On the performance of a two-server queueing network under stress

G. Hazari, M. P. Desai

May 28, 2007

Abstract

We consider the optimal load assignment problem in a two-server queueing system operating under stress, where the servers have deterministic service times. In this model, the two-server queueing network with finite queueing is servicing a sequence of i.i.d. requests characterized by probabilities of accessing the two servers. We study the problem of determining the probability distribution of the requests for which the system throughput is maximized. We develop a Markov chain model which models the behaviour of this system exactly, and show the following:

1. For balanced servers, the optimal distribution of requests is balanced, *irrespective* of the amount of queueing available at each of the two servers.
2. When the servers are not balanced, the optimal distribution of requests depends on the amounts of queueing at the two servers.

We contrast the optimal distribution under this model with standard memory-less approximations of the same system and demonstrate that the first result holds only on this model.

1 Introduction

In this paper, we study the queueing network shown in Fig. 1 (we will refer to this network as a *fork*). This network consists of two servers (each with a server queue) having deterministic service times which are serving a single

input queue with head-of-line blocking. The following model describes the behaviour of the system:

- Each job in the queue is destined for one of the servers and each job's destination is independent of all other jobs.
- We assume that the fork is under stress i.e. there is always some job waiting in the input queue.
- When a server becomes free, it picks up the job at the head of its server queue. A new job can enter the server queue at the same instant of time. The server remains busy for a fixed amount of time (the service time) after it picks up a job.
- At each instant in time jobs can move out from the input queue to the server queues as long as there is room in the respective server queues.
- We assume that there is head-of-line blocking in the input queue; i.e. if the job at the head of the queue does not have place in the destination server queue, then it blocks subsequent jobs from moving out.
- At start-up time, we assume that both server queues are empty and both servers are idle and ready to accept jobs.

As an example, this model can be used to describe the memory sub-system in a VLSI system. The servers correspond to memories, and the input queue represents the users of the memory sub-system. The under stress model is applicable because the users are typically much faster than the memory sub-system [10] [12]. However, this queueing structure appears in a large class of electronic and computing systems as well as industrial operations which are operating under high loads.

Let us label the servers as S_1 and S_2 and their service times (defined as the time required to finish a job) as d_1 and d_2 respectively. The depths of the queues in front of the respective servers are m_1 and m_2 respectively. Let x be the probability that a job is destined for S_1 (thus, $1-x$ is the probability that a job is destined for S_2). We define the throughput T of the fork to be the steady state rate at which jobs cross the cut C. Our objective is to determine the value of x that maximizes the throughput of the fork.

The first queueing networks to be analyzed (by Erlang and Jackson) assumed Poisson arrivals, exponential server service times, infinite queue depths

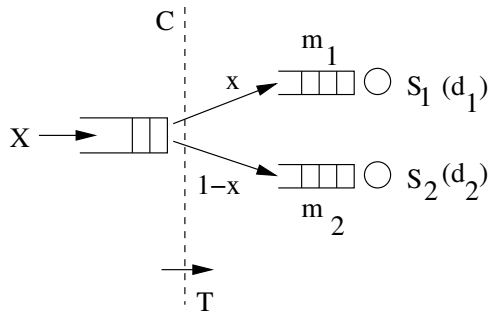

Figure 1: The Queueing Fork

and i.i.d. job distributions [1] [2] (systems with finite queue depths were assumed to drop jobs when the queues got filled, for the purpose of analysis). This combination of attributes makes the system completely memory-less. Most of the research on queueing networks with discrete arrival processes and service time distributions have modeled the system using Bernoulli and Geometric random variables [3], which also have the memory-less property.

On the other hand, the queueing fork that we analyze has servers with deterministic service times, finite queue depths with head-of-line blocking and an i.i.d. arrival process that stresses the system. (In this paper, we will also compare the results on our model with standard memory-less models for the same queueing network, for representative cases.)

Poisson queueing networks with blocking have been studied in [4]. There have been some studies of deterministic queues with respect to packet transmission over communication networks [5] [7] [6] [8] and ATM networks [9] [3]. The queueing structure we have studied has some similarities to the structures in [7] and [6].

To analyze this system, we will use a discrete parameter Markov chain model. We first present an analysis of the balanced server case (where server speeds are equal) (in Section 2), and then the case where the servers are not balanced (in Section 3). We are able to solve the Markov chains analytically for the balanced server case. For the unbalanced server case, we use an automated procedure to generate and solve these Markov chains and obtain the throughput for a given workload assignment. In these cases, we can numerically determine the optimal workload (up to a specified level of precision). (We have extended this procedure to the memory-less models

studied as well.)

In this paper, we show the following:

- When servers are balanced (but queue depths are arbitrary), distributing the workload equally will maximize throughput. This lack of dependence on queue depths in the balanced server case is surprising.
- When server speeds are not equal, the optimal distribution depends on the queue depths. In particular the optimal distribution under the assumption of finite queueing will in general be different from that under the assumption of infinite queueing, however we observe (over a large number of cases) that the faster server will get a larger share of the load irrespective of the queue depths.

This technique can, in principle, be applied to forks with a larger number of servers. However the number of states in the Markov chain increases exponentially with the number of servers, and solving the Markov chains becomes intractable on general purpose computers.

2 The Balanced Server Case

In this section, we will first demonstrate our analysis procedure on a fork with servers having identical service times. Without loss of generality, we may assume that the service time is equal to 1 time unit (due to the head-of-line blocking and the fact that the servers are idle at start-up time, the instants of time at which there is movement in the system will be multiples of the common service time). We will refer to the unit time instants as *clocks*. At every clock, jobs move out of the input queue until the job at the head of the queue is destined for a server whose queue is full. Thus, the state of the system at that clock (after the jobs have moved out of the input queue) will be such that at least one of the server queues is full, and the state of the input queue will be such that the job at the head of the queue will be destined for a server whose queue is full.

We define the state of the system as the ordered pair of lengths of the queues for the two servers (counting the jobs being served and the job waiting at the head of the input queue). Because of the stress assumption, we need to consider only those ordered pairs for which at least one of the queues is saturated. We can hence list the state space as follows:

$$S_{DF2} = \{(\alpha, \beta) : \alpha = m_1+2, \beta \leq m_2+1 \quad or \quad \beta = m_2+2, \alpha \leq m_1+1\} \quad (1)$$

We label and order the states as follows:

$b_1 = (m_1 + 2, 0)$
$b_2 = (m_1 + 2, 1)$
$b_3 = (m_1 + 2, 2)$
$\vdots \qquad \vdots$
$b_{m_2+2} = (m_1 + 2, m_2 + 1)$
$b_{m_2+3} = (m_1 + 1, m_2 + 2)$
$b_{m_2+4} = (m_1, m_2 + 2)$
$\vdots \qquad \vdots$
$b_{m_1+m_2+4} = (0, m_2 + 2)$

To obtain the transition probabilities suppose that the present state is $b = (\alpha, \beta)$. By the next clock, one additional slot will become free in each of the server queues and the queues would have $Max(\alpha-1, 0)$ and $Max(\beta-1, 0)$ entries in them; at the next clock new jobs would be admitted into the queues until a state in the state space is reached. Let us take up $b_1 = (m_1 + 2, 0)$ as an example: by the next clock, the queue lengths will be $m_1 + 1$ and 0 respectively. Depending on the entries in the input queue, the the possible next states (with their probabilities) can be determined in the following manner: from b_1 we can remain in b_1 if the next entry in the input queue (after the one waiting at the head) is also headed for the first server; we can move from b_1 to b_2 if the next entry is targeted at server 2, and the next-to-next entry is targeted at server 1; and so on.

Let us list out the transition probabilities for state b_1:

$b_1 : x$
$b_2 : xy$
$b_3 : xy^2$
$b_4 : xy^3$
$\vdots$
$b_{m_2+2} : xy^{m_2+1}$
$b_{m_2+3} : y^{m_2+2}$

All the remaining states are unreachable from b_1 in a single transition.

Hence, the transition probability matrix for arbitrary m_1 and m_2 is:

$$\begin{pmatrix}
x & xy & xy^2 & \dots & y^{m_2+2} & 0 & \dots & 0 \\
x & xy & xy^2 & \dots & y^{m_2+2} & 0 & \dots & 0 \\
0 & x & xy & \dots & y^{m_2} & 0 & \dots & 0 \\
 & & \vdots & & & \vdots & & \\
0 & \dots & x & xy & y^2 & 0 & \dots & 0 \\
0 & \dots & 0 & x^2 & xy & y & \dots & 0 \\
 & & \vdots & & & \vdots & & \\
0 & \dots & 0 & x^{m_1+1} & \dots & xy & y & 0 \\
0 & \dots & 0 & x^{m_1+2} & \dots & x^2y & xy & y \\
0 & \dots & 0 & x^{m_1+2} & \dots & x^2y & xy & y
\end{pmatrix} \tag{2}$$

Then, the stationary probability vector π corresponding to the transition matrix in Eq. (2) is given by

$$\pi(b_i) = \frac{y}{x}\pi(b_{i-1}) \text{ for } i = 2, 3, \dots \tag{3}$$

That is,

$$\pi(b_i) = \left(\frac{y}{x}\right)^{i-1} \pi(b_1) \tag{4}$$

Let

$$\gamma \;=\; \frac{y}{x} \tag{5}$$

Now, two jobs are completed every clock except when in states b_1 and $b_{m_1+m_2+4}$, in which only one job is completed every clock. Hence, the expected throughput R is given by:

$$\begin{aligned}
R &= \pi(b_1) + \pi(b_{m_1+m_2+4}) + 2\sum_{j=2}^{m_1+m_2+3} \pi(b_j) && (6)\\
&= 2 - (\pi(b_1) + \pi(b_{m_1+m_2+4})) && (7)\\
&= 2 - \frac{1+\gamma^{m_1+m_2+3}}{1+\gamma+\gamma^2+\ldots+\gamma^{m_1+m_2+3}} && (8)
\end{aligned}$$

Examining the derivative of this function, we find that the maximum is attained at $\gamma = 1$.

This distribution ($\gamma = 1$) maximizes expected throughput independent of the queue depths. Note, the distribution with $\gamma = 1$ equally balances the workload across the two servers. Thus, we have the following result

(m_1, m_2)	$Model - C$	$Model - G$	$Model - Q$
$(1, 2)$	0.514	0.507	0.505
$(1, 4)$	0.514	0.506	0.510
$(1, 8)$	0.508	0.503	0.513

Table 1: Optimal Workload for Balanced Servers under Memory-less Models

Theorem 1 *When the servers in the deterministic fork are balanced (have equal service times), the optimal load distribution (for an i.i.d. stream of jobs) is also balanced, irrespective of the queue depths in front of the two servers.*

We have also studied the fork under the following behaviour models (which have the memory-less property) [11]:

- the servers have exponential service times and the fork is under stress (labeled as *Model-C*)
- the servers have geometric service times and the fork is under stress (labeled as *Model-G*)
- the servers have exponential service times and the job arrival process is a Poisson process: we determine the workload distribution that will support the maximum job arrival rate (labeled as *Model-Q*).

We have used the automated procedure mentioned in the introduction to obtain the optimal distribution for *Model-C* and *Model-G*. For *Model-Q*, we have derived an expression for the throughput [11] and then determined the point of maximum numerically.

In Table 1, we show the optimal distribution (up to a precision of three decimal places) for the following configurations of the fork with two servers: both servers have identical service time distributions with mean 1 time unit, the queue in front of the first server has depth 1 and the queue in front of the second server has depths $2, 4, 8$ (for *Model-G* we keep the mean of the service time process at 2 clocks because a geometric process with mean 1 reduces to a deterministic process).

As we can see, Theorem 1 does not hold for memory-less models of the fork. In the next section, we take up the case where server speeds are not equal.

3 The Unbalanced Server Case

Let us next consider a fork with servers having arbitrary service times. Without loss of generality, we can select the clock to correspond to the highest common factor of the service times of the two servers (due to the head-of-line blocking and the fact that the servers are idle at start-up time, there will be movement in the system only at time instants which are multiples of this highest common factor).

Let the service times be in the ratio $d_1 : d_2$ (where d_1 and d_2 are mutually prime integers). Now, the state of the system becomes the ordered set of lengths of the queues (counting the jobs being served and the job at the head of the input queue) and the number of pending clocks for the job being served, on each of the servers:

$$\begin{aligned} S_{DF2} &= \{(\alpha, \beta, z_\alpha, z_\beta) : \alpha = m_1 + 2, 1 \leq \beta \leq m_2 + 1 \quad or \\ &\qquad \beta = m_2 + 2, 1 \leq \alpha \leq m_1 + 1, z_\alpha = 1, 2, \ldots d_1, z_\beta = 1, 2, \ldots d_2\} \\ &\bigcup \{(\alpha, \beta, z_\alpha, z_\beta) : \alpha = m_1 + 2, \beta = 0, z_\alpha = 1, 2, \ldots d_1, z_\beta = 0\} \\ &\bigcup \{(\alpha, \beta, z_\alpha, z_\beta) : \alpha = 0, \beta = m_2 + 2, z_\alpha = 0, z_\beta = 1, 2, \ldots d_2\} \end{aligned} \quad (9)$$

Let us take up the case where $d_1 = 1, d_2 = 2, m_1 = 1, m_2 = 1$. We label the states as follows:

$b_1 = (3, 0, 1, 0)$
$b_2 = (3, 1, 1, 2)$
$b_3 = (3, 1, 1, 1)$
$b_4 = (3, 2, 1, 2)$
$b_5 = (3, 2, 1, 1)$
$b_6 = (2, 3, 1, 2)$
$b_7 = (2, 3, 1, 1)$
$b_8 = (1, 3, 1, 2)$
$b_9 = (1, 3, 1, 1)$
$b_{10} = (0, 3, 0, 2)$
$b_{11} = (0, 3, 0, 1)$

The transition probability matrix is as follows:

$$\begin{pmatrix}
x & xy & 0 & xy^2 & 0 & y^3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & x & 0 & xy & 0 & y^2 & 0 & 0 & 0 & 0 \\
x & xy & 0 & xy^2 & 0 & y^3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x & 0 & y & 0 & 0 & 0 & 0 \\
0 & x & 0 & xy & 0 & y^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & x^2 & 0 & xy & 0 & y & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & x^3 & 0 & x^2y & 0 & xy & 0 & y & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & x^3 & 0 & x^2y & 0 & xy & 0 & y & 0
\end{pmatrix} \tag{10}$$

Then the expressions for the stationary probabilities are (we take $\pi(b_1) = 1$ and will scale the expression for expected throughput appropriately later):

$$\pi(b_1) = 1 \tag{11}$$

$$\pi(b_2) = \frac{y}{x^2} \tag{12}$$

$$\pi(b_3) = \frac{y}{x} \tag{13}$$

$$\pi(b_4) = \frac{y^2}{x^4}(1+xy) \tag{14}$$

$$\pi(b_5) = \frac{y^2}{x^3}(1+x) \tag{15}$$

$$\pi(b_6) = \frac{y^3}{x^5}(1+xy) \tag{16}$$

$$\pi(b_7) = \frac{y^3}{x^4}(1+x) \tag{17}$$

$$\pi(b_8) = \frac{y^2}{x^6}(1+xy) - \frac{3y^3}{x^4}(1+x) - \frac{y^4}{x^5}(1+xy) - \frac{y^2}{x} \tag{18}$$

$$\pi(b_9) = \frac{y^3}{x^5}(1+xy) \tag{19}$$

$$\pi(b_{10}) = \frac{y^3}{x^7}(1+xy) - \frac{y^4}{x^5}(2+2x+x^2) - \frac{y^3}{x^2} \tag{20}$$

$$\pi(b_{11}) = \frac{y^2}{x^7}(1+xy) - \frac{3y^3}{x^5}(1+x) - \frac{y^2}{x^2} \tag{21}$$

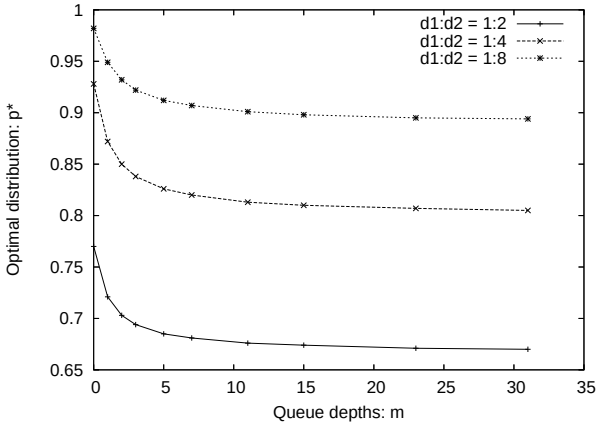

Figure 2: Workload Distribution for Maximum Throughput (I)

Now, one job is completed on the first server in every state where $z_\alpha = 1$ and similarly one job is completed on the second server in every state where $z_\beta = 1$. Hence, the expression for expected throughput for the above configuration is given by:

$$\begin{aligned} R &= \pi(b_1) + \pi(b_2) + 2\pi(b_3) + \pi(b_4) + 2\pi(b_5) \\ &\quad + \pi(b_6) + 2\pi(b_7) + \pi(b_8) + 2\pi(b_9) + \pi(b_{11}) \end{aligned} \tag{22}$$

Numerically, we find that this function attains its maximum at $x = 0.721$ (precision up to three places of decimal).

Next, we have automated the procedure to generate and solve the Markov chain for any given system configuration and job distribution. We have studied the optimal distribution (up to a precision of three decimal places) for a number of configurations of the fork. In Fig. 2 we present the optimal distribution (labeled as p^*) for the following configurations: server service times are in the ratio $1:2, 1:4, 1:8$, both queue depths are kept equal and the joint queue depth (labeled as m) is varied from 0 to 31.

We observe the following trends:

- the optimal distribution has a dependence on queue depths,
- as queue depths are increased the optimal distribution converges towards the distribution that distributes the workload in the inverse ratio of the service times (we will refer to this as the optimal distribution under infinite queueing),

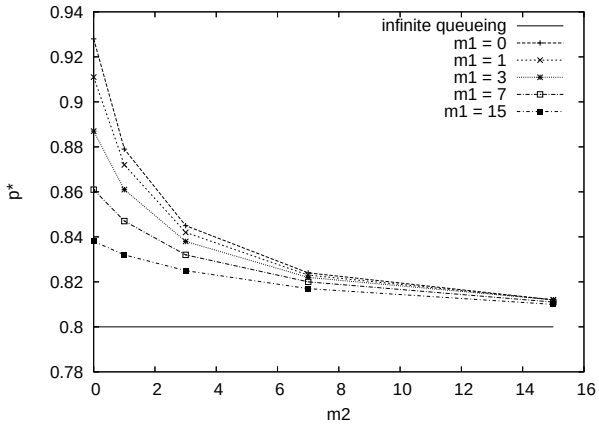

Figure 3: Workload Distribution for Maximum Throughput (II)

d_2	$Original$	$Model-C$	$Model-G$	$Model-Q$
2	0.721	0.845	0.793	0.716
4	0.872	0.964	0.953	0.864
8	0.949	0.991	0.990	0.941

Table 2: Optimal Workload for Queue Depths 1 across all Models

- at smaller queue depths the optimal distribution places a larger share of the load on the faster server.

At this point, let us compare the data for the optimal distribution with the corresponding data using the memory-less models. In Table 2 we present the data for the following configurations: the first server has mean service time 1 and the mean service time for the second server is varied as $d_2 = 2, 4, 8$, both queue depths are 1.

As we can see, the values for optimal distribution depend on the system model being used.

Next, let us return to our original model and look at the trends when queue depths are not equal. (For this presentation we show the plots for service time ratio $1 : 4$ only, however we have verified the trends observed on the configurations with service time ratios $1 : 2$ and $1 : 8$ as well.) In Fig. 3 we show the plots when m_1 and m_2 are varied independently between 0 and 15. We observe that the optimal distribution places a larger share of the load on the faster server as compared to the optimal distribution under

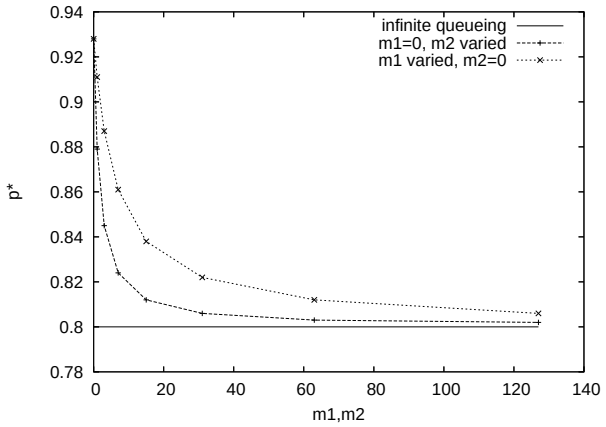

Figure 4: Workload Distribution for Maximum Throughput (III)

infinite queueing, in all cases. In Fig. 4 we show the plots for m_1 fixed at 0 and m_2 varied between 0 and 127, and vice versa. The earlier observation holds for each of these configurations.

It is clear that in the unbalanced server case, the optimal distribution has a strong dependence on queue depths (which contrasts it with the balanced server case).

4 Conclusions

In this paper, we have examined a queueing system (with servers having deterministic service times) under stress. This model describes several real life situations more accurately than the traditional memory-less queueing network models. In particular, memory sub-systems in VLSI systems can be modeled in this manner. For the two server case, we have presented a discrete time Markov chain model in order to determine the workload distribution that optimizes throughput.

Based on the analysis, we have shown that if the servers are balanced, then the optimal load distribution is also balanced, *irrespective* of the queue depths. This result can be contrasted with the behaviour of memory-less queueing systems with finite queueing, in which the optimal load distribution depends on the depths of the queues in front of the servers. Note that this result is true under the assumption that the destinations of the jobs are i.i.d.

Bursty distributions of the jobs would certainly change the conclusion.

For the unbalanced servers case, we find that the optimal distribution depends on the service times as well as the queue depths. Based on our observations, we make the following conjecture: the optimal distribution favours the faster server (as compared to the distribution in the inverse ratio of service times) irrespective of the queue depths.

Thus, it is clear that the two server fork under stress has behaviour patterns that are different from the memory-less queueing networks. That is, the established results from the analysis of queueing networks (M/M/1 to M/G/n) are not applicable in this model. Since this model seems appropriate for many real-life situations, further study of such networks would be valuable. In particular, the case of more than two servers with non-i.i.d. arrivals is of great interest.

References

[1] L. Kleinrock, "Queueing Systems", John Wiley, New York, 1975.

[2] K. S. Trivedi "Probability and Statistics with Reliability, Queueing and Computer Science Applications" Prentice-Hall Inc. New Jersey 1982.

[3] H. Daduna "Queueing Networks with Discrete Time Scale" Springer, Berlin 2001.

[4] H. G. Perros, "Queueing Networks with Blocking", Oxford University Press, New York, 1994.

[5] Cheng-Shang Chang "Stability, Queue Length and Delay of Deterministic and Stochastic Queueing Networks", *IEEE Transactions on Automatic Control* vol. 39, issue 5, May 1994, pp. 999 – 1004.

[6] A. E. Eckberg "The Single Server Queue with Periodic Arrival Process and Deterministic Service Times", *IEEE Transactions on Communications*, vol. COM-27, no. 3, March 1979, pp. 556 – 562.

[7] P. Humblet, A. Bhargava, M. G. Huchyj "Ballot Theorems Applied to the Transient Analysis of nD/D/1 Queues", *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, Feb. 1993, pp. 81 – 95.

[8] K. Lai, M. Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay", *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Stockholm, Sweden, 2000, pp. 283 – 294.

[9] D. D. Kouvatsos, R. Fretwell "Discrete Time Batch Renewal Processes with Applications to the Performance Modeling of ATM Switch Architectures" *Twelfth UK Teletraffic Symposium: Performance Engineering in Telecommunications Networks*, Old Windsor, March 1995, pp. 16/1 – 16/12.

[10] G. Hazari, M. P. Desai, "Towards a Memory Sub-system Design Procedure for Enhanced Performance in SoC Architectures", *PhD. Progress Report, Dept. of Elect. Engg., I.I.T. Bombay,* Aug. 2006.

[11] G. Hazari, M. P. Desai, "Towards a Memory Sub-system Design Procedure for Enhanced Performance in SoC Architectures", *PhD. Progress Report, Dept. of Elect. Engg., I.I.T. Bombay,* (to be submitted) Aug. 2007.

[12] G. Hazari, M. P. Desai, H. Kasture "On the Impact of Address Space Assignment on Performance in Systems-on-Chip", *Proc. VLSI Conference*, Jan. 2007, Bangalore, India.