The Proposed Genetic FPGA Implementation For Path Planning of Autonomous Mobile Robot

O.Hachour

***Abstract*—**This paper proposes Genetic Algorithms (GAs) for path Autonomous Mobile Vehicle (AMV). This approach has an advantage of adaptivity such that the GA works perfectly even if an environment is unknown. First, we present a software implementation GA path planning in a terrain. The results gotten of the GA on randomly generated terrains are very satisfactory and promising . Second, we discuss extensions of the GA for solving both paths planning and trajectory planning using a Single Static Random Access Memory (SRAM) for Field Programmable Gate Array (FPGA). This new design methodology based upon a VHDL description of the path planning has the two (02) advantages : to present a real autonomous task for mobile robots, and being generic and flexible and can be changed at the user demand. The results gotten are promising.

***Keywords*—**Autonomous Mobile Vehicle (AMV), FPGA implementation, Genetic Algorithm (GA) , and VHDL Simulation.

I. INTRODUCTION

Autonomous robots which work without human operators are required in robotic fields. In order to achieve tasks, autonomous robots have to be intelligent and should decide their own action. When the autonomous robot decides its action, it is necessary to plan optimally depending on their tasks. More, it is necessary to plan a collision free path minimizing a cost such as time, energy and distance. When an autonomous robot moves from a point to a target point in its given environment, it is necessary to plan an optimal or feasible path avoiding obstacles in its way and answer to some criterion of autonomy requirements such as : thermal, energy, time, and safety for example. thermal, energy, time, and safety for example. Therefore, the major main work for path planning for autonomous mobile robot is to search a collision free path. Many works on this topic have been carried out for the path planning of autonomous mobile robot.

Motion planning is one of the important tasks in intelligent control of an autonomous mobile robot . It is often decomposed into path planning and trajectory planning. Path planning is to generate a collision free path in an environment with obstacles and optimize it with respect to some criterion. Trajectory planning is to schedule the movement of a mobile robot along the planned path . A wide variety of approaches have been considered, but these can broadly be categorized into on-line and off-line techniques. However, few algorithms have been developed for on-line motion planning in a time-varying or unknown.

Previous research on the path planning can be classified as following one of two approaches: model-based and sensor – based. In general, the model-base approach considers obstacle avoidance globally it uses prior models to describe known obstacles completely in order to generate a collision free path. In contrast, the sensor-based approach aims to detect and avoid unknown.

A robotic vehicle is an intelligent mobile machine capable of autonomous operations in structured and unstructured environment, it must be capable of sensing (perceiving its environment), thinking (planning and reasoning), and acting (moving and manipulating). But, the current mobile robots do relatively little that is recognizable as intelligent thinking, this is because:

1) Perception does not meet the necessary standards.
2) Much of the intelligence is tied up in task specific behavior and has more to do with particular devices and missions than with the mobile robots in general.
3) Much of the challenge of the mobile robots requires intelligence at subconscious level.

A robotic systems capable of some degree of self-sufficiency is the overall objective of an AMR and are required in many fields [2,3]. The focus is on the ability to move and on being self-sufficient to evolve in an unknown environment for example. Thus, the recent developments in autonomy requirements, intelligent components, multi-robot systems, and massively parallel computer have made the AMR very used, notably in the planetary explorations, mine industry, and highways [1,9,10,11,12,13].

This paper deals with the intelligent path planning of AMR in an unknown environment, by applying the principles of the genetic algorithms. The aim of this paper is to develop an AMR for the AMR stationary obstacle avoidance to provide them more autonomy and intelligence .This technology GA based on intelligent computing as becoming useful as alternate approach may be able to replace the classical approaches such as: recognition, learning, decision-making , and action (the principle obstacle avoidance problems). This approach can be realized in efficient manner and has proved to be superior to combinatorial optimization techniques, due to the problem complexity.

Recently, applications of genetic algorithms to path Planning or trajectory planning have been recognized. GA is a search strategy using a mechanism analogous to evolution of

life in nature. The GAs, which are evolutionary have recently emerged from study of the evolution mechanisms and are searching strategies suitable for finding the globally optimal solution in a large parameter space. They are based on learning mechanisms. It has widely been recognized that GA works even for complex problems such that traditional algorithms cannot find a satisfactory solution within a reasonable amount of time.

To benefit from using such technology, another technological advance in hardware has revealed the soft-computing in order to give a high speed processing that could be provided through massively parallel implementation management and so more autonomy requirements such as power, thermal and communication management are obtained in real time. GA approach can be implemented either in analogue or digital way. Analog circuits are sensitive to noise and temperature changes and inter-chip variations make manufacturing functionally identical circuits very difficult. Digital integrated circuits, on the other hand, is easily manufactured and are functionally identical. Nowadays, FPGA are gaining momentum in digital design. They are used for a wide range of application including rapid prototype, glue logic for microprocessor, and hard–wired simulation. Moreover, the FPGA circuit offers attractive features for hardware designer in comparison with other VLSI techniques. Then, this new design methodology of FPGA offers more performance than the software implementation (an easiness of implementation, shortest possible time, and rapid execution by AMR) [9,10,11,12,13]. In this paper, we discuss the possibility to deal with hurdle by proposing a new approach permitting the mapping of an entire navigation approach (planning, intelligent control for obstacle avoidance) into a single Xilinx's.

II. Robots and Motion planning

A Necessity of Intelligent Autonomous Robot

A robot is a "device" that responds to sensory input by running a program automatically without human intervention. Typically, a robot is endowed with some artificial intelligence so that it can react to different situations it may encounter. The robot is referred to be all bodies that are modeled geometrically and are controllable via a motion plan. A robotic vehicle is an intelligent mobile machine capable of autonomous operations in structured and unstructured environment. It must be capable of sensing thinking and acting. The mobile robot is an appropriate tool for investigating optional artificial intelligence problems relating to world understanding and taking a suitable action, such as , planning missions, avoiding obstacles, and fusing data from many sources.

Industrial robots used for manipulations of goods; typically consist of one or two arms and a controller. The term controller is used in at least two different ways. In this context, we mean the computer system used to control the robot, often called a *robot work-station* controller. The controller may be programmed to operate the robot in a number of way; thus distinguishing it from hard automation. The controller is also responsible for the monitoring of auxiliary sensor that detect the presence, distance, velocity, shape, weight, or other properties of objects. Robots may be equipped with vision systems, depending on the application for which they are used. Most often, industrial robot are stationary, and work is transported to them by conveyer or robot carts, which are often called autonomous guided vehicles (AGV). Autonomous guided vehicles are becoming increasingly used in industry for materials transport. Most frequently, these vehicles use a sensor to follow a wire in the factory floor. Some systems employ an arm mounted on an AGV.

Robot programmability provides major advantages over hard automation. If there are to be many models or options on a product, programmability allows the variations to be handled easily. If product models change frequently; as in the automotive industry, it is generally far less costly to reprogram a robot than to rework hard automation. A robot workstation may be programmed to perform several tasks in succession rather than just a single step on a line. This makes it easy to accommodate fluctuations in product volume by adding or removing workstations. Also; because robots may be reprogrammed to do different tasks; it is often possible to amortize their first cost over several products. Robots can also perform many applications that are poorly suited to human abilities. These include manipulation of small and a large object like electronic parts and turbine blades, respectively. Another of these applications is work in unusual environments like clean rooms, furnaces, high-radiation areas, and space. Japan has led the world in the use of robots in manufacturing. The two sectors making heaviest use of robots are the automotive and electronics industries. Interest in legged locomotion has been stimulated by application in traversing rough terrain and in unmanned exploration of unknown environment. Aside from electronic motivation, there are many unanswered scientific question about how biological organism produce the remarkable sensor motor behaviour that we observe. Finally, the notion of simulating biological organism has a certain instinctive reproductive appeal and offers the possibility of satisfying our curiosity as to how come to be as we are.

Robot programming is the mean by which a robot is instructed to perform it task .The guiding for example, is the process of moving a robot trough a sequence of motion to "*show it*" what it must do. One guidance method is to physically drag around the end effectors of the robot, while it records joint position at frequent intervals along the trajectory. The robot then plays back the motion just as it was recorded. An alternative is a master-slave or teleoperation configuration. Early systems of this type were first used or manipulate radioactive material remotely. Teloperator techniques are now employed to guide the space shuttle manipulator. Guiding may also be applied using a *teach pendant*; which is a box keys that are used to command the robot. Several modes of operation are often available on the each pendant. Guiding is limited as a

robot-programming technique, because it does not provide conditionality or iteration. Some systems provide called extended guiding capability that include teaching in a coordinate system that may be moved at run-time and conditional branching between motion sequence.

Robot programs must command robots to move: thus, the way in which motion is specified is important. Also, the program use information obtained from sensors. One way of using sensory information is to monitor sensor until prescribed condition occur and then perform or terminate a specified action is to use feedback from sensors to modify the robots behaviour continuously. A different approach is used to describe the behaviour of the system in terms of relationships that are to be maintained with respect to force, velocity, position, and another measured and controlled quantities.

The issue of concurrent execution of program statements has received widespread attention in computer science. In robotics, the problem is computational processes. Many of the considerations are very similar, but there are important differences, too. Physical processes cannot; in general, be suspended and resumed as computations can, because of physical laws (like Newton's laws) and the presence of external force (like gravity). Concurrency is required for speed, because it is highly undesirable to serialize a set of mechanical task.

Nowadays, Intelligent Autonomous Vehicles (IAV) can carry out task in various environments by themselves like human. These intelligent autonomous systems are very useful in many fields as industry and planetary explorations. However, they are all semi-autonomous and need some human operators. In fact, the future factory needs all flexible and robust IAV.

Additionally, Intelligent Autonomous Vehicles (IAV) are becoming more and more interesting for underwater, terrestrial, and space applications. These mechanical systems are constructed to respond any traditional working as in construction and agriculture. Previously, certain industrial operations required human skills may be tedious and exceptionally hardly ever. Above all, repetitive operations can result in reductions in quality control, as in visual inspections tasks. Also, these repetitive actions may be hazardous health risks as exposure to unsafe materials like radioactive and high pressure in underwater applications. So, the presence of human workers in these environments may be perilous which need the necessity to be replaced by intelligent systems, these systems can move, react, and carry out tasks in various environments by themselves like human.These untethered systems become then free of constraints than is currently used with remotely operated systems. This involves that intelligence and sufficient power must be integrated at a higher level while the robot communicates with the environment. So, AMR must be able to perform purposeful behaviours in the real-world. These vehicles capable of coping with real dynamic worlds populated by other robots and humans constitute powerful test-beds to study and exercise the multi-faceted aspects of intelligence. Using a robot, a Computer Aided Design (CAD) system is typically used to model the robot workstation, arts, and auxiliary equipment. Then the simulated robot is programmed and its task executed in the simulated environment. Collisions between object may be checked by using a collision-detection algorithm. The utility of off-line programming for debugging robot program is limited by the inability of most commercial solid modelling program to include error tolerance information in geometrical model. If a model does not include uncertainties in part position, part dimensions, and robot position, the simulation will succeeds in situations where a real application would fail. Another difficulty with simulation is that force sensing must be modelled by collision detection, which is computationally expensive. This make it inconvenient to model guarded moves and all but impossible to model force-guided compliant motion such as surface following.

The environment force prevents the robot from moving and turning towards obstacles by giving the user the distance information between the robot and the obstacle in a form of force. This force is similar to the traditional potential force field for path planning of mobile robot. However, the environment force is different from the potential force in some aspects. First there is no attention to a goal since we assume that the goal position is unknown. Secondly, only obstacles in the "relevant" area (according to the logical position of the interface) are consider, i.e. the obstacles that are far, or in the direction opposite to the movement of the robot are not relevant. In this context, a full range of advanced interfaces for vehicle control has been investigated by the researchers. These works demonstrate that obstacle detection and collision avoidance is improved with good results.

Classical artificial intelligence systems presuppose that all knowledge is stored in a central database of logical assertions or other symbolic representation and that reasoning consist largely of searching and sequentially updating that database. While this model has been successful for disembodied reasoning system, it is problematic for robots. Robots are distributed systems; multiple sensory, reasoning, and motor control processes run in parallel, often on separate processor hate rate only loosely coupled with one another. Each of these procure necessarily maintains its own separate, limited representation of the world and task; requiring them to constantly synchronize with the central knowledge base is probably unrealistic. Automated reasoning systems are typically built on a transaction-oriented model of computation .knowledge of the world is stored in a database of assertion in some logical language, indexed perhaps by predicate name.
For one problem discussed, for example, the robot should have reflective pools that give the robot access to its own internal state:

The behavioural pool

This one holds binding between tag and specific robot behaviour. Each behaviour continually compare it tag to the tag on a global call signal. Whenever a behavioural detects a match, it activate itself. Active behaviour also drive a global

running signal with bit-vector of their tag. The signal therefore hold the tag of all running behaviour, allowing any part of the system monitoring the signal to determine whether the behaviour bound to a given tag is running.

The proposition pool

The pool generates a true Signal comprised of the set of all tag bound to proposition that are presently true. This allow one component of the system to "pass" a signal to another component by binding it to tag that has been agreed upon in advance. The receiving component can then monitor the signal by inspecting the appropriate bit of the true signal.

The predicate pool

The predicate pool generates vector of signal, indexed by role, whose elements hold the extension of all bound predicates – role 0 in element 0 , role 1 in element 1, etc. again, this provides an indirection facility for passing signal between components . In this context, we can include a marker- passing semantic net. Node within the net can be bound to role tag and then propagated a marker along links in the net to perform retrieval and inference from long-term memory.

It is important to understand that a given object or concept might be represented in several of these pools simultaneously, with each pool representing different aspect of the object. This is supported in part by allowing element of different pool to share a sing tag register. For example, the lexicon pool entry of the word "show" , the behaviour SHOW, and the semantic net node representing information about the behaviour all share a common tag register. Therefore, when the parser bind "show" to a role, the behaviour that can implement the verb is automatically bound to the same role at the same time. A several works were demonstrated in this domain, many researchers have attended this problem to give successful reasoning systems. They have discussed a lot of an alternate class of architectures-tagged behaviour-based systems- that support a large subset of the capabilities of classical artificial intelligence architecture, including limited quantified inference, forward and backward-chaining, simple natural language question answering and command following, reification, and computational reflection, while allowing object representation, to remain distributed across multiple sensory and representational modalities.

B. Autonomy requirements

Several autonomy requirements must be satisfied to well perform the tasks of AMR, this is summarized in some in the following section.

Thermal

To carry out tasks in various environments as in space applications, the thermal design must be taken into account, especially when the temperature can vary significantly. At ambient temperatures, the limited temperature -sensitive electronic equipment on-board must be placed in a thermally insulated compartments. The thermal environment of Mars challenges the thermal control system. In the course of a Martian day the temperature can vary from 140K to 300K.

Energy

For a specified period, AMR can operate autonomously, one very limited resource for underwater and space applications are energy. So, AMR usually carry a rechargeable energy system, appropriately sized batteries on-board.

Communication Management

The components on-board the vehicle and on-board the surface station must be interconnected by a two-way communication link. As in both underwater and space applications, a data management system is usually necessary to transfer data from AMR to terrestrial storage and processing stations by two-way communication link. Indeed, the data management system must be split between components of the vehicle and surface station. Thus, the vehicle must be more autonomous and intelligent to perform and achieve the tasks. Due to the limited resources and weight constraints, major data processing and storage capacities must be on the surface station. Although individual vehicles may have wildly different external appearances, different mechanisms of locomotion, and different missions or goals, many of the underlying computational issues involved are related to sensing and sensor modelling spatial data representation, and reasoning.

Mechanical design

The mechanical design of Intelligent Autonomous Robots is the result of an integration approach considering several criteria related with perception, control, and planning issues in addition to structural design and other mechanical requirements.

C. Criteria to satisfy by vehicles to be autonomous and intelligent

To evaluate the performance of AMR which are intelligent and autonomous vehicle, the robot must perform the following criteria:

Intelligence

A robotic system capable of some degree of self-sufficiency is the primary goal of Intelligent Autonomous Vehicles . Thus, the robot must achieve his task with more autonomy and intelligence. Also, the vehicle reacts to unknown static and dynamic obstacles with safety not to endanger itself or other objects in the environment. Near Safety, the reliability is taken into account in the field of robotics; it is the probability that the required function is executed without failure during certain duration.

Navigation

Navigation is the ability to move and on being self-sufficient. The AMR must be able to achieve these tasks: to avoid obstacles, and to make one way towards their target. In fact, recognition, learning, decision-making, and action constitute principal problem of the navigation. One of the specific characteristic of mobile robot is the complexity of their environment. Therefore, one of the critical problems for the mobile robots is path planning, which is still an open one to be studying extensively. Accordingly, one of the key issues in the design of an autonomous robot is navigation, for which, the navigation planning is one of the most vital aspect of an autonomous robot. Therefore, the space and how it is represented play a pivotal role in any problem solution in the domain of the mobile robot, because:

- It provides the necessary information to do path planning.
- It gives information for monitoring the position of the robot during the execution of the planned path.

Several models have been applied for environment where the principle of navigation is applied to do path planning. For example, a grid model has been adopted by many researchers, where the robot environment is dividing into many line squares and indicated to the presence of an object or not in each square. On line encountered unknown obstacle are modelled by piece of "wall", where each piece of "wall" is a straight-line and represented by the list of its two end points. This representation is consistent with the representation of known objects, while it also accommodates the fact the only partial information about an unknown obstacle can be obtained from sensing at a particular location.

Many researches which have been done within this field, some of them used a "visibility graph" to set up a configuration space that can be mapped into a graph of vertices between which travel is possible in a straight line. The disadvantage of this method is time consuming. At the opposite, some researches have been based on dividing the world map into a grid (explained before) and assign a cost to each square. Path cost is the sum of the cost of the grid squares through which the path passes. A grid model has been adopted by many authors, where the robot environment is divided into many squares and indicated to the presence of an object or not in each square[6,9]. A cellular model, in other hand, has been developed by many researchers where the world of navigation is decomposed into cellular areas, some of which include obstacles. More, the skeleton models for map representation in buildings have been used to understand the environment's structure, avoid obstacles and to find a suitable path of navigation. These researches have been developed in order to find an efficient automated path strategy for mobile robots to work within the described environment where the robot moves. In the figure 1 we present one example of navigation approach using a square cellule grid for the movement. Another example is presented in the figure 2 to find an optimal path to navigate intelligibly avoiding the obstacles. This example shows the way on which the scene of navigation is decomposed. The figure 3 illustrates one model of navigation where the polygonal model is used for the navigation.

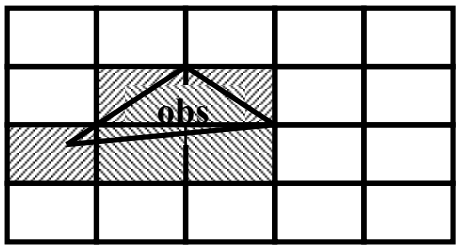

Fig. 1 an example of a square cellule grid navigation

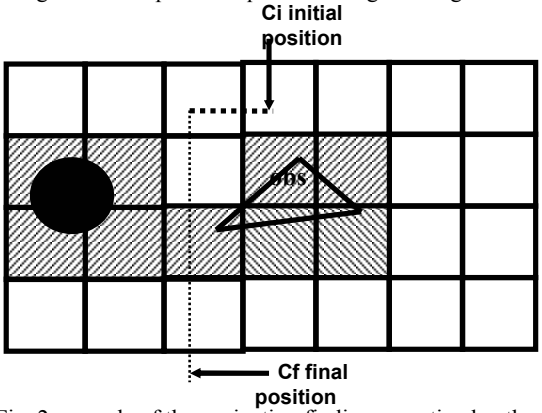

Fig. 2 example of the navigation finding an optimal path

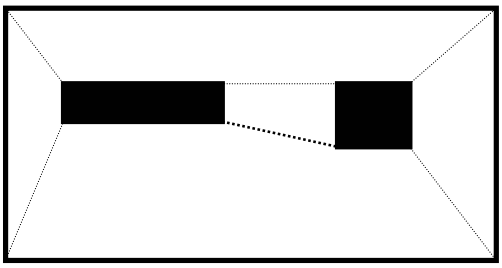

Fig. 3 Example of the polygonal model of navigation

Motion planning

The goal of the navigation process of mobile robots is to move the robot to a named place in a known, unknown or

partially known environment. In most practical situations, the mobile robot can not take the most direct path from the start to the goal point. So , path planning techniques must be used in this situation, and the simplified kinds of planning mission involve going from the start point to the goal point while minimizing some cost such as time spent, chance of detection, or fuel consumption.

Often, a path is planned off-line for the robot to follow, which can lead the robot to its destination assuming that the environment is perfectly known and stationary and the robot can rack perfectly. Early path planners were such off-line planners or were only suitable for such off-line planning.

However, the limitations of off-line planning led a researcher to study on-line planning, which relies on knowledge acquired from sensing the local environment to handle unknown obstacles as the robot traverses the environment.

One of the key issues in the design of an autonomous robot is navigation, for which, the navigation planning is one of the most vital aspect of an autonomous robot. Therefore, the space and how it is represented play a primary role in any problem solution in the domain of mobile robots because it is essential that the mobile robot has the ability to build and use models of its environment that enable it to understand the scene navigation's structure. This is necessary to understand orders, plan and execute paths.

Moreover, when a robot moves in a specific space, it is necessary to select a most reasonable path so as to avoid collisions with obstacles. Several approaches for path planning exist for mobile robots, whose suitability depends on a particular problem in an application. For example, behavior-based reactive methods are good choice for robust collision avoidance. Path planning in spatial representation often requires the integration of several approaches. This can provide efficient, accurate, and consistent navigation of a mobile robot.

The major task for path-planning for single mobile robot is to search a collision –free path. The work in path planning has led into issues of map representation for a real world. Therefore, this problem considered as one of challenges in the field of mobile robots because of its direct effect for having a simple and computationally efficient path planning strategy. For path planning areas, it is sufficient for the robot to use a topological map that represents only the different areas without details such as office rooms. The possibility to use topological maps with different abstraction levels helps to save processing time. The static aspect of topological maps enables rather the creation of paths without information that is relevant at runtime. The created schedule, which is based on a topological map, holds nothing about objects which occupy the path. In that case it is not possible to perform the schedule. To get further actual information, the schedule should be enriched by the use of more up-to date plans like egocentric maps.

Topological path planning is useful for the creation of long distance paths, which support the navigation for solving a task. Therefore, those nodes representing for example, free region space are extracted from a topological map, which connect a start point with a target point. The start point is mostly the actual position of the robot. To generate the path, several sophisticated and classical algorithms exist that are based on graph theory like the algorithm; of the shortest path. To give best support for the path planning it could be helpful to use different abstraction levels for topological maps. For example, if the robot enters a particular room; of an employee for postal delivery, the robot must use a topological map that contains the doors of an office building and the room numbers.

Topological maps can be used to solve abstract tasks, for example, to go and retrieve objects whose positions are not exactly known because the locations of the objects are often changed. Topological maps are graphs whose nodes represent static objects like rooms, and doors for example. The edges between the nodes are part's relationships between the objects. For example, an abstract task formulated

The navigation planning is one of the most vital aspects of an autonomous robot. Navigation is the science (or art) of directing the course of a mobile robot as the robot traverses the environment. Inherent in any navigation scheme is the desire to reach a destination without getting lost or crashing into any objects. The goal of the navigation system of mobile robots is to move the robot to a named place in a known, unknown, or partially known environment. In most practical situations, the mobile robot can not take the most direct path from start to the goal point. So, path finding techniques must be used in these situations, and the simplest kinds of planning mission involve going from the start point to the goal point while minimizing some cost such as time spent, chance of detection, etc. When the robot actually starts to travel along a planned path, it may find that there are obstacles along the path, hence the robot must avoid these obstacles and plans a new path to achieve the task of navigation.

Systems that control the navigation of a mobile robot are based on several paradigms. Biologically motivated applications, for example, adopt the assumed behavior of animals. Geometric representations use geometrical elements like rectangles, polygons, and cylinders for the modeling of an environment. Also, systems for mobile robot exist that do not use a representation of their environment. The behavior of the robot is determined by the sensor data actually taken. Further approaches were introduced which use icons to represent the environment.

Several approaches for path planning exist for mobile robots, whose suitability depends on a particular problem in an application. For example, behavior-based reactive methods are good choice for robust collision avoidance. Path planning in spatial representation often requires the integration of several approaches. This can provide efficient, accurate, and consist navigation of a mobile robot. It is sufficient for the robot to use a topological map that represents only the areas of navigation (free areas , occupied areas of obstacles). It is essential the robot has the ability to build and uses models of its environment that enable it to understand the environment's structure. This is necessary to understand orders, plan and execute paths.

Path planning in spatial representation often requires the integration of several approaches. This can provide efficient, accurate, and consist navigation of a mobile robot. It is sufficient for the robot to use a topological map that represents only the areas of navigation (free areas, occupied areas of obstacles). It is essential the robot has the ability to build and uses models of its environment that enable it to understand the environment's structure. This is necessary to understand orders, plan and execute paths [4].

In this paper, a simple and efficient navigation approach for autonomous mobile robot is proposed in which the robot navigates, avoids obstacles and attends its target. Note that, the algorithm described here is just to find a feasible and flexible path from initial area source to destination target area, flexible because the user can change the position of obstacles it has no effect since the environment is unknown. This robust method can deal a wide number of environments and gives to our robot the autonomous decision of how to avoid obstacles and how to attend the target. More, the path planning procedure covers the environments structure and the propagate distances through free space from the source position. For any starting point within the environment representing the initial position of the mobile robot, the shortest path to the goal is traced. The algorithm described here therefore is to develop a method for path planning by using simple and computationally efficient-way to solve path planning problem in an unknown environment without consuming time, lose energy, un-safety of the robot architecture. The shortest path is obtained by using GA, this biological process has an advantage of adaptivity such that the GA works perfectly even if an environment is unknown. GAs combine survival of the littest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.

III. The Proposed GA For Motion Planning Of AMR

GAs are search algorithms based on the mechanics of natural genetics. A genetic algorithm for an optimization problem consists of two major components. First, GA maintains a population of individual corresponds to a candidate solution and the population is a collection of such potential solutions. In GA , an individual is commonly represented by a binary string the mapping between solutions and binary strings is called a "coding". GA has been theoretically and empirically proven to provide robust search capabilities in complex spaces offering a valid approach to problems requiring efficient and effective searching [7].

Before the GA search starts, candidates of solution are represented as binary bit strings and are prepared. This is called a population. A candidate is called a chromosome (in our case: the path is a "chromosome" and positions are the "genes"). Also, an evolution function, called fitness function, needs to be defined for a problem to be solved in order to evaluate chromosome. As fitness function, we should define distance for each chromosome to give an evaluation function. This evaluation is the goal of the GA search and goes as follows: two (02) chromosomes are chosen randomly from populations are mated and they go through operations like the crossover (see the figure 4)to yield better chromosomes for next generations. This is repeated until about twelve populations with new chromosomes. To determine execution of the GA, we must specify a stopping criterion, in our case; it could be determined by a probabilistic function: as we have four chromosomes and we choose randomly two chromosomes, to combine and to compare one path with itself. The crossover is the comparison operator. Therefore, after several generations of GA search (The problem of mutation, see the figure5), relatively low fitness of chromosomes remains in a population and some of them are chosen as the solution of the problem (the most preferable path.

GAs are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the littlest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. An occasional new part is tried for good measure avoiding local minima. While randomized, GAs are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance. In the figure 6 we present genetic algorithm chart :

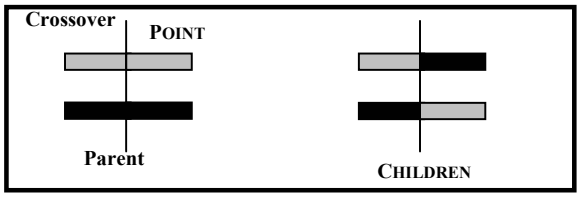

Fig. 4 Example of Crossover on singlepoint

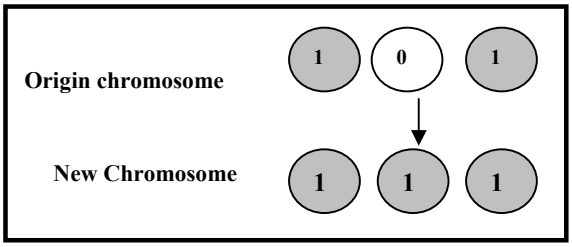

Fig. 5 Example of Mutation in the second bit

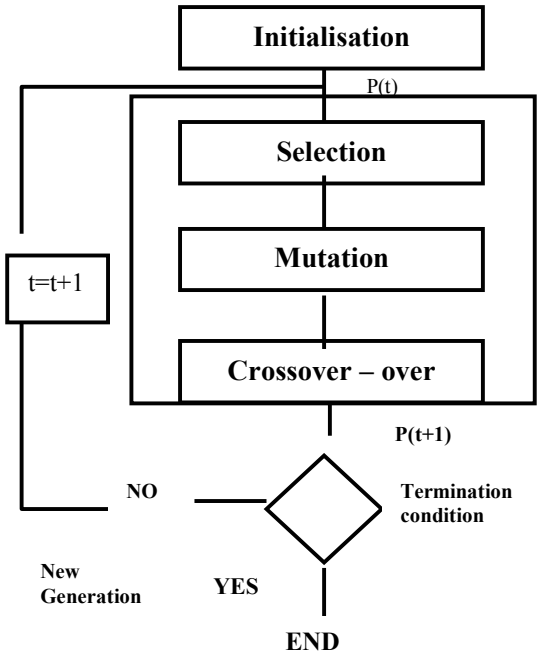

Fig. 6 genetic algorithm chart

A. Crossover and Mutation process

We have designed an operator that swaps two substrings of two genitors. However two children paths are generated which inherit some of the properties (substrings) of their two parents. Let two paths A_1 and A_2. Two points $S_1 \in A_1$ and $S_2 \in A_2$ are randomly selected, from which two new paths A_3 and A_4 from start to goat are generated by using the linking and concatenation operators (see the figure 5).

We note that: the path is chromosome and the positions are genes. In theory any path of the S_{AB} can be obtained by the crossover operator. However, due to the chosen probabilistic distribution in the calculus. The children have a great probability of lying in the region bounded by the parents that demonstrates the need of mutations to explore new regions.. Let us to explain more :

path A_3 :
1- path A_1 from A to S_1.
2-New path from S_1 to S_2.
3- path A_2 from S_2 to B.

Path A_4 :
1-path A_2 from A to S_2.
2-new path from S_2 to S_1.
3-Path A_1 from S_1 to B.

The selection is done according to a probability proportional to the performance (« ***fitness*** » in the classical literature). The selection is given by P :

$$p = \frac{2^n - 4}{2^m} \qquad (1)$$

Where *n* is the code number of paths designed to be candidates of selection of two paths***, m*** is the code number of all paths . At the beginning, a population of paths is created by the mutation operator between A1 = A and A2= B . The size n of the population is a critical parameter of all the genetic algorithms : if the number of individuals is small , the region of the terrain explored at the beginning of the search are limited and then the population iteratively generated for optimizing a performance index may tend to include paths neighboring a local minimum. On the other hand, a large population allows the generation of many individuals covering most of the terrain, and has a good chance to find all the optimal and near – optimal solution, but the population will also include less interesting solutions and furthermore the computing time may be high. The better chromosome which has less cost of path (the shortest path) yields after progenitor (several mutations and generation).The criteria of progenitor is stopped after (2^n-4) of generations, where n is the number of paths, we substract with number 4 because we cannot compare and combine a path with itself. The fitness function for each path is the number of pixels belonging to this path. (in the literature the fitness function is the performance that evaluates and gives a meaning of each chromosome). For improving iteratively the performances of the individuals in the population, the best individuals are preferred to serve as parents serve as parents in the next crossover operations.

A. Path planning

Assume that path planning is considered in a square terrain and a path between two locations is approximated with a sequence of adjacent cells in the grid corresponding to the terrain. The length A(α, β) from cell ″α″ to its adjacent cell ″β″ is defined by the Euclid distance from the center cell ″α″ of one cell to the center cell ″β″ of another cell. Each cell in this grid is assigned of three states: *free, occupied,* or *unknown* otherwise. A cell is *free* if it is known to contain no obstacles, *occupied* if it is known to contain one or more obstacles. All other cells are marked *unknown*. In the grid, any cell that can be seen by these three states and ensure the visibility constraint in space navigation. We denote that the configuration grid is a representation of the configuration space. In the configuration grid starting from any location to attend another one, cells are thus belonging to reachable or unreachable path. Note that the set of reachable cells is a subset of the set of free configuration cells, the set of unreachable cell is a subset of the set of occupied configuration cells. By selecting a goal that lies within reachable space, we ensure that it will not be in collision and it exists some "feasible path" such that the goal is reached in the environment. Having determined the reachability space, the algorithm works and operates on the reachability grid This one specifies at the end the target area. The detection of the three states is done by the different color

of pixels of those belonging to the area obstacle. Generally, the detected different colors of pixels have the same luminous intensity for every free path (a less difference). The other color neighbors are belonging to obstacle area. This detection is based on the game of every detected color of pixel. We separate between the set of luminous intensity of free path of pixels with those belonging to the set of luminous intensity of obstacle and unknown area. This separation is very useful to get a meaning of segmentation.

A grid of (i X j) dimension of free path is denoted by "***X***," an occupy grid of (l X k) is denoted by "***Y*** " . An obstacle is collection of hazardous cells in the "***Y***" ***grid*** **.**A path from start cell "C" to destination cell "D" that the detected color of "***X***" does not interest any detected color "***Y***". the path is said to be monotone of free cells "***X***" with respect to i-coordinates if no lines parallel to k-axis cross the j-axis (see figure 8).

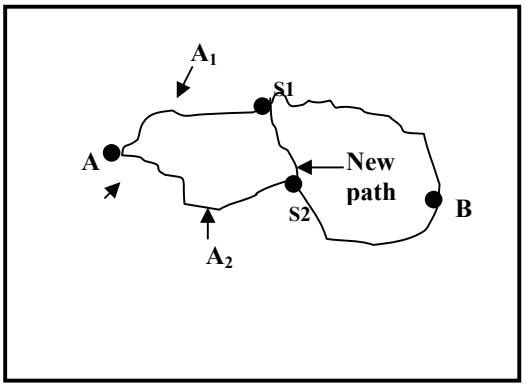

Fig. 7 crossover operation

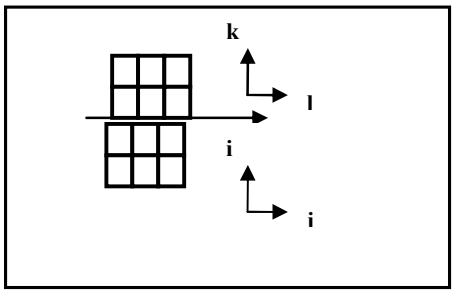

Fig. 8 an example of no intersecting in unknown environment

The proposed algorithm here relies on number of cells and iterates, as follows :

1) i by j grid, start cell a in the grid.
2) Detect free destination in the grid (free cells).
3) Detect the collection of cells in the grid corresponding to obstacle area (hazardous occupied cells area) and unknown cells.
4) A path from "C" to "D" such that the total of neighboring cells are detected free.
5) If the collection of free cells is continuous, detect all neighboring on the same destination until the target is reached.
6) If the collection of free cells is discontinuous, change the direction and continue on another free continuous collection of cells.

To maintain the idea ; we have created several environments which contain many obstacles. The search area (environment)is divided into square grids. Each item in the array represent one of the squares on the grid, and its status is recorded as walkable or unwalkable area (obstacle).The robot can identify three colors inside our environment: dark, yellow and green. The dark color is interpreted as an obstacle area; whereas the yellow color represents the free trajectory to attend the given target, and the green color refers to the area target (this is the first part of the main project to be after developed more, i.e., we start by this principle to give after more intelligence to our AMR). The robot starts from any position then it must move by sensing and avoiding the obstacles. The trajectory is designed in form of a grid-map, when it moves it must verify the adjacent case by avoiding the obstacle that can meet to reach the target. We use an algorithm containing the information about the target position, and the robot will move accordingly. To determine the nature of space of navigation, and as we have illustrated before, cells are marked as free or occupied; otherwise unknown. We can therefore divide our search area into free and occupied area . note that all free space cells represent the walkable space and unwalkable in occupied space. Each free cell is able of lying all the neighbor free cell within a certain distance "d". this distance "d" is usually set to a value greater than or equal to the size of cell. Note that the set of free cells is a subset of the of free cells, which is in turn a subset of the set of free occupancy cells. Thus, by selecting a goal that lies within free space , we ensure that the free sub-path will not be in collision with the environment, and that there exists some sub-paths to get the target. We use probabilistic concept to select the sub paths to get the target, given by :

α1: randomly probability of initial location space.
α2: randomly probability in free space area (walkable area).
α3: randomly probability of intermediate sub-positions in walkable area.
α4: randomly probability of final sub-positions in walkable area .
α5: randomly probability of destination location in free space area (walkable area).

Every $\boldsymbol{\alpha}_i$ belonging to occupied area (unwalkable area) is removed. Note that these probabilities $\boldsymbol{\alpha}_i$ are done in order to trace without collision the free trajectory and not to be in unwalkable area stopped with inside obstacles. Note, we determine the free resultant cells within free space to get a Feasible path during navigation. For unwalkable space (Occupied space) we just develop a procedure of avoiding danger. The figure 9 shows an example of walkable or unwalkable space .

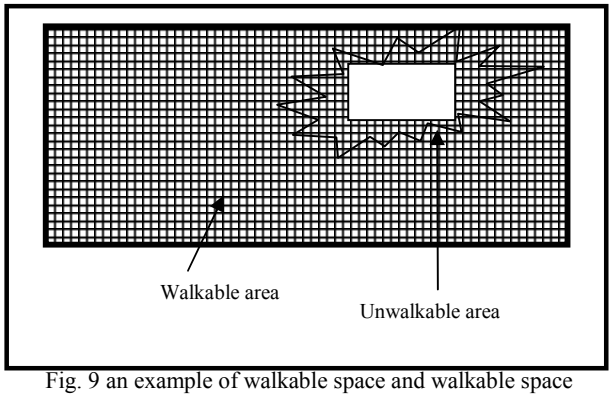

Fig. 9 an example of walkable space and walkable space

For unwalkable space, we compute the total size of free cells around danger (obstacle) area. This total may be at least greater or equal than to the length of architecture of robot. This is ensure the safety to our robot to not be in collision with the obstacle, and that the path P has enough security SE to attend it target where it is given by P±SE (S is size of security). In principle, we generate a plan for reaching safety area for every neighboring danger area. The safety distance is generated to construct the safety area building to the navigation process, to be near without collision within this one.

C. Path planning based on GA

First, we need to choose a coding which maps a path from start cell "C" to destination cell "D" into a binary string . We can represent an arbitrary path by using a binary string of "variable length" . To simplify the problem, we assume that a path from C to D is either i-monotone or j-monotone (but not necessarily both). Obviously, not all paths are monotone. The path can be represented by a column-wise (or row-wise) sequence of (i-1,j-1) pairs of direction and distance such that each pair corresponds to each column (or each row, respectively respectively) . Thus, the path can encode into a binary whose length is proportional to j and fixed. The first bit B1 indicates that path is x-monotone if B1 =1, and is y-monotone if B1=0. A block of (n+1) bits represents distance and direction on each column or row, where (n+1) are pairs of distance and direction. In case of B1 =1 : the first 2 bits of a block denote the direction ;e.g., 00(vertical), 01 (diagonal), 10 (horizontal). In the case of B1=0; we denote 00 (vertical), 10 (diagonal) and 01 (horizontal) (see figure 10). The aim of this technique is clearly obvious when we see the figure 11 (the sub- procedure for obstacle avoidance). The other bits of the block denote the distance is denoted by 1 if free, 0 if it is belonging to obstacle area. The population size is computed by $2^{(n+1)}$ bits. The likelihood of optimality which is the estimated probability of finding an optimal solution within g generations computed by : ($2^{(n)}/\ 2^{(g+1)}$). The mutation rate is a=0,1 crossover rate b=0,9 and win rate w=(1-a)---(1-b).this implies that GA can find an optimal solution with high probability if it is executed respectively and to a realized and b realized.

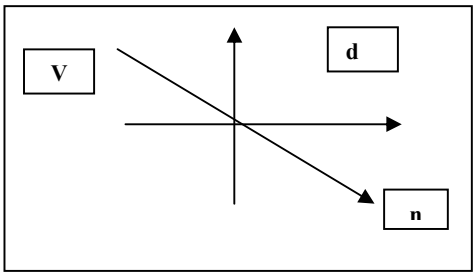

Fig. 10 an example of pairs of distance and

In order to evaluate, the average performance of our GA over various environments, we observed simulation of the GA for great number of environments. We can change the position of obstacles so we get other different environments. These environments generated .To find a new optimal path after insertion of deletion of an obstacle, we measure the number of generations of candidates. The coding of GA is to affect label 0 for free cell and 1 for hazardous cell. This way of work is very useful later if the substring is inherited to new generations by genetic operators. We generate (04) four paths and we detect the best one to be optimized between them. The fourth are shown in the Fig.11 and Fig.12. These fourth paths are called sub optimal paths. Every suboptimal is presented by its fitness function (number of detected pixels of free path) and is quite different from each other. Hence, a mobile robot detects unknown hazardous obstacle on the path. The generation of several paths gives at the end the best optimal path with low fitness function (e.g. the shortest path).

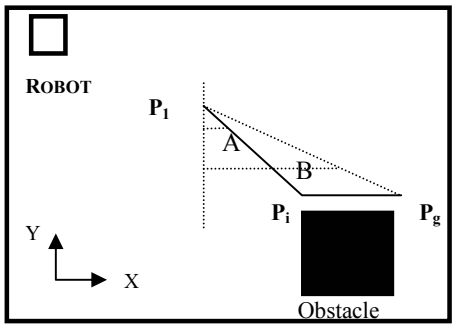

Fig. 11 Vehicle obstacle avoidance-mode

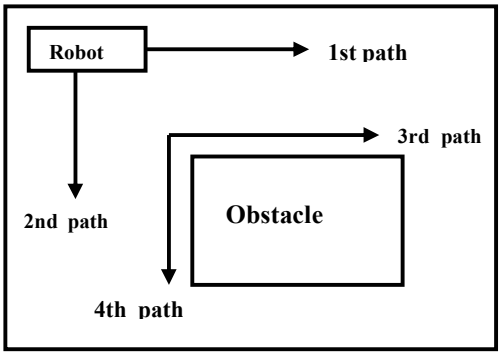

Fig. 12 an example of generation of the fourth paths

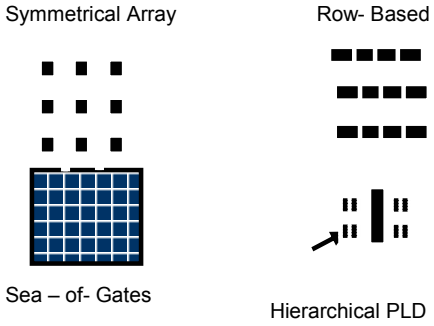

Fig. 14 the fourth class of FPGA

IV A VLSI PATH PLANNING OF AMR

A. FPGA concepts

Field Programmable Gate Arrays (FPGAS) are digital Integrated Circuits (ICs) that contain configurable (programmable) blocks of logic (CLB) along with configurable interconnects between these blocks. Depending on the way in which they are implemented , some FPGAs may only be programmed a single time, while others may be reprogrammed over and over again.(SEE figure 13).Not surprising, a device that can be programmed only one time is referred to as "one-time-programmable OTP". The" field programmable" portion of the FPGA's name refers to the fact that its programming takes place "in the field" (As opposed to devices whose internal functionality is hard-wired by the manufactures". If a device is capable of being programmed while remaining resident in a higher –level system, it is referred to as being In-system programmable (S P).the figure 14 presents the fourth class architectures of FPGAs in the industrial site of microproducts. In the figure 15 and figure 16, we present some CLB architectures (top down of CLB) of some products of FPGA Xilinx these architectures illustrate how it is the main body of some XILINX FPGA architecture.

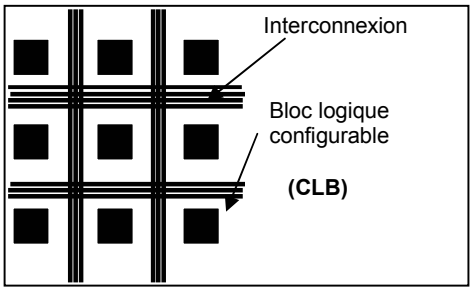

Fig. 1**3** FPGA Architecture

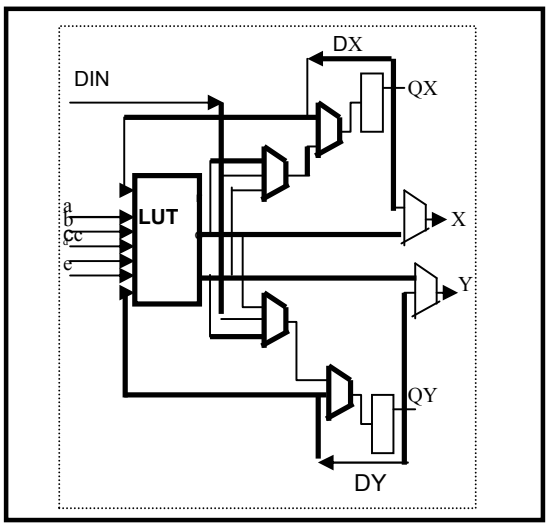

Fig. 15 CLB Xilinx – familly XC3000 Architecture.

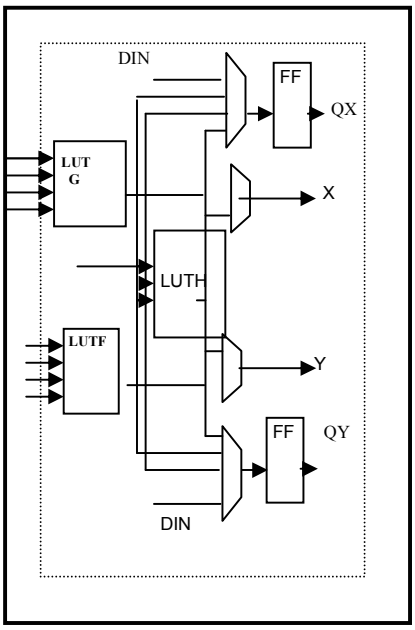

Fig 16. CLB Xilinx Familly XC4000Architecure

The complex issue of programming FPGA may be approached in a wide range of ways. One extreme is to consider that the designer hold only have to ketch his design in an abstract way, leaving to automatic tools as much of the implementation job as possible, with as little human intervention as possible. This hands-off approach reduces development time and costs, at their expense of the performance of the implementation. At the other extreme, when performance is critical, the designer has to intervene in the whole design process. This may include low-level implementation work and require important expert knowledge and much longer development time. Usually the implementation of a design on FPGA fall somewhere in the middle of these two extremes. The tools, while increasingly useful, still require a lot of technology-dependent knowledge from the designer performance than the software implementation (an easiness of implementation, shortest possible time, and rapid). We can have the mapping of an entire approach into a single Xilinx.

FPGAs. Implementation of some designs is very attractive because of the high flexibility that can be achieved through the reprogramability nature of these circuits. In addition, the synthesis tools allow designers to realize the mainly reasons : the need to get a correctly working systems at the first time, technology independent-design, design reusability, the ability to experiment with several alternatives of the design, and economic factors such as time to market. This new design methodology based upon VHDL description and using a synthesis tool Galileo. More, the parallelism, performance, flexibility and their relationship to silicon area are achieved by this new technique.

B. VHDL Concepts

VHDL allows for the description of hardware behavior from system to gate levels. The system level focuses on the description of the functionalities of the system (what is does) and tries to avoid it implementation description (how it is constituted). The notion of time is essentially a notion of causality: one action implies another. A constant is to forge useless details, which would imply architectural choices too early in the design methodology. Too detailed a system description is a drawback for it restrict further architectural choices or implies a given technology. Therefore, hiding the information structure is desirable and the notion of concurrency may not be necessary at this phase.

To fit this level of description, the language has to offer lager degrees of abstraction, a powerful algorithmic, wide capabilities for merging different description levels, an easy expression of causality, and also the possibility of introducing non determinism, which may be an interesting feature. To date, this level of description has not been synthesizable: no explicit architecture is described and no tool on the market offer a real and an efficient architectural synthesis (except for some specific target architecture). The synthesizable level is the potential input for synthesis tools. Here some implementation choices have already been made:

-Architecture, or at least an architecture family is targeted and is implied by the code structure.
-The widths of datapath are known.
-Time can be expressed in term of clock or sequentially (one statement is executed after the previous one).

The language must allow a description of the model at this level with a sufficient level of abstraction toward the physical level. Clock, sequentiality, dataflows, and combinational art have to be easily expressed. A large degree of parameterization is also required. The netlist level is the potential output of synthesis output of synthesis tool. It is structural view appearing as a collection of model instantiation. This kind of description involves the existence of model libraries (see the figure17). The notion of time is often present in the description of these models, from the notion of propagation delay trough a gate to very sophisticate delays (using slopes, temperature, etc.). These delays are either provided by the library and are therefore only indicative, or are relevant to a given technology and therefore more accurate. They can even be deduced from the enlist using a back notation mechanism (probably outside the VHDL world) .At this step, the language has to offer an optimal flexibility in terms of timing configuration or technology. These two aspects are most often linked.

The notion of time

The notation of time, which is carefully described in the LRM, is only related to simulation. This time is discrete: Only event have a date. Indeed, the notion of time does not exist between two of these date. The simulator is event driven: it kip from one event to the following one without exploring what happen in between. No synthesis semantic is defined in the VHDL LRM. Therefore, it is not possible to directly express implementation timing constraints using the language. The only exiting notion is the delay: “this output takes this value after this exact delay”. Moreover, no MIN / MAX simulation mechanism is deduced in the language.

Modularity

A VHDL description is never monolithic, modularity is everywhere. The first structuring level is the design unit. When compiling a VHDL source file, this file notion does not exit after compilation: each contained design unit once successfully analyzed is independently stored within a VHDL library.The only structuration of the original file which exists after compilation in the VHDL world is the notion of design unit. A VHDL source file is seen a collection of design units. A design unit link between design units written in the same source files, and there is no implicit link between design units written in the same source file (see the figure 18).

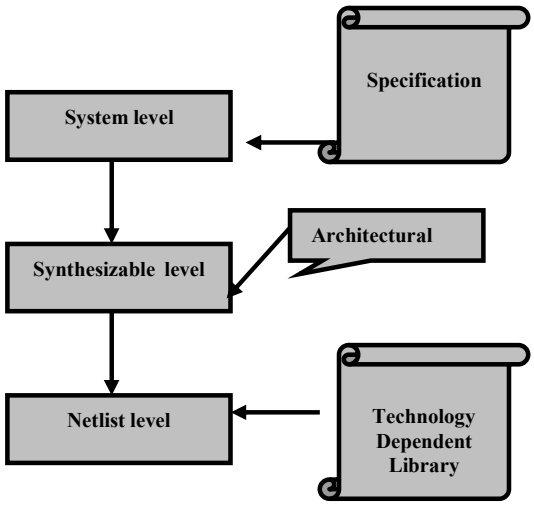

Fig. 17 Different level of Description

There are five kinds of design unit. They can be roughly split into two parts: those that describe the hardware hierarchy (i.e. , the structure of the model) and those that are more of software oriented. This operation is very important. An orthogonal classification shows that three deign unit are the external world. A secondary unit is the implementation (internal view) of its primary. A varying number of design unit, possible of different kinds, constitute a library each design unit is self –compilable but may use objects of other design unit, possibly stored in other libraries.

Portability

Portability was one of the main guidelines during the VHDL language design phase. The widespread uses of this language is mainly due to the fact that it is a standard. A standard is the only way for users to be free of the potential precariousness of a proprietary language. New prospect in the next new year; new synthesis techniques will have more and more impact on system design methodology. The synthesis process by itself is not the source of such a modification. Synthesis is only of the potential targets of a hardware description. Modeling ("modelware", that is the act to describe he behavior of a system as a whole) , is really be changing design methods. The remarkable points is that a single language. VHDL, is now able to cover the entire design cycle from functional specification to low—level structural design**.**

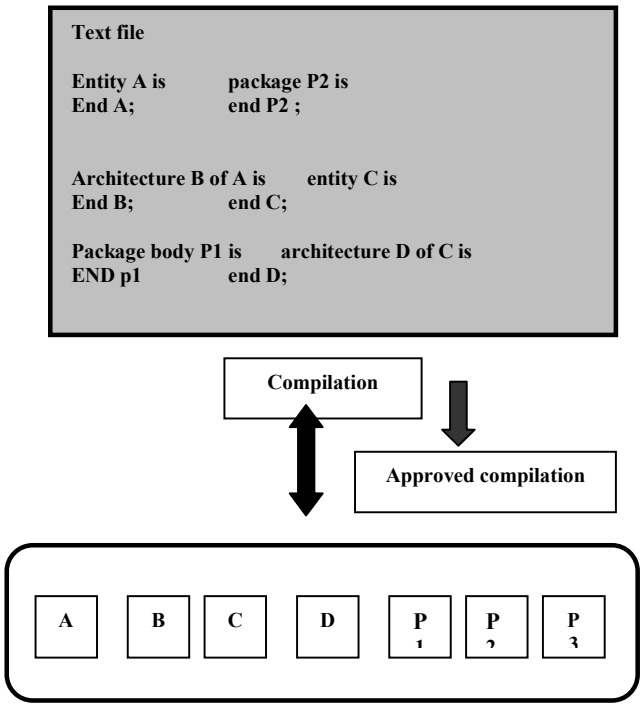

Figure. 18 Compiling source file into design units

Functional specification phase

The goal here is to translate the requirements into – a formal description. These requirements may have multiple aspect, document with text mixed with schemes references to international standard (protocol), set of equations, algorithm. Etc. even the completeness of the specification itself is not always ensured. The need for a functional description as reference model is therefore obvious. This model will be the essential starting point of the design cycle and will allow, by comparison of simulation result (and perhaps in the future using well-known that such a functional model allows the designer an in-depth understand of the problem to be solved. Fundamental question, such as the completeness of data transforms, arise during this phase.

Architecture choice phase

There is no universal tool for this phase. Experience and designer know-how is the key to success. All the potential targets must be taken into account. Their number is quite large: from software to ASICs (Application specific Integrated circuits), from programmable devices to standard circuits. Indeed, synthesis description is also one of these targets. By it very nature, is fuzzy and abstract. The goal of this book is to help the designer achieve a better understanding of it and therefore use it more efficiently study in the reference where there is different ways of expressing hardware in VHDL. Therefore, the designer will better understand the real capabilities of today synthesis tool.

As discussed in the previous section, the synthesis domain is vast. At the highest level, it is possible to express a et of interpedently treatments (dataflow graph) without any specific control. The only requirement is to respect constraint if any.

This is the architectural synthesis domain. On the other hand, logic synthesis requires the complete description of the control and use of resources in time. Thus, by itself is to appear to be a consistent continuation of the architectural synthesis process, once the resource allocation has been performed. Beyond all doubt, these two synthesis level (architectural and logic) will in the future merge into one.

Consistency between simulation and synthesis

One of the main objectives of all methodology is to make the transformation between two steps as safe as possible. Checks can be provided for this. Synthesis consists of the transformation of an initial description, as abstract as possible, into a structural description using well-identified hardware resources. Therefore, in terms of methodology, comp ring both description is essential. The power of description of VHDL is able, without any constraint, to support the two levels of description. Indeed, there is no real problem at this stage, but much more trouble I related to the transformation technology. In other word, the way of interpreting VHDL text source for synthesis purpose (the semantics for synthesis) is fundamental and critical; and it determine the quality of the result.

Text source architecture

An *entity* is the external view of a model: ports (inputs/outputs) and parameter (named generic), as well as static check on parameter values (such a range or minimal value verification) and dynamic check on port (such a set –up or hold time verification) are described in this design unit. The structure of a model, its behavior, or any mixture of both structure and behavior are described within it architecture. The syntax of such a design unit is

```
Architecture A of entity-b is
Architecture _Declarative_part
Begin
….
End;
```

More than one architecture can be associated with a given but only is selected for each model instantiation at simulation time. This selection is the main goal of the configuration mechanism that is discussed below**.** Architecture is implicitly dependent on it associated entity: all objects defined are known within the architecture. form, a shown in the example the ports are seen a signal and can be assigned within the architecture. The architecture body description consists of a set of concurrent statements.

Notion of component

This is a powerful VHDL notion. Unfortunately; the notion of component is not natural for beginner. In a first approach, we can seen that a model may be instantiated a certain number of time: a nand gate model can be instantiated three time Therefore, a single model, i.e. a single description, generates different instantiation, each of them having it own behavior. To be more general, VHDL offer a general and flexible mechanism: the instantiation is applied to the idea we have of a model, and not a in the case of direct instantiation to the model itself. It is therefore possible, either to directly instantiated a pair entity/ architecture or to initiate an intermediate object called component. Indeed, this notion of component represents the external view of a desired model. This desire may be quite different from the actual external view of the model we will finally use, and adaptation mechanisms are provided. There is no behavior attached to he notion of component, but it has on be great advantage: it allow compilation. So, many static checks may be performed even if the entity/ architecture pair that will finally be used is not known (or does not even exist yet).

This mechanism allows for a top-down methodology with real decoupling between the component library (what is desired) and the model library (what we have).Using the notion of component implies three fundamental operations: declaration, instantiation, and configuration of the component. Fortunately, and especially in the logic synthesis domain, any tool generate this source code automatically. Nevertheless, considering the component declaration as a simple redundancy of the entity declaration error. When using already exiting libraries or design units written by some body else, the power of the component notion appears obvious adaptation is possible.

Component configuration

Associating entity/ architecture pairs to component instances is the goal of this operation. It can be performed in the architecture where components are instantiated or within a specific design unit the configuration unit. In both cases, it allows an adaptation of the component to the model (the pair entity / architecture). This adaptation may consist in changing the name of the ports or parameters, also called generics, modifying their order, and even making some of them disappear. Continuing our analogy, this operation can be seen a plugging a circuit into a socket. Each socket corresponds to a component instantiation. Adapting the socket to the circuit is possible during this operation. The flexibility provided by the notion of component is very powerful. Selecting an entity / architecture pair is possible very late in the design cycle (just before simulation or synthesis) and switching from one library to another to change one model into another is a straightforward operation.

Mapping VHDL to hardware

The synthesis modeling style is mainly characterize by reducing the VHDL possibilities to a subset in which
-delay expression (***after*** clauses, ***wait for*** statement) are ignored
-Certain restrictions on the writing of process statement occur .
-Only a few types are allowed.
-Description is oriented towards synchronous styles by explicating clocks.

The result of synthesis process is conceptually a netlist of component. These component are predefined within a " proprietary" model library. There is no standard here, and the content of such libraries as well as their abstract level are rather different. some vendors even propose different libraries, such as the following :
1- Libraries accurately characterize and integrating a back-annotation mechanism, they allow an expensive but accurate simulation
2- Libraries more roughly described with timing value (elementary time unit), they nevertheless allow a quick check of the logical behavior of the model

C .The proposed GA hardware

The GA principle and work are tested in more unknown environments. The creation starts with empty environment (no intelligent navigation) and finishes until we get the high order of dispositions (e.g. the maximum number of obstacles). The operation examines the crossover and mutation operators and various parameters values for each operator. Simulation results in 16 or more unknown environments shown that after 2^4 generations the optimal path is gotten (4 is number of paths, 2 : is two paths). The parent (02 paths) are combined together that crossover operator is applied regarding to their fitness functions) (shortest path) is realized, then, the short one is stored with low fitness function (low number of pixels). We choose between two parents (two paths) and we combine between them (regarding their fitness function) until we get the best one suitable path to be taken into account to navigate intelligibly in unknown environment.

V. Digital Implementation Of GA

Configurable hardware is an approach for realizing optimal performance by tailoring its architecture to the characteristics of a given problem. FPGAs Implementation of *GA* is very attractive because of the high flexibility that can be achieved through the reprogramability nature of these circuits[9]. The complexity of VLSI circuits is being more and more complexes. Nowadays, the key of the art design is focused around high level synthesis which is a top down design methodology, that transform an abstract level such as the VHDL language (acronym for Very High Speed Integrated Circuits Hardware Description Language) into a physical implementation level[5,6,8,9]. In addition, the synthesis tools allow designers to realize the mainly reasons: the need to get a correctly working systems at the first time, technology independent design, design reusability, the ability to experiment with several alternatives of the design, and economic factors such as time to market. In this section, we present a new design methodology of GA based upon a VHDL description and using a synthesis tool Galileo. The result is a netlist ready for place and root using the XACT. The intended objective is to, realize architecture that taxes into account the parallelism, performance, flexibility and their relationship to silicon area. In this section, we discuss the possibility to permitting the mapping of an entire *GA* into a single Xilinx's FPGA.

A. Design methodology

The proposed design method for the *GA* implementation is illustrated in fig. 19 as a process to follow.This status is followed by the VHDL description of the navigation approach. Then the VHDL code is passed through the synthesis tool Galileo. The result is a netlist ready for place and root using the XACT tool. At this level, verification is required before final FPGA implementation. The simplified model of GA is presented in Fig.20. Thus, we can represent it in its hardware equivalent model, and the top view of architecture is represented in Fig.21.

B. VHDL Description of GA

Synthesis of this design is achieved by using the VHDL language with Register Transfer Logic (RTL) style description. The choice of VHDL comes from its emergence as an industry standard. The RTL style is used because it is well adapted for synthesis. The description of the *GA approach* begins by creating the components; the flexibility of the design is introduced by *generic* statement. Libraries that are functional and that target potentially accelaratable primitives.

Other option may be offered and are part of the usual environment available on VHDL synthesis platforms. The main synthesis issue with enumerated type is the encoding: there is no consensus for encoding enumerated type. Very often, the enumeration values are encoded by default into bit vectors whose length is the minimum number of bit required to code the number of enumerated value. This policy is applied to he predefined enumerated type BOOLEAN; where FALE and TRUE are respectively encoded a '0' and '1'. The solution is not systematically chosen, especially when the enumerated type is used for defining the state of a finite state machine. In that case, another way of encoding has to be chosen to optimize the result.The use of enumerated type in modeling is strongly recommended and is one way of remaining at an abstract and relying one the synthesis tool to choose the right strategy when optimizing and encoding. Some datatypes are not useful for synthesis purposes (for example, all physical type defined by the designers are not supposed). Predefine physical type TIME is not expressed. Even timing constraint for synthesis is not expressed in VHDL with time expressions. They are external constraints give, to the synthesis tool using attributes or proprietary mechanisms.

C. Tool dependency

Concerning tool dependency, one good point is that VHDL code can be the same for many synthesis tools. However the way to define a non synthesizable part of code tool dependent. Moreover the constraint descriptions differ from one tool to another one. Their syntaxes and also their semantic are defined for only one synthesis context. In fact, except for the architecture description written in VHDL, all information concerning the hardware context is described according to the interface provided by the synthesis tool. For example, if port defined by the entity at the top level of hierarchy have to become real inputs/output of the circuit, special hardware must be provided. This kind of information is described to the synthesis tool in it own language.

When the desired hardware is synchronous, the main constraint to be respected by the synthesis process is the period of the clock. After describing the electrical characteristics of the port (trough to drive, capacitance), the synthesis tool may accept the period to be respected as a constraint. It is not always the case, some tools do not support this data, and the designer has to check the critical path delays. If the constraints cannot be respected, the only solution, for a given library, is to choose architecture to find improved optimizations. Consequently a new VHDL code must be written and then synthesized. If all constraints are accepted, the hardware result of synthesis must be the smallest in terms of area.

D. Synthesis and implementation

The described methodology has been used for FPGA using the synthesis tool Galileo. At this level, and depending on the target technology, which is in our case, the FPGA Xilinx XC4000 family, the synthesis tool proceeds to estimate area in term of CLBs (Configurable Logic Blocs). The table I present the best synthesis results for *GA Approach* (given by the process). The output (XNF file) generated by the synthesis tool is passed trough the XACT place and tool. The resulting FPGA implementation of navigation approach of the whole architecture has been into a single FPGA mapped. Using the XACT, the netlist are placed and rooted and the area of configurable logic blocs is almost occupied by the architecture (the small juxtaposed squares)m the detail of this Implementation will be explained more in next article where the choice of Xilinx Family and satisfaction with silicon area. e try several synthesis to get the best chip of this application, the selection of the best is done automatically by he tool of development. More, The tool of programming selects the best circuit according the characteristics of the FPGA taking into account the architecture which will be implemented.

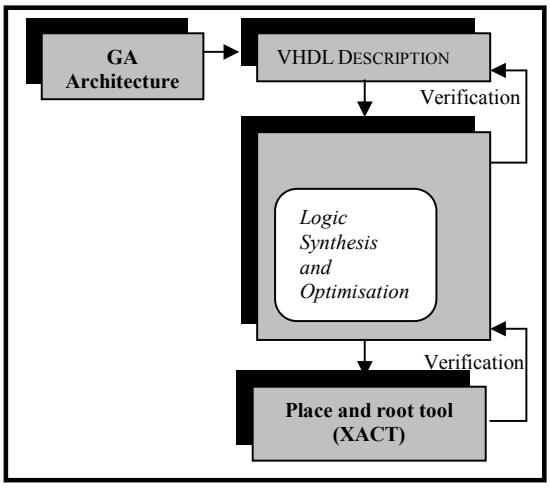

Fig.19 Design methodology of the *GA approach*

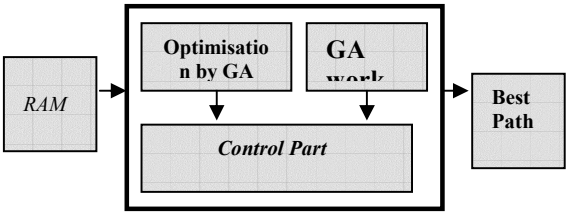

Fig. 20 Architecture of GA

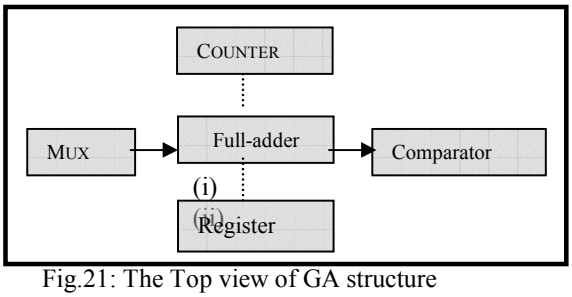

Fig.21: The Top view of GA structure

Design	***AREA (in term of CLBs)***
GA	***192***

Table I : Synthesis results

VI. CONCLUSION

The theory and practice of AMV are currently among the most intensively studied and promising areas in computer science and engineering which will certainly play a primary goal role in future. These theories and applications provide a source linking all fields in which intelligent control plays a dominant role. Cognition, perception, action, and learning are essential components of such-systems and their use is tending extensively towards challenging applications (service robots, micro-robots, bio-robots, guard robots, warehousing robots). In this paper, we have presented a hardware implementation of navigation approach of an autonomous mobile robot in an

unknown environment. The proposed approach can deal a wide number of environments. This system constitutes the knowledge bases of *GA approach* allowing to recognize situation of the target localization and obstacle avoidance, respectively. Also, a new design methodology of *GA* is introduced based upon a VHDL description and using a synthesis tool Galileo. The top down of this design based on logic synthesis has allowed a single chip FPGA implementation. The proposed VHDL description has the advantage of being generic and can be changed at the user demand. Depending on the final performance requirements, *GA* can be implemented using software tools supported by the standard microprocessor or by the hardware tools using FPGA technology, but the earlier offers more performance, flexibility and high speed processing into hardware, using the VLSI Technology FPGA (the concept of specification task and the reduce time of treatment). The primary results of the digital implementation show that the "intelligence-autonomy-economy" is achieved. However in the future, it is necessary to use a "micro-robot" in hostile environment and space exploration or other applications by using advanced micro-product control systems.

REFERENCES

[1] D. Estrin, D. Culler, K. Pister, *PERVASIVE Computing* IEEE, 2002,pp. 59-69.

[2] T. Willeke, C. Kunz, I.Nourbakhsh, The Personal Rover Project : The comprehensive Design Of a domestic personal robot, Robotics and Autonomous Systems (4), Elsevier Science, 2003, pp.245-258.

[3] L . Moreno, E.A Puente, and M.A. Salichs, : World modelling and sensor data fusion in a non static environment : application to mobile robots, in *Proceeding International IFAC Conference Intelligent Components and Instruments for control Applications*, Malaga, Spain, 1992, pp.433-436.

[4] S.Florczyk, *Robot Vision Video-based Indoor Exploration with Autonomous and Mobile Robots*, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2005.

[5] R. Airiau, J.M Berger, V. Olive, *Circuit synthesis with VHD,* Kluwer Academic Publishers, 1994.

[6] S.D.Brown, R.J., J.Francis Rose, and Z.G. Vranesic : *Field-Programmable Gate Array*, *Kluwer Academic Publishers*, 1997.

[7] A. Gonzalez and R. Perez. : SLAVE : A genetic learning System Based on an Iterative Approach, *IEEE, Transaction on Fuzzy systems*, Vol 7, N.2, April 1999, pp.176-191.

[8] J. Legenhausen, R. Wade, C.Wilner, and B. Wilson,: *VHDL for programmable logic*, Addison- Wesley, 1996.

[9]O. Hachour and N. Mastorakis, IAV : A VHDL methodology for FPGA implementation, *WSEAS transaction on circuits and systems,* Issue5, Volume3,ISSN 1109-2734, pp.1091-1096.

[10] O. Hachour AND N. Mastorakis, FPGA implementation of navigation approach, *WSEAS international multiconference 4th WSEAS robotics, distance learning and intelligent communication systems (ICRODIC 2004),* in Rio de Janeiro Brazil, October 1-15 , 2004, pp2777.

[11] O. Hachour AND N. Mastorakis, Avoiding obstacles using FPGA –a new solution and application ,*5th WSEAS international conference on automation & information (ICAI 2004) , WSEAS transaction on systems ,* issue9 ,volume 3 , Venice , italy15-17 , November 2004 , ISSN 1109-2777, pp2827-2834 .

[12] O. Hachour AND N. Mastorakis Behaviour of intelligent autonomous ROBOTIC IAR", *IASME transaction*, issue1, volume 1 ISSN 1790-031x WSEAS January 2004,pp 76-86.

[13] O. Hachour AND N. Mastorakis, Intelligent Control and planning of IAR, 3rd WSEAS International Multiconfrence on System Science and engineering, in Copacabana Rio De Janeiro, Brazil, October 12-15,2004.www.wseas.org.