FPGA based Combinatorial Architecture for Parallelizing RRT

Gurshaant Singh Malik, Krishna Gupta, K Madhava Krishna and Shubhajit Roy Chowdhury

***Abstract*—Complex tasks are often handled through software implementation in combination with high performance processors. Taking advantage of hardware parallelism, FPGA is breaking the paradigm by accomplishing more per clock cycle with closely matched application requirements. With the aim to minimise computation delay with increase in map's size and geometric constraints, we present the FPGA based combinatorial architecture that allows multiple RRTs to work together to achieve accelerated, uniform exploration of the map. We also analyse our architecture against hardware implementation of other scalable RRT methods for motion planning. We observe notable furtherance of acceleration capabilities with the proposed architecture delivering a minimum 3X gain over the other implementations while maintaining uniformity in exploration.**

I. INTRODUCTION

In the field of robotics, FPGA is capable of delivering tightly packed, energy efficient infrastructures adept in fast real time performance. FPGA implementation of Image processing descriptors [1], [2], Random Decision Tree Body Part Recognition Using FPGA [3] and FPGA based collision avoidance [4] further validate our belief. An FPGA allows gate level control of system architecture. This allows the designer to tap the potential of hardware design, allowing control over minute details of arithmetic design, real time parallelization, pipelining of sequential processes. The hardware level flexibility afforded by an FPGA results in designs that are not only fast, but also small and power efficient. FPGA boards generally consume small physical area, making it an ideal solution for robotic systems with constrained dimensions.

Sampling based methods for motion and path planning have gained more interest during the last decade, as computer power has increased. In the field of robotics, the Rapidly exploring Random Tree (RRT) has become the customary algorithm for solving mathematically complex single-query motion planning problems [5] involving Kinematic, Holonomic, Non-Holonomic or Kino-dynamic closure constraints [5], [6], [7]. Apart from the domain of robotics, it has proven itself as an effective substructure in the field of structural biology and medicine, manufacturing, virtual prototyping to name a few. For improving the performance of RRT, techniques like biased exploration [8], [9], controlled sampling [10], and faster nearest neighbour search [11] have been employed. Considerable research effort has also gone into achieving speedup of sampling based motion planning by parallelizing it [12], [13].

Thus the main contribution of the paper lies in exploiting the hardware flexibility offered by an FPGA to achieve a faster and more homogeneous exploration of the workspace by parallel growing RRTs. The proposed architecture is compared with distributed [12] and K-distributed architectures [13]. This architecture delivers an acceleration gain that is 3X to the best of the state of the art architectures with a far more uniform per-unit area coverage of the map. This is made possible through a novel multi-port hardware architecture that allows for zero latency read/write access to the N RRTs that grow in parallel. With the proposed architecture, apart from significant acceleration in exploration of the map, multiple RRTs work together in parallel to achieve a more homogeneously explored map as compared to a single RRT. As shown in Fig. 1, by allowing each of the 16 RRTs to explore the entire map in parallel but limiting the sampling space to evenly distributed, exclusive sub-spaces, an accelerated, uniform exploration of the map should be expected. It should be observed that an RRT can latch on to the explored portion of the entire map. Only the sampling is limited to its sub-space.

978-1-4673-9163-4/15$31.00 © 2015 IEEE

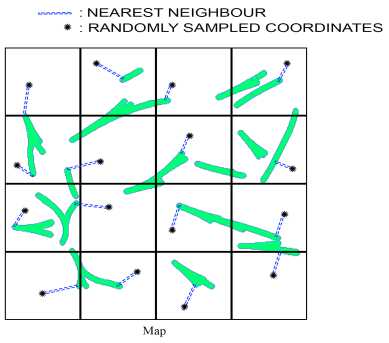


Fig. 1. Distribution of sampling space for N=16

Employing the proposed architecture (realizable only in a hardware implementation), for the same amount of time, the explored map becomes denser and homogeneous with the increase in number of RRTs working in parallel, as shown in Fig. 2. This observation will be delineated in more detail in the results section.

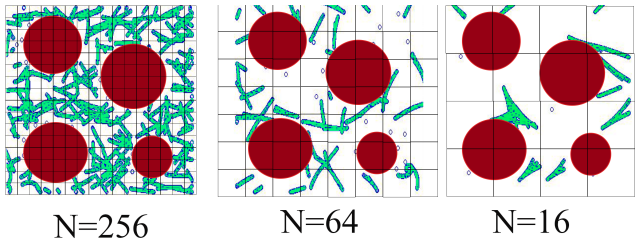


Fig. 2. Demo of combinatorial architecture for differentially steered system

II. ARCHITECTURE FOR PARALLELIZING RRT

A. Current Challenges

Since RRT involves randomized exploration of the map, ours and many proposed algorithms [12], [13], use the principle of exploratory decomposition [14] as their foundation. In other words, each RRT produces its own output and through different write mechanisms, the outputs are integrated to build

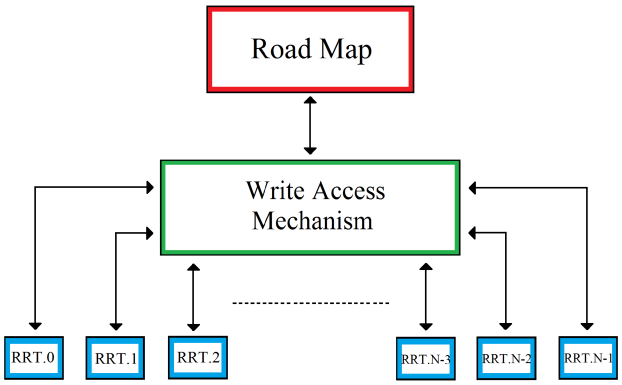


Fig. 3. Schematic illustration of exploratory decomposition

the explored map. Fig. 3 provides an overview of this design philosophy.

In such a parallel RRT design, an important issue is to decide the write access mechanism that integrates the data from multiple RRTs and then updates the global explored map. There are 2 general philosophies: 1) Distributed and 2) Shared. The distributed philosophy employs a scheme by which each RRT will have its own local explored map. As a result, changes made by it to its local explored map will have no effect on other RRT's local explored maps. Hence, as shown in Fig. 4, we need a mediator system that updates each RRT's local explored map to changes made by other RRTs. This will incur significant inter-RRT communication time in case of large scale parallelization.

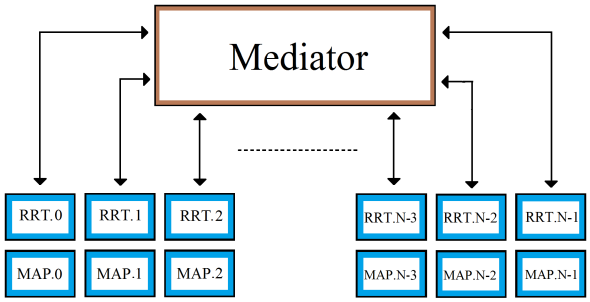


Fig. 4. Schematic illustration of distributed philosophy

The shared design philosophy, shown in Fig. 5, allows all the RRTs to have access to the same global explored map. Hence, there is virtually no inter-RRT communication. But, since all RRTs will have access to the same global explored map, large scale parallelization, without scheduling, can geometrically increase traffic on global address space, leading to data collisions.

B. Proposed Write Access Mechanism

Owing to the probability reliant exploration of RRT, accurate prognosis of data arrival time is an ambiguous task. As shown in Fig. 6, N RRTs working in parallel can result in 2^N possible cases during a write window to the global explored map.

Hence, as shown in Fig. 7, the first part of the architecture is a multi-port random access memory with the ability

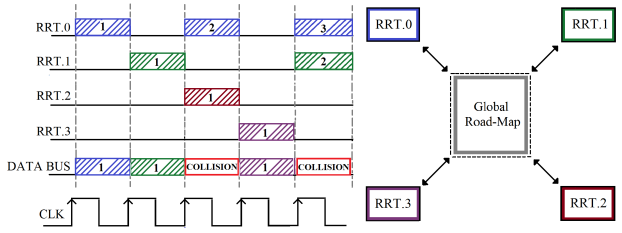


Fig. 5. Schematic illustration of the Shared design philosophy

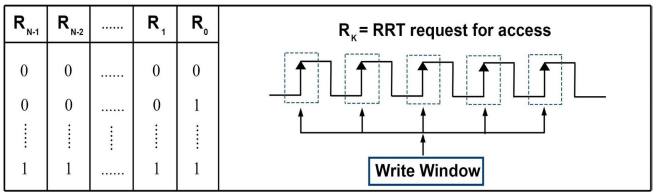


Fig. 6. Illustration of possible cases during a write window for N RRTs

to handle [0, N] variable, asynchronous write and/or read transactions during a write and/or read window. Each line of data in this memory will store the explored M degrees of freedom of the mobile robot in the form [DOF_1, DOF_2, ..., DOF_M].

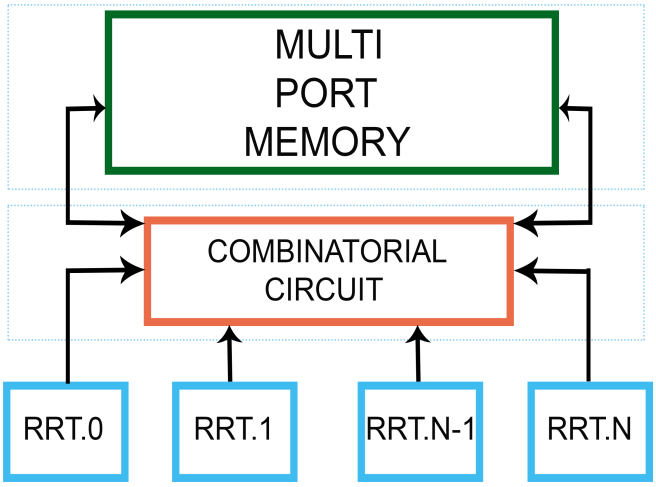


Fig. 7. Schematic illustration of combinatorial architecture

As shown in Fig. 7, the second part of the architecture is a combinatorial circuit that ascertains the current case of the 2^N cases during the write window and feeds the appropriate write control signals to the memory.

This allows each of the N RRTs to have access to the write window with zero latency/scheduling since this write access mechanism combinatorially accounts for all the possible 2^N cases. This results in accelerated exploration of the map and preserves integrity of the freshly explored data. It should be observed that only a hardware(FPGA/ASIC) implementation of this mechanism is realizable since the parallel writes need to be triggered in real time, a luxury not afforded to conventional software implementations.

III. FPGA IMPLEMENTATION OF THE COMPLETE SYSTEM

Optimised for high-performance logic and DSP with low power serial connectivity, the design test platform, Virtex™-6XC6VLX75T FPGA delivers 6.6 Gbps GTX transceivers and built-in PCIe and tri-mode Ethernet MAC blocks that help meet higher bandwidth and performance demands with less power. The table below provides system parameters.

System Parameters			
Slices	11,640	**Total BRAM(Mb)**	5.48
Logic Cells	74,496	**DSP48E1**	288
CLB Flip Flops	93,120	**Area**(mm^2)	23*23

Register transfer level(RTL) design removes the mathematical abstraction of integers and forces the designer to work on binary sequences of bits. For our design, the cartesian coordinates are represented as 32bit long, fixed-point, 2s compliment binary strings where the 24 MSB represent the integer part and the 8 LSB represent the fractional part. This representation provides an incremental resolution of 0.00390625 in decimal format. The geometrical angle is represented as a 16bit long, fixed point, 2s compliment binary string where the 3 MSB represent the integer part and the 13 LSB represent the fractional part, affording an incremental resolution of 0.00012207 radians.

The 3 sub-systems : 1.) RRT, 2.) The combinatorial circuit and 3.) The multi-port memory work together to constitute the complete system. On account of the scale of the FPGA implementation, in order to clearly delineate, the behavioral modelling of each is explained in separate subsections.

A. Implementation of Single RRT

As Algorithm 1 explicates, a single RRT consists of 4 steps : 1.) Random sampling, 2.) Nearest neighbour search, 3.) Collision detection and 4.) Kinematic path extension.

Algorithm 1: Rapidly Exploring Random Tree

input : map M, root q_{init}, $rrtStop$
output: new explored map M
while $\sim$*rrtStop* **do**
  $q_{rand} \leftarrow$ sampleRandomStateC ;
  $q_{near} \leftarrow$ nearestNode(C,q_{rand}) ;
  if *collisionFree(C)* **then**
    $q_{new} \leftarrow$ kinematicExtend(q_{near}, q_{rand}) ;
  wait.Access $\rightarrow$ update.C $\rightarrow$ release.Access ;

1) Random Sampling: Pseudo-Random and Pseudo-Independent numbers from a uniform distribution are generated using a modified multiplicative congruential algorithm. These numbers are then synthesised and mapped to a 640 KB Read Only Memory(ROM), as shown in Fig. 8(I), subsequently used as the source for random numbers.

2) Nearest Neighbour Search: As shown in Fig. 8(II), 32 bit input combinatorial subtractors feed the coordinate difference to dedicated, zero latency multipliers to generate distance numerals, which are then subsequently fed to the combinatorial comparator to identify the nearest neighbour.

3) Collision detection: The original map, including the objects, is cached to a Read Only Memory(ROM), the size of which is equal to the dimensions of the map. The data is stored in binary, where '0' represents a free space unit and '1' represents an obstacle unit. Subsequent AND operation between the ROM and the robot's configuration ascertains the collision state, as shown in Fig. 8(III).

4) Kinematic Path Extension: The implementation of this sub-module is directly correlated to the kinematics and dynamics of the robot. In general, CORDIC cores are deployed for computation of trigonometric functions. DSP48E1 slices are used for high speed multiplication and addition requirements, as shown in Fig. 8(IV)

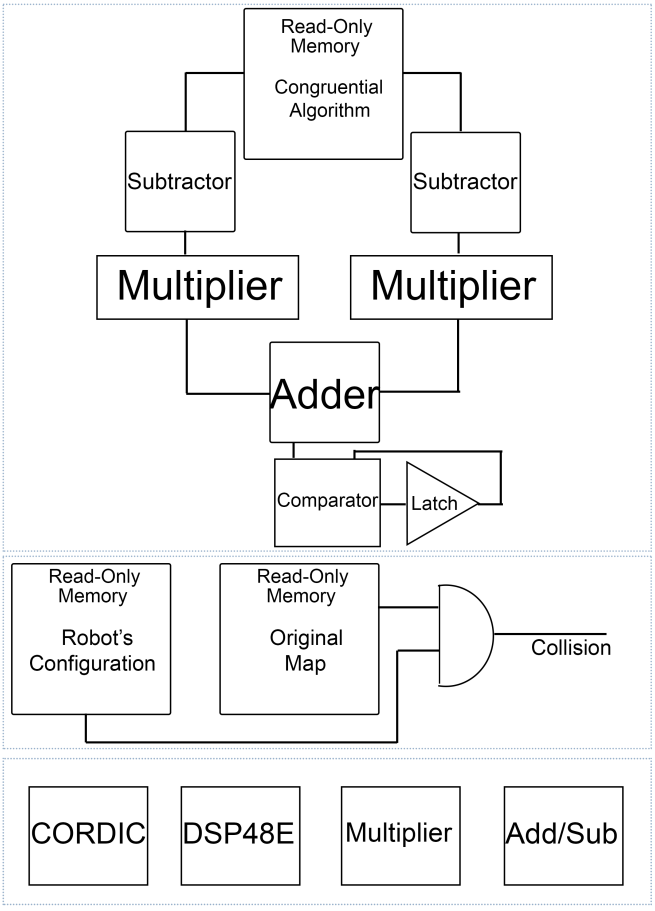


Fig. 8. Top to bottom : Sampler, Nearest Neighbour, Collision Check, Kinematic Extension

B. Implementation of Combinatorial Circuit

For N RRTs, the 2^N possible cases and the corresponding control signals of the multi-port memory are mapped to cascaded look up tables(LUTs). An N bit string, where each bit corresponds to a RRT, is used as input. A '1' bit means that the corresponding RRT is requesting access and a '0' bit means otherwise. The outputs of this module are the control signals of the multi-port memory, as shown in Fig. 9.

C. Implementation of Multi-port Memory

With a global address space, the multi-port memory is implemented as a heap of N distributed, single channel mem-

ories, each of size $(400 * M)/N$ KB, where M is the number of degrees of freedom of the robot and N is the number of RRTs. The read and write channels are designed asynchronous to enable independent read and write transactions. Auxiliary multiplexers on the read and write channels apportion the global address space to local address spaces, as shown in Fig. 9.

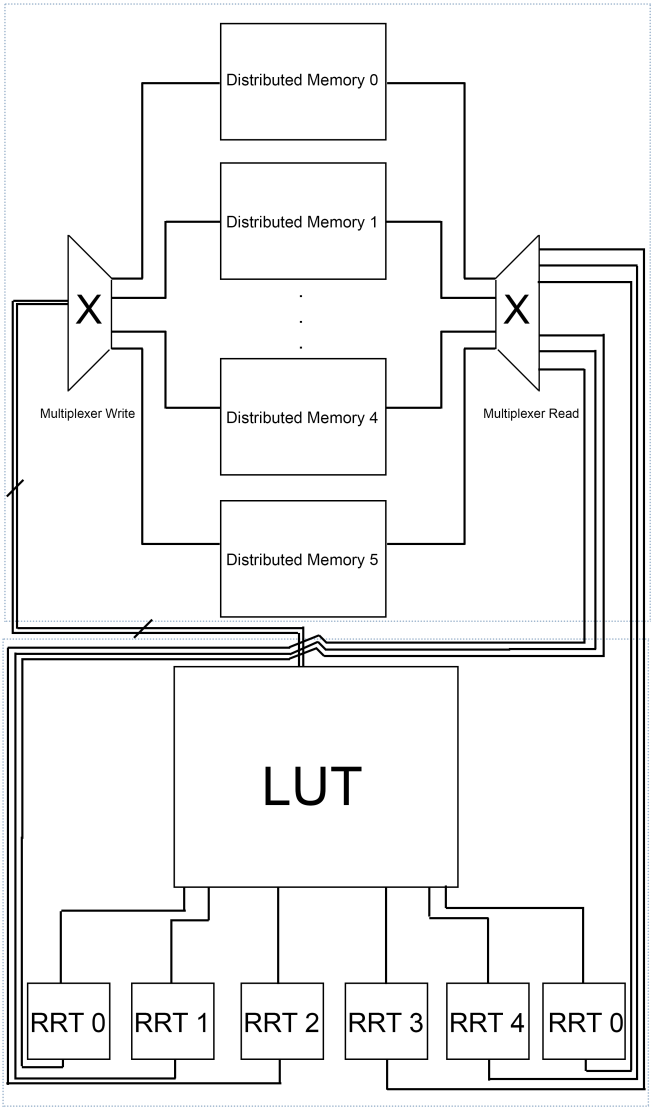


Fig. 9. Top to bottom : Memory, Combinatorial circuit

Fig. 9 details the complete architectural abstract of the system for N = 6. N RRTs can result in 2^N different transaction combinations. As delineated in Fig. 10, the combinatorial circuit and the multi-port memory guarantee that each of these 2^6 combinations complete in a single write window. During first write window, one RRT(R1) requests for access. 3 RRTs(R3, R2, R1) request for access during second window. All 6 RRTs request access during the next window and so on. To sum it up, this ensures that each RRT is granted instant access to the write window upon request, culminating in an accelerated exploration of the global map.

#	R5	R4	R3	R2	R1	R0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
62	1	1	1	1	1	0
63	1	1	1	1	1	1

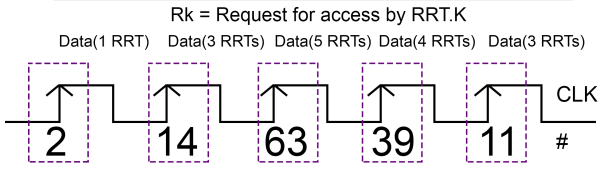


Fig. 10. Combinatorial architecture's instant access to write window

IV. RESULTS

The architecture was designed to attain accelerated, homogeneous levels of exploration that are in proportion to levels of parallelism of the system. We deployed a three wheeled mobile robot equipped with a differentially steered drive system and carried out the trials in 3 maps, each measuring $80X80\ unit^2$, detailed in Fig. 11.

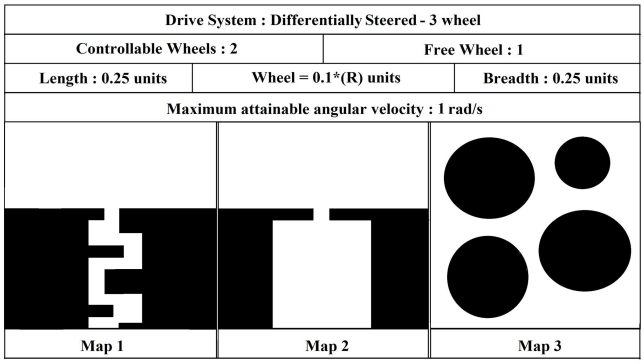
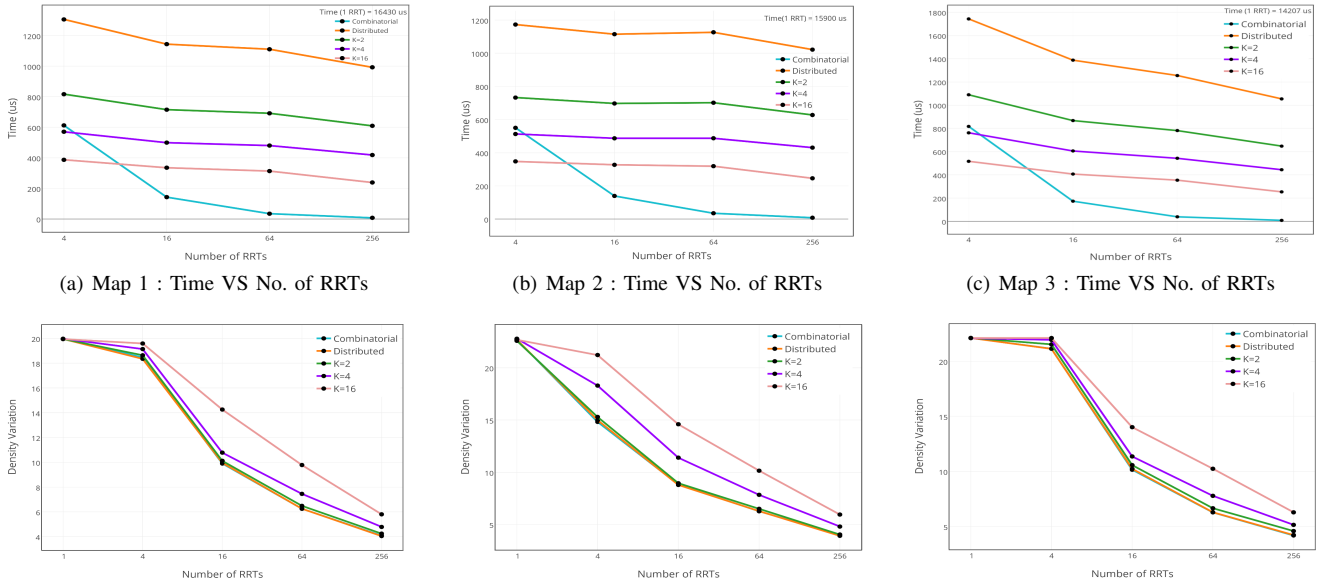


Fig. 11. Details of the test system deployed for quantification of analysis

We analyse the results against 2 different architecture for parallel RRTs : 1.) Distributed RRT [12] and 2.) K-Distributed RRT [13]. Distributed RRT proposes the use of distributed memory architecture that calls for inter-RRT scheduling for access to the global map. With increase in number of RRTs, the traffic on the map bus increases and leads to increased inter-RRT scheduling times. The K-distributed RRT introduces a variable K that quantifies the number of nodes an RRT adds to its local map before requesting for access of the global map. With increase in the value of K, although the inter-RRT scheduling time drops, the global explored space trends towards localised exploration, leading to non uniformly explored map.



(a) Map 1 : Time VS No. of RRTs (b) Map 2 : Time VS No. of RRTs (c) Map 3 : Time VS No. of RRTs

(d) Map 1 : Density variation VS No. of RRTs (e) Map 2 : Density variation VS No. of RRTs (f) Map 3 : Density variation VS No. of RRTs

Fig. 12. Comparative analysis of Combinatorial, Distributed and K-Distributed for the three different test environments

With the 3 maps as the test environment, Fig.12 quantifies the performance of the 3 different architectures of RRTs, with 1.) Acceleration in exploration, 2.) Uniformity of exploration and 3.) Power consumption serving as benchmarks of evaluation. We discuss the performance in these categories in separate subsections.

A. Acceleration in exploration

For quantification of results for each map, time taken to add $10,000$ nodes to the explored space was measured for N = 1, 4, 16, 64, 256 RRTs. Owing to the probabilistic nature of RRT, each iteration was performed 1000 times to get mean values, which are presented in Fig. 12 (a), (b), (c).

Increase in inter-RRT scheduling time with increase in number of RRTs renders distributed RRT to be the slowest architecture of the three. For low number of RRTs, K-distributed RRT for K = 4 and K = 16 performs faster exploration compared to combinatorial architecture. The explanation behind this lies in the 4-way handshake between an RRT and the memory during memory capture. An RRT transfers each node to memory in combinatorial architecture whereas an RRT transfers K nodes in K-distributed architecture. Hence, the handshake occurs once every K nodes in K-distributed architecture compared to combinatorial architecture, where it occurs during every node transfer. The extra cycles spent on handshakes slows down the combinatorial architecture compared to K-distributed architecture for low number of RRTs. But, with increase in number of RRTs, the combinatorial architecture overtakes the K-distributed architecture. The reason behind this observation is that inter-RRT scheduling in K-distributed architecture, reduced as compared to distributed architecture, still increases with increase in number of RRTs. This scheduling eventually weighs out the cycles saved on handshakes. Combinatorial architecture, which suffers from no such scheduling, keeps accelerating with increase in number of RRTs. For N = 256, the combinatorial architecture displays an acceleration gain of 30.81 (Map 1), 30.86 (Map 2) and 30.73 (Map 3).

Please note that the reason behind combinatorial architecture transferring every node instead of K nodes in burst is explained in next subsection.

B. Uniformity of exploration

To calculate the variation in density, the explored space was discretized into 6400 cells. 10,000 nodes were then added to the map. Variance of the obstacle free cells was calculated. Please note that the mean is calculated just for the cells that do not completely contain an obstacle.

$$\sigma^2 = \sum_{i=1}^{6400}(Nodes.cell_i - Mean)^2,\ cell_i \text{ != } obstacle\ .$$

As Fig. 12 (d), (e), (f) explicate, the variance in density of exploration exhibits a downtrend with increase in number of RRTs, indicating increased uniformity. The K-distributed RRT, with increase in values of K, has the least uniformly explored map. The variable K directs this observation. A higher K value will have less scheduling, as shown in Fig. 12 (a), (b), (c) but the RRT will be forced to latch on to an outdated nearest neighbour from the global map, leading to a localised exploration by that RRT. Distributed and combinatorial architecture, however, transfer each and every node (K = 1). With the global explored map updated with the latest data, it allows the RRTs to latch on to the correct nearest neighbour, resulting in an exploration that is more uniform. The same is observed in Fig. 12 (d), (e), (f), with combinatorial and distributed architecture showcasing nearly identical, minimum variance in density.

C. Power consumption by equivalent hardware

Fig. 13 presents the dynamic power consumption by the FPGA realised hardware implementations of the 3 architectures. For small to large scale($N = 1, 4, 16, 64$) parallelization,

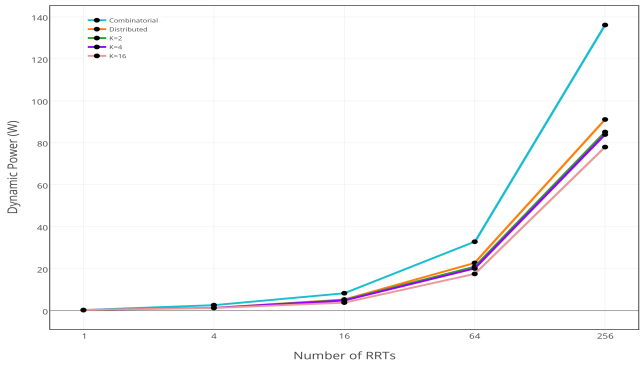


Fig. 13. Dynamic power consumption (Quiescent level = 3.36 W)

the 3 architectures consume nearly identical power levels. But with transition into ultra scale($N = 256$) parallelization levels, the power consumption levels of the combinatorial architecture rise relatively faster compared to the other two architectures. This behaviour is attributed to the increase in size and switching activity of the combinatorial circuit with increase in number of RRTs. The architectural implementation of distributed and K-distributed architectures does not grow in size and switching activity because of its tolerance towards inter-RRT scheduling, leading to lower power consumption levels.

To summarise, the combinatorial architecture's ability to account for all the 2^N cases with no inter-RRT scheduling makes it the fastest architecture for medium to ultra scale parallelization. The architecture was designed to transfer every node with $K = 1$ that results in homogeneous exploration of maps, including maps with tight, constrained pathways like Map 1 and Map 2. The trade off for relatively accelerated, homogeneous exploration is the higher power consumption levels due to the combinatorial accounting of the 2^N cases.

V. CONCLUSIONS

Implementation of a software algorithm on an FPGA enables fundamental hardware level optimizations that offers real time performance and economical power consumption levels. This paper proffered the combinatorial architecture that benefits from the inherent parallel abilities of the FPGA. Quantitative benchmarking of this architecture on maps with tight geometric constraints exhibited the architecture's ability of fast and uniform exploration. The trade off was the higher power consumption levels with the transition to the domain of ultra scale parallelization. These quantitative results were also supplemented with qualitative reasoning. In conclusion, this architecture presents a very favourable combination of speed, uniformity and power consumption in the domain of medium to large scale parallelization.

As part of our future work, we would like to map the optimal number of RRTs required to the features of the map. This would allow us to maintain decent levels of speed and uniformity while keeping the power consumption levels low. Owing to the complex challenges involved in implementing a system on an FPGA, we would like to extend our implementation to robots with higher degrees of freedom. We are confident that the architecture will scale very well with increase in robot's kinematic complexity.

REFERENCES

[1] Jan Fischer, Alexander Ruppel, Florian Weisshardt, and Alexander Verl. A rotation invariant feature descriptor o-daisy and its fpga implementation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2365–2370. IEEE, 2011.

[2] Jan Svab, Tomas Krajnik, Jan Faigl, and Libor Preucil. Fpga based speeded up robust features. In *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*, pages 35–41. IEEE, 2009.

[3] Jason Oberg, Ken Eguro, Ray Bittner, and Alessandro Forin. Random decision tree body part recognition using fpgas. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 330–337. IEEE, 2012.

[4] Roopak Dubey, Neeraj Pradhan, K Madhava Krishna, and Shubhajit Roy Chowdhury. Field programmable gate array (fpga) based collision avoidance using acceleration velocity obstacles. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 2333–2338. IEEE, 2012.

[5] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[6] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[7] Juan Cortés and Thierry Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In *Algorithmic Foundations of Robotics VI*, pages 75–90. Springer, 2005.

[8] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2859–2865. IEEE, 2009.

[9] Russ Tedrake. Lqr-trees: Feedback motion planning on sparse randomized trees. 2009.

[10] Léonard Jaillet, Anna Yershova, Steven M La Valle, and Thierry Siméon. Adaptive tuning of the sampling domain for dynamic-domain rrts. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2851–2856. IEEE, 2005.

[11] Mikael Svenstrup, Thomas Bak, and Hans Jørgen Andersen. Minimising computational complexity of the rrt algorithm a practical approach. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5602–5607. IEEE, 2011.

[12] Didier Devaurs, Thierry Siméon, and Juan Cortés. Parallelizing rrt on distributed-memory architectures. In *Proc. IEEE ICRA'11*, pages pp–2261, 2011.

[13] Nick Stradford, Sam Ade Jacobs, and Nancy M Amato. Scalable parallel rrt method for motion planning.

[14] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Publishing Company Redwood City, CA, 1994.