Implementation of Robotic Navigation Algorithms Using Partial Reconfiguration on Zynq SoC

ECS Transactions, 107 (1) 13887-13901 (2022)
10.1149/10701.13887ecst ©The Electrochemical Society

Implementation of Robotic Navigation Algorithms Using Partial Reconfiguration on Zynq SoC

Mudasar Basha[a, b], Siva Kumar M[a] and Chinnaiah M C[b]

[a] Department of ECE, Koneru Lakshmaiah Educational Foundation (Deemed to be university), Green Fields, Vaddeswaram, Andhra Pradesh, India
[b] Department of Electronics & Communication Engineering, B V Raju Institute of Technology, Narsapur, Medak, Telangana, India

The purpose of the research work is to design a heterogeneous method for adaptive, task-based computer hardware reconfiguration for adaptive mobile robotic applications. The proposed method uses the dynamic reconfiguration technique to modify the navigation system to meet the requirements of various navigation algorithms for accomplishing an assigned task. In this work, we have proposed an efficient architecture that satisfies the requirements with adequate hardware and software resources by dynamically reconfiguring the system. The ability to change resources allows robots to collaborate more easily, to enhance performance & fault tolerance. The approach is endorsed through case studies in which a multiple mobile robots is loaded with bitfile and their behaviors are dynamically modified during run-time. This paper also proposes the hardware implementation of navigation algorithms on Zynq SoC, XC7Z020CLG484-1, using Xilinx Vivado 2017.3 to simulate and synthesize the design. Verilog is a real-time hardware description language, which is used to deploy multiple robots. Software Development Kit (SDK) is used to facilitate the generation of integrated software applications for Xilinx embedded processors. One of the important aspects of this work adopting an FPGA-based robot in service-based applications for assisting humans in their daily lives.

1. Introduction

The next step in robotics research is the development of even more adaptive mobile robot teams. With surveillance patrols to space exploration, mobile robots are already being used in wide range of applications. One of the significant challenges in mobile robotics is navigation and obstacle avoidance. Among the most significant functions of mobile robots is navigation. The established mobile robotic systems are constructed to perform well in a specific classic, such as a structured indoor or office with sufficient lighting for assist humans in accomplishing the task.
Under real-time restrictions, a mobile robot navigation system needs to respond for changing locations and emergencies. In order to obtain the minimal cost of the path a process of navigation or path planning is determined with free from obstacles. As per the classic path planning is classified as static & dynamic**.** Static path planning refers to when obstructions move their location relative with time; whereas dynamic path planning refers

to when obstructions alter, their location and direction with respect to time are differentiated as two types a) Global Navigation b) Local Navigation. Prior information of the surroundings must be known in global navigation such as, the Voronoi graph, Artificial potential field, Dijkstra algorithm, trees & graph, grids & cell decomposition method follow global navigation model. During local navigation, the robot can decide or regulate its mobility and orientation freely using ultrasonic sensors, IR sensors and camera. Various studies have successfully used fuzzy logic, neural networks, neuro-fuzzy, genetic, pso, aco, simulated annealing are used to tackle the local navigation problem (6). The prototype of hardware-efficient solutions for specified robotic activities has previously been addressed. As a result of our research, we present a universal and versatile platform using FPGA technology. For studying and designing a full robot, navigation system for various computational needs. Partial Reconfiguration (PR) extends this versatility by allowing and loading a modified partial configuration bitfile on fpga. Using PR, a portion of the configurable logic blocks can be diversified at runtime without impacting the integrity of logic operating on the other sections of the device. As a result, PR can enable diverse robotic applications to share a portion of an FPGA, resulting in improved energy and performance and making the FPGA an appropriate computing platform in dynamic and complicated robotic workloads. It is worth noting that robotics is not a single technology, but rather a collection of them. The robotic system's stack, as depicted in figure 1, is made up of three key components: Sensing, perception, localization, motion planning, and control are examples of application workloads; a software edge subsystem, which includes the operating system and runtime layer; and computing hardware, which includes microcontrollers and companion computers. This system may be utilized to satisfy the wide range of computational needs of autonomous robotic applications. In practice, energy efficiency was ensured by employing a low-complexity FPGA with a small number of logic cells. Furthermore, all sophisticated mobile navigation operations were embedded on the robot using dynamic reconfiguration (2).

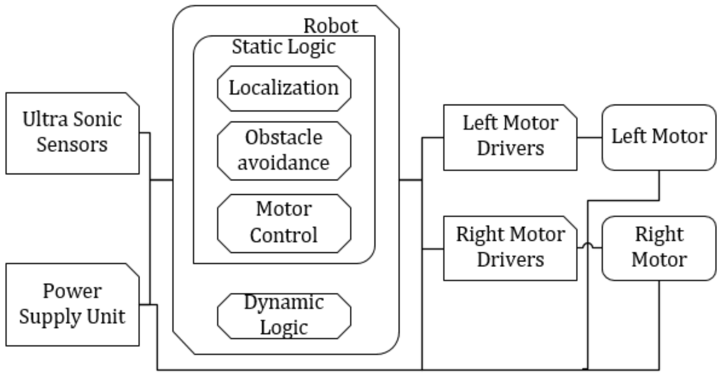


Figure 1. Robotic System.

A customizable real-time system - on - chip architecture with the parameters, such as capable of robust application-specific tractability, exploit, computation, is needed. So many researchers attempt to develop improved control and processing approaches for mobile robot navigation, during run time. Hence, excessively high costs of developing a new robot is much reduced for different applications. In order to enhance the said parameters, we proposed a dynamically reconfigurable real-time System on Chip that has high-end processing system, programmable logic for robots to inculcate to changing needs on the float to improve performance and capability loss.

The following are the contributions of our work i) constructing a classic grid environment ii) navigation safety & path length by avoiding the obstacles in the path.

The rest of this work is structured as follows: The literature survey is presented in Section 2. The Partial reconfiguration controller for our proposed computing algorithm utilized for mobile robot navigation is discussed in Section 3. Section 4 will explain the proposed methodology. Section 5 will discuss about the hardware implementation and its results, finally this work concludes with a conclusion in Section 6.

2. Literature Survey

Intelligent mobile robots have been designed to work increasingly complex tasks in a variety of applications these days. The navigation challenge, obstacle avoidance, wall following, or rather sophisticated parking have all been the subject of extensive research and numerous studies. The actuators are used to control the mobility of mobile robots with the information acquired from the sensors in a classic environment to achieve the goal. As a result, the electronic components chosen must be precise in order to control the desired position, location to produce real-time data for executing multiple tasks. Many researchers have been working on the problem of path planning for the recent decades, and numerous solutions have been devised.
On the Xilinx FPGA Spartan 3A board, a re-configurable cognitive controller for a mobile robot were developed by calculating distance by a set point with known settings; the robot performs obstacle detection and avoidance. The Levenberg-Marquardt algorithm uses data to train the intelligent neural network (NN) controller & the black box model in the MATLAB environment (1) represents the response of hardware-based controller. The author has presented a flexible platform based on the FPGA technology, for studying and conceiving complete unified robot navigation system (2). The author has uses the concept of integrating the advantages of a localization based on an odometer and localization based on camera data and make them complementary which will enable the robot to accurately localize itself to manage complex tasks that other embedded platforms cannot guarantee especially for complex vision applications (2). Odometer doesn't guarantee of accurate positions especially for the indoor environment applications and image processing may utilize more number of FPGA resource (2). An intelligent autonomous robot is required in various applications such as space, transportation, industry and defense for mobile robots to perform tasks like material handling, disaster relief, patrolling, and rescue operation, faiza Gul et al has made an effort has to study navigation techniques that are well suited for the static and dynamic environments (3). Hassani et al. used the free segment and turning point technique to design the path of a robot in a congested environment. The turning point method looks for a safe way for the robot to travel from point A to point B without clashing with obstacles furthermore, a model predictive controller was proposed to stabilise the robot so that it can follow the intended trajectory (4). The author has attempted to present a survey to bring together the wide area of work in the specific domain of dynamic & partial reconfiguration from the perspectives of architectures, tools, and applications, with a detailed discussion of efforts to date, and key research challenges standing in the way of widespread adoption, some of the comparative architectural features of commercially available FPGAs supporting PR has been addressed (5).
The author has presented about the various techniques by various researchers in the past two decades used for mobile robot navigation and obstacle avoidance as one of the

fundamental problems in mobile robotics are working to solve the robotic challenges by general classification of the Deterministic algorithm, Nondeterministic (Stochastic) algorithm, and Evolutionary algorithm, which are implemented for mobile robot navigation has been addressed (6). J.Baca et al. has presented a System is based on a stock of simple, low-cost fabrication modules. It provides the end-forward effector's kinematics (FK) in relation to the frame at the point of union with power control module body's wtih tasks involving movement one of the most common activities performed by robots is displacement. (7). The author has presented a cooperative multi-robot exploration algorithm that combines a frontier-based approach with the ability to put precise hard and soft limits upon that robot through with a central planning algorithm, the author attempted to show the change of exploration priority for both individual robots and the entire group for a defined environment (8). M.L.Silva et al. has presented a proof-of-concept & its implementation was used to create partial configurations at run-time for 20 circuits with up to 21 components and 288 connections with different topologies necessary to meet the conflicting goals of shorter running time and more flexible placement and routing (9). S. Commuri et al. has proposed an approach, which allows for the development of efficient controllers for each robot task, allowing for improved operational efficiency by using fixed computer complexity, several case studies in which a group of robots is deployed and their behavior is dynamically adjusted at runtime, validated with this method and efficiency of FPGA resource has been presented (10).

A simplified architectural description for the urban surveillance experiment under development within the URUS project is a functional layer through which the system has basic operational capabilities, and the authors present a human layer that concentrates the systems related to human-robot interaction (11). J.C.Palma et al. has tried to present an approach to solve problems related to the dynamic interconnection of hard IP cores reuse and configuration inside VLSI reconfigurable devices with data path widths greater than 1 bit (12).

To provide fault tolerance, Cobleigh et al presented tree - structured self-adaptation models for such mobile robots (13). In response to observed conditions, Gafni et al [1] presented an architectural style for real-time systems in which dynamic reconfiguration is implemented via a synchronous control task (14). The author has proposed comparison of various Biological Inspired algorithms the navigation over static and dynamic condition is analysed (for single and multiple robot systems) and it has been observed that the reactive approaches are more robust and perform well in all terrain when compared to classical approaches (15). N. Ivanova et al. has presented a strategy for constructing an occupancy map by probabilistic model of an ultrasonic sensor, during robot indoor navigation, a gaussian function is used for modelling the ultrasonic sensors with the aim of reaching higher precision of the distance measured for each obstacle and the author has restricted to simulation (16). M. B. Subramanian et al. has presented an algorithm based on cell branching method, called Breadth First Search (BFS) algorithm for a MRA on grid map based dynamic environment using RoboSim Simulator tool with the experimental results shows that BFS algorithm can realize the path planning under the dynamic environment very optimal and avoid the obstacles, and reach to its target point (17). The author expresses the map building system to process the environment readings by stochastic force to planned trajectory when sum of forces is zero along the end circumference of the robot by a series of simulation results (18). A Grid-based navigation approach is proposed by the author, which uses minimal environmental infrastructure and only two sensors on the mobile device, which works by starting from a predefined tile and determining its destination in the grid (19).

As per the authors knowledge the proposed solution addresses two robot objectives: navigation safety & path length and guarantee the effective use of FPGA with Partial Reconfiguration as an effective strategy for lower optimization of resources.

3. Partial Reconfiguration Navigation Controller

The terminology "partial reconfiguration" means changing one or more elements of the FPGA logic while leaving the rest alone. Field Programmable Gate Arrays (FPGA) offer speed and flexibility equal to or higher than a general processor. As shown in Figure 2, all FPGA devices are conceptually made up of two separate layers: the configuration memory layer as well as the hardware logic layer. This composition gives FPGAs their distinctive re-programmability and versatility.

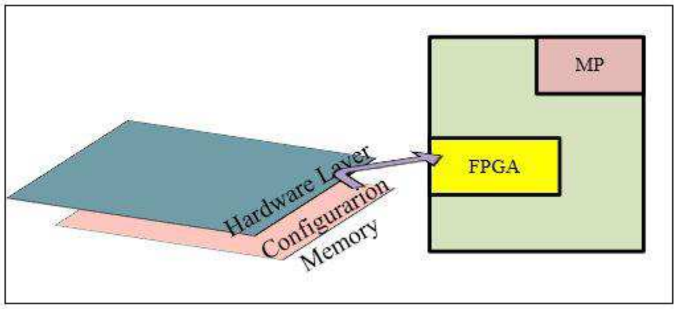


Figure 2. FPGA Configuration Layers.

Lookup tables (LUTs), flip-flops, digital signal processing (DSP) blocks, memory blocks, transceivers, and perhaps other computational hardware resources are all found at the hardware logic layer. The routing resources & switch boxes that enables components to be joined to build a circuit are also included in this tier. The FPGA configuration information can be stored in the configuration memory layer via a binary file known as a configuration file or bit stream. The values stored in the LUTs, the initial set & reset status of flip-flops, embedding process value system for memories, voltage standards of such output ports, & routing information for such programmable interconnect to enable the resources to form the described circuit are all contained in this binary file (5). The hardware logic layer's function is thus entirely dictated by the values recorded with in configuration memory (5).

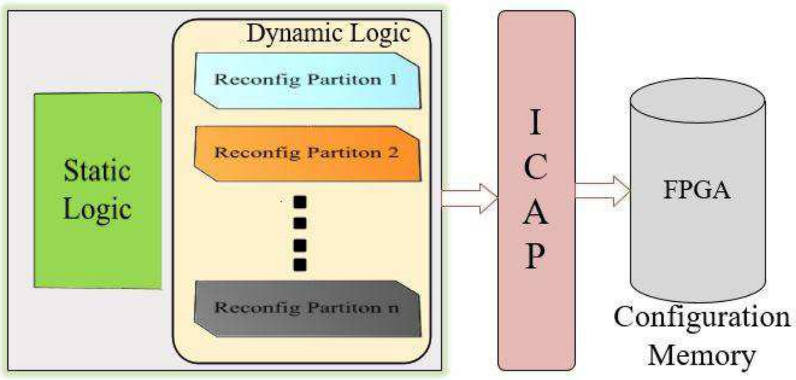


Figure 3. Dynamic Bit Stream Configuration on FPGA.

FPGA designs can benefit from PR in a number of ways. First, by time-multiplexing hardware resources among mutually incompatible computations, the chip's effective logic density can be improved, allowing a larger application to be housed on a smaller chip. PR also has the advantage of being cost-effective. Because this duration is directly proportional to the size of the reconfiguration, it takes less time than a full reconfiguration. The size of the configuration file is proportional to the size of the configuration file. Chip is being reprogrammed as a result; reconfiguration can be used in systems that are time-critical requirements. PR is best useful in adaptable hardware systems for Robotic applications, because it allows them to adapt computations to a changing environment while continuing to analyse data in a changing environment.

Internal Configuration Access Port (ICAP) on Xilinx devices, are used to support PR. The Xilinx Zynq SoC FPGA has the partial reconfiguration features, which can provide parallel and pipeline processing for Data, Image processing and matching. Hardware reconfigurable computing has grown to an important and large field of research. The fastest solution for reconfiguration is through context switching. The major innovative architecture design proposed in this paper is applying context switching on Multiple Navigation Paths with the multi-robots paradigm to embedded systems.

We examine the possibility of developing a multi-path architecture for robot navigation systems. We make the following assumptions: that each robot with a path can work independently, communicate with other robots, and can have a finite number of possible paths to go through at any one time. The architecture avoids having to wait for loading the selected path states from low-cost hardware to high-performance software. As a part of the reconfigurable resources can be used at any given time. The architecture for multi path navigation system is a hybrid based with hierarchical design. The feasible paths of robotic navigation system is clearly explained in the next section. Further, the following assumptions are strictly followed throughout this work.

i) Classic Grid Environment of 5 X 5 meters is assumed as a track for robot to navigate.
ii) Environment may consists of both Static and Dynamic Obstacles. (As a part of this work Obstacle avoidance methods are not shown here)
iii) Robots are known with number of tasks i.e., the number of tasks robot can accomplished is assumed as three. (In this work task, assignment procedure is not focused).
iv) Initially Robots are defined at an Origin ‘P’, and after completing the task, the robots will return to the Origin station.

4. Proposed Methodology

The main goal is to construct the classic environment utilizing fusion of sensor data. A novel FPGA based Partial Reconfiguration hardware architecture is being designed for static and dynamic environments where the robot need the most information on the vertex of obstacles, path planning, and destination information using Hamilton methods like Depth first search and Breadth first search algorithms and the dynamic function exchange among the robots has been proposed using Partial Reconfiguration method. The Static logic represented in FPGA is to follow depth first search algorithm as a default logic. Once the user has configured the respective bit file of the navigation model will be put on to fpga.

a) Depth First Search Algorithm

Depth First Search Algorithm (DFS) is a recursive algorithm that uses backtracking to traverse the graph. It requires searching all nodes thoroughly, advancing ahead if possible and retracing if necessary.
When searching in a graph, the searching is limited to of two operations in an approach:

(a) Go to a graph node and canvas it;

(b) Availability to the nodes adjacent to the recently visited node that is profitable.

To traverse a nodes and graphs, the depth first search method defines a Stack-based structure to store immediate results. The operation is carried out in parallel using the hardware-based navigation algorithm as follows.

i) Choose a projection matrix.
ii) In the classic, compute the Source & Target goal, task specifics, & load the set.
iii) Keep track of the node's current stack position.
iv) Decrement the stack and load the present value to validate with the task.
v) If the task for the given node has been completed, end the search and return a result.
v) If not, go back to step iii
vii) If the stack is empty, this indicates that all nodes have been traversed, and the search operation should be terminated with the result "nothing found."
viii) If the stack is not empty, go through the steps again iii.

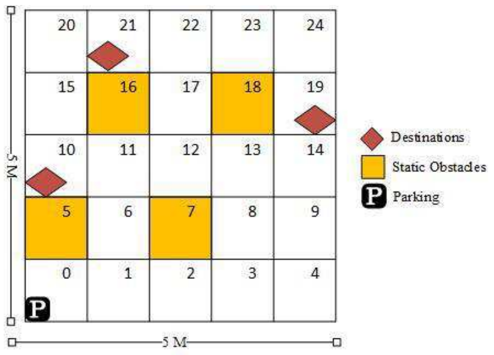


Figure 4. Classic Environment

All the robots are originated at node 0 defined as ‘P’ Parking. There are three destinations represented at node 10, node 19, node 21, Yellow colour is represented as static obstacles at node 5, node 7, node 16, node 18.Environment may consists of dynamic obstacles also. The dynamic Path planning approach is explained as follows:

Navigation 1: Initially, the robot travels from the origin to node 0, and if no obstacle exists at node 0, the robot follows no obstacle path to reach the target at node 10. At node 1, the robot looks for obstacles in the probable nodes that follow. When there is an obstacle at node 6, it would choose node 2, otherwise it will prefer node 6 and follow the algorithm's shortest path to the target. Similarly, it assesses the occurrence of barriers at node 3 and decides whether to go to node 8 or node 4. The feasible path directions, both without and with obstacles, are depicted here.

Static obstacles identification: at node 0 and node 7

Original path of the robot:

Origin –> 0^* –> 1^* –> 6 –> 11 –> 10.

No obstacle identified:

Origin –> 0 –> 5 –> 10.

which is the minimum and shortest path for reaching the task 1.

If obstacle identified:

Origin –> 0 –> 1 –> 2^* –> 7 –> 12 –> 11 –> 10.

Navigation 2: Second robot, configured by the task, will now perform the navigation. While travelling the path, the robot must detect barriers and then choose a path depending on the algorithm that used create the map.

The robot reached the following probable nodes at node 6, node 11, node 12, and node 17, only to find that it was stuck in a position that robot cannot move further, so robot will return back to find the next possible way to accomplish the task. Below are the path planning choices with and without obstacles.

Static obstacles identification at node 5 and node 6.

Original path of the robot:

Origin –> 0^* –> 1 –> 6 –> 11^* –> 10 –> 15 –> 20 –> 21

No obstacle:

Origin –> 0 –> 5 –> 10 –> 15 –> 20 –> 21

Navigation 3: Similarly, the other robot will start as per the task scheduled at node 19 as per the algorithm and the best possible ways are shown below.

Static obstacles identification: at node 5, node 7 and node 8

Original path of the robot:

Origin –> 0^* –> 1^* –> 6–> 11 –> 12 –> 13 –> 14 –> 19

If no obstacle:

Origin –> 0 –> 5 –> 10 –> 11 –> 12 –> 13 –> 14 –> 19

If obstacle:

Origin –> 0 –> 1^*–> 2^* –> 3^* –> 8 –> 13 –> 14 –> 19

Navigation 4: This navigation is the basic robot stationary module where the robot is to remain idle after completing the task. The clear path directions are shown below

Obstacle identification: at node 5

Original path of robot:

Origin –> 0

b) Breadth First Search Algorithm:

The breadth-first search (BFS) algorithm is used to search a tree data model for a node that meets a specific criterion. It begins at the root of the tree and investigates all nodes so at current depth level before moving on to vertices at the next predetermined depth. To maintain track of the leaf node that have been encountered but has not yet been visited, more memory, generally a queue, is required.
To traverse the nodes and graphs, the breadth first search method defines a queue -based structure to store immediate results and it is explained as follows with the help of Pseudo code.

```
Initial BFS(G, root) is
Let us consider a Q be a queue for storing the vertex and edge information
Keep track of label root as explored
Store Q.enqueue (root) which has been visited
While Q is not empty, do
Y: = Q.dequeue ()
Check for the goal v has reached then
Return the value v
Visit all the edges from v to w in G.adjacentEdges (v) do
If the node has not visited, then mark as not labelled and explore the root
Keep track of label w as explored
Store Q.enqueue (w)
Return the value x
```

The navigation strategy explained for different path approaches is same as depth first search traversal algorithm with a queue based structure and the path planning of the corresponding bit file will be loaded.

c) The concept of the dynamic reconfiguration

With time-sharing the reconfigurable logic with different algorithms specified above, parts of a path planning logic, dynamic reconfiguration could be employed to improve FPGA resource consumption. The static portion of our proposed depth first search algorithm. Whenever a different localization system is necessary, the core FPGA fabric can be dynamically modified at runtime to deploy on the fly by loading the partial bits of the algorithm on to the FPGA. This approach enables for the optimization of the configuration surface as well as the management of FPGA resources across many robot applications. The utilization of the algorithm is explained in the next section. The dynamic reconfiguration has been accomplished while the gadget is in use: certain portions of the device can be altered while others remain operational.

d) Design Flow

In this section we offered a completely autonomous system procedure for PR, that includes support & automation throughout the process of reconfiguration management as shown in the figure 4.With the help of Finite state machines, states reflect system configurations, the state transitions indicate the system's flexible behaviour. RTL source files should be accessible for all modules utilized in various system setups. The user also defines the time constraints that has to be met.

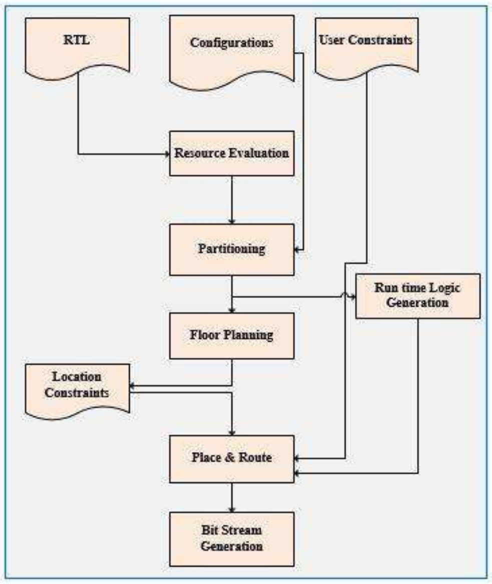


Figure 5. Flow chart of Design

By synthesizing separate modules, the tool first evaluates their resource utilization. The design is then partitioned. This requires estimating out how many reconfigurable regions (RRs) there are and assigning modules to them. Partitioning should be done in such a way that overall reconfiguration time is minimised (20). We accomplish this using a modified hierarchical clustering technique that groups modules with a higher chance of coexisting into the same RR. The partitioning tool's result is then provided to the floor planner, which determines where the RRs are physically located on the FPGA. As a result, the implementation tools can construct partial bitstreams using a set of area limitations. The floor planner's main goal is to shows a design flow that has been suggested. Reduce the amount of resources wasted.

Xilinx tools are then used to perform low-level place & route operations, then partial bitstreams corresponding to each RR's configuration are created. The dynamic system, which controls reconfiguration and performs the designer's dynamic behaviour, is likewise created as part of the pipeline. After partitioning, the design is immediately updated with a high-performance ICAP controller. In a modified finite state machine, the user-described behaviour of flow-produced partitions are mixed. When the required requirements are met, the ICAP-controller acquires the necessary partial bitstreams and configures the appropriate region (s).

5. Hardware Implementation & Results

The partial reconfiguration strategies is applied on the multiple mobile robots in the classic environment discussed in section 3. The hardware implementation is deployed using Vivado 2017.4 on Zynq 7000 Series FPGA and the embedded software application is developed in Xilinx Software Development Kit (SDK).

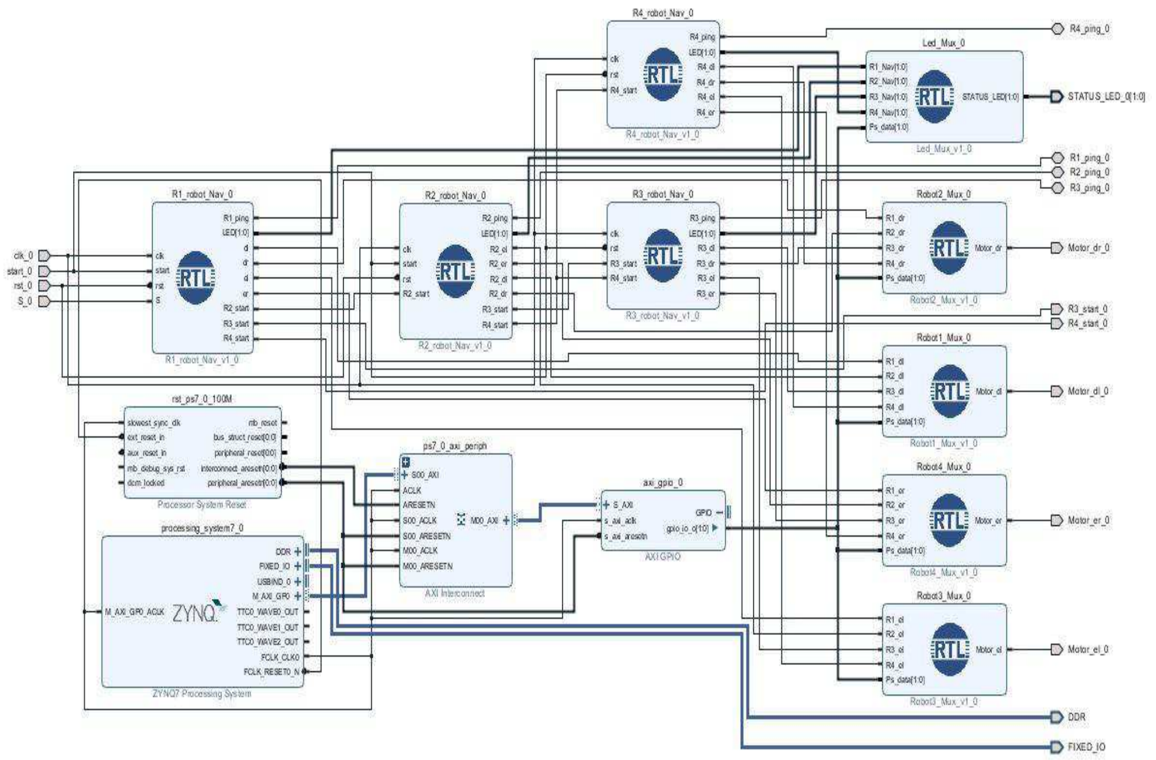


Figure 6. Block Diagram of Design

Figure 6 shows the block architecture of the system. It includes Zynq processing system with processor reset configuration and ICAP IP to manage the partial bitstreams on FPGA. RTL Design files used for selecting the partial portion of the design.

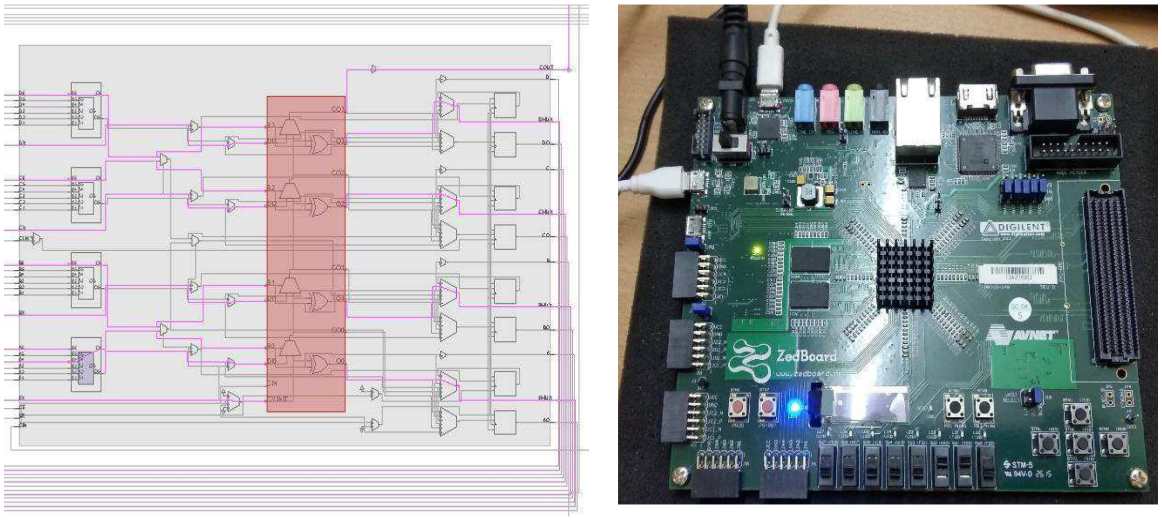
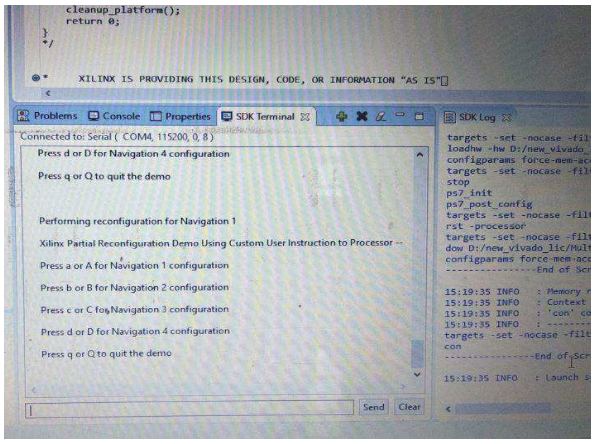


Figure 7. a) Reconfigurable architecture b) Static Logic

The partial portion of the reconfigurable logic is shown in figure 7a with its architecture. Once the static portion of bitstream is loaded on FPGA, the default logic is executed as shown in figure b.as per the algorithm stated in section 4. The user defined that there is no partial bit stream loaded to perform specific function.

Once the user wants to implement the desired functionality then user has to choose the operation to be performed by entering the specified instruction on SDK application as shown in Figure 8a. Once the bitstream is loaded, then corresponding LED has blown; for the convenience, we have used the onboard Leds for indication of robot navigation 1 using DFS. Similarly, the user can upload the bitstreams for the desired function & can achieve the corresponding navigation of the individual robots as shown in figure c, d, e, f.


```
        cleanup_platform();
        return 0;
    }
    */

 *     XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
```

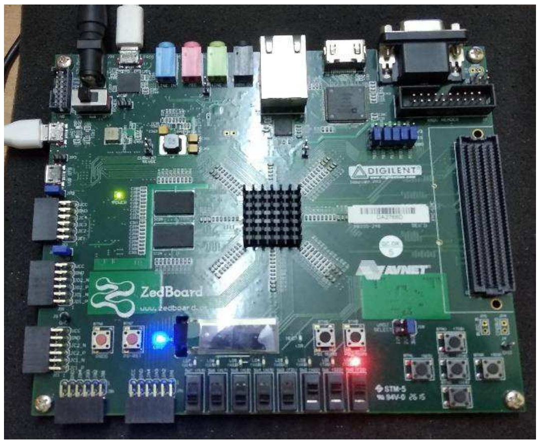
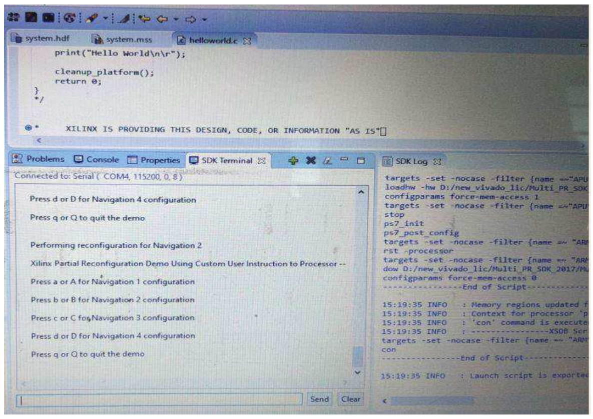
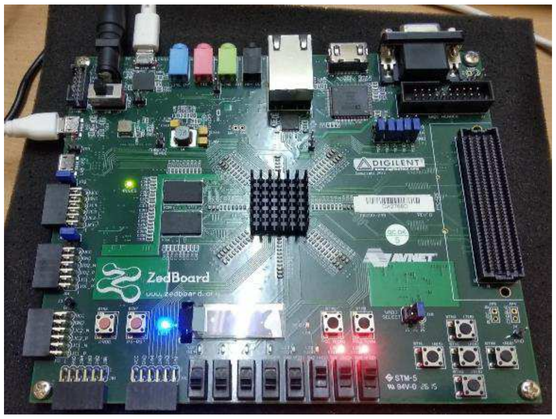
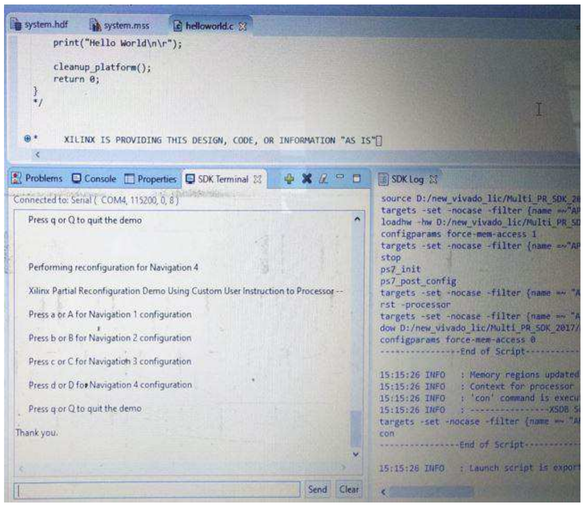
```
Connected to: Serial ( COM4, 115200, 0, 8 )
  Press d or D for Navigation 4 configuration

  Press q or Q to quit the demo


  Performing reconfiguration for Navigation 1

  Xilinx Partial Reconfiguration Demo Using Custom User Instruction to Processor --

  Press a or A for Navigation 1 configuration

  Press b or B for Navigation 2 configuration

  Press c or C for Navigation 3 configuration

  Press d or D for Navigation 4 configuration

  Press q or Q to quit the demo
```
```
targets -set -nocase -fil
loadhw -hw D:/new_vivado_
configparams force-mem-ac
targets -set -nocase -fil
stop
ps7_init
ps7_post_config
targets -set -nocase -fil
rst -processor
targets -set -nocase -fil
dow D:/new_vivado_lic/Mult
configparams force-mem-acc
----------------End of Scr

15:19:35 INFO    : Memory r
15:19:35 INFO    : Context
15:19:35 INFO    : 'con' co
15:19:35 INFO    : --------
targets -set -nocase -filt
con
----------------End of Scr

15:19:35 INFO    : Launch s
```


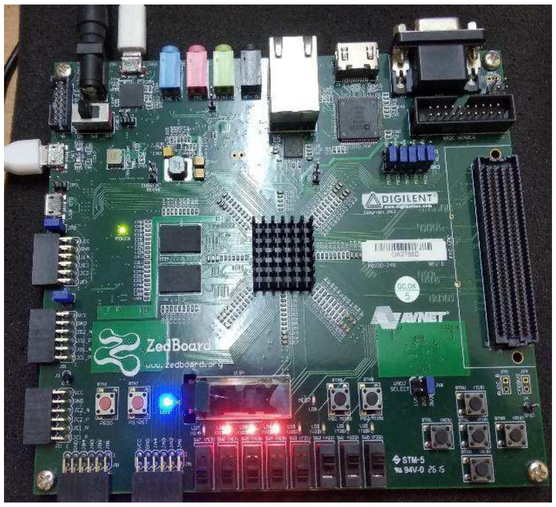
Figure 8. a) User Input through SDK application b) Navigation 1 is performed



```
        print("Hello World\n\r");

        cleanup_platform();
        return 0;
    }
    */

 *     XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
```
```
Connected to: Serial ( COM4, 115200, 0, 8 )
  Press d or D for Navigation 4 configuration

  Press q or Q to quit the demo

  Performing reconfiguration for Navigation 2

  Xilinx Partial Reconfiguration Demo Using Custom User Instruction to Processor --

  Press a or A for Navigation 1 configuration

  Press b or B for Navigation 2 configuration

  Press c or C for Navigation 3 configuration

  Press d or D for Navigation 4 configuration

  Press q or Q to quit the demo
```
```
targets -set -nocase -filter {name =~"APU
loadhw -hw D:/new_vivado_lic/Multi_PR_SDK
configparams force-mem-access 1
targets -set -nocase -filter {name =~"APU
stop
ps7_init
ps7_post_config
targets -set -nocase -filter {name =~ "AR
rst -processor
targets -set -nocase -filter {name =~ "AR
dow D:/new_vivado_lic/Multi_PR_SDK_2017/M
configparams force-mem-access 0
----------------End of Script-----------

15:19:35 INFO    : Memory regions updated f
15:19:35 INFO    : Context for processor 'p
15:19:35 INFO    : 'con' command is execute
15:19:35 INFO    : ----------------XSDB Scr
targets -set -nocase -filter {name =~ "ARM
con
----------------End of Script-----------

15:19:35 INFO    : Launch script is exporte
```


c) User Input through SDK application d) Navigation 2 is performed



```
        print("Hello World\n\r");

        cleanup_platform();
        return 0;
    }
    */

 *     XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"
```
```
Connected to: Serial ( COM4, 115200, 0, 8 )
  Press q or Q to quit the demo


  Performing reconfiguration for Navigation 4

  Xilinx Partial Reconfiguration Demo Using Custom User Instruction to Processor --

  Press a or A for Navigation 1 configuration

  Press b or B for Navigation 2 configuration

  Press c or C for Navigation 3 configuration

  Press d or D for Navigation 4 configuration

  Press q or Q to quit the demo

Thank you.
```
```
source D:/new_vivado_lic/Multi_PR_SDK_20
targets -set -nocase -filter {name =~"AP
loadhw -hw D:/new_vivado_lic/Multi_PR_SD
configparams force-mem-access 1
targets -set -nocase -filter {name =~"AP
stop
ps7_init
ps7_post_config
targets -set -nocase -filter {name =~ "A
rst -processor
targets -set -nocase -filter {name =~ "A
dow D:/new_vivado_lic/Multi_PR_SDK_2017/
configparams force-mem-access 0
----------------End of Script----------

15:15:26 INFO    : Memory regions updated
15:15:26 INFO    : Context for processor
15:15:26 INFO    : 'con' command is execu
15:15:26 INFO    : ----------------XSDB S
targets -set -nocase -filter {name =~ "A
con
----------------End of Script----------

15:15:26 INFO    : Launch script is expor
```


e) User Input through SDK application f) Navigation 4 is performed

Figure 9 describes the power report of the model. A dynamic power consumed by FPGA is 1.552 W with static power as 0.145 W. The power analysis shown here is robot for static portion & respective partial configuration of the logic. Similarly, for the group of robots we can define the power consumption is same until unless the logic of the respect to the configuration or design has been changed. Therefore, the authors has assured the power is optimal for the design for this application.

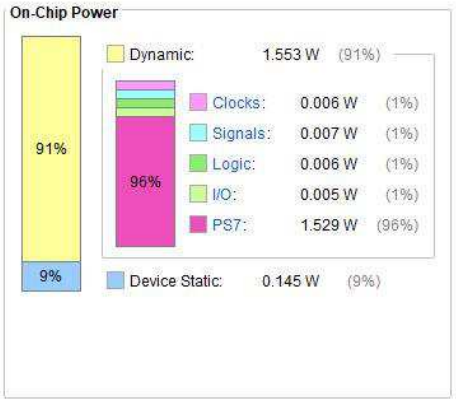
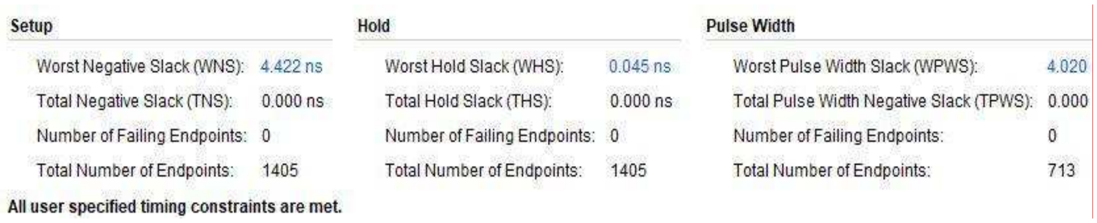


Figure 9. Power report of the Design



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4.422 ns	Worst Hold Slack (WHS):	0.045 ns	Worst Pulse Width Slack (WPWS):	4.020
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1405	Total Number of Endpoints:	1405	Total Number of Endpoints:	713

All user specified timing constraints are met.

Figure 10. User Specified Timing Constraints

The above figure shows the timing Summary report after adding user specified timing constraints (Timing constraints are included based on SDC Constraint approach followed by the industries).It shows the all user specified timing constraints were met, hence the author guarantee the best possible solution of the partial reconfiguration approach used for robotic navigation algorithms. The constraints may vary accordingly

Table.1 Synthesis result of the Design

Device Resource Zynq 7000 Series	Utilization	Available	Utilization %
LUT	2532	53200	4.76
LUTRAM	66	17400	0.38
FF	2216	106400	2.08
IO	16	200	8.00

The resource utilization of fpga is shown in the table 1. It shows the only 4.76 % of Look up tables were used for the implementation of the DFS and BFS algorithm for the robot navigation using Xilinx Configurable Logic Block architecture. The implementation shown above is for only one robot, hence we can deploy the same on other robots, which has the same configuration to perform specific task. Therefore, with the Partial reconfiguration method the resource utilization of fpga fabric can be minimized & can built more complex functions for high-end applications.

6. Conclusion

Thanks to advancements in FPGA technology, it is now possible to integrate a comprehensive navigation system on a single board installed on small self-governing robots. In robotic vision applications, the FPGA investigation is currently underutilized. In this paper, we use technology to analyse and design a complete integrated robot navigation system. The process can be simply tweaked to meet the robot localization application's varied computational requirements. We implemented two different proposed technique on the FPGA: Depth first search, Breadth first search, and we took advantage of Zynq SoC fpga for dynamic reconfiguration to dynamically adjust the system based in order to meet the energy-surface ratio with the freed resources for the implementation of other robot tasks. The authors have focused mainly on different navigation strategies to enhance or improve the optimal path cost for multi robotic applications, which is one of the challenges in robotics. In future, the author will focus on different strategies of robot model using partial reconfiguration method to define the low cost effective robotic system.

Acknowledgments

The Authors thank and acknowledge the Management, Principal of B V Raju Institute of Technology Narsapur, India, for their constant encouragement in all phases of this work.

References

1. A. H. Issa, A. T. Humod, and S. A. Gitaffa, “FPGA implementation of reconfigurable intelligent controller for mobile robot,” J. Mech. Eng. Res. Dev., vol. 44, no. 1, pp. 254–264, 2020.
2. A. Ghorbel, N. Ben Amor, and M. Jallouli, “Design of a flexible reconfigurable mobile robot localization system using FPGA technology,” SN Appl. Sci., vol. 2, no. 7, pp. 1–14, 2020, doi: 10.1007/s42452-020-2960-4.
3. Faiza Gul, Wan Rahiman & Syed Sahal Nazli Alhady | Kun Chen “A comprehensive study for robot navigation techniques,” Cogent Engineering, 6:1, DOI: 10.1080/23311916.2019.1632046
4. Hassani, Imen & Maalej, Imen & Rekik, Chokri “Robot Path Planning with Avoiding Obstacles in Known Environment Using Free Segments and Turning Points Algorithm,” Mathematical Problems in Engineering. 2018. 1-13. 10.1155/2018/2163278.
5. K. Vipin and S. A. Fahmy, “FPGA Dynamic and Partial Reconfiguration,” ACM Comput. Surv., vol. 51, no. 4, pp. 1–39, 2018, doi: 10.1145/3193827.
6. A. Pandey, “Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review,” Int. Robot. Autom. J., vol. 2, no. 3, pp. 96–105, 2017, doi: 10.15406/iratj.2017.02.00023.
7. J. Baca, M. Ferre, and R. Aracil, “A heterogeneous modular robotic design for fast response to a diversity of tasks,” Rob. Auton. Syst., vol. 60, no. 4, pp. 522–531, 2012, doi: 10.1016/j.robot.2011.11.013.

8. J. Butzke and M. Likhachev, “Planning for multi-robot exploration with multiple objective utility functions,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 3254–3259, 2011, doi: 10.1109/IROS.2011.6048764.
9. M. L. Silva and J. C. Ferreira, “Creation of partial FPGA configurations at run-time,” *Proc. - 13th Euromicro Conf. Digit. Syst. Des. Archit. Methods Tools, DSD 2010*, pp. 80–87, 2010, doi: 10.1109/DSD.2010.14.
10. S. Commuri, V. Tadigotla, and L. Sliger, “Task-based hardware reconfiguration in mobile robots using FPGAs,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 49, no. 2, pp. 111–134, 2007, doi: 10.1007/s10846-007-9131-3.
11. J. Sequeira, “Architectural aspects of networked robotic systems,” *Proc. 2008 IEEE Int. Work. Safety, Secur. Rescue Robot. SSRR 2008*, no. February, pp. 83–88, 2008, doi: 10.1109/SSRR.2008.4745882.
12. J. C. Palma, A. V. De Mello, L. Möller, F. Moraes, and N. Calazans, “Core communication interface for FPGAs,” in *Proceedings - 15th Symposium on Integrated Circuits and Systems Design, SBCCI 2002*, 2002, pp. 183–188, doi: 10.1109/SBCCI.2002.1137656.
13. J. Cobleigh et al, “Containment units: a hierarchically composable architecture for adaptive systems,” ACM SIGSOFT Software Engineering Notes, 27(6):159-165, November 2002.
14. V. Gafni, “Robots: a real-time systems architectural style,” In Proc 7th European software engineering conference, pages 57-74, Toulouse, 1999.
15. B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” Def. Technol., vol. 15, no. 4, pp. 582–606, 2019, doi: 10.1016/j.dt.2019.04.011.
16. N. Ivanova, V. Gugleva, M. Dobreva, I. Pehlivanov, S. Stefanov, and V. Andonova, “We are IntechOpen,” the world ’ s leading publisher of Open Access books Built by scientists , for scientists TOP 1 %,” Intech, vol. i, no. tourism, p. 13, 2016.
17. M. B. Subramanian, K. Sudhagar, and G. Rajarajeswari, “Intelligent Path Planning Of Mobile Robot Agent By Using Breadth First Search Algorithm,” vol. 3, no. 3, pp. 1951–1955, 2014.
18. J. Velagic, B. Lacevic, and N. Osmic, “Efficient path planning algorithm for mobile robot navigation with a local minima problem solving,” Proc. IEEE Int. Conf. Ind. Technol., pp. 2325–2330, 2006, doi: 10.1109/ICIT.2006.372707.
19. C. Buschmann, F. Müller, and S. Fischer, “Grid-Based Navigation for Autonomous, Mobile Robots,” Proc. Work. Positioning, Navig. Commun. (WPNC 2004), no. April 2004, pp. 157–162, 2004.
20. K. Vipin and S. A. Fahmy, “Efficient region allocation for adaptive partial reconfiguration,” in *Proceedings of Int. Conf.on Field Programmable Technology (FPT)*, 2011.