TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. TX-92/1245-5F	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle The Simulation of Area Wide Real Time Traffic Control--Final Report		5. Report Date November 1991, Revised June 1992 6. Performing Organization Code
7. Author(s) Junguk L. Kim, Steve J-C. Liu, Yilong Chen, Ying Hao, Soonjung Lee, Taesoon Park, Prabaharan Swarnam		8. Performing Organization Report No. Research Report 1245-5F
9. Performing Organization Name and Address Texas Transportation Institute Texas A&M University System College Station, Texas 77843		10. Work Unit No. 11. Contract or Grant No. Study No. 2-1-90/1245
12. Sponsoring Agency Name and Address Texas State Department of Transportation P.O. Box 5051 Austin, Texas 78701		13. Type of Report and Period Covered Final Report September 1990-August 1991 14. Sponsoring Agency Code

15. Supplementary Notes

Research performed with all State Funds.
Research Study Title: Texas Advanced Transportation Technology Project

16. Abstract

This study was performed to develop the Area-Wide Real-Time Traffic Control (ARTC) System which uses real-time traffic flow information and design signal timing plans online. The ARTC system consists of microcontrollers at each intersection to perform signal timing computations which are connected to each other by a communication network. The signal controllers exchange traffic flow data to aid in the signal timing computations. A simulation of the ARTC system was performed and compared with a fixed time control which was computed using the TRANSYT7F program. Results of the simulation show significant savings in delay and stops by ARTC over fixed time control.

Under the ARTC system, signal controllers must process large amounts of real-time traffic information and communicate with neighboring signal controllers while making traffic control decisions. A modification of the TYPE179 controllers to suit the needs of the ARTC system and future traffic management systems may be expensive and infeasible. A design of the signal controller using the microcontroller MC68332, which has the desired features for a typical signal controller, is also presented.

17. Key Words Traffic Signal, Traffic Adaptive, Real-time Signal Control, Traffic Management Center	18. Distribution Statement No Restrictions. This document is available to the public through the National Technical Information Service 5285 Port Royal Road Springfield, Virginia 22161

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 59	22. Price

Form DOT F 1700.7 (8-69)

SIMULATION STUDY OF THE ARTC SYSTEM

by

Junguk L. Kim
Assistant Research Specialist

Steve J-C. Liu
Assistant Research Specialist

and
the following Research Assistants

Yilong Chen

Ying Hao

Soojung Lee

Taesoon Park

Prabaharan Swarnam

Texas Transportation Institute

Research Report 1245-5F

Research Study Number 2-1-90/1-1245
Texas Advanced Transportation Technology Project

Sponsored by

Texas Department of Transportation

Texas Transportation Institute
The Texas A&M University System
College Station, Texas 77843

June 1992

METRIC (SI*) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

LENGTH

Symbol	When You Know	Multiply By	To Find	Symbol
in	inches	2.54	centimetres	cm
ft	feet	0.3048	metres	m
yd	yards	0.914	metres	m
mi	miles	1.61	kilometres	km

AREA

Symbol	When You Know	Multiply By	To Find	Symbol
in²	square inches	645.2	centimetres squared	cm ²
ft²	square feet	0.0929	metres squared	m²
yd²	square yards	0.836	metres squared	m²
mi²	square miles	2.59	kilometres squared	km²
ac	acres	0.395	hectares	ha

MASS (weight)

Symbol	When You Know	Multiply By	To Find	Symbol
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams	Mg

VOLUME

Symbol	When You Know	Multiply By	To Find	Symbol
fl oz	fluid ounces	29.57	millilitres	mL
gal	gallons	3.785	litres	L
ft³	cubic feet	0.0328	metres cubed	m³
yd³	cubic yards	0.0765	metres cubed	m³

NOTE: Volumes greater than 1000 L shall be shown in m³.

TEMPERATURE (exact)

Symbol	When You Know	Multiply By	To Find	Symbol
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

APPROXIMATE CONVERSIONS TO SI UNITS

LENGTH

Symbol	When You Know	Multiply By	To Find	Symbol
mm	millimetres	0.039	inches	in
m	metres	3.28	feet	ft
m	metres	1.09	yards	yd
km	kilometres	0.621	miles	mi

AREA

Symbol	When You Know	Multiply By	To Find	Symbol
mm²	millimetres squared	0.0016	square inches	in²
m²	metres squared	10.764	square feet	ft²
km²	kilometres squared	0.39	square miles	mi²
ha	hectores (10 000 m²)	2.53	acres	ac

MASS (weight)

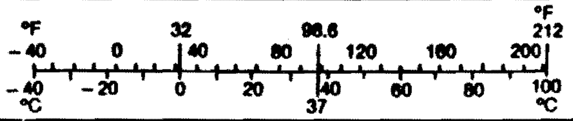
Symbol	When You Know	Multiply By	To Find	Symbol
g	grams	0.0353	ounces	oz
kg	kilograms	2.205	pounds	lb
Mg	megagrams (1 000 kg)	1.103	short tons	T

VOLUME

Symbol	When You Know	Multiply By	To Find	Symbol
mL	millilitres	0.034	fluid ounces	fl oz
L	litres	0.264	gallons	gal
m³	metres cubed	35.315	cubic feet	ft³
m³	metres cubed	1.308	cubic yards	yd³

TEMPERATURE (exact)

Symbol	When You Know	Multiply By	To Find	Symbol
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



These factors conform to the requirement of FHWA Order 5190.1A.

* SI is the symbol for the International System of Measurements

Abstract

This study was performed to develop the Area-Wide Real-Time Traffic Control (ARTC) System which uses real-time traffic flow information and design signal timing plans online. The ARTC system consists of microcontrollers at each intersection to perform signal timing computations which are connected to each other by a communication network. The signal controllers exchange traffic flow data to aid in the signal timing computations. A simulation of the ARTC system was performed and compared with a fixed time control which was computed using the TRANSYT7F program. Results of the simulation show significant savings in delay and stops by ARTC over the fixed time control.

Under the ARTC system, signal controllers must process large amounts of real-time traffic information and communicate with neighboring signal controllers while making traffic control decisions. A modification of the TYPE179 controllers to suit the needs of the ARTC system and future traffic management systems may be expensive and infeasible. A design of the signal controller using the microcontroller MC68332, which has the desired features for a typical signal controller, is also presented.

Disclaimer

Summary

Research Report 1245-4 developed the ARTC concept. The ARTC system consists of "intelligent" signal controllers at each intersection, which are microcontrollers capable of performing intensive computations. These signal controllers are connected with each other by a communication network and hence can exchange traffic flow data to aid in the signal timing computations. The traffic system, under ARTC's control, is divided into regions and each region contains a set of the signal controllers which have one regional controller which periodically collects traffic flow data for providing congestion control and progression. The signal control protocols which were designed for the signal controllers and the regional controllers are believed to improve the performance of the traffic flow. To justify this belief we developed a simulation model and have performed an intensive simulation of this system.

The current simulation of the traffic network is not capable enough to reflect the effects of the ARTC communicating signal controllers. Hence an expansion of the traffic network will significantly improve the performance limit of the system. Moreover, simulation and comparison with other on-line control algorithms is essential to justify the performance of the ARTC system.

Contents

1 Introduction **1**

2 Simulation System **3**

2.1 Design of the Simulation System 5

2.2 Graphics Interface 6

3 Survey of Traffic Control Methods **8**

3.1 Webster's Model 8

3.1.1 Delay Equation 9

3.1.2 Split and Optimal Cycle Length 12

3.1.3 Evaluation 13

3.2 OPAC System 13

3.2.1 Optimal Signal Settings 14

3.2.2 Implementation 15

3.2.3 Evaluation 15

4 Simulation of the Traffic System **17**

4.1 Traffic Control Strategies 17

4.2 ARTC Version-I Algorithms 18

4.2.1 Queue Estimation 22

4.3 Performance Evaluation 25

4.4 Analysis 26

4.5 Regional Controller 29

5 Design of ARTC Signal Controllers **38**

5.1 Upgrading TYPE179 to Meet the ARTC Requirements 38

5.2 Design of ARTC Signal Controller Using VMEbus 41

6 Summary and Future Work **52**

List of Figures

1 Geometry of Traffic Network Simulated 4
2 Module Interactions in the ARTC Simulation System 6
3 The Calculation of the Webster's Formula 10
4 OPAC Algorithm 16
5 Computation of Extension for Split 21
6 Optimal Point for D_{green} and D_{red} 22
7 Queue Estimation Using CFP and Flow Rates 24
8 Two Phase Sequence 26
9 Varying Arrival Rate With 2% Left Turn From Arterial 30
10 Varying Arrival Rate With 10% Left Turn From Arterial 31
11 Fixed Arrival Rate With 2% Left Turn From Arterial 32
12 Fixed Arrival Rate With 10% Left Turn From Arterial 33
13 Block Diagram of ARTC Signal Controller 44
14 Block Diagram of Control Module 46
15 Conversion Between VMEbus Control Signals and MC68332 Control Signals 47
16 Design of the Back-up Mechanism in ARTC Controller 49
17 Block Diagram of the Input/Output Modules 50
18 I/O Expansion in Input/Output Module 51

1 Introduction

In the first phase of the Texas Advanced Traffic Management Project the conceptual model of the ARTC was developed; in this phase we present the simulation model and the design of the signal controller. The ARTC system consists of "intelligent" signal controllers at each intersection, which are microcontrollers capable of performing intensive computations. These signal controllers are connected with each other by a communication network and hence can exchange traffic flow data to aid in the signal timing computations. The traffic system, under ARTC's control, is divided into regions and each region contains a set of the signal controllers which have one regional controller which periodically collects traffic flow data for providing congestion control and progression. The signal control protocols which were designed for the signal controllers and the regional controllers are believed to improve the performance of the traffic flow. To justify this belief we developed a simulation model and have performed an intensive simulation of this system.

Various traffic control systems which have been developed and successfully implemented in the past are SCOOT, SCAT and OPAC. The difference between these systems and the ARTC system are that the signal timing computation in these systems is performed by one central controller, which is the regional controller, and the signal controllers execute the timing plan as computed by the regional controller. Hence, when the regional controller fails, the signal controllers switch back to the primitive form of fixed time traffic control. The signal controllers have the authority to only extend or shorten the phase duration by a few seconds, thus the autonomy of the signal controllers is very limited. However, in the ARTC system the signal controllers compute the timing plans by directly communicating with their neighbors and with data from the detectors. Moreover, the regional controller in the ARTC system periodically collects data and provides information about traffic flow paths which tend to show an increase in traffic, thus warning the signal controllers to lengthen or shorten the green time appropriately. A restriction in green time can prevent the occurrence of congestion by limiting the incoming flow rate to match the output flow rate at the exit points of the traffic network. The lengthening of green time can provide progression to the vehicles in those paths if the flow can be drained at the exits. In SCOOT, congestion is controlled by varying the cycle time by extending or reducing it by only a few seconds, but the resulting cycle does not efficiently serve sudden surges in traffic which will need an immediate large increase in the cycle length.

Under the ARTC system, signal controllers must process large amounts of real-time traffic information and communicate with neighboring signal controllers while making traffic control decisions. Thus a systematic design approach is required for the signal controllers to meet the requirements of ARTC and the anticipated needs of the future traffic management systems. This design of the signal controller for the ARTC system was introduced in the first phase of this project and the details of this design are presented in this report. The *Traffic Control Hardware Type* 179 (TYPE179), which is used in New York and California, has been reviewed and it does not satisfy the needs of the ARTC system. A modification of the TYPE179 to suit the needs of the ARTC system and future traffic management systems may be expensive and infeasible. One limitation in the TYPE179 is that it cannot perform extensive computations and recover from failure of components. These required qualities exist in a microcontroller MC68332, which can be used in the design of the signal controller. Moreover, the MC68332 has unique features to support real-time event management and is highly reliable in operation.

This report is organized as follows: in Section 2 we present the description of the simulation system and the graphics system which we used to evaluate the performance of the traffic strategies. A review of two traffic control methods, the Webster's equation and the OPAC system, are presented in Section 3. The ARTC version-I control algorithm is discussed in Section 4. The input parameters for the simulation and the performance index measures and analysis of the results are also given in Section 4. Two approaches of the ARTC signal controller designs are presented in Section 5. The report concludes in Section 6, where the future direction of our work is described.

2 Simulation System

A simulation of the ARTC was performed and compared with the results of an optimized fixed time signal control plan on an arterial model. The timing computation for this fixed time control was done off-line with the use of the TRANSYT7F program to provide the cycle and split durations for specified arrival rates and geometry of the intersection.

The model which we simulated consisted of an arterial with four intersections, as shown in Figure 1. Three lanes were provided for each approach, with the left lane being exclusively used for vehicles making left turns and the right lane being shared by vehicles turning right and straight ahead. The center lane was used by vehicles moving straight ahead only. All vehicles were modeled to travel at a uniform fixed speed and were allowed to change lanes. The vehicles were assigned lanes on a normal distribution with a prespecified average value for each lane. The vehicles were allowed to change into a lane if the gap between the vehicles in that lane was sufficient to accommodate the lane changing vehicle. In our simulation we used a gap of 5 vehicle positions between vehicles was used to accommodate a lane changing vehicle in that gap. The characteristics of the drivers were assumed to be rule abiding and no untoward incidents were assumed. Moreover, one segment of the arterial stretch, indicated in Figure 1, was designed with a length higher than the rest of the links

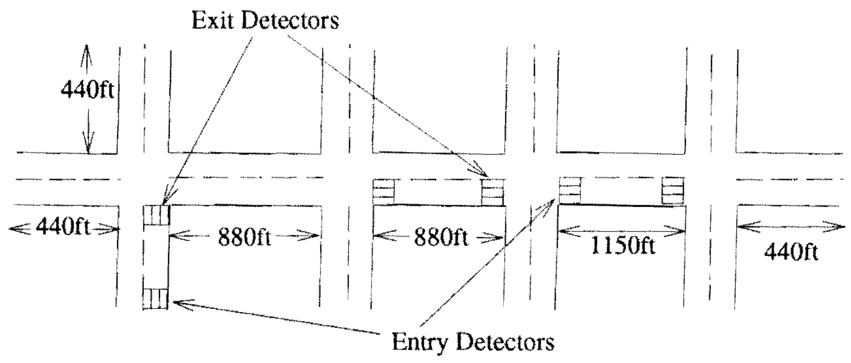


Figure 1: Geometry of Traffic Network Simulated

in the arterial stretch thus making the model asymmetric. Entry detectors were placed at the entry point of each lane and exit detectors were placed at the exit point of each lane, which are shown in Figure 1. The arrival rates of the vehicles, which is specified as an input parameter, is generated at the prespecified fixed rate using an exponential distribution. For example, if it was specified that an average of 400 cars per hour would enter a lane from the exterior of the system, then on a random basis 400 cars would be generated per hour. Moreover, platoons formed at the intersections were dispersed uniformly, *i.e.*, the platoon moved at the uniform speed and a few vehicles from the platoon exited the arterial stretch at the appropriate intersections.

A simple two-phase sequence was used in the timing plan, and two all red periods, which were 3 seconds long, were provided between each phase. This 3 seconds of all red represented the loss time experienced by the vehicles during each phase change. The drivers were assumed to respond immediately to a signal change from red to green. However, since we did not have a yellow phase, the loss time experienced by the vehicles due to deceleration and acceleration was included in the all red period.

2.1 Design of the Simulation System

The traffic system is broken into various objects such as cars, roads, signals, etc. These objects in our system are classified into *active* and *passive* objects; the active objects generate events during the course of the simulation which affect other objects in the system, and the passive objects are manipulated by the active objects.

The active objects in our system are cars and signal controllers which generate events to indicate the changes they will make and also modify the passive objects like the data structures which represent roads, detectors and signal lights. The events generated by the active objects are interdependent, and hence the state of the active objects are interlinked and care has to be taken to reflect these changes in the simulation of the traffic system. For the simulation three event lists are used. The interactions between the various modules in the simulation system are shown in Figure 2.

In Figure 2, the three main objects which play an important role in the simulation are the modules *car.c, signal.c* and *controller.c*. The basic functions required for the simulation are provided in the *simulator.c* module. These functions are used to report current system time, create a event list, get an event from a list and insert an event in the list, etc. The other modules use these functions to manipulate the lists during the simulation.

The *input.c* module is called by the *main* module at the start of the simulation to gather the simulation input parameters. The input module is designed to accept the parameter values, by prompting the user, either through a input parameter file containing the values or through an interactive mode where the user enters the values. The *initdata.c* module initializes the simulation system to start the simulation by reading the values from the input file and allocating data structures for the different objects.

The modules *car.c, signal.c* and *controller.c* perform the actual movements of vehicles,

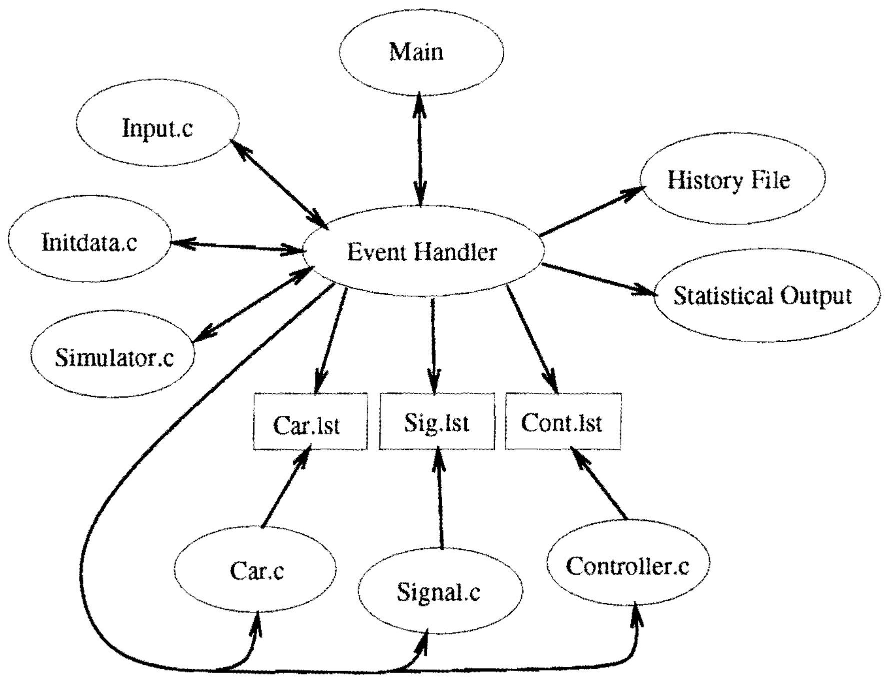


Figure 2: Module Interactions in the ARTC Simulation System

changing of signal lights and the optimization of the signals. These modules produce the appropriate positions of the vehicles in the road network and status of the signal lights which are put in the history file. This history file is then used by a graphics routine to display the animation of the vehicles which were simulated. Moreover, the values of the performance output parameters, which are of interest, are stored to display statistics of the simulations

2.2 Graphics Interface

We have separated the graphics display from the main simulation for the purpose of solving the real-time display lag and for saving time. The interface for the graphics system will be through a history file, which keeps track of the changes in the traffic system during the simulation. The graphics system takes the history file and displays the results according to the time indicated for each event in the history file. The events in the history file are ordered according to time.

History File Format:

The history file will be created during the course of the simulation. The necessary data for the graphics display for the cars and signals are:

- ID # - is unique for each car and signal.
- Event - the various events are move, stop, turn etc. For a signal this indicates the traffic light combinations.
- Position - the position at which it is located on the screen.
- Type - the different types of vehicles are car, truck, etc.

This essential information has to be stored in the history file in a way so as to be accessed easily by the graphics program. The only events which affect the display pattern are the generation of cars, the exiting of cars, the move event, the stop event, the turn event, and the signal change event. During the simulation, when one of these events is generated for an object, the event indicates a change in the display pattern. These events are saved with the event time, which is the simulated time, to identify the events. With the graphics interface we can view the animation of the simulated traffic data and the statistical graphs reflecting the performance of the traffic system.

We now describe the specifications of the various display screens which we plan to provide in the next stage of the graphics development.

- Interactive animation of the data

  We plan to provide interactions during animation to perform various functions such as tracing the path of a vehicle, to redisplay or fast forward the display as required.

- Snapshot of the ongoing animation

  This will freeze the display either at that instant or at a prespecified time or when a specified situation occurs.

- Various menus and interactions

  The user can choose any existing simulated data outputs or can run the simulation program from the graphics environment to produce a new set of data for a defined input parameter specification.

3 Survey of Traffic Control Methods

To obtain the optimal signal settings in traffic systems, the control strategy has to adapt to the real traffic situation of a traffic system. In this section a review of two control strategies is presented.

3.1 Webster's Model

Webster's equation, which is an off-line control strategy, is used to calculate the optimal signal settings for an intersection based on the average influx traffic volume into the intersection for each approach. Webster proposed a mathematical model to evaluate the average delay of vehicles in an isolated intersection. In this model, each approach leading to an intersection is modeled as an ideal queue, that is, the physical structures of the approaches are not considered in the model. Therefore, the calculated vehicular delay is the time elapse between the time the vehicle stops in the approach and the time it starts again.

As known, the signal in an intersection is set in such a way that a green signal can only be given to non-conflict approaches, that is, signals are set by phases. In the four-phase plan, for example, the queues numbered i can only be served during phase i ($i = 1, 2, 3, 4$). It will be complicated to solve this queuing model since the states of queues are dependent with each other. In their work, Webster simplified the model by first assuming that the queues are independent of each other and then analyzing their interactions through the relations of the phases.

Based upon the model, an optimal signal setting for this intersection is obtained which minimizes the average vehicular delay in the intersection.

3.1.1 Delay Equation

We first survey the method used in Webster's to obtain the average queuing delay for an arbitrary queue in an intersection.

Assume that an intersection has a cycle length c with effective green time g and red time r, thus $c = g + r$. Further, we assume the average flow rate, i.e., the number of vehicles passing a point in an unit time, in the approach to be q and the saturation flow of the approach to be s. It is obvious that on the average the total number of vehicles entering the intersection during a cycle time should be less than the number of vehicles which can pass the intersection during the effective green time, *i.e.*, $qc < gs$, so that the system is not in the over-saturated situation. Let d denote the average delay each vehicle in this approach experiences, λ denote the proportion of cycle which is effectively green for the phase under consideration, i.e. g/c, and x as the degree of saturation, *i.e.*, $x = qc/sg = q/\lambda s$.

The following assumptions are also made in the model to simplify the mathematical

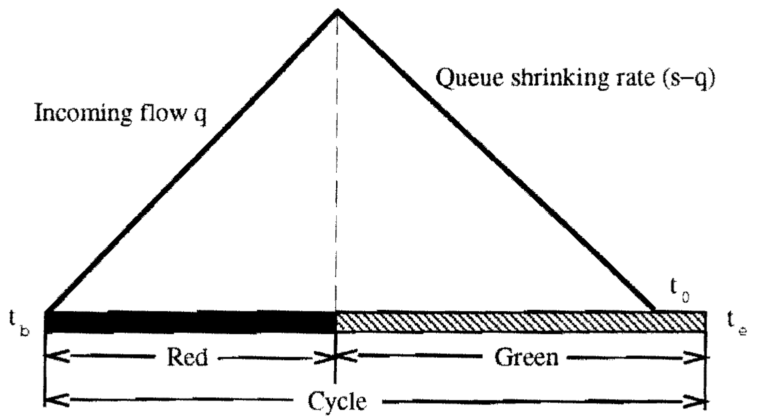


Figure 3: The Calculation of the Webster's Formula

manipulations:

1. Traffic flow to an intersection is independent of current signal settings of the intersection.

2. After the end of a (effective) green phase, all of the vehicles, which were piled up during the red phase, will be totally discharged. Equivalently, at the time when the red phase begins, no vehicle is waiting in the approach.

3. The intersection is considered as an ideal queue, i.e., no physical structure of the road is considered in the model.

So, vehicles arrive in an intersection with an average headway $1/q$, they pile up and form a queue during the red time r. Then, during the subsequent green time, the vehicles leave with the regular headway $1/s$ until the end of green time or the queue is discharged, say at t_0, as shown in Figure 3. From the assumptions, it is straightforward that $t_0 \leq t_e$.

The vehicular delay in an intersection is caused by two factors: the blocking in red time and the randomness of vehicle headway. It may be noted that when the degree of saturation x is low, i.e., the average number of vehicles flowing into the intersection, in a cycle, is small compared to the vehicles flowing out of the intersection during the green time at the saturation flow rate. The random behavior of the arrival pattern has little effect on the behavior of the queuing delay, since in this case the behavior of individual vehicles will not affect the behavior of other vehicles significantly. When the degree of saturation is high, however, the random pattern of arriving vehicles will have a very significant effect on the behavior of other vehicles and thus the average queuing delay. Therefore, Webster evaluated the vehicular delay in two terms. The first term evaluates the delay considering only the effect of blocking by the red phase and the second term evaluates the delay considering only the effect of randomness of the vehicle headway without considering the blocking effect.

We first show how the first term is derived. As shown in Figure 3, vehicles arriving between t_b, at which the cycle begins, and t_0 are blocked and thus experience delay in the queue. Vehicles arriving between t_0 and t_e will not experience any delay in this case. In this figure the building rate of the queue is the vehicle arrival rate q thus qr vehicles arrive during the red time. In the green time the queue is discharged at the rate of s with arriving rate being still being q. Thus the actual discharge rate of the queue will be $s - q$. Without considering the random behavior, the total delay will be the area of the triangle in Figure 3, i.e., $d_{total} = \frac{qr^2}{2(1-\lambda x)}$, so, the average delay per vehicle will be $d_1 = d_{total}/qc = \frac{c(1-\lambda)}{2(1-\lambda x)}$ since the total arrival in a cycle is qc.

The second term incorporates the effect of the randomness of headway into the model by considering the arrival pattern as a Poisson stream, and the service time in the whole cycle as deterministic with a value g/cs. This term does not consider the red time (blocking phase) separately but takes the cycle as a whole, so it is only suitable in a situation with a high degree of saturation. The second term is derived by using $M/D/1$ queue with mean arrival rate q and the service rate sg/c, which is $d_2 = \frac{x^2}{2q(1-x)}$. Combining these two terms, the Webster's delay formula for one queue is obtained as:

$$d = d_1 + d_2 = \frac{c(1-\lambda)}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)}.$$

3.1.2 Split and Optimal Cycle Length

Assume we have a n-phase signal setting, the average vehicular delay in the intersection is the summation of the average delay for vehicles in each approach and in each phase. As shown in the delay equation, the delay in a approach is decided only by the cycle time of the intersection and the effective green time of the phase. Thus the average vehicular delay in the intersection can be denoted as a function of cycle length c and effective green time of each phase g_i $(i = 1, 2, \cdots, n)$, given the vehicle flow rate in each approach. However, by using the suggestion from the traffic engineering handbook, *the least delay to traffic is obtained when the green periods of the phases are in proportion to the corresponding ratios of flow to saturation flow.* So, $g_i = (c - L)\frac{q_i/s_i}{\sum_j^n q_j/s_j}$, where c is the cycle length, and L is the lost time in a cycle. In such a way, the average vehicular delay in an intersection is denoted as the function of only c.

Now we show how the optimal cycle length is derived. For simplicity, in the phase i of a cycle, only the approach with maximum utilization, i.e., the ratio of flow to saturation flow, is selected to reflect the average vehicular delay in the phase and let y_i denote the value q_i/s_i. Therefore, the average delay a vehicle experiences in phase i is:

$$d_i = \frac{c(1-\lambda_i)}{2(1-\lambda_i x_i)} + \frac{x_i^2}{2q_i(1-x_i)}$$

with $\lambda_i = g_i/c$, $x_i = q_i/\lambda_i s$ and $g_i = (c_L)q_i/s$. So the average delay per vehicle in the intersection is $D = \frac{\sum_{i=1}^n d_i q_i}{\sum_{i=1}^n q_i}$.

Webster solves the linear optimization problem by directly using the *first order necessary condition*, i.e., $dD/dc = 0$ and gets the optimal cycle length c_0 with some necessary approximation. That is,

$$c_0 = \frac{2L}{1-Y}\{1 + \frac{\sqrt{Y^2 - Y + 1/E} - Y}{2}\},$$

where, $Y = \sum_{i=1}^{n} y_i$ and

$$E = \frac{L(1+Y)^2}{16nY^5} \sum_{i=1}^{n} \frac{y_i s_i}{1-y_i}\{4(Y-y_i)^2 - y_i(1-Y)^2\}.$$

3.1.3 Evaluation

The Webster model provided a solution to get the optimal signal setting in an isolated intersection. This formula is easy to compute and thus very significant to the early signal controller, in which very little computing power exists. The derivation of the vehicular delay, however, is based totally on the average traffic flow, and the vehicle travel time is not considered when deriving the optimal cycle length. The method is thus not adequate to be used for the abnormal traffic situation.

3.2 OPAC System

Some interesting work has been done using Optimized Policies for Adaptive Control (OPAC). A dynamic programming concept is employed in the system to obtain the optimized signal settings. This method discards the classic concepts of split and cycle, and signals are set dynamically without fixed cycle. That is, given the real traffic profile of a time period, the signal is set in such a way that the total delay in this period is minimized. So, the objective of optimal dynamic traffic control is to determine optimal signal settings for a time period given the traffic profile of this time period.

OPAC discards the classical concept of the split and cycle length and uses dynamic programming concept to decide the optimal splits for one cycle.

The following assumptions, which are similar with the Webster model, are made for a traffic system.

1. Intersections are independent of each other. Approaches of an intersection are also independent to each other.
2. Traffic flow to an intersection is independent of system state.
3. The traffic profile can be obtained from the detector embedded in the upstream of each approach.

3.2.1 Optimal Signal Settings

In order to use the dynamic programming, the time is considered as slotted in the OPAC system. Each time slot is 5 seconds and thus a cycle time in conventional traffic system, which has a time period of 50-100 second, is divided into 10-20 slots. In OPAC system, the 10-20 (denoted as k) continuous slots (or intervals) are called a stage. So, a stage is about the same length as that of the cycle length in a conventional system. Further, it is assumed that at least one and at most three signal changes (splits) can occur in this k-slot time interval and the split must happen at the beginning of a slot. The problem is to find the three time instants (s_1, s_2, s_3) at which the signal lights have to be changed such that the vehicle delay in that cycle is low.

3.2.2 Implementation

As assumed before, the traffic profile of an approach can be obtained from the upstream detector of the approach. Therefore, if we can get the traffic profile of the k stage, we can get the optimal settings s_1, s_2, s_3 by minimizing the traffic delays in this cycle. However, it is impractical to obtain the k-slot traffic data because the street between two adjacent intersections is not long enough to allow 50-100 seconds. So, a method, called *rolling horizon approach* is used to solve this problem.

In this approach, we just use a portion of real traffic data (say r slots) to set the signals in the k-slot cycle. This r slots are on the head portion of the k-slot as illustrated in Figure 4. The traffic flow rate in the rest of $k - r$ slots is assumed to be average flow rate. Dynamic programming is used in this k slots to find the optimal settings s_1, s_2, s_3. However, only s_i which falls into the first r slots will be applied to the control system. After this step, the optimal settings of interval $2r, \ldots, k + r$ will be calculated as shown in Figure 4. So, the upstream detector in the system will send the traffic profile every r slot time to the signal controller, and the controller calculates the optimal settings and applies to the intersection.

3.2.3 Evaluation

OPAC sets the traffic signal optimally based on the current traffic situation. Three field tests for isolated intersections have been conducted. The first test uses the two-phase signal plan with light traffic flow. It is reported that the average delay decreased by 3.9 percent and the average number of stops decreased by 1.6 percent compared with the actuated control. The second test also uses the two-phase plan but with heavy traffic flow. It is reported that the average delay decreased by 15.9 percent but the average number of stops

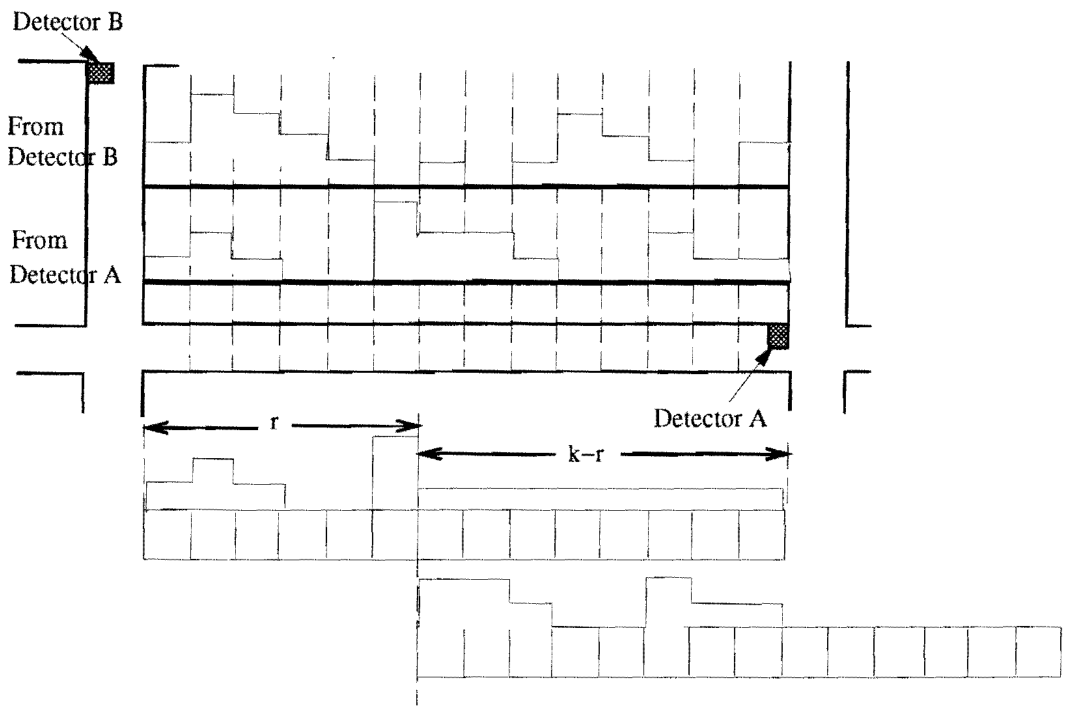


Figure 4: OPAC Algorithm

increased by 3.9 percent. The third test uses the 8 phase dual-ring plan and it is reported that the average delay decreased by 7.7 percent and the number of stops increased by 9.5 percent.

In OPAC, the optimization is done on an isolated intersection. In fact, the states of neighboring intersection have to be considered. Therefore, the optimality of intersections is interdependent and such optimality cannot be achieved by using OPAC of its computational complexity.

4 Simulation of the Traffic System

The signal timing computations in the ARTC include the computations by the signal controllers and the commands from the regional controller which influence the signal timing decisions at the signal controllers. Moreover, the signal controllers perform the signal computations and compute the split and cycle, using data from the detectors and from their neighboring controllers. The signal controllers in the ARTC system, being adaptive, have to vary the split, *i.e.* the length of the green phase, for a link depending on the traffic flow in that link. In this section we explain the simulated version of the signal computation method used by the signal controllers and an overview of the computation performed by the regional controller. In the appendix we have explained a simulation of ARTC with a fixed time control with a arterial model and have included the results.

4.1 Traffic Control Strategies

Two forms of signal control were simulated and compared; a fixed time control and an on-line control. The on-line control, which we named ARTC version-I, was further refined and the ARTC version-I(extension) was developed and simulated. The timing computation for the fixed time control was done off-line with the use of the TRANSYT7F program to provide the cycle and split durations for specified arrival rates and geometry of the intersection. Another off-line control which was also simulated was the split computed using the classic Webster's formula.

In the simulated model of the on-line control, we use the the *actual* current flow rates in the links, measured at the entry detectors, for each split computation. Since the measured flow rates at the detectors might vary from cycle to cycle, depending upon the neighboring signal controller, we use the average of the flow rates measured for the past few cycles. For a two-phase sequence, the cycle consists of two splits, a split for each approach, and two all red periods which are 3 seconds long. This 3 seconds of all red represents the lost time experienced by the vehicles during each phase change.

4.2 ARTC Version-I Algorithms

The basic philosophy in this first step towards implementing ARTC was the following:

1. No vehicle stops more than once at an intersection
2. The entire queue must be dissolved in the computed split

The split duration for each phase change in the version-I control is computed just before the actual change of phase as follows.

1. The current flow rate entering the approaches, which are currently in the red phase, are calculated using the entry detector data. The flow rates for each link are defined as the vehicles which have entered the link since the last signal change at that intersection in the time-period of vehicle counting.

2. Using the flow rate data measured in the past few cycles (our implementation uses the data from the past five cycles), the average flow rate in the approaches

is calculated, which is f_{red}.

3. The queue lengths of the vehicles piled up at the approaches, which are currently in the red phase, are calculated and the longer queue value Q_{red}, among these queue values, is used to compute the split time.

4. The time taken to dissolve this queue, which is the split, is computed using the following equation.

   The queue at the approach will start flowing with a saturation flow S, when the signal changes green. However, the flow f_{red} will still be entering the link. Hence, the rate at which the queue Q_{red} will disperse is $S - f_{red}$.

   $Split = \frac{Q_{red}}{(S - f_{red})}$

The split duration resulting from the above method is sufficient to dissolve the queue present in the approach and the vehicles which will append to the queue due to the arrival rate f_{red}.

However, this approach is a greedy one since it does not consider the flow rate of the cross link during the phase change. Hence, a modification to the above method, which we called version-I(extension), performed the following steps for the split computation, which is also shown in Figure 5. An extension factor was used, which determined whether the current phase should be extended or changed, depending upon the flow rates and queue at time of phase change. This computation for determining whether to extend or not was performed at the time of the original scheduled change of the phase. The version-I(extension) ARTC computation performed the following steps.

As shown in Figure 5, t_{red} and t_{green} represent the red time and the green time for the approaches lying along the North-South direction. The upward slopes indicate the growth of vehicles in the queue when an approach faces a red light. The flow rate in the approach lying along the North-South direction is f_{green} and the flow rate in the approach lying along the East-West direction is f_{red}. The longest queue value of one of the approaches which is currently red is denoted by Q_{red}. The longest queue value during the past red phase of the approach which is currently in green is represented by Q_{green}. Initially t_{ext}, which is the extension time, is set to 0 and is incremented by discrete amounts of extension time given by a prespecified constant *EXT-TIME.*

1. At scheduled change for phase, compute the flow rates for both directions, *i.e.* f_{red} and f_{green}.
2. Find the split duration of the current green phase t_{green}.
3. Find the split duration of the past red phase t_{red}
4. Compute the Delay per car in the link currently at the green phase. This is given by $D_{green} = \frac{delta_0}{f_{green}*(t_{red}+t_{green}+t_{ext})}$ where $delta_0 = \frac{Q_{green}*(t_{red}+t_{green})}{2}$.
5. Similarly compute the delay per car in the link currently at the red phase. This is computed by using the values of $delta_1$, $delta_2$ and $delta_3$, which are the areas representing delays in Figure 5. *i.e.* $D_{red} = \frac{delta_1+delta_2+delta_3}{f_{red}*(t_{red}+t_{green}+t_{ext})}$.
6. If $D_{green} \leq D_{red}$ then implement the phase change or else extend the phase by changing $t_{ext} = t_{ext}+$ *EXT-TIME* and go to step 1.

From the above control strategy it can be seen that an increase by t_{ext} decreases the delay value in the link which is at green phase. Since the value of $delta_0$ remains same for any value of t_{ext} because the queue has been dissolved and all further vehicles pass the intersection without any delay. However, the delay for the vehicles in the link which is at the red phase increases in proportion to t_{ext}. Hence an increase in t_{ext} will result in a

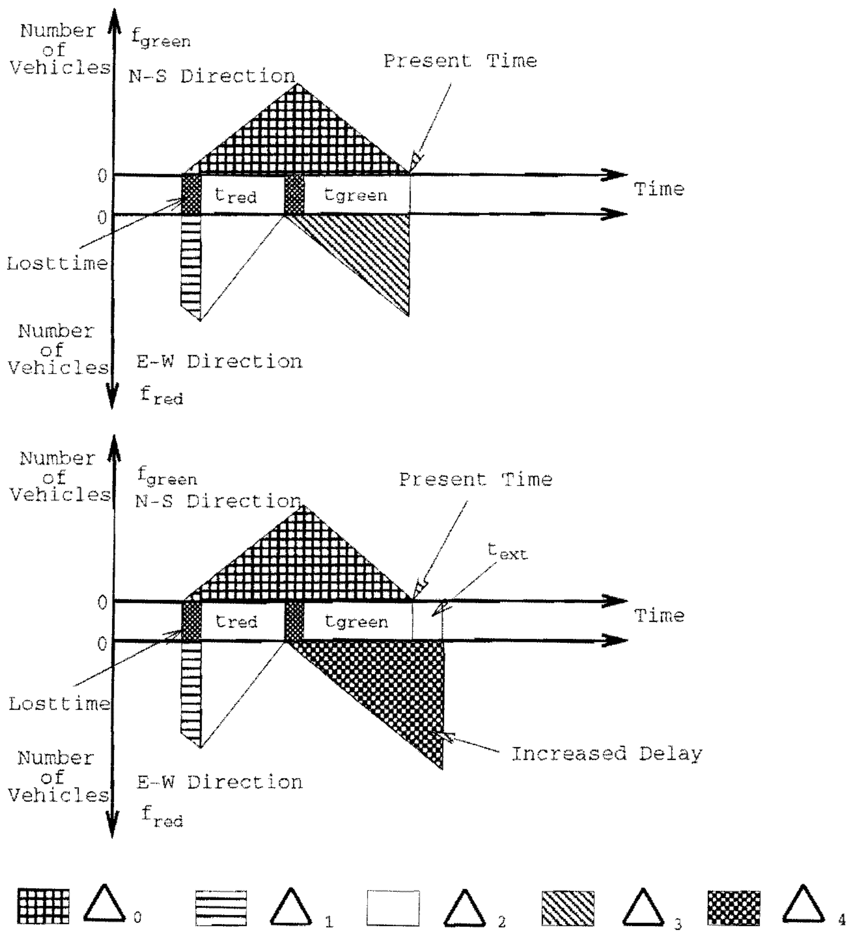


Figure 5: Computation of Extension for Split

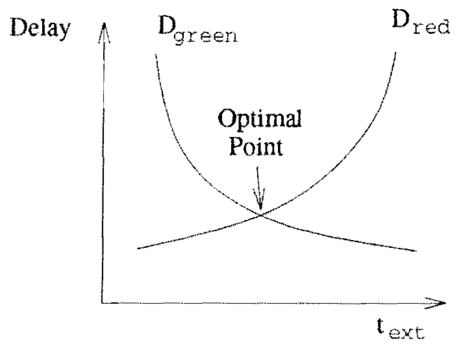


Figure 6: Optimal Point for D_{green} and D_{red}

decrease in D_{green} and an increase in D_{red}. An optimal value of these delays can be reached which is $D_{green} = D_{red}$, and this point is shown in Figure 6.

4.2.1 Queue Estimation

In the above strategies the queue length is estimated by the following strategy. The information from the entry detectors and the exit detectors are used in estimating this queue value. Moreover, the average flow rates and the percentage of vehicles exiting from each lane are measured at the exit detector. The CFP matrix is used in keeping track of the vehicles which stay in the link after they have been detected at the entry detector. Shown in Figure 7 is a CFP data structure which is used for the link pictured above reflects the presence of vehicles in the entire link. The CFP is divided into slots such that each slot reflects the presence of at most 3 vehicles. That is, the slot of the CFP is a compressed form of the link with the capacity of all the three lanes folded into one, which can be clearly seen from the figure.

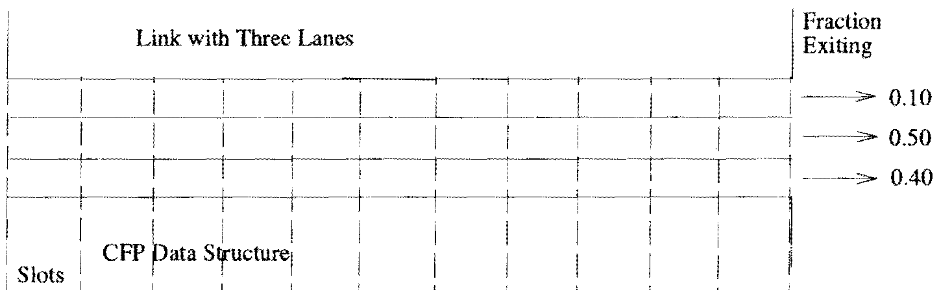
As time progresses the CFP data structure contents are shifted towards the direction of the intersection so that the movement of the vehicles is appropriately reflected. However, while shifting the count of the vehicles at the slot close to the intersection, the status of the signal light facing that approach is checked. If the status of the signal is red than the vehicles at the end of the CFP, which is close to the intersection, cannot exit the approach and hence have to form a queue. Thus if the status of the signal is red, the count of the vehicles which are at the end of the CFP, close to the intersection, is shifted and added into a variable called $QUEUE$ which holds all these vehicles which are guaranteed to have formed a queue.

To estimate the queue the following steps are performed, and this is shown in Figure 7.

1. Measure the average flow rate $flow$ at the entry detector.
2. Measure the average percentage $frac$ of vehicles which exit the lane for which we need to compute the queue value. This percentage value can be obtained from the exit detector.
3. Measure the value in the $QUEUE$ variable.
4. The value $frac * QUEUE$ gives the minimum number of cars which will be present in the queue, we call the $basequeue_1$.
5. Recursively perform the following:

   Step 1. Count the vehicles present in the CFP from the end close to the intersection upto the $basequeue_1$th slot.

   Step 2. Multiply this value with $frac$ to get $basequeue_2$ and perform the above step by counting the number of cars in the CFP between the $basequeue_1$th slot and $basequeue_2 + basequeue_1$th slot and so on until $basequeue_i$ is equal to $basequeue_{i+1}$. At this point the sum of the $basequeue_i$ values will give the estimated queue.



Arrival	10	9	8	7	6	5	4	3	2	1	0	QUEUE	Signal Status
1	1	0	0	1	2	2	0	1	1	2	3	0	GREEN
2	1	1	0	0	1	2	2	0	1	1	2	0	GREEN
2	2	1	1	0	0	1	2	2	0	1	1	2	RED
3	2	2	1	1	0	0	1	2	2	0	1	3	RED
0	3	2	2	1	1	0	0	1	2	2	0	4	RED
	0	3	2	2	1	1	0	0	1	2	2	4	RED

At this point of time estimate center queue.
QUEUE value = 4
Estimated basequeue_1 = 4*0.50 = 2

Sum of vehicles in CFP slot 0 to 1 (i.e. 2 slots) = 4
Estimated basequeue_2 = 4*0.50 = 2

Sum of vehicles in CFP slot 2 to 3 = 1
Estimated basequeue_3 = 1*0.50 = 0.5 Approx = 1

Sum of vehicles in CFP slot 4 = 0

Final Queue = basequeue_1 + basequeue_2 + basequeue_3 = 2 + 2 + 1 = 5

Figure 7: Queue Estimation Using CFP and Flow Rates

4.3 Performance Evaluation

To compare the performance of the traffic control methods described above we identified two important performance parameters which we explain below. These output parameters concisely describe the quality of the traffic control when compared with the performance results on other traffic systems under the same environment.

- **Mean number of stops:** This parameter accounts for the average number of stops a vehicle has to experience while traveling through the road network which is under control by the traffic control system being simulated. The higher the value of mean number of stops indicates that the vehicle stops too often at the intersections or is possibly lined up at queues at the intersection. The performance of the system is better with a lower number of stops. The formula to compute the mean number of stops is given below, where n is the total number of cars.

$$mean\text{-}number\text{-}of\text{-}stops = \frac{\sum_{i=1}^{n} number\text{-}of\text{-}stops\text{-}for\text{-}each\text{-}car}{n}$$

- **Average Delay per Intersection:** A vehicle comes to a stop at a intersection if the signal light for its direction is red and hence waits at the intersection until it changes green. This waiting time at the intersection is the delay experienced by the vehicle and should be minimized. The average delay is the sum of the delays of the vehicles at an intersection divided by the total number of vehicles which passed that intersection. This value of the average delay is the waiting time which a vehicle will experience in the intersection on an average. The formula to compute this value is given below, where k is the number of cars which passed the intersection.

$$average\text{-}delay = \frac{\sum_{i=1}^{k} delay\text{-}for\text{-}car\text{-}i}{k}$$

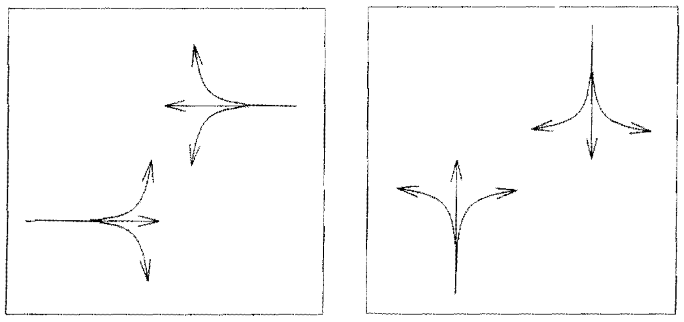


Figure 8: Two Phase Sequence

4.4 Analysis

After identifying these parameters the simulation was performed with the fixed time control and the on-line control. The following types of arrival rates were given as input parameters.

1. **Fixed Arrival Rate:** The arrival rates of the vehicles, which is specified as an input parameter, is generated at the prespecified fixed rate. For example, if it was specified that an average of 400 cars per hour would enter a lane from the exterior of the system, then the number of cars generated in a random fashion would be 400 cars per hour

2. **Varying Arrival Rate:** The arrival rate here, though specified, will vary within a range such that the overall average will correspond to the specified arrival rate. That is, the vehicle arrival rate will not be uniform but rather in surges of traffic. This method of generating the vehicle arrival corresponds to the traffic pattern in reality.

Using these arrival rates we performed the simulation with a two phase sequence which is shown in Figure 8.

Online Control Strategy

The online control strategies simulated were the two methods explained earlier, where one method used the queue estimation based on the CFP, which we called the CFP based, and the other used the flow rates for the queue estimation, which we called the flow based. The simulation program for the CFP based approach, estimated the queues in the arterial as explained in the previous section and for estimating the queues in the crossing approaches it used the flow based. The signal timing plans were computed at each controller in such a way that a progressive service would be provided for the traffic flow which is heavy. In our model the arterial stretch carried a heavier flow of traffic most of the time and hence was provided with a progressive platoon service.

Moreover, the online control method used by ARTC performed the following steps to control congestion, in addition to the steps described in the previous section to compute the split. The following steps are executed when any direction is to be provided with a green phase.

1. Obtain information from neighboring controllers about the number of vehicles present in the link between these intersections.
2. If the neighbor's load is lower than a $THRESHOLD$ level, which is a value of the allowable load, then it is assumed safe to provide a progression and the split is computed and executed.
3. If the neighbor's load exceeds the $THRESHOLD$ value, then a congestion control is enforced by bounding the split time such that the number of vehicles which can pass in the split is less than or equal to the $ALLOWANCE$ value. This $ALLOWANCE$ value is fixed by the neighbor who reports a low exit rate.

The performance of the traffic control methods described above are compared based on the performance indices (a) Average delay per vehicle (b) Mean number of stops, which have been explained in the report in Section 2.

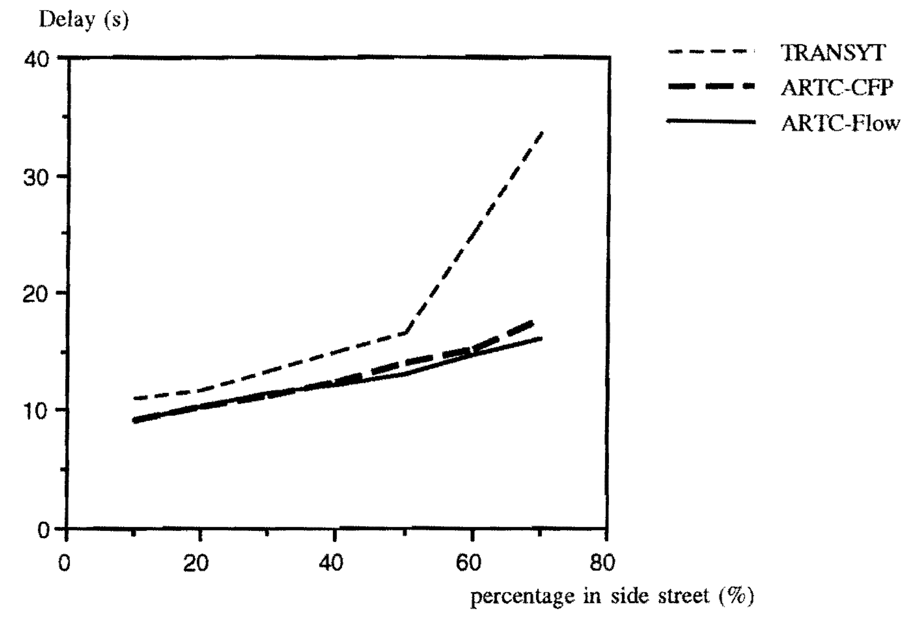
The simulation for both the systems was performed with different sets of traffic flow data for the approaches. The two main cases under which the system was simulated were, a fixed traffic flow rate and a varying traffic flow rate. The fixed traffic flow rate, generated vehicles on all approaches at a predefined rate on a random basis for intervehicle gaps. The arrival rate used in our simulation was 800 vehicles per hour per lane. The flow rates for the arterial stretch and the crossing streets were specified as a percentage of this fixed flow rate. The varying traffic flow rate, generated vehicles at all approaches such that the arterial stretch varied from 30% of the predefined flow rate value to 170% and the crossing approaches were varied from 5% of the predefined rate to 50%. The percentage of vehicles turning left from the arterial to the crossing approaches was set at 10% and 2% for each of the fixed and varying flow rate simulations, thus providing us with a total of four different cases. For each of these cases, the TRANSYT program was run to produce the signal timing plan, by specifying the arrival rates and probabilities at which the vehicles were expected to turn.

The performance result graphs of these parameters plotted against time are shown below. As can be seen from the plots of delay against time, the performance of ARTC is significantly better than TRANSYT with a lower delay value in the varying arrival rate. As the percentages of the crossing approach traffic increases, ARTC performs better with the CFP based and the flow based methods. This varying traffic can be observed in reality, where the traffic flow in the network fluctuates depending on time of day and other factors, and we believe that the ARTC system would perform much better and can exploit its use of on-line split computation in these situations. With a fixed arrival rate, we can see that the flow based ARTC and TRANSYT perform much better than the CFP based ARTC, which is because the CFP based ARTC uses the delay time as the performance index and hences provides shorter cycle times. Moreover, for a fixed arrival rate, a flow based approach will provide the necessary split which will be sufficient for this average fixed flow. Moreover, simulation results indicate that as the traffic flow increases, in the crossing approaches for the fixed flow cases, the performance of both ARTC methods and TRANSYT produce almost similar values of delay and stops.

It can also be seen that the number of stops increases with increasing flow rate values. This is due to the fact that the vehicles approaching the intersection in the East-West direction are provided a progression when the North-South flow rate is low, but as it is increased the East-West direction is not always provided a progression. Moreover, a shorter cycle length would also increase the number of stops and is the reason why the number of stops for TRANSYT increases as the North-South flow rate increases. This improved performance of ARTC indicates that an on-line control is better for traffic control with fluctuations in the arrival patterns and a flow based ARTC is better for a fixed arrival pattern.

4.5 Regional Controller

Changes in the flow pattern in the traffic network are detected by the regional controller by collecting data from all signal controllers in its region periodically. Although signal controllers cooperate with each other, it is not perfectly guaranteed that congestion does not occur. In handling the congestion problem, a more efficient way than to detect the congestion is to prevent the occurrence of congestion by maintaining a moderated traffic



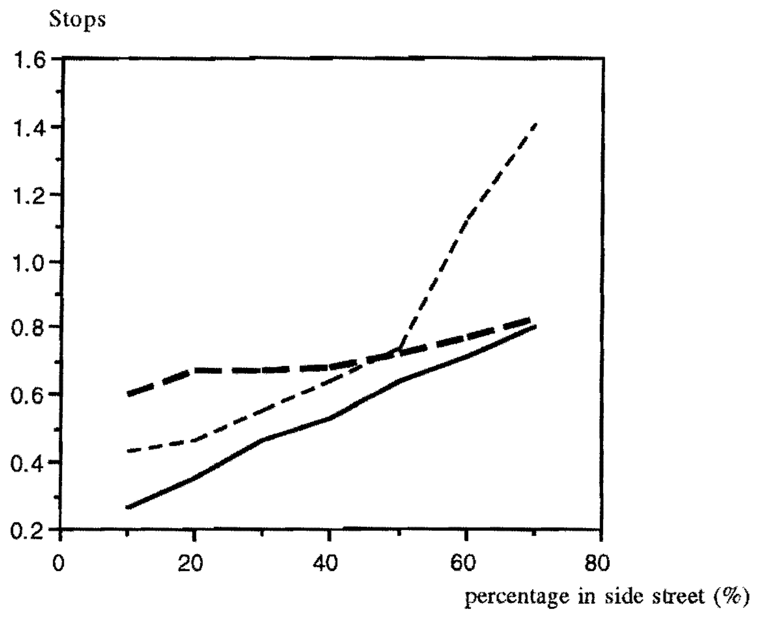


Figure 9: Varying Arrival Rate With 2% Left Turn From Arterial

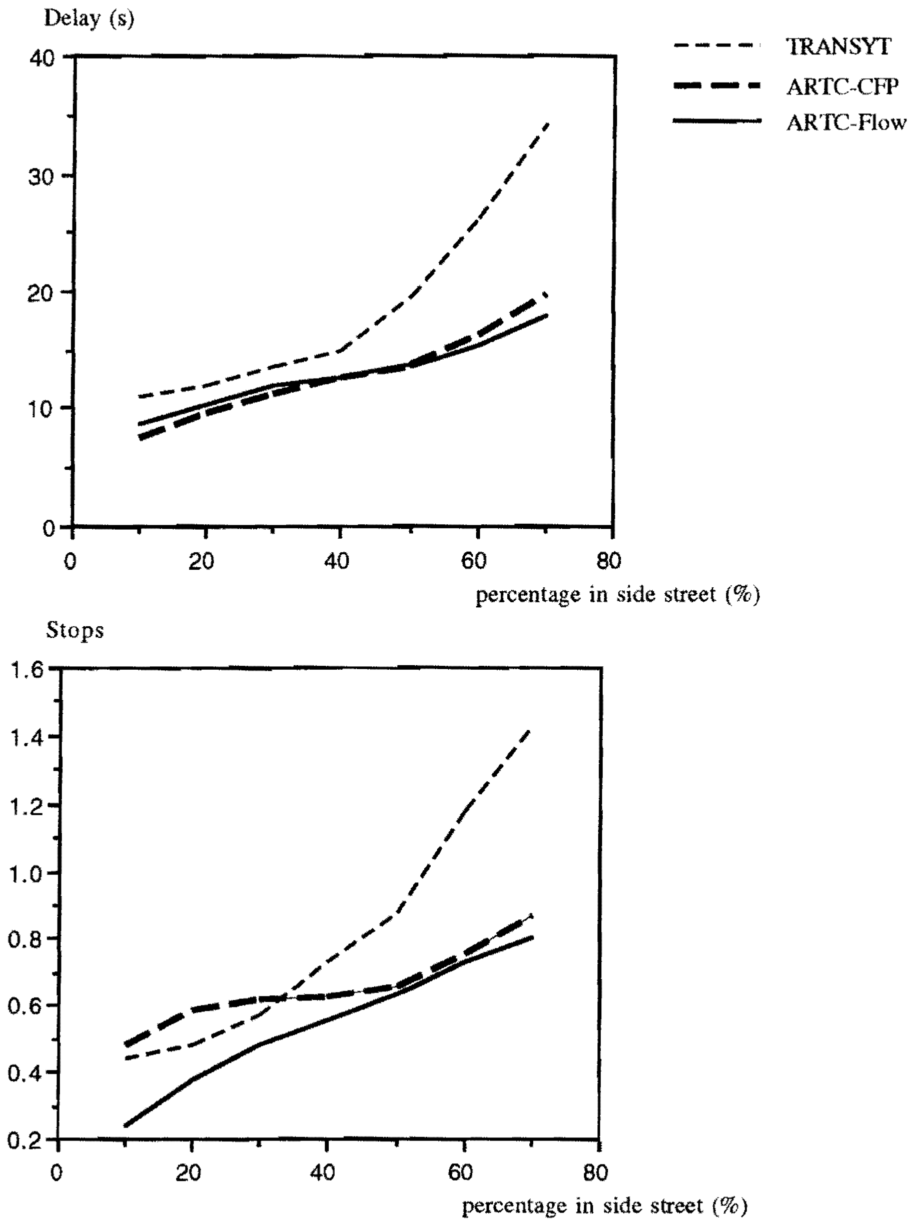


Figure 10: Varying Arrival Rate With 10% Left Turn From Arterial

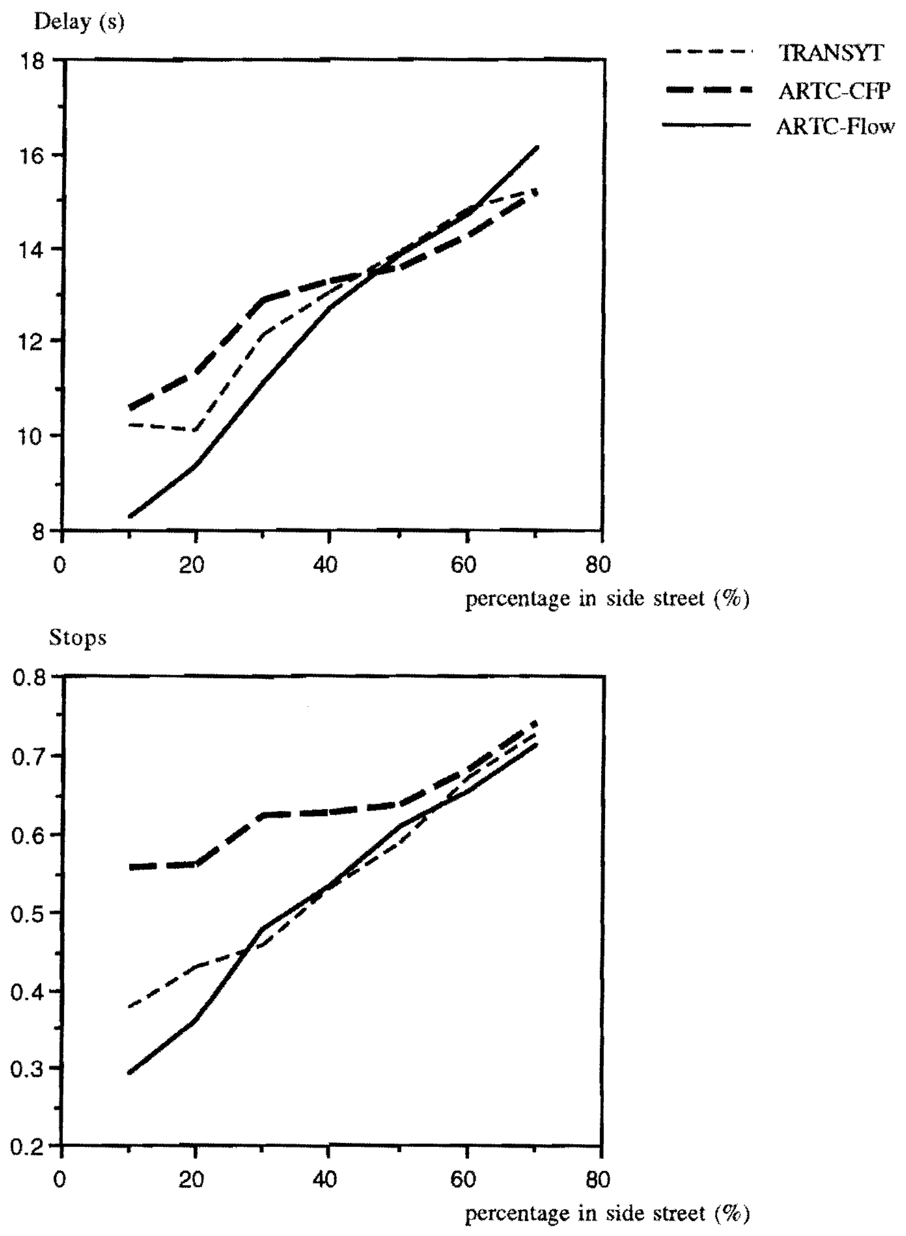


Figure 11: Fixed Arrival Rate With 2% Left Turn From Arterial

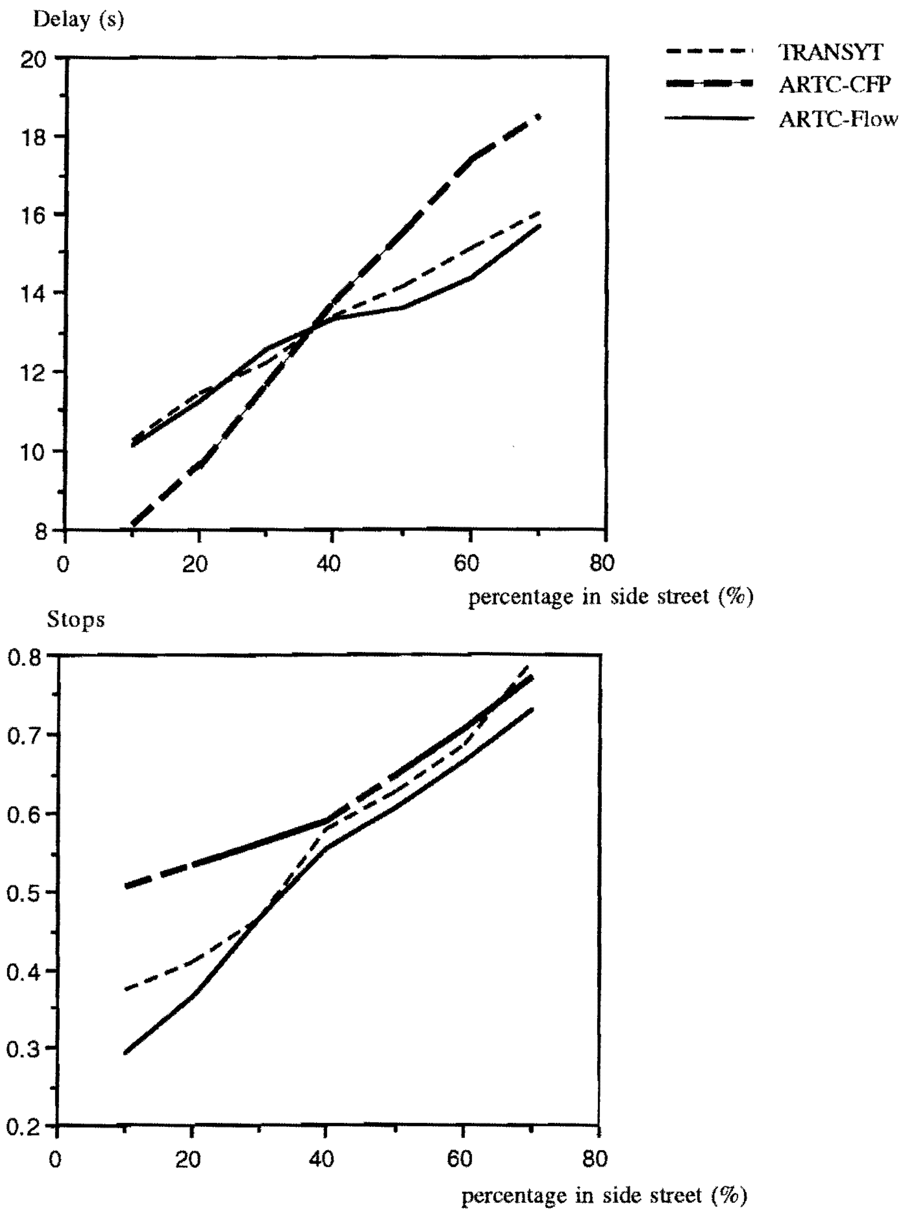


Figure 12: Fixed Arrival Rate With 10% Left Turn From Arterial

flow in the entire region. Thus, the regional controller periodically checks the traffic flow of the region, detects any traffic path which shows a critical increase, and then tries to reduce that flow. Since the overall traffic flow is maintained at a moderate level, the congestion may not be propagated to the neighboring approaches, even if an approach is temporarily congested due to some problems.

A. Traffic Flow Information

A signal controller maintains the traffic flow information for incoming approaches and this information is periodically updated and collected at the regional controller for every time period δt. To indicate degree of traffic flow of an approach during a time period δt, we use *vehicle volume ratio*, VVR, which indicates the average ratio of the number of vehicles compared to the maximum number of vehicles that an approach can carry.

At a given time T, the vehicle volume ratio of an approach is calculated as follows:

$$\text{Vehicle Volume Ratio(VVR)} = \frac{\sum_{t=T-\delta t}^{T} Number\ of\ Vehicles\ at\ time\ t}{\sum_{t=T-\delta t}^{T} Maximum\ Vehicle\ Capacity\ at\ time\ t}$$

Maximum vehicle capacity of an approach at a given time is determined by the length and width of the approach and the status factor which is determined according to the condition of the approach at that time. For this, a signal controller maintains the statistical data for the maximum vehicle capacity of the approaches according to the time of day. A signal controller can determine the number of vehicles in each of its approaches by using the CFP data. Thus, vehicle volume ratio can be easily calculated at each signal controller for the approaches.

We now use the VVR data to identify the heavy flow approaches by following the steps specified below. We define a range of VVR to be a critical zone, which implies that the traffic in the approach is dangerously high if its VVR value is in the critical zone. For safer estimation we represent each discrete range of the VVR in levels. For example if we divide the VVR into 10 levels, then level 0 denotes that VVR lies between 0 and 0.1 and level 1 denotes that VVR lies between 0.1 and 0.2 and so on. An increase in the calculated level indicates an increase in traffic volume.

B. Selecting Potentially Critical Approach

An approach is said to be in the potentially critical state if its current VVR level is found to be higher than the critical level. Upon receiving a information collection request from the regional controller, the signal controller sends a reply message which carries the current VVRs of approaches incident to it.

C. Computation

The regional controller periodically collects information about the traffic levels from the signal controllers in its region. Each time this information is collected, the regional controller forms the traffic information graph denoted by TIG. Each node in this graph represents a controller of an intersection, and a directed edge between two nodes represents the approach between two intersections. Each edge in the graph represents the level of flow of the corresponding approach.

The TIG reflects the current traffic status in the region. In brief, the TIG at the regional is used to:

1. detect a part or parts of a region which carries heavy traffic flow and then
2. smoothly reduce the incoming flow into these parts so that the traffic flow stays at a moderate level over the entire region.

If the level of flow is in the critical zone then the edge is called a *critical* edge. When a set of such edges forms a path, then it is identified. A path formed by the critical edges is called a *critical path*. When a critical path is shown on the TIG, the regional controller computes the duration of the green phase for the controllers in that path to reduce the flow along the path. The duration of the green phase of the controllers in that path reduces gradually from the head to tail. If multiple paths merge at an intersection, the influx into the downstream approach of the intersection must reflect the turning flows. In this way, the traffic flowing into the path is gradually reduced experiencing a moderately longer delay but avoiding imminent congestion.

The regional controller may participate in the offset timing planning for paths experiencing outstanding but not yet critical level of traffic flow. For example, if two outstanding paths cross each other, then the intersection at the crossing point may become a bottleneck. In this case, the offset timing plan computed for each path should depend on the split timing at the intersection. If the regional controller does not provide this service, then it is possible that signal controllers cannot provide a smooth flow to the traffic.

It may also be desirable to use the regional controller to calculate the offset timing plan if the path with outstanding flow is detected, hence the flow in the progressive path can be better and smoother.

One of the major functions of the information collection at the regional controller is provision of such information to the central control center. With our approach, we believe that the information can be effectively collected, displayed, and used for various purposes.

D. Detection of Critical Paths

The regional controller uses a list structure to represent the TIG. Each edge, with an unique identifier represented in the data structure used for the computation, is as follows:

Edge ID	Source	M	Destination	P1 set	P2 set

P1 set is the set of pointers to outgoing critical edges from the destination.

P2 set is the set of pointers to incoming critical edges from the source.

M represents the number of incoming critical edges to the source.

To detect the critical paths, the regional controller performs the following steps:

(1) All non-critical edges are removed.

(2) Select a critical edge.

Mark this edge as H_{edge}.

If no such edge is found, go to (4).

- (2.1) Visit the node incident on the tail of the edge.
- (2.2) Select one critical edge incident on the node.
- (2.3) Repeat (2.1) and (2.2) until a node with no critical incoming edge is visited.
- (2.4) (Now, the search continues in a reverse direction beginning at H_{edge}.)

  Visit node incident on the head of H_{edge}.
- (2.5) Select one critical edge incident on the node.

  If no such edge is found, go to (3).
- (2.6) Repeat (2.4) and (2.5) until a node with no critical outgoing edge is visited.
- (2.7) Store the node visited at the last step in the above process as the head of the path.

As an edge is visited, it is stored in a linked list, and then it is removed from the TIG.

(3) Go to step (2).

(The operational complexity of the above steps is $O(E)$, where E is the number of edges in TIG. Now, each linked list represents each critical path.)

(4) Visit the node with more than one incoming critical edge, N_1 and find the path that has N_1 as its head.

Merge the two lists.

(5) Search the lists for critical paths to find a node at a tail of a path whose identifier matches N_1.

(6) Repeat (4) and (5) until all critical paths are visited.

The computation described thus far takes $O(P)$ operations, where P is the number of critical paths in the graph. The detection of paths with outstanding but not yet critical traffic levels can be done in a similar manner.

5 Design of ARTC Signal Controllers

In this section, two different approaches in the ARTC signal controller design are discussed: In the first approach, the ARTC algorithm can be implemented by upgrading existing signal controllers such as TYPE179 or Eagle Programmable Actuated Controller Unit (EPAC300). In the another approach, the ARTC signal controller is designed using the state-of-the-art VLSI technology to satisfy the computational need for current and future computation.

5.1 Upgrading TYPE179 to Meet the ARTC Requirements

The TYPE179 is an 8-bit microprocessor (MC6809) based system which has a modular system architecture, and modules in the system are interconnected through a conventional backplane bus called the SS50 bus. Major modules of the TYPE179 include the *controller unit module, coprocessor and modem module, input/output expansion module, memory expansion module,* and other peripheral modules such as detection and isolation modules. The controller unit has full control over other modules in the system, and all data and control messages are transferred between the central microprocessor and peripheral devices through the backplane bus. Since the TYPE179 uses semiconductor technology developed in the early 1980s, it has a lower computing speed (1.8MHz) and small physical memory space (64KB) so that the complex memory management mechanism has to be used.

It is obvious that this mature signal controller has some limitations when used in ARTC. Even though we do not take the higher reliability into consideration, the computation and communication capabilities have to be increased to meet the ARTC needs. It should be noted that in the ARTC, a large amount of sensor information transformation, arithmetic computation, and inter-controller communication have to be timely performed. Software design of the signal controller will become very complex and unreliable if there is not enough peripheral supports such as the hardware timers in the system. Thus, enhancing the TYPE179 should be done from its computation core and control mechanism.

To make full use of the existing resources in the TYPE179, an efficient design by enchancing it has been performed and is explained below. In this design, we keep the original *controller unit* in the TYPE179 as a peripheral supporting board rather than the input/output controlling and handling the communication between controllers. Therefore, all the peripheral expansion modules including input/output, detector, isolation, protection module, etc., can be employed in the new system through the *controller unit* module. For the computation function, we use a newly designed module instead of the MC6809 based microprocessor system. This module can be constructed around a powerful microcontroller like MC68332 or other microcontrollers with very few chips but sufficient memory space. It is expected that it will provide sufficient computation power and convenient timing support. In the original design of TYPE179, there exists a coprocessor module for coordinating with the *controller unit* module in computing and communicating. However, we think the efficiency of this module will be very low in the ARTC use. Instead of the usage of the coprocessor module in the TYPE179 enhancement, we can partially employ the message transferring mechanism originally occupied by this module in the coordination between the new computation module and the *controller unit* module. Since the *controller unit* serves as an intelligent controller for the peripheral functions in the new design, a sharing memory and priority interrupt mechanism can efficiently support this coordination. Thus, the *controller unit* will collect the traffic data and inter-controller coordination message, send it to the computation module, and then, after the computation module makes the traffic decision, execute the traffic control functions.

Besides the computation module, a special communication module has to be developed for the enhancement of the TYPE179 controller. The original TYPE179 only supports one or two ports of long distance communication through the external modem, and the mechanism handling this communication is not very efficient since there is not the need of frequent message exchange in its original design. In the ARTC network, we have to keep at least five communication ports to support the message exchange with its four neighboring controllers in the normal situation and the regional coordination. Such a communication module will be installed in the SS50 bus and controlled by the *controller unit.*

The EPAC300, introduced by the state of Texas, is another choice for an upgrading approach. This signal controller consists of an I/O board and a processor board which is a dual-microprocessor based system. In the processor board, the MC68008, a 16-bit microprocessor, functions as the control core which performs the traffic control algorithms and controls the input/output board. The MC68121, a programmable microcontroller, manages the keyboard and LCD display. The I/O board in the EPAC300 is composed of some data latchers and voltage converters which provide the interface with the detectors and signal lights. This board is completely controlled by the processor board like that in the TYPE179. The unique feature in the EPAC300 is that all programming, including the timing setting and function chosing, can be done via its front panel with a simple menu operation. For ARTC use, we have to incorporate another new computation board to perform the ARTC algorithms and communication function, into the original signal controller, to avoid changing the original design of this controller since it has a comparatively powerful computation capability. This is because that the EPAC300 does not have a modular structure and is an object-oriented design.

5.2 Design of ARTC Signal Controller Using VMEbus

The new generation of ARTC signal controllers can have a single processor or multiprocessor architecture. A single processor architecture is easy to design, but its capacity is limited only to simple applications. In view of the rapid evolution of electronics technology, and the future demand on various applications such as the *Intelligent Vehicle-Highway Systems* (IVHS) functions, a multiprocessor architecture is a more suitable candidate for the ARTC signal controller design. In the multiprocessor architecture, an inter-modular information exchange protocol has to be defined. Then, new modules can be added to the system with little interference to existing modules.

In addition to its expandability, a multiprocessor architecture provides flexibility to the design of advanced sensor and signal light control subsystems, because (autonomous) intelligent I/O modules can coordinate with processor modules through the intermodular information exchange protocols. Without such protocols, it is very costly and difficult to modify existing systems for accommodating new I/O subsystems, because the complex low level I/O signals have to be directly processed by the computation processor. Thus, any change in the I/O technology requires major changes to the traffic control programs.

To support a multiprocessor architecture, an appropriate system backplane bus has to be selected. It is a time-consuming and very involved process to define a new bus standard for the next generation signal controllers. To avoid such a high design overhead, we chose a popular industrial bus standard, the VMEbus, which evolved from Motorola's VERSAbus, for the ARTC signal controller design. The VMEbus can efficiently support multiprocessor architecture, and it has been successfully used in many industrial applications. It is expected that the VMEbus should satisfy the performance requirements of ARTC signal controllers. However, if new features are needed for the ARTC signal controllers that cannot be effectively supported by the VMEbus defined signals, the large number of *user-defined* signal pins on the VMEbus can be easily defined for the new applications.

Useful features of the VMEbus include dynamic data width, a large address space (16MB or more), bus arbitration mechanisms, priority interrupts and system error interrupt. The VMEbus supports 8, 16 or 32-bit data width, and its address space ranges from 16MB or 1GB, depending on the number of VMEbus connectors used. Thus, I/O modules of different data widths can be supported easily. The large address space allows different modules to be assigned to different portions of the global address space without using a complex overlay memory structure as the one in the TYPE179.

The bus arbitration and interrupt mechanisms can be configured in different ways for various applications. Since most of the functions in the ARTC signal controllers have a fixed processing sequence or importance in their nature, they can be assigned with different priorities in bus accessing and interrupt processing. That is, a module with a higher priority is serviced before the one with a lower priority, i.e., the *daisy chain* structure in backplane bus design.

The information exchange protocol between the processors in the VMEbus system is a key design issue. Through this protocol, information can be efficiently exchanged between modules. The communication protocol can be a *shared memory* based protocol or a *mailbox* based protocol. In the shared memory based protocol, a global memory module is assigned for information exchange between modules. The shared memory architecture is more efficient but is less reliable, because a failed module can easily corrupt the shared memory area and cause a system crash.

To provide better system reliability and system expandability, mailbox based communication is recommended for the ARTC signal controller. Processors in a mailbox based system are loosely coupled. That is, information that has to be sent to one module has to be deposited in its mailbox, which can be a memory area or register area of that module. In this way, only modules that are configured to communicate with each other can exchange information. A module can reject the message sent by other modules unless authorized. Thus, the chance of the mailbox being corrupted by a failed module is much lower than the shared memory counterpart.

An ARTC signal controller can be organized into three classes of intelligent modules: Control Module, Input/Output Module and Communication Module. Each module is built on a separate board and with an independent processor. Fig. 13 shows the structure of an ARTC signal controller.

The control module is based on an advanced microcontroller MC68332 which integrates

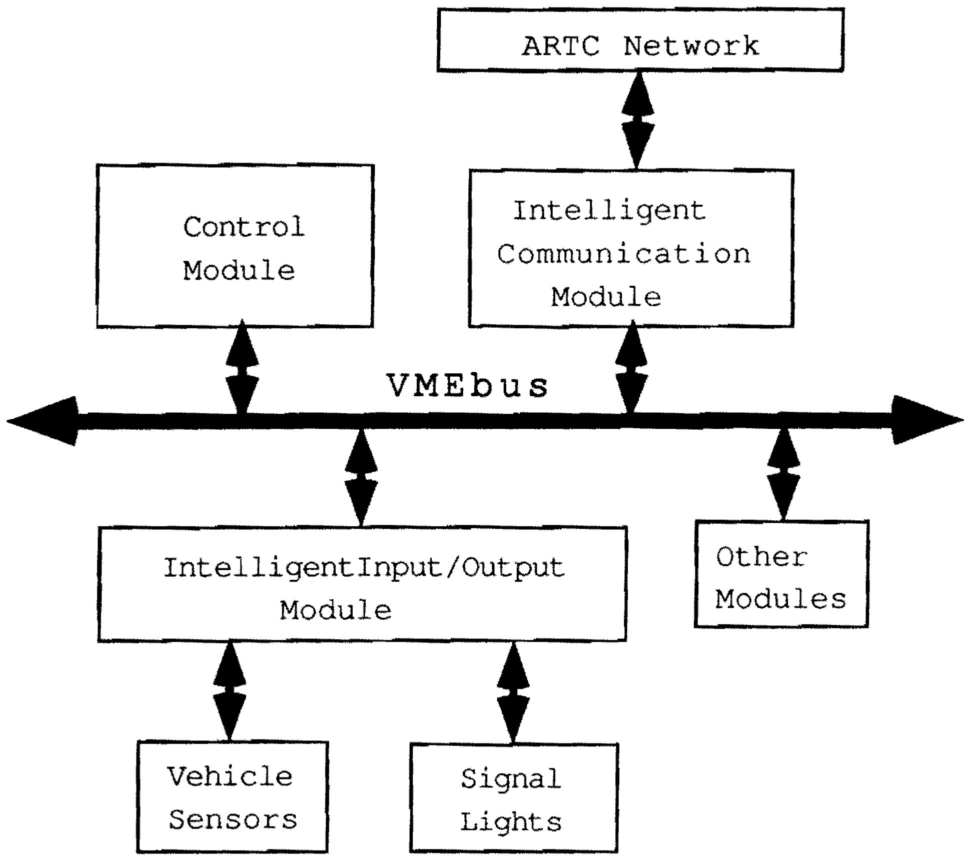


Figure 13: Block Diagram of ARTC Signal Controller

a 32-bit microprocessor and many peripheral functions circuits into one chip. The functions of MC68332 include high-performance data processing, time management (many timing channels), self-testing and peripheral interface, etc. It has nearly all the features needed for ARTC signal controller except for some external memory and supports the standard VMEbus interface. Thus, this chip is a good candidate for the ARTC computation and control module in both design approaches.

Moreover, the control module also needs some random logic circuits to implement the prioritized VMEbus access arbitration mechanism and match low level data transfer signals between the VMEbus and the microcontroller. Although the MC68332 has 16MB of address space, we plan to accommodate 1 MB of memory (ROM and Static RAM) on the control module. The memory is for storing timing decision information, and part of the memory is designated as mailbox which can be accessed by other modules. The memory chips can be interfaced to the MC68332 without using extra interface circuits. The basic block digram of the control module is given in Fig.14.

There are some basic differences in the control signals for data transferring in the MC68332 microcontroller and the signals defined on the VMEbus. Thus, control signals have to be made compatible with each other through a proper conversion circuit. On the VMEbus, the data width is dynamically determined by the combination of three special signals called the data strobe *DS0**, *DS1** and *LWORD**. In MC68332, however, the data width is determined by the value of the address line, *A0*, size indicating signals, *SIZE0* and *SIZE1*, and two data strobe acknowledgement signals, *DSACK0** and *DSACK1**. A design to convert these signals from one format to another, and vice versa, is shown in Fig. 15.

A main difference between our originally proposed architecture and the current design is that, in our current design, instead of using two extra redundant modules, only one back-up

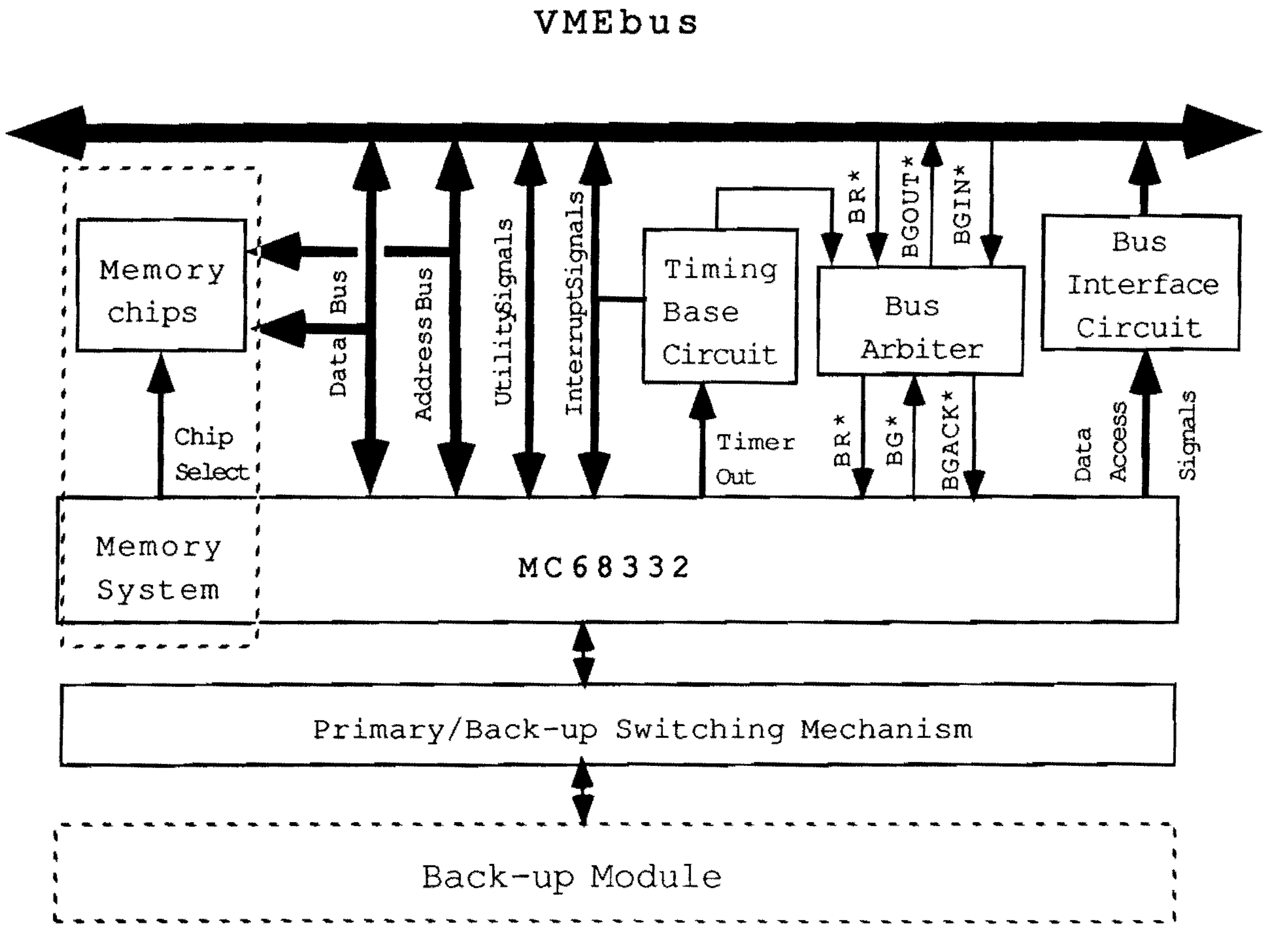


Figure 14: Block Diagram of Control Module

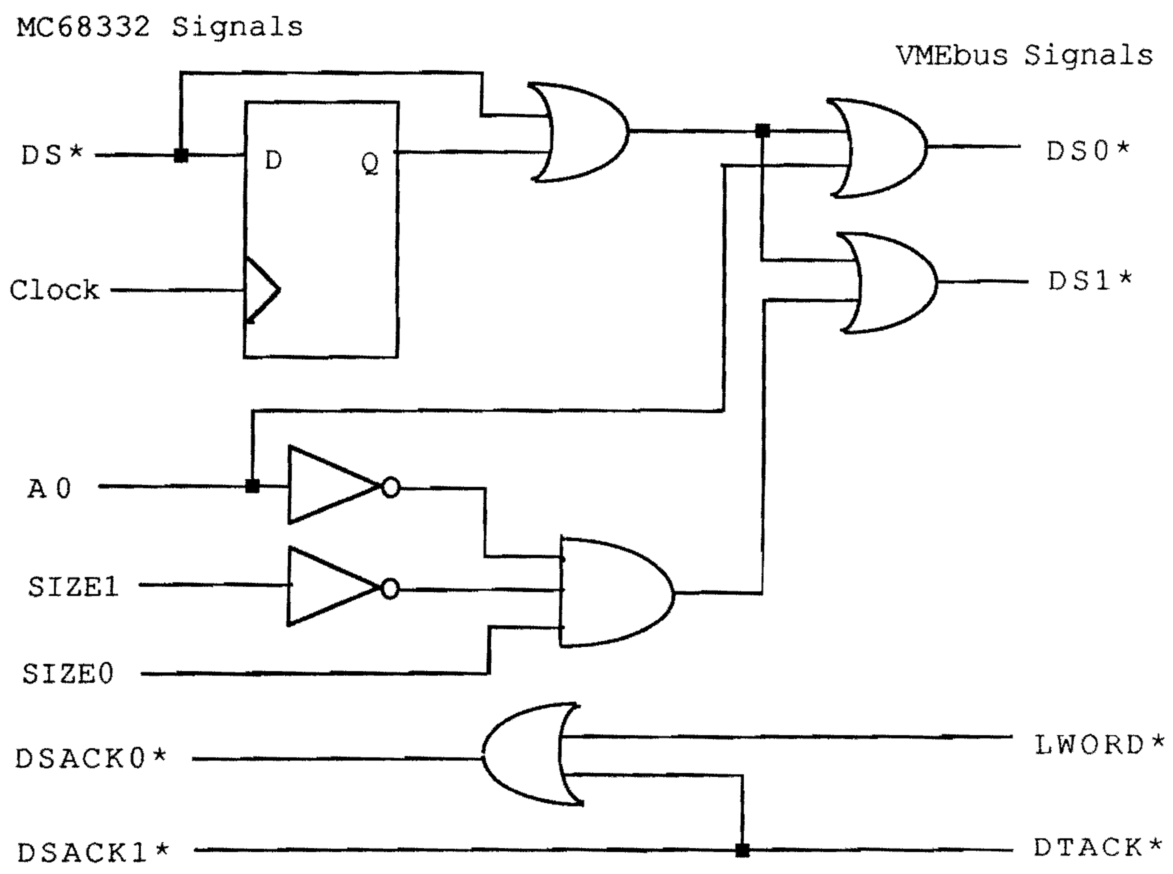


Figure 15: Conversion Between VMEbus Control Signals and MC68332 Control Signals

control module is needed because of the self-testing capability of the MC68332. With its self-testing capability, one control module can determine its correctness in one instruction without interacting with other modules. Both the primary and back-up copies are monitored by a watchdog monitor/arbitrator. When the primary module fails to reset the watchdog monitor, it is disabled by the monitor, and the back-up module prepares to resume the control functions of the primary module. In the mean time, the watchdog monitor must inform other modules to be aware of the switching process and enters the fail-safe mode until the back-up process is successfully completed. During the processor switching period, the primary and back-up processors have to test themselves for any failure. Fig. 16 illustrates the operation of this mechanism.

The input/output module in the ARTC signal controller is a (low-end) microcontroller-based independent module. Since the I/O modules do not perform such heavy computations as the computation module, one single chip microcontroller, which has on-chip memory, timers, and some I/O ports can meet the requirement. As an example, the MC68HC11 8-bit microcontroller is used to construct I/O modules. MC68HC11 is designed for single chip operation. It includes an 8-bit microprocessor, 8KB on-chip ROM, 512-byte on-chip EEPROM, 256-byte RAM, four 8-bit parallel I/O ports, and an 8-bit A/D converter. Typical control signals such as interrupt, clock, reset circuit and a watchdog timer are also integrated into this chip. Based on this modular structure, and the multiprocessor architecture, we can easily handle future expansion by adding more I/O modules, including advanced sensor based traffic information collection modules. The structure of this module is illustrated in Fig. 17.

In a typical intersection, we may have 36 sensors and 36 signal lights, each of which needs one I/O line. Moreover, 36 lines for signal light detectors and several more lines for

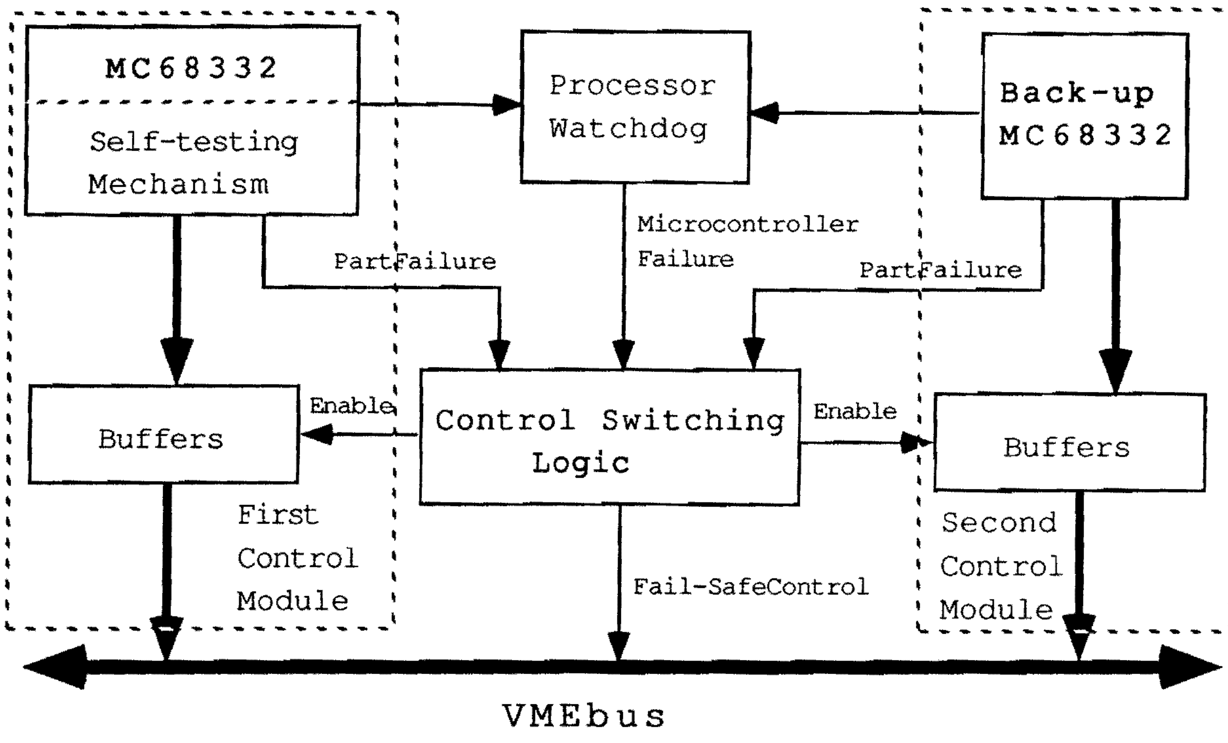


Figure 16: Design of the Back-up Mechanism in ARTC Controller

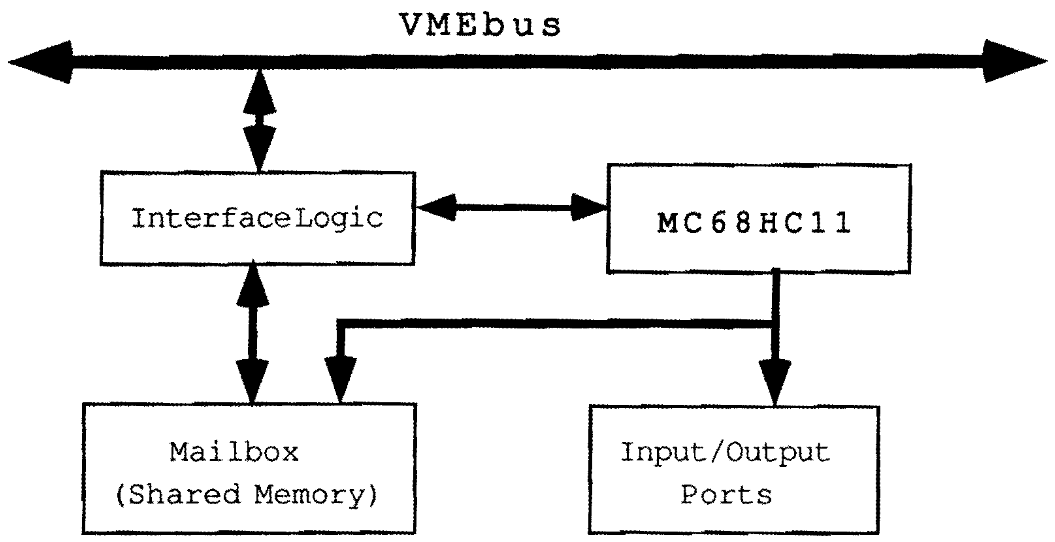


Figure 17: Block Diagram of the Input/Output Modules

I/O device reset are also necessary. Thus, a total of 108 I/O lines should be included in the input/output module. Although the MC68HC11 has only 32 I/O pins on the chip, more input/output ports can be expanded to accommodate the sensor inputs and signal light control outputs using the expansion technique provided by the MC68HC11 as shown in Fig. 18. Finally, a *mailbox*, which is essentially a memory block, is needed in the I/O module for inter-modular communication. This can be done easily by adding a static RAM chip to the microcontroller. The interface logic in this module performs the functions of VMEbus buffers, VMEbus address decoder, bus arbitration daisy-chain and shared memory protection.

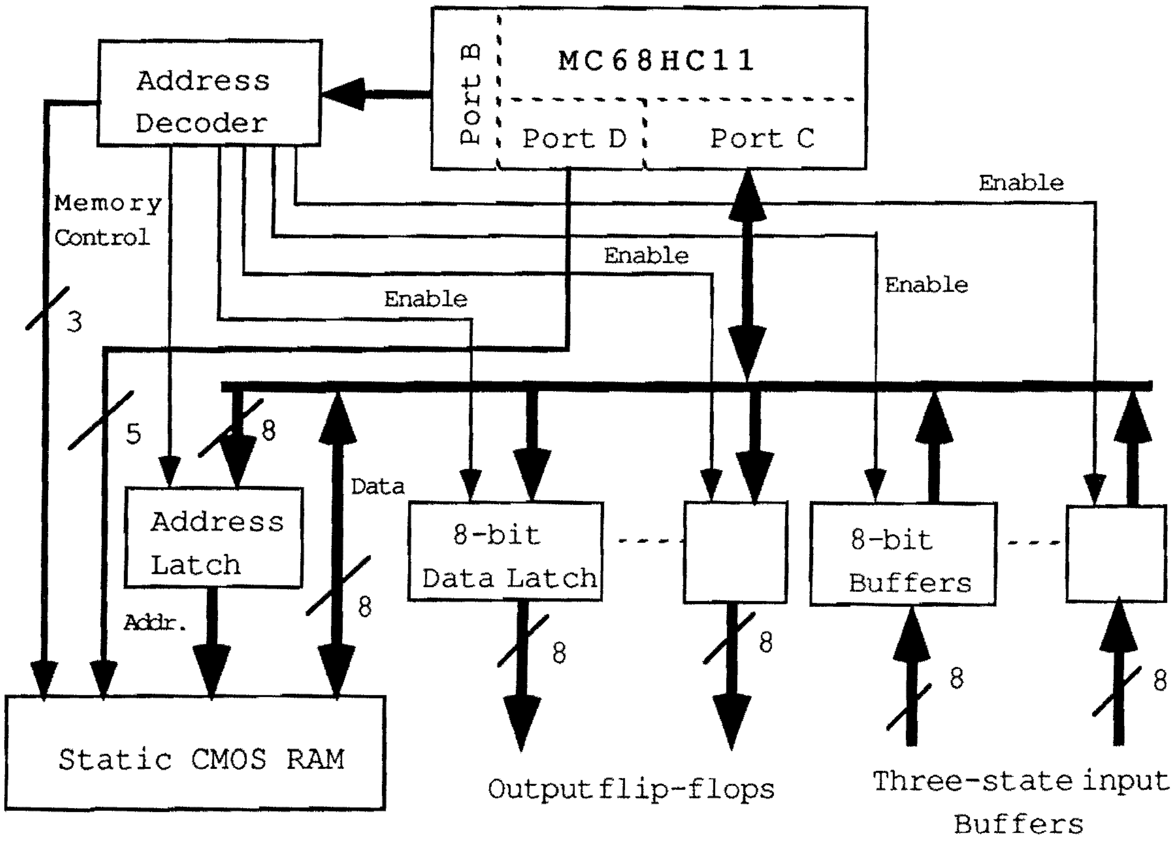


Figure 18: I/O Expansion in Input/Output Module

6 Summary and Future Work

The current simulation of the traffic network is not capable enough to reflect the effects of the ARTC communicating signal controllers. Hence an expansion of the traffic network will significantly improve the performance limit of the system. Moreover, simulation and comparison with other on-line control algorithms is essential to justify the performance of the ARTC system.

At present the ARTC signal controllers work in an isolated manner and hence the effect of communication is not visible. In the next stage of this project, we plan to develop the detailed algorithm specifically oriented to exploit the communication between the signal controllers and hence provide better traffic control. The regional controller will also be brought into effect to provide congestion and progression control commands. This regional controller will also be capable of detecting abnormal traffic pattern changes and take necessary action.

The current simulation system has been designed to simulate the traffic control mechanisms on one machine and is hence limited in terms of processing speed and memory in the machine. We plan to extend the simulation into a distributed level which will help significantly reduce the processing time and minimize the memory problems. The graphics display will be enhanced with better human interaction features which will ease the use of the system.

To meet the current and future need for handling heavy computation and communication activity in the IVHS environment, a multiprocessor architecture is proposed for the ARTC signal controller. This new generation of system architecture can be implemented using the VME backplane bus as the base, with the system level resource arbitration mechanism optimized for integrating the real-time traffic control and other information query and communication functions into the same system. To design the new generation signal controller, microprocessors used in the system must be highly reliable and have a high performance.

After an extensive search of microprocessors as the candidate of the ARTC signal controller, the microcontroller MC68332 was chosen for an in-depth evaluation. The MC68332 has comprehensive built-in functions, in addition to its regular data processing functions, for time management, exceptional events processing, and self-diagnosis. It is believed that a MC68332 based microcomputer system, with a less or similar number of microchips as that of the TYPE179, should be able to provide more computing power than the TYPE179 or other similar systems.

The reliability of a MC68332 based system would not be comprised for its improved performance, because first, the system failure rate would be reduced for a number of microchips; and second, the MC68332 processor has on-chip self-testing capability, which is not found in any other microprocessors. With its self-testing capability, the system maintenance problem can be significantly improved in both long run and short run. It is natural, and necessary, as the next step to implement a prototype of the signal controller to prove the feasibility and correctness of our system concepts. For these two kinds of approaches for implementing the ARTC signal controllers the future work is somewhat different. For the MC68332 approach, a 2X2 signal controllers prototype with all the designed functions will be built and its functionality and reliability will be tested. For the upgrading approach, we have to enhance the communication capabilities by developing some new expansion board and the computation capability of the TYPE179 by adding some special computing module.