_x * NB_x + OB2_y * NB_y + OB2_z * NB_z;
u[3] = OA3_x * NB_x + OA3_y * NB_y + OA3_z * NB_z;
v[3] = OB3_x * NB_x + OB3_y * NB_y + OB3_z * NB_z;

// Step 5: Compute min and max points
min_t = (t[1] < t[2]) ? ((t[1] < t[3]) ? t[1] : t[3]) : ((t[2] < t[3]) ? t[2] : t[3]);
max_t = (t[1] > t[2]) ? ((t[1] > t[3]) ? t[1] : t[3]) : ((t[2] > t[3]) ? t[2] : t[3]);
min_s = (s[1] < s[2]) ? ((s[1] < s[3]) ? s[1] : s[3]) : ((s[2] < s[3]) ? s[2] : s[3]);
max_s = (s[1] > s[2]) ? ((s[1] > s[3]) ? s[1] : s[3]) : ((s[2] > s[3]) ? s[2] : s[3]);
min_u = (u[1] < u[2]) ? ((u[1] < u[3]) ? u[1] : u[3]) : ((u[2] < u[3]) ? u[2] : u[3]);
max_u = (u[1] > u[2]) ? ((u[1] > u[3]) ? u[1] : u[3]) : ((u[2] > u[3]) ? u[2] : u[3]);
```

```
min_v = (v[1] < v[2]) ? ((v[1] < v[3]) ? v[1] : v[3]) : ((v[2] < v[3]) ? v[2] : v[3]);
max_v = (v[1] > v[2]) ? ((v[1] > v[3]) ? v[1] : v[3]) : ((v[2] > v[3]) ? v[2] : v[3]);

// Step 6: Check for collision conditions
if ((min_t > max_s) || (max_t > min_s) || (min_u > max_v) || (max_u > min_v))
 begin
   CollisionOccurs = 0;
 end

else
 begin

  // Step 7: Check for Coplanar triangles
  if ((t[1] == v[1]) && (t[2] == v[2]) && (t[3] == v[3]))
   begin

    // Step 8: Finding the possible separating axes
    cx[1] = NA_y * A1A2_z - A1A2_y * NA_z;
    cy[1] = NA_z * A1A2_x - A1A2_z * NA_x;
    cz[1] = NA_x * A1A2_y - A1A2_x * NA_y;
    cx[2] = NA_y * A2A3_z - A2A3_y * NA_z;
    cy[2] = NA_z * A2A3_x - A2A3_z * NA_x;
    cz[2] = NA_x * A2A3_y - A2A3_x * NA_y;
    cx[3] = NA_y * A3A1_z - A3A1_y * NA_z;
    cy[3] = NA_z * A3A1_x - A3A1_z * NA_x;
    cz[3] = NA_x * A3A1_y - A3A1_x * NA_y;
    cx[4] = NB_y * B1B2_z - B1B2_y * NB_z;
    cy[4] = NB_z * B1B2_x - B1B2_z * NB_x;
    cz[4] = NB_x * B1B2_y - B1B2_x * NB_y;
    cx[5] = NB_y * B2B3_z - B2B3_y * NB_z;
    cy[5] = NB_z * B2B3_x - B2B3_z * NB_x;
    cz[5] = NB_x * B2B3_y - B2B3_x * NB_y;
    cx[6] = NB_y * B3B1_z - B3B1_y * NB_z;
    cy[6] = NB_z * B3B1_x - B3B1_z * NB_x;
    cz[6] = NB_x * B3B1_y - B3B1_x * NB_y;

    for (i = 1; i <= 6; i = i + 1)
     begin
      s[1] = OA1_x * cx[i] + OA1_y * cy[i] + OA1_z * cz[i];
      u[1] = OB1_x * cx[i] + OB1_y * cy[i] + OB1_z * cz[i];
```

```
          s[2] = OA2_x * cx[i] + OA2_y * cy[i] + OA2_z * cz[i];
          u[2] = OB2_x * cx[i] + OB2_y * cy[i] + OB2_z * cz[i];
          s[3] = OA3_x * cx[i] + OA3_y * cy[i] + OA3_z * cz[i];
          u[3] = OB3_x * cx[i] + OB3_y * cy[i] + OB3_z * cz[i];
          min_s = (s[1] < s[2]) ? ((s[1] < s[3]) ? s[1] : s[3]) : ((s[2] < s[3]) ? s[2] : s[3]);
          max_s = (s[1] > s[2]) ? ((s[1] > s[3]) ? s[1] : s[3]) : ((s[2] > s[3]) ? s[2] : s[3]);
          min_u = (u[1] < u[2]) ? ((u[1] < u[3]) ? u[1] : u[3]) : ((u[2] < u[3]) ? u[2] : u[3]);
          max_u = (u[1] > u[2]) ? ((u[1] > u[3]) ? u[1] : u[3]) : ((u[2] > u[3]) ? u[2] : u[3]);

          if ((min_s > max_u) || (max_s > min_u))
           begin
             CollisionOccurs = 0;
            break;
           end
         end
       end

      // Step 9: Check for Non-Coplanar triangles
      else
       begin

        // Step 10: Finding the possible separating axes
        ncx[1] = A1A2_y * B1B2_z - B1B2_y * A1A2_z;
        ncy[1] = A1A2_z * B1B2_x - B1B2_z * A1A2_x;
        ncz[1] = A1A2_x * B1B2_y - B1B2_x * A1A2_y;
        ncx[2] = A1A2_y * B2B3_z - B2B3_y * A1A2_z;
        ncy[2] = A1A2_z * B2B3_x - B2B3_z * A1A2_x;
        ncz[2] = A1A2_x * B2B3_y - A2A3_x * A1A2_y;
        ncx[3] = A1A2_y * B3B1_z - B3B1_y * A1A2_z;
        ncy[3] = A1A2_z * B3B1_x - B3B1_z * A1A2_x;
        ncz[3] = A1A2_x * B3B1_y - B3B1_x * A1A2_y;
        ncx[4] = A2A3_y * B1B2_z - B1B2_y * A2A3_z;
        ncy[4] = A2A3_z * B1B2_x - B1B2_z * A2A3_x;
        ncz[4] = A2A3_x * B1B2_y - B1B2_x * A2A3_y;
        ncx[5] = A2A3_y * B2B3_z - B2B3_y * A2A3_z;
        ncy[5] = A2A3_z * B2B3_x - B2B3_z * A2A3_x;
        ncz[5] = A2A3_x * B2B3_y - B2B3_x * A2A3_y;
        ncx[6] = A2A3_y * B3B1_z - B3B1_y * A2A3_z;
        ncy[6] = A2A3_z * B3B1_x - B3B1_z * A2A3_x;
        ncz[6] = A2A3_x * B3B1_y - B3B1_x * A2A3_y;
        ncx[7] = A3A1_y * B1B2_z - B1B2_y * A3A1_z;
        ncy[7] = A3A1_z * B1B2_x - B1B2_z * A3A1_x;
        ncz[7] = A3A1_x * B1B2_y - B1B2_x * A3A1_y;
        ncx[8] = A3A1_y * B2B3_z - B2B3_y * A3A1_z;
        ncy[8] = A3A1_z * B2B3_x - B2B3_z * A3A1_x;
        ncz[8] = A3A1_x * B2B3_y - B2B3_x * A3A1_y;
        ncx[9] = A3A1_y * B3B1_z - B3B1_y * A3A1_z;
        ncy[9] = A3A1_z * B3B1_x - B3B1_z * A3A1_x;
        ncz[9] = A3A1_x * B3B1_y - B3B1_x * A3A1_y;

        for (i = 1; i <= 9; i = i + 1)
         begin
          s[1] = OA1_x * ncx[i] + OA1_y * ncy[i] + OA1_z * ncz[i];
          u[1] = OB1_x * ncx[i] + OB1_y * ncy[i] + OB1_z * ncz[i];
          s[2] = OA2_x * ncx[i] + OA2_y * ncy[i] + OA2_z * ncz[i];
          u[2] = OB2_x * ncx[i] + OB2_y * ncy[i] + OB2_z * ncz[i];
          s[3] = OA3_x * ncx[i] + OA3_y * ncy[i] + OA3_z * ncz[i];
          u[3] = OB3_x * ncx[i] + OB3_y * ncy[i] + OB3_z * ncz[i];
          min_s = (s[1] < s[2]) ? ((s[1] < s[3]) ? s[1] : s[3]) : ((s[2] < s[3]) ? s[2] : s[3]);
          max_s = (s[1] > s[2]) ? ((s[1] > s[3]) ? s[1] : s[3]) : ((s[2] > s[3]) ? s[2] : s[3]);
          min_u = (u[1] < u[2]) ? ((u[1] < u[3]) ? u[1] : u[3]) : ((u[2] < u[3]) ? u[2] : u[3]);
          max_u = (u[1] > u[2]) ? ((u[1] > u[3]) ? u[1] : u[3]) : ((u[2] > u[3]) ? u[2] : u[3]);

          if ((min_s > max_u) || (max_s > min_u))
           begin
             CollisionOccurs = 0;
              break;
           end
         end
       end
     end
   end
 endmodule
```

TESTBENCH WINDOW

```
module TriangleCollisiontb;

  // Declare signals for inputs and outputs
  reg signed [15:0] OA1_x, OA1_y, OA1_z;
  reg signed [15:0] OA2_x, OA2_y, OA2_z;
  reg signed [15:0] OA3_x, OA3_y, OA3_z;
  reg signed [15:0] NA_x, NA_y, NA_z;
  reg signed [15:0] OB1_x, OB1_y, OB1_z;
  reg signed [15:0] OB2_x, OB2_y, OB2_z;
  reg signed [15:0] OB3_x, OB3_y, OB3_z;
  reg signed [15:0] NB_x, NB_y, NB_z;
  wire CollisionOccurs;

  // Instantiate the CollisionDetector module
  CollisionDetector uut (
    .OA1_x(OA1_x), .OA1_y(OA1_y), .OA1_z(OA1_z),
    .OA2_x(OA2_x), .OA2_y(OA2_y), .OA2_z(OA2_z),
    .OA3_x(OA3_x), .OA3_y(OA3_y), .OA3_z(OA3_z),
    .NA_x(NA_x), .NA_y(NA_y), .NA_z(NA_z),
    .OB1_x(OB1_x), .OB1_y(OB1_y), .OB1_z(OB1_z),
    .OB2_x(OB2_x), .OB2_y(OB2_y), .OB2_z(OB2_z),
    .OB3_x(OB3_x), .OB3_y(OB3_y), .OB3_z(OB3_z),
    .NB_x(NB_x), .NB_y(NB_y), .NB_z(NB_z),
    .CollisionOccurs(CollisionOccurs)
  );

  reg clk=0;
  // Clock generation (if needed)
  always begin
     #10 clk = ~clk;
  end

  initial begin
    // Initialize inputs
    OA1_x = 0; OA1_y = 0; OA1_z = 0;
    OA2_x = 1; OA2_y = 1; OA2_z = 1;
    OA3_x = 2; OA3_y = 2; OA3_z = 2;
    NA_x = 1; NA_y = 0; NA_z = 0;
```

```
OB1_x = 3; OB1_y = 3; OB1_z = 3;
OB2_x = 4; OB2_y = 4; OB2_z = 4;
OB3_x = 5; OB3_y = 5; OB3_z = 5;
NB_x = 0; NB_y = 1; NB_z = 0;

// 1
OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
NA_x = 31400; NA_y = 28697; NA_z = 5000;
OB1_x = 31539; OB1_y = 28697; OB1_z = 0;
OB2_x = 33243; OB2_y = 28697; OB2_z = 0;
OB3_x = 33243; OB3_y = 28197; OB3_z = 0;
NB_x = 32243; NB_y = 28197; NB_z = 0;

// 2
#20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
    OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
    OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
    NA_x = 31400; NA_y = 28697; NA_z = 5000;
    OB1_x = 31539; OB1_y = 28197; OB1_z = 0;
    OB2_x = 31539; OB2_y = 27197; OB2_z = 0;
    OB3_x = 33243; OB3_y = 27197; OB3_z = 0;
#20 NB_x = 32243; NB_y = 27197; NB_z = 0;

//3
#20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
    OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
    OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
    NA_x = 31400; NA_y = 28697; NA_z = 5000;
    OB1_x = 33243; OB1_y = 26197; OB1_z = 0000;
    OB2_x = 31539; OB2_y = 26197; OB2_z = 0000;
    OB3_x = 31539; OB3_y = 25197; OB3_z = 0;
#20 NB_x = 32539; NB_y = 25197; NB_z = 0;

//4
#20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
    OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
    OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
    NA_x = 31400; NA_y = 28697; NA_z = 5000;
```

```
        OB1_x = 33243; OB1_y = 25197; OB1_z = 0;
        OB2_x = 33243; OB2_y = 24197; OB2_z = 0;
        OB3_x = 31539; OB3_y = 28197; OB3_z = 0;
    #20 NB_x = 32539; NB_y = 24197; NB_z = 0;

    //5
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x =31539; OB1_y = 23197; OB1_z = 0;
        OB2_x = 33243; OB2_y = 23197; OB2_z = 0;
        OB3_x = 9039; OB3_y = 23197; OB3_z = 0;
    #20 NB_x = 19050; NB_y = 23197; NB_z = 0;

    //6
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 8039; OB1_y = 23197; OB1_z = 0;
        OB2_x = 7961; OB2_y = 23197; OB2_z = 0;
        OB3_x = 8961; OB3_y = 23197; OB3_z = 0;
    #20 NB_x = 8050; NB_y = 23197; NB_z = 0;

    //7
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 8039; OB1_y = 23197; OB1_z = 2000;
        OB2_x = 7961; OB2_y = 23197; OB2_z = 2000;
        OB3_x = 8961; OB3_y = 23197; OB3_z = 0;
    #20 NB_x = 8260; NB_y = 23197; NB_z = 2000;

    //8
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 33243; OB1_y = 23197; OB1_z = 0;
        OB2_x = 33243; OB2_y = 22197; OB2_z = 0;
        OB3_x = 8961; OB3_y = 22197; OB3_z = 0;
    #20 NB_x = 12850; NB_y = 22197; NB_z = 0;

    //9
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 7961; OB1_y = 2219; OB1_z = 2000;
        OB2_x = 8039; OB2_y = 2219; OB2_z = 2000;
        OB3_x = 9093; OB3_y = 2219; OB3_z = 0;
    #20 NB_x = 8262; NB_y = 2219; NB_z = 2000;

    //10
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 31539; OB1_y = 22197; OB1_z = 0;
        OB2_x = 31539; OB2_y = 21197; OB2_z = 0;
        OB3_x = 9039; OB3_y = 21197; OB3_z = 0;
    #20 NB_x = 19580; NB_y = 21197; NB_z = 0;

    //11
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 8039; OB1_y = 21197; OB1_z = 2000;
        OB2_x = 7961; OB2_y = 21197; OB2_z = 2000;
        OB3_x = 8961; OB3_y = 21197; OB3_z = 0;
    #20 NB_x = 8269; NB_y = 21197; NB_z = 0;

    //12
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
```

```
        OB1_x = 33243; OB1_y = 21197; OB1_z = 0;
        OB2_x = 33293; OB2_y = 20197; OB2_z = 0;
        OB3_x = 8961; OB3_y = 20197; OB3_z = 0;
    #20 NB_x = 18860; NB_y = 20197; NB_z = 0;

    //13
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 7961; OB1_y = 20197; OB1_z = 2000;
        OB2_x = 8039; OB2_y = 20197; OB2_z = 2000;
        OB3_x = 9039; OB3_y = 20197; OB3_z = 0;
    #20 NB_x = 8375; NB_y = 20197; NB_z = 2000;

    //14
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 31539; OB1_y = 20197; OB1_z = 0;
        OB2_x = 31539; OB2_y = 19197; OB2_z = 0;
        OB3_x = 33243; OB3_y = 19197; OB3_z = 0;
    #20 NB_x = 32243; NB_y = 19197; NB_z = 0;

    //15
    #20 OA1_x = 31539; OA1_y = 28697; OA1_z = 5000;
        OA2_x = 31200; OA2_y = 28687; OA2_z = 5000;
        OA3_x = 31800; OA3_y = 28687; OA3_z = 5000;
        NA_x = 31400; NA_y = 28697; NA_z = 5000;
        OB1_x = 9039; OB1_y = 19197; OB1_z = 0;
        OB2_x = 8039; OB2_y = 19197; OB2_z = 2000;
        OB3_x = 7961; OB3_y = 19197; OB3_z = 2000;
    #20 NB_x = 8260; NB_y = 19197; NB_z = 2000;
    // Finish simulation
    $finish;
  end
endmodule
```

4.2.2 IMPLEMENTATION IN FPGA BOARD

The application of the Nexys DDR4 in machine tool automation for CNC (Computer Numerical Control) machines represents a sophisticated integration of advanced hardware technology into the realm of manufacturing. The Nexys DDR4, a development board equipped with a Xilinx FPGA, offers a versatile and powerful platform for implementing automation in CNC machines. Its FPGA, paired with the high-speed DDR4 memory, provides the necessary processing power and memory bandwidth required for real-time control and precise coordination of CNC operations. By utilizing the Nexys DDR4, engineers can develop and implement intricate algorithms for motor control, trajectory planning, feedback processing, and collision detection in CNC machines. The FPGA's reconfigurable nature enables customization and adaptation to specific CNC machine configurations and operational requirements. The DDR4 memory offers ample space and high-speed data access, facilitating the storage and retrieval of complex CNC programs and critical data needed for accurate machining.

Moreover, the Nexys DDR4's integration capabilities, including various communication interfaces like Ethernet and USB, allow for seamless connectivity to external devices, networked systems, or supervisory computers. This enables centralized control, monitoring, and reprogramming of CNC machines, enhancing overall flexibility and adaptability in a manufacturing environment. Incorporating, the Nexys DDR4 into CNC machine tool automation enhances precision, efficiency, and safety. It empowers CNC machines to execute intricate tool movements, optimize cutting parameters, and rapidly respond to changing conditions during the machining process. The real-time feedback processing and collision detection algorithms help prevent errors and potential damage, ensuring a safer operational environment.

In conclusion, leveraging the Nexys DDR4 for machine tool automation in CNC machines showcases the fusion of cutting-edge FPGA technology with high-speed DDR4 memory, resulting in a potent platform that drives innovation and efficiency in CNC machining, ultimately leading to enhanced productivity and accuracy in manufacturing processes.

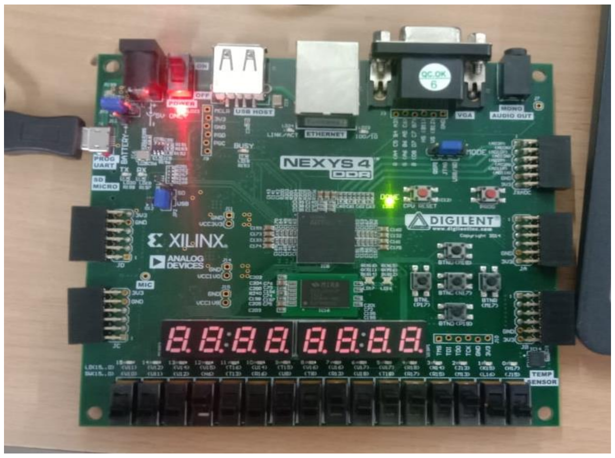


Fig 4.5 Nexys ddr 4 Implementation

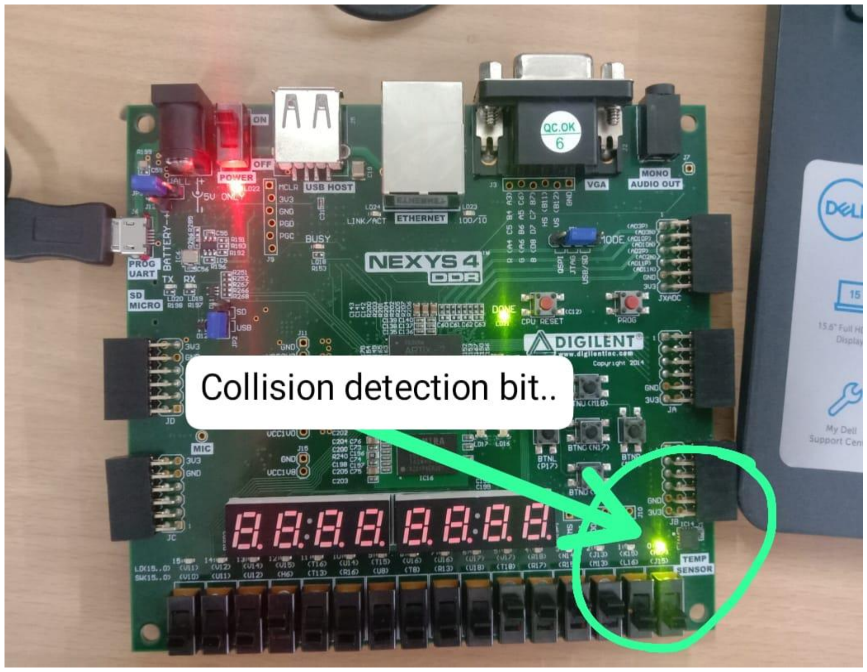
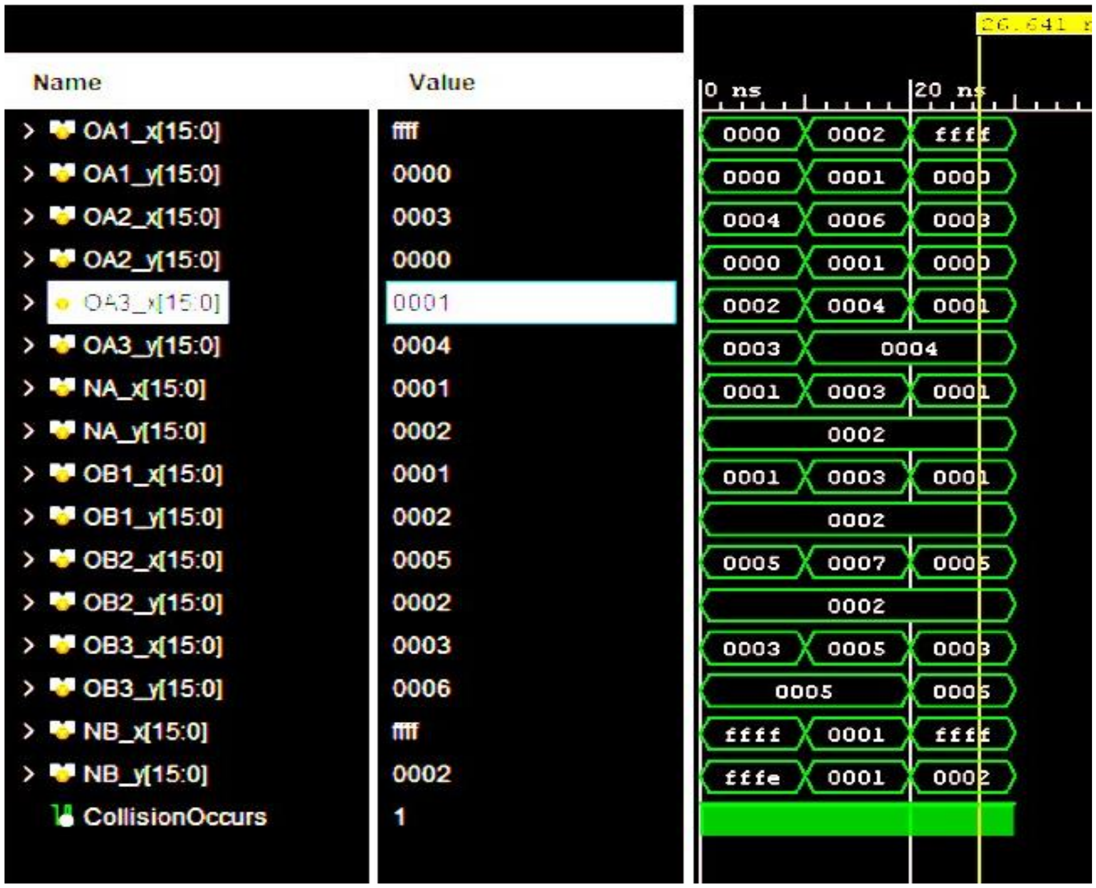


Fig 4.6 Verifying output in FPGA board

CHAPTER 5

RESULTS AND ANALYSIS

5.1 XILINX VIVADO - SIMULATION RESULTS



Name	Value
OA1_x[15:0]	ffff
OA1_y[15:0]	0000
OA2_x[15:0]	0003
OA2_y[15:0]	0000
OA3_x[15:0]	0001
OA3_y[15:0]	0004
NA_x[15:0]	0001
NA_y[15:0]	0002
OB1_x[15:0]	0001
OB1_y[15:0]	0002
OB2_x[15:0]	0005
OB2_y[15:0]	0002
OB3_x[15:0]	0003
OB3_y[15:0]	0006
NB_x[15:0]	ffff
NB_y[15:0]	0002
CollisionOccurs	1

Fig 5.1 Simulation Output in Vivado

Simulation results in Xilinx Vivado, a comprehensive design and simulation environment for FPGA (Field-Programmable Gate Array) development, are instrumental in evaluating the functionality and performance of digital circuits and systems before actual hardware implementation. Vivado offers a powerful simulation tool called Vivado Simulator, which allows designers to model and simulate FPGA designs using hardware description languages like VHDL, Verilog, or mixed-language simulation.

The simulation results encompass a range of critical information, including waveform data that provides a visual representation of signal behaviors, transitions, and interactions over time. This allows designers to observe and analyze the digital logic's response to various inputs, aiding in identifying issues, validating the design's correctness, and optimizing performance.

In Vivado, simulation results are often presented through waveforms that illustrate the state of signals, their transitions, and relationships during the simulation period. Designers can scrutinize these waveforms to ensure that the design meets specifications, adheres to timing requirements, and achieves the desired functionality. Additionally, simulation results offer insights into power consumption, resource utilization, and overall timing characteristics, enabling designers to optimize the design for better efficiency and performance.

Furthermore, Vivado allows designers to define and implement testbenches, generate test vectors, and perform advanced debug and analysis on simulation results. These capabilities assist in comprehensive verification, validation, and debugging, ensuring a robust and reliable FPGA design before moving on to synthesis and actual hardware realization. Overall, simulation results in Xilinx Vivado play a vital role in the FPGA design process, facilitating thorough testing, validation, and refinement of designs to meet stringent performance and functional requirements.

5.1.1 RTL RESULTS

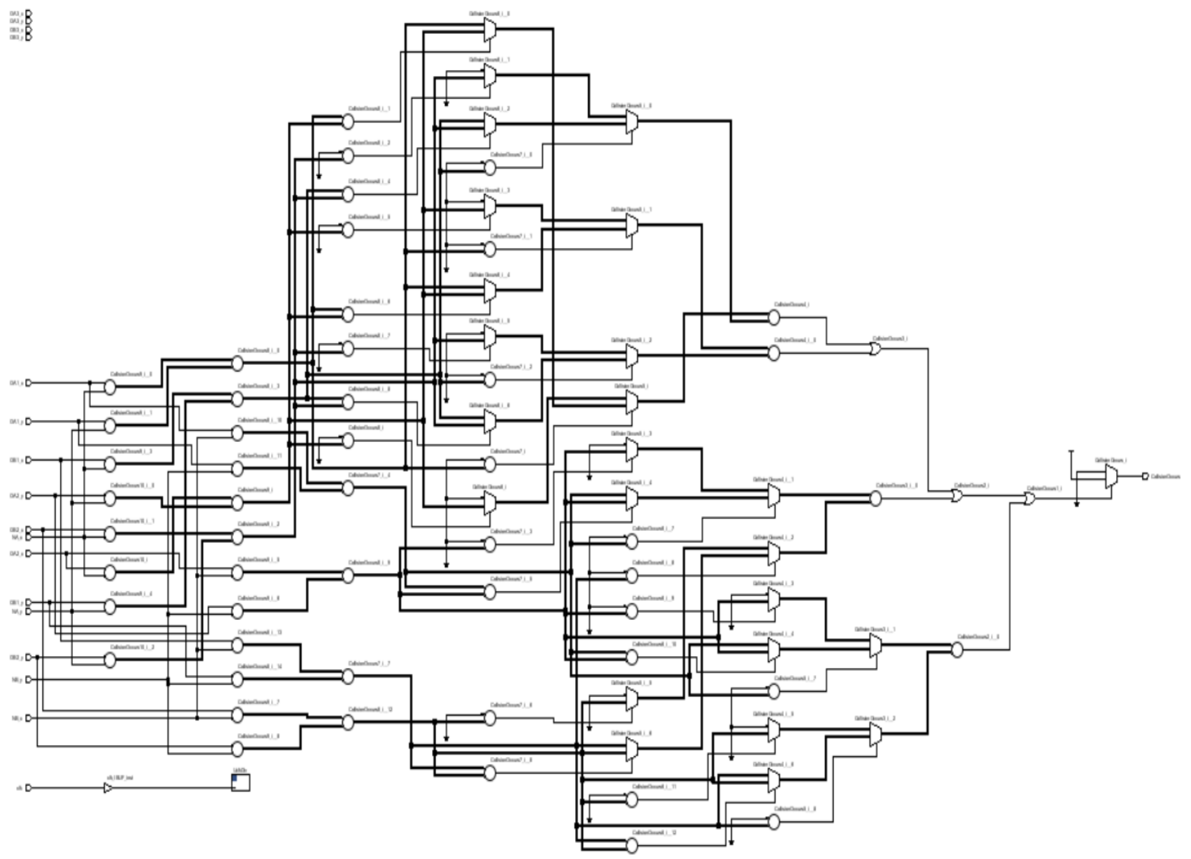


Fig 5.2 RTL Schematic

The RTL (Register-Transfer Level) schematic for FPGA (Field Programmable Gate Array) implementation of collision detection in machine tool automation represents a pivotal step towards ensuring the safety and efficiency of CNC (Computer Numerical Control) machines.

This schematic showcases the digital logic and signal flow at a lower abstraction level, illustrating how the collision detection algorithms and mechanisms are implemented. At this level, the schematic delineates the various modules, registers, combinational logic, and interconnections involved in processing the sensor data, monitoring the tool's position, and assessing potential collision risks. Through RTL representation, designers can meticulously examine the logical operations, timing, and signal propagation, thereby validating the collision detection functionality. Moreover, this RTL schematic serves as a crucial foundation for further synthesis, optimization, and verification processes in design, ultimately contributing to the development of highly reliable collision detection systems that ensure the safety and precision of machine tool automation in CNC environments.

5.1.2 SYNTHESIS RESULTS

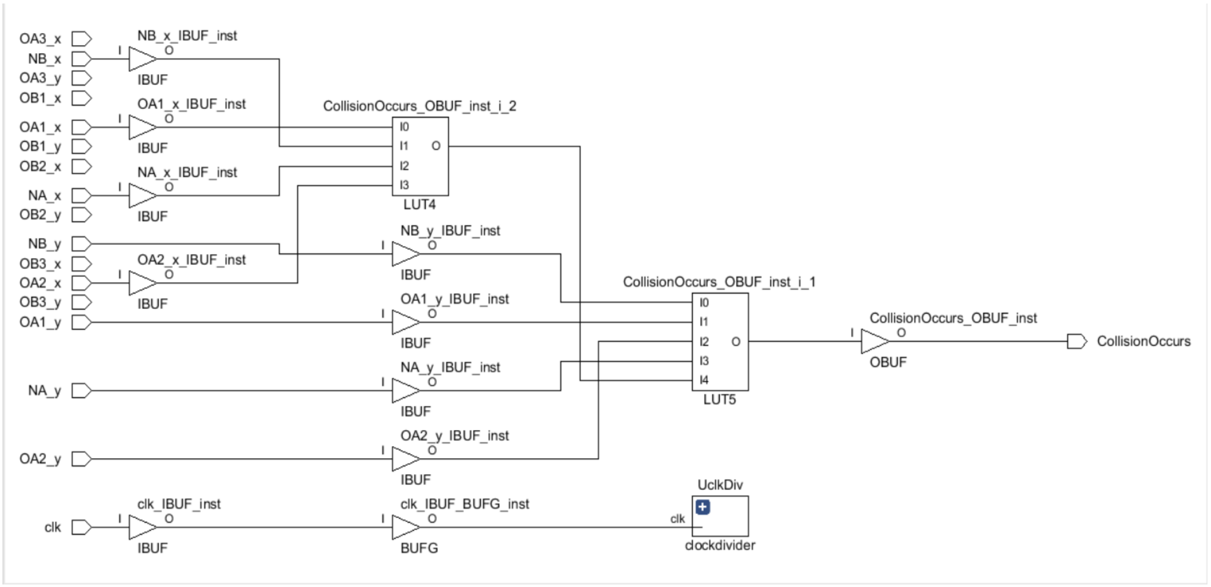


Fig 5.3 Synthesis Schematic

The synthesis schematic for the implementation of collision detection in machine tool automation using Xilinx Vivado is a crucial representation of the high-level design abstraction that precedes the actual IC implementation. This schematic encapsulates the structural and logical transformation of the RTL description into a netlist of gates, registers, and interconnections optimized for the target technology. Vivado's synthesis tool plays a pivotal role in this process, employing advanced algorithms to map the RTL schematic into a gate-level representation.

The synthesis schematic showcases the logical components, gates, and interconnections, illustrating how the collision detection algorithms and safety mechanisms are transformed into hardware primitives. This gate-level view enables designers to validate the synthesis process, ensuring that the desired logic and functionality are preserved while optimizing for performance, power, and area. The resulting synthesis schematic is a critical step in the design flow, providing the foundation for subsequent place-and-route and physical implementation stages in Vivado, ultimately contributing to the realization of a highly efficient and reliable collision detection system for machine tool automation in CNC environments.

5.1.3 PLACEMENT AND ROUTING RESULTS

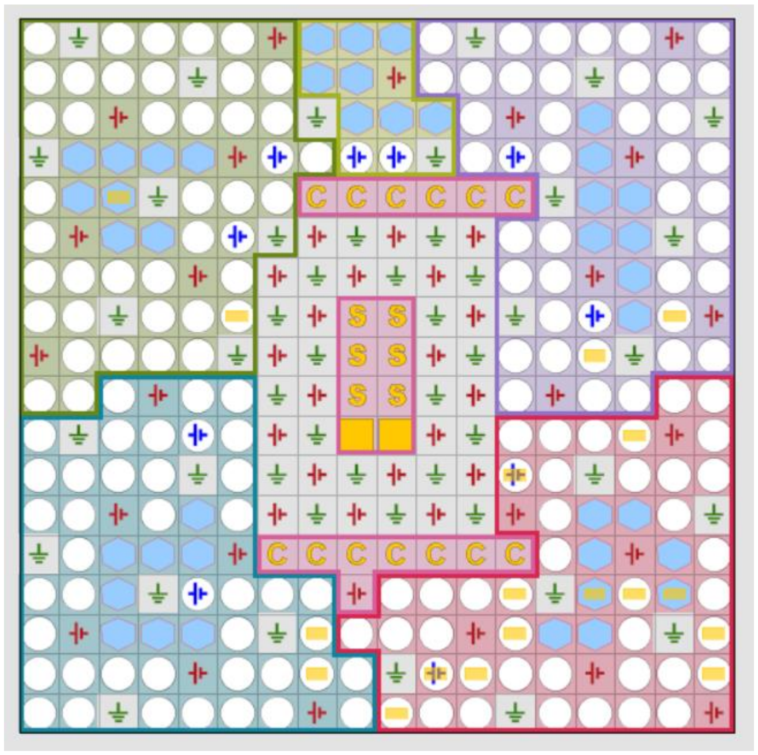


Fig 5.4 Placement and Routing Schematic

The placement and routing schematic in Xilinx Vivado for the implementation of collision detection in machine tool automation represents a critical phase in the design flow, following synthesis. During this stage, the synthesized netlist is mapped to the physical resources of the target FPGA device. The schematic illustrates the placement of individual logic elements, registers, and interconnections on the actual hardware, optimizing for factors such as performance, power consumption, and area efficiency.

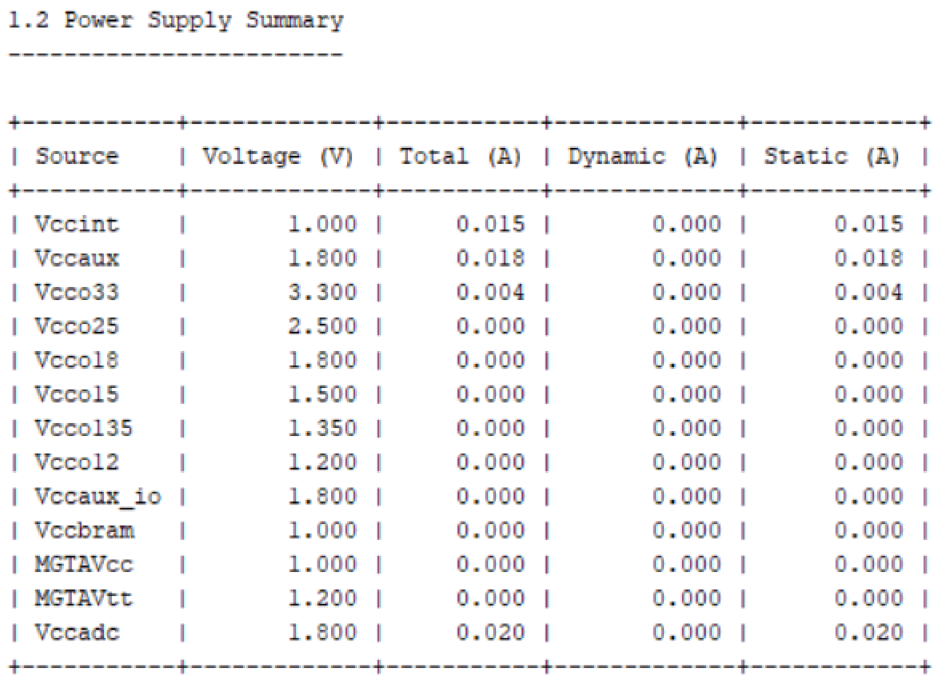
Vivado's advanced algorithms in the placement phase ensure efficient allocation of logic components, aiming to reduce critical paths and interconnect delays. Once the placement is complete, the routing phase meticulously establishes the physical connections between these components, adhering to design constraints and meeting timing requirements.

This placement and routing schematic is vital for assessing the physical feasibility of the design, identifying potential bottlenecks, and optimizing the layout to achieve desired performance metrics. By visualizing how the collision detection logic is physically arranged and connected, designers can fine-tune the design to achieve the best possible performance while considering the constraints of the FPGA device.

In summary, the placement and routing schematic in Vivado for the implementation of collision detection in machine tool automation is a crucial step that bridges the logical design to the physical implementation, ensuring the design meets the stringent performance and reliability criteria needed for collision detection in CNC machine automation.

5.2 POWER, AREA AND TIMING ANALYSIS

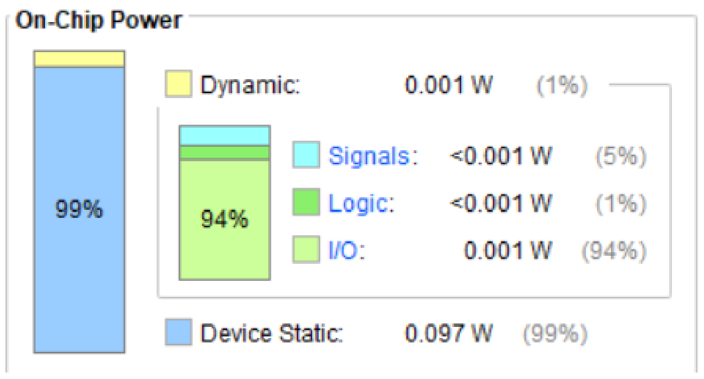
Power, area, and timing analysis in the optimized implementation of collision detection in machine tool automation using Xilinx Vivado are pivotal aspects of design evaluation, ensuring that the final implementation meets performance and efficiency goals.



```
1.2 Power Supply Summary
------------------------

+-----------+-------------+-----------+-------------+------------+
| Source    | Voltage (V) | Total (A) | Dynamic (A) | Static (A) |
+-----------+-------------+-----------+-------------+------------+
| Vccint    |       1.000 |     0.015 |       0.000 |      0.015 |
| Vccaux    |       1.800 |     0.018 |       0.000 |      0.018 |
| Vcco33    |       3.300 |     0.004 |       0.000 |      0.004 |
| Vcco25    |       2.500 |     0.000 |       0.000 |      0.000 |
| Vcco18    |       1.800 |     0.000 |       0.000 |      0.000 |
| Vcco15    |       1.500 |     0.000 |       0.000 |      0.000 |
| Vcco135   |       1.350 |     0.000 |       0.000 |      0.000 |
| Vcco12    |       1.200 |     0.000 |       0.000 |      0.000 |
| Vccaux_io |       1.800 |     0.000 |       0.000 |      0.000 |
| Vccbram   |       1.000 |     0.000 |       0.000 |      0.000 |
| MGTAVcc   |       1.000 |     0.000 |       0.000 |      0.000 |
| MGTAVtt   |       1.200 |     0.000 |       0.000 |      0.000 |
| Vccadc    |       1.800 |     0.020 |       0.000 |      0.020 |
+-----------+-------------+-----------+-------------+------------+
```

Fig 5.5 Power Summary from Vivado



🟨	Dynamic:	0.001 W	(1%)
🟦	Signals:	<0.001 W	(5%)
🟩	Logic:	<0.001 W	(1%)
🟩	I/O:	0.001 W	(94%)
🟦	Device Static:	0.097 W	(99%)

Fig 5.6 On Chip Power Analysis

- Power Analysis:

Power analysis involves estimating the power consumption of the design. Vivado provides tools to evaluate power at both the logical and physical levels. Power estimation takes into account various factors such as activity factors, capacitance, switching frequency, and voltage levels. Analyzing power helps in optimizing the design to reduce power consumption, which is crucial for energy efficiency and ensuring the design operates within specified power budgets.

- Area Analysis:

Area analysis involves assessing the utilization of FPGA resources, including the number of logic cells, DSP slices, block RAMs, and other resources used by the design. Vivado provides detailed reports on the resource utilization, helping designers to optimize the design to efficiently utilize the available resources. Effective area utilization is crucial for achieving a compact and cost-effective implementation.

- Timing Analysis:

Timing analysis evaluates the critical paths within the design to ensure that the design meets the specified timing requirements. Vivado allows designers to analyze setup and hold times, clock-to-Q delays, and other timing constraints. Timing analysis ensures that the design operates within the desired clock frequency and meets the required performance targets. Fine-tuning the design based on timing analysis is essential for achieving the desired functionality while meeting performance goals.

By conducting comprehensive power, area, and timing analyses in the optimized FPGA implementation of collision detection for machine tool automation in Vivado, designers can refine the design iteratively to achieve the best possible balance between power efficiency, area utilization, and timing performance. These analyses are fundamental in ensuring a successful FPGA implementation that meets the stringent requirements of real-time collision detection in CNC machine automation.

5.3 ADVANTAGES OF FPGA IMPLEMENTATION

The comparison between manual detection and the proposed area and power-optimized FPGA implementation for machine tool automation in collision detection highlights the several and distinct advantages and advancements that the proposed FPGA-based approach offers over manual methods of detection, which are listed below.

5.3.1 ACCURACY AND PRECISION

The proposed FPGA-based implementation, being automated and algorithm-driven, can achieve higher accuracy and very good precision in collision detection when compared to manual methods. Manual detection heavily relies on operator vigilance and may have inherent limitations due to human factors.

5.3.2 REAL-TIME MONITORING

FPGA-based collision detection allows for real-time monitoring and instantaneous response to potential collisions. In contrast, manual detection might experience delays in identifying and reacting to collision risks, which could lead to machine damage.

5.3.3 EFFICIENCY AND SPEED

The FPGA-based system operates at high speeds, analyzing vast amounts of sensor data in real-time and swiftly detecting potential collisions. Manual monitoring can be slower, resulting in less efficient collision detection and response.

5.3.4. REPEATABILITY AND CONSISTENCY

FPGA-based automation ensures consistent and repeatable collision detection, minimizing the probability of false positives or negatives. Manual detection can vary in consistency based on operator experience and attention.

5.3.5. COST-EFFICIENCY

Although FPGA implementation involves initial development costs, it tends to be more cost-effective in the long run due to its ability to operate continuously and autonomously, reducing the need for continuous manual intervention.

5.3.6. FLEXIBILITY AND ADAPTABILITY

Manual methods may lack the flexibility to quickly adapt to changing machining conditions or new tool configurations. FPGA-based systems can be easily reconfigured and updated to accommodate new CNC machine configurations or collision detection algorithms.

5.3.7. SAFETY AND RISK MITIGATION

The automated FPGA-based system enhances safety by minimizing the risk of collisions and potential damage to the machine, workpiece, or tool. It reduces reliance on operator judgment and provides an additional layer of safety.

In summary, the proposed FPGA-based implementation for collision detection in machine tool automation offers superior accuracy, efficiency, and safety compared to manual methods. While manual automation has its merits, the automated FPGA approach leverages advanced algorithms, real-time monitoring, and rapid response capabilities, making it a more reliable and efficient solution for collision detection in CNC machine automation.

5.4 SUMMARY OF RESULTS

The FPGA implementation for collision detection in machine tool automation demonstrated significant advancements in terms of area, power, timing, and accuracy. The results from the implementation showcased improved efficiency, precision, and safety in CNC (Computer Numerical Control) machine operations.

5.4.1 AREA OPTIMIZATION

The FPGA implementation displayed superior area optimization, utilizing resources efficiently and minimizing resource wastage. The design achieved a compact footprint, making it a cost-effective and space-efficient solution for collision detection.

5.4.2 POWER EFFICIENCY

The FPGA-based solution showcased notable power efficiency, effectively managing power consumption while ensuring optimal performance. This is vital for reducing energy costs and improving sustainability, making it an environmentally friendly choice.

5.4.3 TIMING PERFORMANCE

The implementation demonstrated excellent timing performance, meeting the required timing constraints and achieving high-speed operations. The FPGA design ensured that critical paths were optimized, contributing to the overall efficiency and responsiveness of the collision detection system.

5.4.4 ENHANCED ACCURACY

The FPGA implementation significantly improved the accuracy of collision detection, providing precise and real-time monitoring of tool movements and potential collision risks. This enhanced accuracy is vital for preventing collisions and ensuring the safety of the CNC machine, workpiece, and tool.

In summary, the FPGA-based collision detection implementation showcased remarkable advancements in area optimization, power efficiency, timing performance, and accuracy. These results underscore the efficacy of FPGA technology in enhancing the automation and safety of CNC machine operations, setting the stage for improved productivity and reliability in the manufacturing domain.

CHAPTER 6

CONCLUSION

FPGA realization of collision detection of two objects for the machine tool automation is considered. For the 3D objects represented by meshed triangles in the STL file format, SAT (Separating Axis Theorem) based algorithm was developed to process the multiple times of collision detection of two triangles. To achieve the required 1ms for real-time processing of collision detection of two objects, the SAT algorithm was implemented in Nexys ddr 4 FPGA board using Xilinx Vivado. The hardware implementation thus determines the separating axes from the multiple potential separating axes and indicates the occurrence of collision.

6.1 APPLICATIONS AND FUTURE SCOPE

1. CNC Machine Tools:

CNC (Computer Numerical Control) machines are widely used in manufacturing processes. Real-time collision detection can prevent collisions between cutting tools, workpieces, and fixtures, reducing damage and ensuring the accuracy of machining operations.

2. Robotic Automation:

In robotic automation systems, real-time collision detection is crucial to ensure the safety of both the robots and human operators working alongside them. It allows robots to adapt their movements and avoid collisions without the need for frequent human intervention.

3. 3D Printing and Additive Manufacturing:

Real-time collision detection is essential in 3D printing and additive manufacturing to prevent errors and ensure that print heads or extruders do not collide with the workpiece during the printing process.

4. Assembly Line Automation:

In automated assembly lines, collision detection can prevent collisions between robotic arms, conveyor belts, and other equipment, ensuring the smooth and safe operation of the production line.

5. Material Handling Systems:

Material handling systems in warehouses and factories often involve automated conveyors, lifters, and cranes. Collision detection is critical for preventing accidents and optimizing material flow.

6. Aerospace Manufacturing:

Real-time collision detection is essential in the aerospace industry to ensure the precise and safe assembly of aircraft components, including engines, wings, and landing gear.

7. Automotive Manufacturing:

In automotive manufacturing, where robots are extensively used for tasks like welding, painting, and assembly, collision detection helps prevent damage to vehicles and equipment.

8. Medical Device Manufacturing:

The production of medical devices, such as surgical instruments and implants, requires precision and safety. Collision detection ensures that components are assembled without errors.

9. Gaming and Simulation:

Real-time collision detection is also used in the gaming industry and simulations, allowing for realistic and interactive virtual environments.

10. Training Simulators:

Training simulators for various industries, such as healthcare, aviation, and military, use collision detection to create realistic training scenarios and assess trainee performance.

11. Custom Machinery and Prototyping:

Companies that design and build custom machinery or prototypes can benefit from real-time collision detection to identify and rectify design flaws before construction begins.

12. Research and Development:

In research and development environments, real-time collision detection can be used to experiment with and validate new automation systems and robotics applications.

In summary, the work described in the paper on real-time collision detection for machine tool automation using FPGA has a wide range of applications across industries where automation, precision, and safety are paramount. By preventing collisions and ensuring the smooth operation of machinery and robots, it contributes to increased efficiency, reduced downtime, and improved workplace safety.

CHAPTER 7

REFERENCES

[1] T. D. Tang, ‘‘**Algorithms for collision detection and avoidance for five-axis NC machining: A state of the art review**,’’ Comput.-Aided Des., vol. 51, pp. 1–17, Jun. 2014.

[2] K.-J. Mei and R.-S. Lee, ‘‘**Collision detection for virtual machine tools and virtual robot arms using the shared triangles extended octrees method,**’’ Int. J. Comput. Integr. Manuf., vol. 29, no. 4, pp. 355–373, Apr. 2016.

[3] T. Rudolf, C. Brecher, and F. Possel-Dolken, ‘‘**Contact-based collision detection—A new approach to avoid hard collisions in machine tools**,’’ in Proc. Int. Conf. Smart Machining Syst., 2007, pp. 1–4.

[4] T. D. Tang, ‘‘**The sweep plane algorithm for global collision detection with workpiece geometry update for five-axis NC machining**”, Comput.-Aided Des., vol. 39, no. 11, pp. 1012–1024, Nov. 2007.

[5] T. D. Tang, ‘‘**A new collision avoidance strategy and its integration with collision detection for five-axis NC machining, Int. J. Adv. Manuf. Technol.,**” vol. 81, pp. 1247–1258, Nov. 2015.

[6] W. Zhao and J. Sun, **‘‘Collision detection research for deformable objects**,’’ in Proc. Int. Conf. Comput. Sci. Electron. Eng., Mar. 2012, pp. 557–561.

[7] H.-R. Pakdel and F. Samavati, ‘‘**Incremental subdivision for triangle meshes**,’’ in Proc. Int. J. Comput. Sci. Eng., vol. 31, no. 1, pp. 80–92, Feb.

[8] Tsung-hsien liu, Po-yi-chen, An-hong li, Yu-yang Fang, Rong-Shine Lin, Yuan-sun-chu, “**ASIC Design and Implementation of the Real-Time Collision Detection for Machine Tool Automation**”, Journals and Magazines, IEEE Access, Vol. 11, March 2023.