A Continuous Time Markov Decision Process based On-Chip Buffer Allocation Methodology

S. Kallakuri,N. Thepayasuwan, A. Doboli
ECE Department
Stony Brook University
Stony Brook , NY 11794
contact email: elsanky@ece.sunysb.edu

E. A. Feinberg
AMS Department
Stony Brook University
Stony Brook , NY 11794
email: Eugene.Feinberg@stonybrook.edu

ABSTRACT

We have presented an optimal on-chip buffer allocation and buffer insertion methodology which uses stochastic models of the architecture. This methodology uses finite buffer space and presents a method to distribute this finite space in an optimal fashion. Such a methodology is useful in managing the scarce buffer resources available on chip as compared to network based data communication which can have large buffer space. The methodology also uses Continuous Time Markov Decision Processes CTMDPs. The modeling of this problem in terms of a CTMDP framework lead to a nonlinear formulation due to usage of bridges in the bus architecture. We present a methodology to split the problem into several smaller, though linear systems and we then solve these subsystems.

1. INTRODUCTION

We have applied CTMDP (Continuous Time Markov Decision Processes) to optimise the buffer space used in SoC architectures. This involves using continuous time queueing models for the architectures. The use of such continuous time stochastic models is necessary due to the continuous time nature of tasks when they are executed on the IP cores and the shift from RTL level design to system level design. A finite amount of buffer space has to be distributed among a set of processors talking to a bus and the continuous time modeling allows incorporating how long certain amounts of buffer space have to be allotted as well as how much of the space should be allotted to processor. The division of the finite buffer space by certain stochastic policies generated through the CTMDP based solutions [2] could lead to an optimal division of the buffer resources. We found this optimal distribution of buffer space different from simple division of the space depending on traffic ratios.

While attempting to solve the buffer sizing problem we encountered a problem when there were bridges between buses. A typical example of such an architecture has been shown in Figure 1 where buses b,f and g talk to each other apart from processors. The architectures in which two buses are connected by a bridge which is a typical example in the AMBA and CoreConnect systems. For such a scenario with bridges the model developed for CTMDPs was nonlinear and the system of quadratic equations were not solvable for a test example shown in Figure 1. We solved this problem by splitting the system to smaller subsystems and solving linear equations obtained from CTMDP based methods for the subsystems.

In stochastic control methods probabilities are defined over the states of the system as well as the possible actions that can be taken in these states [7]. In general policies implemented have actions which could depend on a history of states and actions [2]. We have attempted to use similar stochastic modeling for optimal buffer sizing as well as distribution of the finite buffer space. he novelty of our method is an optimal buffer sizing scheme which isn't available with the other on chip communication system design methods as well as the modeling of bridges and insertion of buffers in order to allow communication without loss through them. The use of CTMDPs in general and the Kswitching policy [2] in particular to get optimal buffer allocation as well as the buffer insertion for bridges to split the communication system into linear subsystems are the novel ideas presented in this paper. The design of communications architectures has been dealt with in [6] and [5],in which, the bus architectures have been designed but the insertion and sizing of buffers has not been attempted. Once the bus architecture is obtained it is possible to know where these buffers will be inserted into the bus architecture. Tuning of on chip resources in [4] has also encompassed the idea of managing buffer space among other on-chip resources. The buffer management is based on deadlines of tasks whereas we are concentrating on the optimal distribution of a finite amount of buffer space rather than analysing the buffer requirements to meet hard timing requirements or loss based QoS requirements. The experiments that we have conducted have observed the effect of varying constraints on total on-chip buffer size on this distribution methodology.

The effect of packet drops and communication faults on latency requirements and energy dissipation has been studied by [1]. These methods were developed for tile based SoC which have a dedicated FIFO channel between any two neighbouring IP cores that communicate. The use of continuous time modeling has been exploited in these methods and CTMDPs have been used for the optimisation of the

GLSVLSI'05, April 17–19, 2005, Chicago, Illinois, USA.

policies for transmitting packets. One of the methods for reducing buffer size based on data shaping was developed by Taylor et al [8]. This methodology called Orbit, exploits the varying information content in video frames and selects a few important frames which could be used to reconstruct the video stream rather than decoding the entire output of the video encoder thus allowing some reduction in buffer size.

The paper is organised in the following manner. The second section is a discussion of the modeling and the formulation of the CTMDP . The third section is an explanation of how we tackled bridges by inserting buffers. The fourth section is a set of experiments and discussion of the results followed by the conclusion in section five.

2. MATHEMATICAL MODELING OF THE COMMUNICATION SYSTEM

For convenience in comprehension of the following equations we present the definition of a CTMDP. As per [3] a continuous-time Markov Decision Process can be defined by a set $\{I,A,A(.),q,K,r\}$ where I is a finite state space;A is a finite action set;$A(i)$ is a set of actions available at state $i \in I$; $q(i,j,a)$ is a transition rate from state $i \in I$ to state $j \in J$, if an action $a \in A(i)$ is selected ,where $q(i,j,a) \geq 0$ for $i \neq j$ and $\sum_{j \in I} q(i,j,a) = 0$ for all $i \in I$, K =0,1.... is the number of criteria; $r_k(i,a)$ is a reward rate for criterion k =0,..,K, if an action a is selected in a state i.

The manner in which we have formulated the optimal buffer sizing problem in a CTMDP framework is as follows. The state of the system is given by a set of processors for every bus in the system. Thus an action would be to give a certain processor control over the bus. The rates $q(i,a)$ are the request rates of the processors and the rates $q(i,j,a)$ are the rates of a certain processor i giving up a bus a for another processor j. The rewards associated with each state action pair are the loss rates obtained from presimulation of the queueing model with a certain arbitration policy.

We are using LP based methods for solving the CTMDP and this involves setting up the following sets of equations. The equations can be split into 5 categories equality constraints, inequality constraints, cost function, lower bounds and upper bounds. The constraints in this set of equations physically relate to the finite buffer space available for a bus and the optimal distribution of this space as well as insertion of extra buffer space would be obtained from the solution of this LP. The solution would essentially be state action pair probabilities which could then be translated into meaningful physical quantities; in our case for this paper it would be buffer space. The rewards $r_o(i,a)$ are infact costs and are basically the average loss rates obtained from the simulation of the architecture with constant amount of buffer space allotted to every processor bus pair. The traffic rates $q(i,j,a)$ and $q(i,a)$ were obtained from the simulation too. The arrival rates used in the simulation were obtained from the core graph in [6] and the service rates were related to the bus bandwidth. The inequality constraints on the system are used to ensure the used buffer space is less than certain fixed quantity the coefficients $r_k(i,a)$ are the average queue lengths encountered in the preliminary simulation .

The different actions will be associated with different reward rates due to the different rates available to the Markov chain when a separate action is chosen . These rewards as well as the transitions rates with constraints obtained from the first simulation run will be given to a set of equations shown below [2]. The reward function is made up of the possible expected rewards that can be obtained by choosing an action as well as the rewards that are earned while in that particular state. The constraints are on the time spent by requests in the queues and the queue lengths that will occur while the buses serve the processors.

$$maximize \sum_{i \in I} \sum_{a \in A(i)} r_o(i,a) x_{i,a} \tag{1}$$

The rewards in this case are the loss rates obtained from initial simulation of the arbitration policy on the queueing model of the architecture. These quantities are actually costs hence we assign negative values to them and then try to maximise the total expected reward [3].

$$S.T \sum_{a \in A(j)} q(j,a) x_{j,a} - \sum_{i \in I} \sum_{a \in A(i)} q(i,j,a) x_{i,a} = 0, j \in I, \tag{2}$$

The set of LP equations has a set of equality constraints derived from the steady state equations of the Markov chains and a concept called uniformisation which is a method to reduce the continuous time Markov process to a discrete time Markov process which is equivalent to the continuous time process in terms of the rewards or reward rates associated with the states as explained in [3] [2].

$$\sum_{i \in I} \sum_{a \in A(i)} r_k(i,a) x_{i,a} \leq C_k, k = 1, ..., K, \tag{3}$$

There is also a set of inequality constraints as shown below, its coefficients are obtained from the simulations exploration loop and bounds on the space the LP can find a solution in.

$$\sum_{i \in I} \sum_{a \in A(i)} x_{i,a} = 1, \tag{4}$$

$$x_{i,a} \geq 0, i \in I, a \in A(i), \tag{5}$$

The output of the LP is a set of state action pair probabilities which are the long run probabilities of choosing a certain action given that the system is in a particular state.

The probabilities obtained from the LP can be converted to lengths of space using Kswitching policy [2] over which the system will perform a particular action while in a particular state. Thus the processors will get alloted the lengths of buffer space which depend upon the state action pair probabilities i.e which bus it is talking to. After fixing a certain ordering of the actions, the policy will cycle through the whole set actions.

$$sn_l(i,x) = \frac{\ln(1 - q(i,a(i,l)) x_{i,a(i,l)}}{\sum_{m=l}^{n(i,x)} q(i,a(i,m)) x_{i,a(i,m)})} \tag{6}$$

$$s_l(i,x) = -\frac{sn_l(i,x)}{q(i,a(i,l))} \tag{7}$$

$$\psi(i,t) = a(i,l), if A_x(i) \neq and S_{l-1}(i,x) \leq t < S_l(i,x)$$

$$a, \text{where a is an arbitrary element of} A(i) if A_x(i) = \phi$$

In our system this would mean that the buffer space will get divided into certain ratio in proportion with the probabilities which are given from the LP. The above equation signifies that between the epochs $S_{l-1}(i,x)$ and $S_l(i,x)$ the system will always choose action a(i,l) given it is in state i. The Kswitching strategy hence will have the same average

rewards as the randomised stochastic policy. The reason for using a Kswitching strategy is that LP may return a deterministic stochastic policy which isn't optimal and in case the LP can return only deterministic policies then it is an NP hard problem to find an optimal policy among the deterministic policies. In other words a stationary optimal policy may not exist for a certain problem as in our case to have a constant division of the buffer space is not the best policy. In such cases the task of finding the best sub optimal policy is an NP hard problem. In order to avoid solving this problem we use Kswitching policy which is a piecewise linear approach towards solving this problem and does give us the best sub optimal policy. To get more insight into this problem we would recommend reading [2].

3. BUFFER INSERTION AND SPLITTING OF COMMUNICATION SYSTEM

In Figure 1 the architecture has buses that are connected only to processors like bus a, as well as buses b,f and g which are connected to other buses too. Thus communication between processors 2,3 and 5 will involve insertion of buffers and will require the controller to take into account traffic from all three processors while making arbitration decisions for any of these three buses. One of the problems with designing such an arbiter is that it would require equations which would be quadratic in nature due to the interaction between two buses. In case the buses talk to each other through bridges the equality constraints and the cost function have quadratic terms. The number of quadratic terms depend on how many points in the bus topology are there in which buses are connected to each other and an equation may have more than one quadratic term. An attempt was made to solve the nonlinear equations by using the nonlinear solver from Matlab ver. 6.1. but we were not able to get solutions for them.

Example : In order to illustrate what kind of equations would exist in case the buses talk to buses we present below examples of the equations for the bus b in the architecture in Figure 1.

The equations for bus b in architecture 1 are:

Maximise

$$r_{2,b}x_{2,b} + r_{3,b}x_{3,b} \tag{8}$$

Equality Constraints

$$q_{2,b}x_{2,b} = q_{2,2,b}x_{2,b} + q_{3,2,b}x_{3,b} \tag{9}$$

$$q_{3,b}x_{3,b} = q_{3,3,b}x_{3,b} + q_{2,3,b}x_{2,b} \tag{10}$$

The equations for bus b in architecture 2 which have quadratic terms are:

Maximise

$$r_{2,b}x_{2,b} + r_{3,b}x_{3,b} + r_{5,f}x_{5,f}x_{5,b} + r_{5,f}x_{5,g}x_{5,b} \tag{11}$$

Equality Constraints

$$q_{2,b}x_{2,b} = q_{2,2,b}x_{2,b} + q_{3,2,b}x_{3,b} + q_{5,2,b}x_{5,f}x_{5,b} + q_{5,2,b}x_{5,g}x_{5,b} \tag{12}$$

$$q_{3,b}x_{3,b} = q_{3,3,b}x_{3,b} + q_{2,3,b}x_{2,b} + q_{5,3,b}x_{5,f}x_{5,b} + q_{5,3,b}x_{5,g}x_{5,b} \tag{13}$$

The solution we propose for this problem is to split the bus architecture into a set of linear systems which are separated from each other by buffers and solve a set of equations for each one of the independent linear modules. The fashion in which the system could be split is as shown in Figure 1and 2. In order to find the optima for the entire system all the equations shall be solved in one go and not sequentially for each subsystem. In Figure 2 for subsystem 1 initially the buses b,f and g were communicating but after the split bus b becomes a shared resource between buffer b1, buffer b2, and processors 2 and 3 isolating it from buses f and g thus enabling us to write a set of linear equations for it.

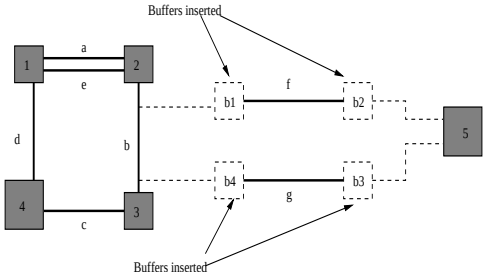

Figure 1: Bus Architecture splitting and buffer insertion

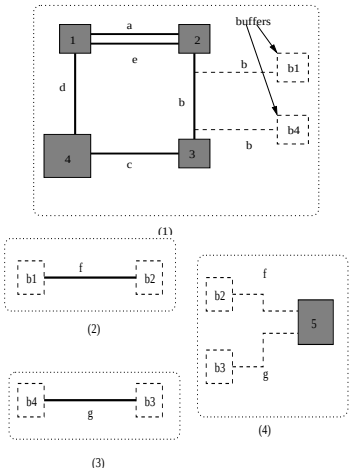

Figure 2: Subsystem 1

Maximise

$$r_{2,b}x_{2,b} + r_{3,b}x_{3,b} + r_{b1,b}x_{b1,b} + r_{b4,b}x_{b4,b} \tag{14}$$

Equality Constraints

$$q_{2,b}x_{2,b} = q_{2,2,b}x_{2,b} + q_{3,2,b}x_{3,b} + q_{b1,2,b}x_{b1,b} + q_{b4,2,b}x_{b4,b} \tag{15}$$

$$q_{3,b}x_{3,b} = q_{3,3,b}x_{3,b} + q_{2,3,b}x_{2,b} + q_{b1,3,b}x_{b1,b} + q_{b4,3,b}x_{b4,b} \tag{16}$$

$$q_{b1,b}x_{b1,b} = q_{b1,b1,b}x_{b1,b} + q_{2,b1,b}x_{2,b} + q_{3,b1,b}x_{3,b} + q_{b4,b1,b}x_{b4,b} \tag{17}$$

This set of equations is for bus b in figure 2. In this set of equations bus b is a shared resource between the two processors Proc. 2 and Proc. 3 as well as two buffers b1 and b4. The state of the system is actually given by a bus and processor pair but since we are writing equations only for bus b we use only the processor using bus b as the state of the system. On solving the CTMDP for this system of equations and translating the state action pair probabilities into buffer space requirements by using the Kswitching policy for a certain processor bus pair the system is resimulated with the new buffer lengths and the losses are compared.

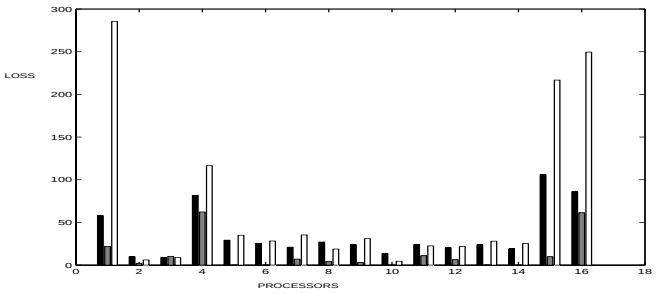

Figure 3: Loss rates before and after sizing

4. EXPERIMENTS

The experiments used a network processor as a test architecture for the buffer insertion and buffer space distribution. The network processors bus architecture was obtained from [6]. It provides opportunity to explore the different scenarios of buses talking to other buses as well as buses that talk only to processors and do not need explicit insertion of buffers. The bridges are essentially connections between buses and would need buffers to be inserted apart from all buses which would have buffers to handle the traffic coming in from the processors. We initially deviced a queueing model for the architecture and initialised with with the appropriate distributions for arrival rates obtained from [6]. The different policies were then simulated on the model to observe the loss that occurred from the resultant distribution of buffer space. The Kswitching policy provides thresholds for the buffer space based on the state action pair probabilities obtained from the LP. These thresholds are encoded into a buffer space controller as shown below.

```
Buffer Space Controller
 Initialise
    for:  every processor i
      for:  every bus j talking to procesor i
           .set threshold buffer space
      end
    end
 While(true)
   for:  every processor i
     for:   every bus j talking to procesor i
           .check for requests}
            if:  space is available
                .service the request
            end
           .update buffer space usage
      end
    end
    if: traffic pattern changes
        .break and
        .reinitialise thresholds
    end
  end
```

In Figure 3 we have plotted the loss rates at the processors before and after the buffer sizing as the first and second bars of Figure 3. We found that though the loss rates decrease drastically for some processors for example processor 16 in Figure 3 they increase slightly for some processors for example processor 1 in case of the the 160 units case as shown in Table 1. The third bar in Figure 3 are the loss rates for a timeout based policy, in which the processors request is not served if the data in the buffer timeout i.e. reaches a threshold time. The threshold time chosen was the average time spent by a request in a buffer. We repeated these experiments for 10 iterations and found that though the loss may increase for some processors if the total buffer space is

Table 1: Loss under varying total buffer size

PROCESSOR	Buf 160		Buf 320		Buf 640	
	pre	post	pre	post	pre	post
1	70	83	41	40	48	0
4	80	100	78	55	74	0
15	107	90	99	12	88	0
16	96	82	84	0	93	0

a very tight bound on the LP, the overall loss of the system decreases by atleast 20% as compared to the constant buffer sizing policy and 50% for the timeout policy. This was true for the case with 160 buffer units where the bound was tight and the loss increased in some processors after the redistribution but the overall loss reduced ,which was our goal. We feel the difference before and after resizing could be improved with better profiling and weighing of the loss at processors i.e. allowing some losses to be more important than the others.

In Table 1 we present the variation in the loss rates before and after sizing the buffers. We have presented the results only for a few processors which show significant variation but a similar trend was observed for the rest of the processors. We observed that some processors loss rates may increase when the buffer space is very limited as in the 160 units case and the redistribution does not provide much improvement as discussed in the previous paragraph. We increased the total buffer space from 160 to 320 and 640. The loss rates after resizing decreased with the increase in buffer space and fell to zero for the total buffer space of 640 units.

5. CONCLUSION

We have presented a methodology to efficiently distribute buffer space by using CTMDPs and stochastic models of the architecture. The use of CTMDP based methods gives us the optimal redistribution of the finite amount of buffer space so that loss is minimised, as seen in the experiments. The use of buffers for bridges can lead to efficient communication between two buses and buses can talk through them with reduced or no loss to other buses used by a different set of processors.

6. REFERENCES

[1] T. Dumitras, S. Kerner, and R. Marculescu. Towards on-chip fault-tolerant communication. *Proc. of ASPDAC)*, pages 225–232, 2003.

[2] E. Feinberg. Optimal control of average reward constrained continuous time finite markov decision processes. *Proceedings of the IEEE Conference on Decision and Control*, pages 3805–3810, 2002.

[3] E. Feinberg and A. Shwartz. *Handbook of Markov Decision Processes methods and applications*. Kluwer, 2002.

[4] K. Lahiri, A. Raghunathan, and S. Dey. Efficient exploration of the soc communication architecture design space. *Proc. of ICCAD*, pages 424–430, 2000.

[5] K. Lahiri, A. Raghunathan, and G. Lakshminarayana. Lotterybus: A new high-performance communication architecture for systems-on-chip designs. *Proc. of ICCAD*, pages 424–430, 2000.

[6] A. N.Thepayasuwan, V.Damle. Bus architecture synthesis for hardware-software co-design of deep submicron systems on chip. *Proc. of ICCD*, pages 126–133, 2003.

[7] Q.Qiu, Q.Wu, and M.Pedram. Dynamic power management of complex systems using generalised stochastic petri nets. *Proc. of the 37th Design Automation Conference*, pages 108–119, 2000.

[8] C. Taylor and S. Dey. Orbit: An adaptive data shaping technique for robust wireless video clip communication. *Proc. of Asilomar Conference on Signals,Systems and Computers*, pages 3081–3085, 2003.