ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

FPGA-based vehicle real-time path planning system

Kai Yuan*, Guangming Li, Xiaojuan Liang

School of Electronic Information and Artificial Intelligence, Shaanxi University of Science and Technology, Xi'an, Shaanxi, 710021, China
**Corresponding author*

Abstract: *In order to realize the requirements of low latency and low power consumption in real-time path planning for private vehicles, this paper designs an FPGA accelerator based on real-time road information and Dijkstra's shortest path algorithm, which is applied to the path planning system of vehicle edge computing. The system is designed using Xilinx High-Level Synthesis (HLS) compiler and is implemented in the programming logic of a Xilinx Zynq FPGA. The experiment is carried out on a city map, and the results show that the system has lower circuit area and power consumption, and its computing performance is 3.8 times that of ARM, which is suitable for edge computing platform.*

Keywords: *intelligent transportation, edge computing, FPGA, path planning*

1. Introduction

Vehicle path planning is an important part of intelligent transportation system [1]. With the development of edge computing and the Internet of Things, vehicles become more intelligent and personalized. Especially with the rise of automatic driving, vehicles, as intelligent terminals, will rely more on real-time traffic information and require more network traffic and bandwidth, which is a challenge for traditional cloud computing architecture[2-3]. The vehicle path planning system based on cloud computing is difficult to meet the security and privacy requirements of private vehicles, so it is necessary to integrate part of the computing into the vehicle itself, so as to relieve the pressure of network communication and cloud servers.

However, vehicle path planning algorithms tend to have high time complexity, especially for large-scale road networks, which are limited by on-board resources and energy consumption, and the calculation of the optimal path often takes a lot of time, which will not be conducive to the real-time update of vehicle path planning during driving[4-5].Traditional general-purpose processors such as CPU and ARM are suitable for process control, but not for algorithm acceleration; GPU algorithm acceleration effect is good, but the cost and power consumption are high; ASICs consume the least power, and because they are dedicated chips, their acceleration performance is also very high, but they lack flexibility; FPGAs, as programmable devices, can accelerate various algorithms under low-power operating conditions[6]. Therefore, this paper uses the heterogeneous platform of ARM+FPGA as the edge computing carrier of the algorithm to realize the real-time solution of the vehicle path planning problem.

2. Methodology

2.1. Vehicle real-time path planning system architecture

The system edge computing system refers to the OpenFog architecture and consists of cloud, network and edge devices[7]. The system downloads real-time traffic network information from the cloud and calculates the optimal vehicle path in FPGA according to the network. The system architecture is shown in Figure 1.

The right side of Figure 1 shows that the equipment in this paper is a Xilinx ZC706 development board, which is equipped with Zynq XC7Z045 FPGA, and is divided into PS and PL terminals. The PS terminal is ARM Cortex-A9 SoC, which is used for communicate with the cloud, and the PL terminal is a programmable logic resource. HLS is used to deploy the algorithm on it for acceleration.

ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

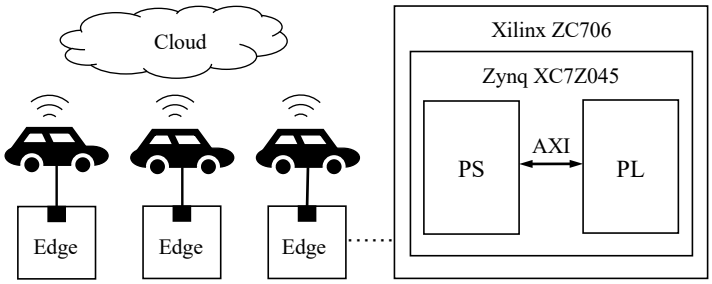

Figure 1: System architecture diagram.

2.2. Vehicle path planning algorithm and acceleration

The path plaining problem is usually described by graph theory and its related data structure. The model is established as (1)

$$\begin{cases} G=(V,E,L) \\ E=\{\langle x,y\rangle \mid x,y\in V\} \\ L=\{l_{xy} \mid \langle x,y\rangle \in R\} \end{cases} \tag{1}$$

Wherein, V is a node set, E is a set of edges, x,y represents the starting and ending points of the road section, L is the set of edge weights. l_{xy} represents the weighted length of the edge, which is affected by the length of the road section, road quality, congestion, cost, etc. For any path, C， which is the weighted path length can be expressed by (2):

$$C=\sum_{i=1}^{n}C_i=\sum_{i=1}^{n}(D_i a_i+(1-a_i)L_i) \tag{2}$$

Wherein, i refers to the number of the road section in the path, D_i refers to the road resistance of the road section, L_i refers to the physical length of the road section, and a_i refers to the proportion weight of the road resistance factor.

There are many vehicle path planning algorithms. For the system architecture proposed in this paper for testing, Dijkstra algorithm is selected for FPGA acceleration. Because this algorithm can obtain an accurate optimal path rather than an approximate solution, it can eliminate random interference, and it has the advantages of both width search algorithm and greedy algorithm, so the efficiency is higher.

Figure 2 shows the C/C++code fragment of Dijkstra algorithm implemented in Vivado HLS compiler. The HLS compiler can expand the loop part of C/C++code, accelerate the algorithm execution efficiency through pipeline or space for time strategy, and finally convert it to RTL level description that can run in FPGA.

```
// AXI4 burst mode copy
memcpy(graph_m,  (const int*)g_addr, n*n*sizeof(int));
int i,n,new_cost;
// perimeter and neighbors array—lists
list_t perim;
isit_t neig;
init_list(&perim) ;    // init 'perim' list size to 0
init_list ( &neig);    // init 'neig' list size to 0
init(path, costs);     // init 'path' and 'costs' static arrays to -1
put_item_prior(&perim, s, 0);
costs[s]=0;
// dijkstra main-loop
while ( !is_empty(&perim)) {
      int v =get_item_prior(&perim);
      if (v ==t)     // early exit
            break;
      // get neighbors of v in graph _m
      neighborhood(&neig,  graph_m, v);
      for (i = 0; i < neig.size; i++) {
            n = neig.data[i];
            new_cost = costs[v] + edge_w(graph_m, v, n);
            if(costs[n] < 0 || new_cost < costs[n]{
                  costs[n] = new_cost;
                  put_item_prior(&perim, n, new_cost);
            }
      }
}
// AXI4 burst mode copy
memcpy((int*)p_addr , path , n*sizeof(int));
```

Figure 2: Core code fragment.

ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

At the beginning of the algorithm, the input road network diagram is mapped to the FPGA block ram for call at any time, thus reducing the transmission delay; Other intermediate variables are mapped to distributed ram, which makes changes more flexible. Figure 3 is the architecture diagram of PL.

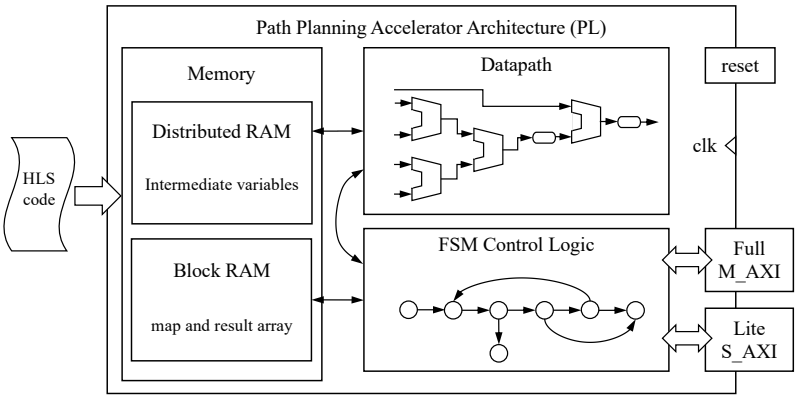

Figure 3: Architecture diagram of PL.

2.3. Interface design

For PS-PL communication interface, two different AXI interfaces are used to maximize the use of FPGA resources. Figure 4 shows the implementation details of the PS-PL side of the FPGA.

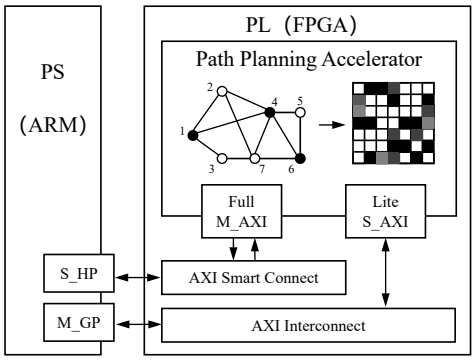

Figure 4: Implementation details of PS-PL terminal.

The AXI4 Lite transmission rate is slow, and it is only suitable for simple, low throughput memory mapping communication, but it takes less resources. Therefore, the system uses AXI4 Lite to transmit control and status signals, and is connected to the general purpose (GP) port on the PS side; AXI4 Full provides high-performance memory mapped PS-PL data transmission, allowing up to 256 bytes of burst mode data transmission, so it is suitable for the transmission of graph nodes, adjacency matrices, and path results, and is connected to the high-performance interface (HP) on the PS side.

3. Experiment and results

3.1. Experimental setup and dataset

The experiment evaluates the performance, power consumption and circuit area of the system in the Xilinx ZC706 development toolbox. The selected data are part of the maps of Beijing obtained from OpenStreeMaps and simplified into directed maps. The nodes include road intersections and passing points of key lines, as shown in Figure 5. The right side of Figure 5 shows the traffic thermodynamic diagram at a certain time.

ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

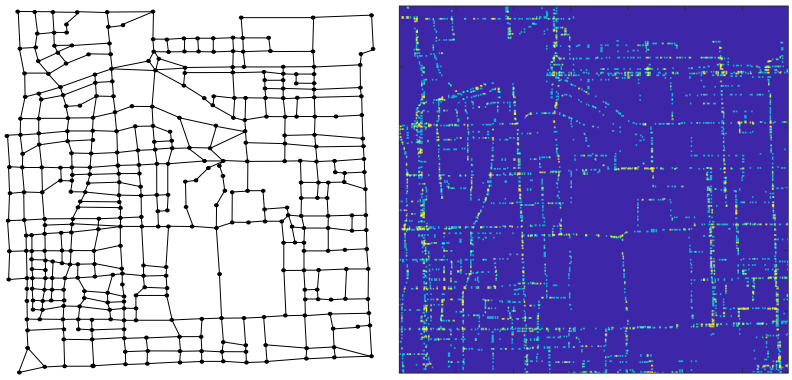

Figure 5: Part of the maps of Beijing.

3.2. System performance

This experiment executes Dijkstra algorithm on FPGA and ARM respectively, and records the number of clock cycles consumed by path planning under different distance conditions, as shown in Table 1.

Table 1: Execution cycles counts of FPGA and ARM.

Cost	ARM	FPGA
5896	22,974,322	6,045,874
5125	19,965,768	5,523,258
3952	12,956,456	4,181,566
2726	5,639,220	2,655,688
1627	2,054,956	1,456,432
969	652,114	934,590
254	325,246	712,566

It can be seen that when the start and finish are close, for example, below 1km, ARM microprocessors can execute Dijkstra algorithm faster than FPGAs. The reason is that the computing scale is small, and the performance is reduced due to additional communication with ARM; However, as the distance between start and finish gradually increases, the computational performance of FPGA gradually highlights. When the distance between the request start and finish is 5896m, the computational performance of FPGA is improved to 3.8 times that of ARM, and the acceleration effect is significant. In addition, since in reality the vehicle path planning is carried out when the starting and ending points are greater than 5km, the acceleration effect of FPGA will be better.

3.3. System power consumption analysis

The power consumption of PS side and PL side during the system operation is measured in the experiment, and each module is subdivided, as shown in Figure 6.

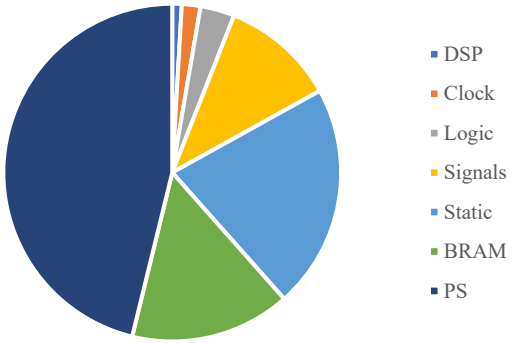

Figure 6: Pie chart of power ratio of each module.

It can be seen that FPGA only takes up a small part of the system power, while the arm side takes up

ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

nearly half. Figure 7 shows the architecture of this paper and the energy consumption change diagram calculated by ARM only when starting and ending points are different. It can be seen that with the increasing difficulty of planning problems, The energy consumption of ARM also increases rapidly; However, the increase of energy consumption on the FPGA side is very small, especially for planning requests that are far from the start and end points, the energy consumption of this architecture can be reduced by more than 10 times compared with ARM processors.

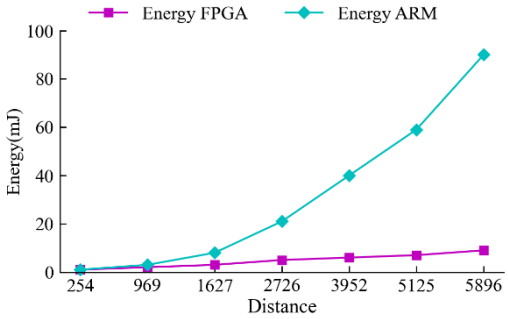

Figure 7: Relationship between request distance and power consumption of ARM and FPGA.

3.4. System circuit area

Table 2 shows the resource occupation of this system. It can be seen that the overall resource consumption of the chip is small, but the utilization rate of BRAM is high. On the one hand, it is necessary to store a large amount of road network data. On the other hand, as an EDA tool, the HLS compiler is difficult to maximize the use of resources on the chip. At the same time, in order to take into account the computing performance and energy consumption, the compiler will make some trade-offs on the consumption of resources on the chip, which leads to the high utilization rate of BRAM in the system, However, it still meets the hardware resource requirements of vehicle edge computing as a whole, which can accelerate the algorithm of vehicle path planning.

Table 2: FPGA resource utilisation.

Resourse	Utilisation	Available	Utilisation(%)
LUT	3954	218600	1.81
LUT RAM	326	70400	0.46
BRAM	4527	437200	1.04
FF	265	545	48.62
DSP	8	900	0.88

4. Conclusion

In this paper, a edge computing method based on FPGA is proposed for real-time path planning of vehicles. By accelerating Dijkstra algorithm on FPGA and using ARM to receive real-time map and information transmission control, real-time update of vehicle path planning is realized. The computing performance is significantly improved, and it has low power consumption and resource consumption, which is suitable for personalized path recommendation of automatic driving in the future. Later, it will be verified on FPGA chips with more resources to play a role in more complex urban road networks.

References

[1] Ming, Y., Li, Y.Q., Zhang, Z.H. and Yan, W.Q. (2022) A survey of path planning algorithms for autonomous vehicles. Sae International Journal of Commercial Vehicles, 14, 97-109.
[2] Boeing, G. (2016) Osmnx: new methods for acquiring, constructing, analyzing, and visualizing complex street. Clinical Orthopaedics and Related Research, 65, 126–139.
[3] Ai, Y., Peng, M. and Zhang, K. (2017) Edge cloud computing technologies for internet of things: a primer. Digital Communications and Networks, 4, 77–86.
[4] Liu, Y., Li, Y., Niu, Y. and Jin, D.P. (2020) Joint optimization of path planning and resource allocation in mobile edge computing. IEEE Transactions on Mobile Computing, 19, 2129-2144.
[5] Ning, Z.L., Zhang, K.Y. and Wang, X.J. (2021) Intelligent edge computing in internet of vehicles: a

ISSN 2616-5775 Vol. 5, Issue 14: 147-152, DOI: 10.25236/AJCIS.2022.051420

joint computation offloading and caching solution. IEEE Transactions on Intelligent Transportation Systems, 22, 2212-2225.
[6] Issa, H.H. and Ahmed, S.M.E. (2019) FPGA implementation of floating point based cuckoo search algorithm. IEEE Access, 7, 134434-134447.
[7] Gebremichael, T., Ledwaba, L.P.I. and Eldefrawy, M.H. (2020) Security and privacy in the industrial internet of things: current standards and future challenges. IEEE Access, 8, 152351-152366.