Performance Analysis of SoC Architectures Based on Latency-Rate Servers

Jelte Peter Vink
Eindhoven University of Technology
Eindhoven, the Netherlands
E-mail: jelte.peter.vink@philips.com

Kees van Berkel
NXP Semiconductors Research
Eindhoven, the Netherlands
E-mail: kees.van.berkel@nxp.com

Pieter van der Wolf
NXP Semiconductors Research
Eindhoven, the Netherlands
E-mail: pieter.van.der.wolf@nxp.com

Abstract

This paper presents a method for static performance analysis of SoC architectures. The method is based on a network calculus theory known as LR servers. This network calculus is extended and applied to make it support SoC performance analysis. Performance requirements of subsystems are elegantly captured as traffic flows and associated latency constraints. The SoC infrastructure is modeled as a set of LR servers to validate that the worst-case delays in handling the traffic flows meet the latency constraints. A multi-channel DVB-T set-top box case study demonstrates the power of the method. Key architecture choices, such as schedule or interconnect variant, can be varied easily to support exploration of architecture options.

1. Introduction

At the heart of consumer products for mobile communications and digital entertainment is typically a complex System-on-Chip (SoC) that performs a broad range of functions. These functions are implemented by subsystems on the SoC, where a subsystem may contain one or more IP blocks like programmable processors and / or dedicated hardware blocks. The subsystems are connected by a SoC infrastructure consisting of interconnect and memory.

A SoC architect has many design options for a SoC architecture that need to be explored. Simulation-based approaches to architecture exploration typically require large efforts for building models of the SoC architectures and have long execution times. Also they do not provide guarantees that performance requirements are met under all circumstances, e.g. for all video content and for all possible interactions in the SoC infrastructure.

We propose a method for static performance analysis that can be used for early exploration of SoC architectures. The method can be used for worst-case analysis, so that the required performance can be guaranteed. Models can be constructed quickly and evaluated with short execution times. The method supports the SoC architect in gaining insight in the design problem at hand by providing valuable feedback for the metrics of interest, such as the required size of a queue in the SoC infrastructure. It helps him to set budgets for execution times, bandwidths, latencies etc that need to be met in downstream design activities to satisfy performance requirements.

Our method for static performance analysis is based on network calculus theory [1] [2]. Performance requirements of subsystems are captured as a set of *traffic flows* with associated *latency constraints*. The SoC infrastructure is modeled as a set of interconnected network elements. We then verify for the specified traffic flows whether the worst case delays incurred by the SoC infrastructure satisfy the latency constraints associated with these traffic flows.

The aim of this paper is to show that an extended form of the "Latency-Rate Servers" network calculus [2] can be applied for fast exploration of SoC architectures. We extend and apply the basic network calculus theory in order to make it support the performance analysis of SoC architectures. We show that the extended network calculus is sufficiently expressive to capture SoC architectures in concise models. We present a realistic design case of industrial complexity to show how different architecture options, such as schedule or interconnect variant, can be modeled and evaluated early, enabling the SoC architect to identify the valid options that meet the required deadlines.

In section 2 we introduce a network calculus theory known as Latency-Rate servers. In section 3 we discuss related work. In section 4 we present our performance analysis method. In section 5 we discuss a multi-channel DVB-T set-top box case study and we present the results of the analysis method. Finally, in section 6 we draw conclusions.

2. Latency-Rate servers

In network calculus, traffic can be characterized by an upper bound. In its basic form, this upper bound is a monotonously increasing function $c(t) = \sigma + \rho \cdot t$, where σ represents the burstiness constraint in $words$ and ρ the rate of the traffic stream in $\frac{words}{s}$. In Figure 1, b_1 is a traffic stream on a particular link during a particular time interval. b_1 is upper bounded by c_1. Network calculus provides an elegant way to describe bounds on traffic for a *sliding window* of *arbitrary size* with just two parameters (i.e. (σ, ρ)).

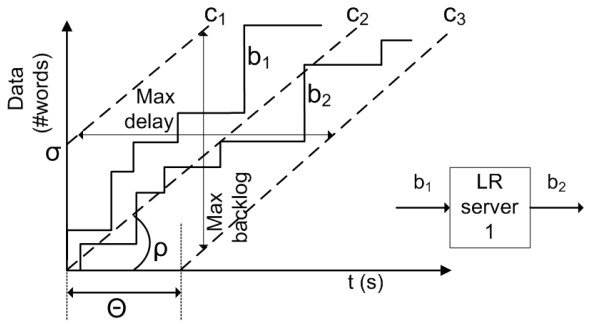

Fig. 1. Traffic modeling of an LR server, if the input is between c_1 and c_2 then the output of the LR server is between c_1 and c_3

978-3-9810801-3-1/DATE08 © 2008 EDAA

Data is sent by making use of packets. If a packet with size L $words$ is transferred over a link with capacity C $\frac{words}{s}$, it takes $\frac{L}{C}$ s to transfer that packet.

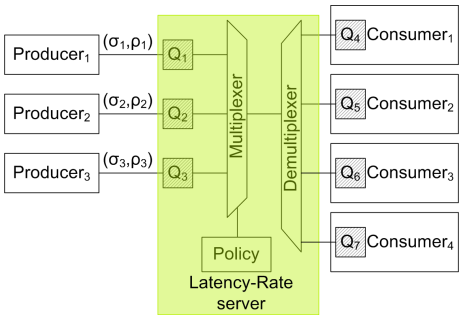

Fig. 2. A system consisting of a Latency-Rate server, 3 producers and 4 consumers [2] ("Q_i" denotes a queue)

For modeling and analyzing communication networks, Stiliadis and Varma [2], [3] have introduced a network element called Latency-Rate server (LR server). In Figure 2 a model of the internal structure of an LR server is shown. More than one producer transmits traffic (represented by σ_i and ρ_i) to an LR server. This traffic is temporarily stored in queues. A multiplexer combines the traffic according to an arbitration policy. Finally, the traffic is demultiplexed and each packet is sent to a consumer.

The behavior of an LR server is determined by the latency (Θ_i) and the allocated service rate in $\frac{words}{s}$ (ρ_i) for input traffic stream i. An LR server guarantees an output service rate ρ_i, a time period Θ_i after receiving packets of stream i (see Figure 1). So, if b_1 is between c_1 and c_2 (i.e. $\rho \cdot t$), then b_2 has a lower bound of c_3 (i.e. $\rho \cdot (t - \Theta)$).

The delay of a packet of stream i (i.e. D_i) for a chain of m LR servers is upper bounded by (see Figure 1)

$$D_i \leq \frac{\sigma_i}{\rho_i} + \sum_{j=1}^{m} \Theta_i^j \tag{1}$$

The maximum backlog in $words$ of the k^{th} LR server in a chain of LR servers for stream i (i.e. Q_i^k) is upper bounded by (see Figure 1)

$$Q_i^k \leq \sigma_i + \rho_i \cdot \sum_{j=1}^{k} \Theta_i^j \tag{2}$$

This determines the required queue size for the k^{th} LR server.

3. Related work

Cruz has pioneered a network calculus [1], originally intended for computer networks. He made a mathematical framework for deriving worst-case bounds on the performance (e.g. the delay). He assumed that $R(t)$ represents the instantaneous rate in $\frac{words}{s}$ of a stream flowing on a link at time t. Then, this traffic can be upper bounded by a monotonously increasing function $c(t)$, such that $\int_{t_1}^{t_2} R(t) \leq c(t_2 - t_1)$, $(\forall t_1, t_2 : 0 \leq t_1 \leq t_2)$, where $c(t) = \sigma + \rho \cdot t$. Stiliadis and Varma extended this work and introduced the LR server as an abstract network element.

Chakraborty and Thiele define bounds for arrival curves and service curves of traffic streams in [4]. They present a task-level model that captures some properties of stream processing applications and supports investigation of task interactions. Jersak et al. [5] also focus on the task interaction. Their event streams are somewhat similar to the traffic model we employ.

In our paper, the focus is on the traffic between the subsystems and the SoC infrastructure, where we assume that the deadlines have been specified for the subsystems and do not depend on the interaction between subsystems. The techniques in [4] may be used for deriving such deadlines. Our techniques can then be applied to derive the actual execution times of tasks in order to check that the deadlines can be met.

In [6] Henriksson applies network calculus for the analysis of memory access latencies. He supports request-response streams and pipeline degrees to obtain tight bounds in the analysis. We adopt these concepts and derive expressions for total delay.

4. Performance Modeling and Analysis

The method of Stiliadis is used as foundation for our performance analysis method. The (σ, ρ) traffic model is used to characterize traffic and LR servers are used for modeling SoC infrastructures. This method was not originally devised for SoC performance analysis. We observed that the method of Stiliadis does not support

- the total delay for handling data consisting of multiple packets starting from the first word of the first packet until the last word of the last packet. Stiliadis only derives delays for individual packets where delay is defined as the time between last-word-in and last-word-out.
- the total delay for sending a request to a subsystem and receiving a response to that request.
- the total delay for sending requests, where the number of outstanding requests is limited.
- a model of a memory system.
- SoC arbitration policies (e.g. TDMA).

In [6] the concepts of request-response stream and pipeline degree are introduced and a DRAM model is derived. The contribution of this paper consists of deriving expressions for the total delays with support for three traffic characteristics (dropping impulse assumption, request-response stream and pipeline degree) and a SoC arbitration policy (TDMA).

For the performance analysis method, we have made the following assumptions.

- There are no dependencies between the different traffic streams of a subsystem.
- The input of a subsystem is according to specified traffic characteristics.
- The output of a subsystem can always be delivered (e.g. buffers are large enough to prevent overflow).

4.1 Dropping impulse assumption

Stiliadis assumes that a packet has been serviced when its last word has left the server. Then, the arrivals and departures of packets are considered as impulses (impulse assumption). Therefore, he uses last-word-in, last-word-out to determine

the delay. Because the total delay is required for sending packets from Subsystem 1 via LR servers to Subsystem 2, first-word-in, last-word-out is required. Then, the arrival time of the first packet in the first LR server is part of the total delay. Therefore, a term $\frac{L}{C}$ is added to the total delay of Equation (1) [7].

4.2 Request-response streams

Subsystems interact via the SoC infrastructure. Stiliadis only determined the maximum delay for a packet traveling from one subsystem to another. In some cases, a subsystem sends requests via the SoC infrastructure to another subsystem and waits for responses from that subsystem. This type of traffic stream is called a "request-response stream" [6]. An example is a load request and a corresponding load response from a memory system. We extend the method of Stiliadis with request-response streams and derive an equation for the delay of such streams.

Assume the situation of Figure 3. Subsystem 1 produces requests and sends these requests via a chain of m LR servers, with a total latency of $\sum_{j=1}^{m} \Theta_{req}^{j}$, to Subsystem 2. Subsystem 2 produces responses to the requests and sends these responses via a chain of m' LR servers, with a total latency of $\sum_{j=1}^{m'} \Theta_{resp}^{j}$, back to Subsystem 1. Furthermore, Subsystem 2 requires D_{proc} s to produce a response, after a request is received. Finally, for each request exactly one response is generated.

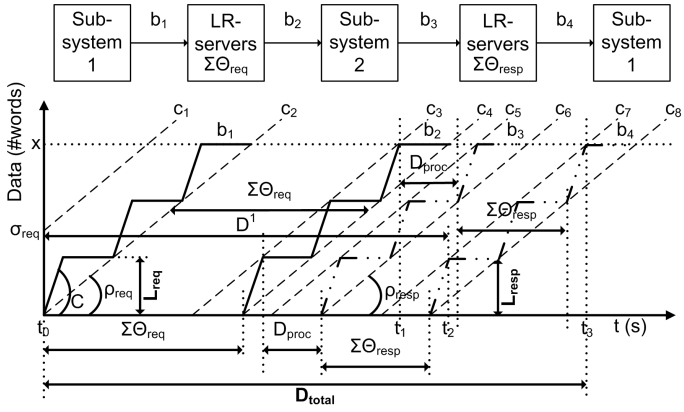

Fig. 3. Request-response stream, $L_{req} = L_{resp}$ for clarity

Assume that the request flow b_1 is characterized by σ_{req}, ρ_{req} and packet size L_{req} $words$. Furthermore, the response flow b_3 is characterized by σ_{resp}, ρ_{resp} and L_{resp}. Finally, assume that $\frac{L_{req}}{\rho_{req}} = \frac{L_{resp}}{\rho_{resp}}$. Then, the number of requests and responses per time unit is the same.

If the request flow is bounded by c_1 and c_2 (see Figure 3), the LR servers guarantee that these requests arrive at Subsystem 2 $(\frac{L}{\rho} - \frac{L}{C})$ s before c_4. Then, at most D_{proc} later, the responses can be sent back to Subsystem 1. Therefore, the response stream b_3 is lower bounded by c_6. The responses arrive at Subsystem 1 before c_8, because of the guarantees of LR servers.

Then, the total delay for x $words$ of requests (i.e. D_{total}) is upper bounded by [7]

$$\begin{aligned} D_{total} &\leq t_3 - t_0 \\ &= (t_1 - t_0) + (t_3 - t_1) \\ &= \left(\left\lceil \frac{x}{L_{req}} \right\rceil \cdot \frac{L_{req}}{\rho_{req}} + \sum_{j=1}^{m} \Theta_{req}^{j} - \frac{L_{req}}{\rho_{req}} + \frac{L_{req}}{C} \right) \\ &\quad + \left(D_{proc} + \sum_{j=1}^{m'} \Theta_{resp}^{j} + \frac{L_{resp}}{C} \right) \end{aligned} \quad (3)$$

where $\lceil \frac{x}{L_{req}} \rceil \cdot \frac{L_{req}}{\rho_{req}}$ is the delay for sending x $words$, $\sum_{j=1}^{m} \Theta_{req}^{j}$ the latency of the first chain of LR servers, $\frac{L_{req}}{\rho_{req}}$ the improved delay bound as described in [2], $\frac{L_{req}}{C}$ the delay of sending a request packet, D_{proc} the processing delay of Subsystem 2, $\sum_{j=1}^{m'} \Theta_{resp}^{j}$ the latency of the second chain of LR servers and $\frac{L_{resp}}{C}$ the delay of receiving the last response.

The backlog of the LR servers can still be determined by the original equation of Stiliadis (Equation (2)).

4.3 Pipeline degree

For most subsystems in SoC architectures, the number of outstanding requests is limited. In [6] the pipeline degree of a traffic stream is defined as the maximum number of outstanding requests. A higher pipeline degree can decrease the maximum delay of a traffic stream, but increases the complexity of the subsystem. We introduce the pipeline degree into the method of Stiliadis and derive an equation for the delay.

Assume that n is the pipeline degree. Then, the $(n+k)^{th}$ request cannot be sent earlier than the k^{th} response has been received. Assume the same situation as in Figure 3, but now the number of outstanding requests is limited. Then, the maximum delay for x $words$ of requests can be split into periods of sending n requests and waiting until the first response arrives (i.e. D_1, see Figure 3). Therefore [7],

$$D_1 \leq \frac{L_{req}}{C} + \sum_{j=1}^{m} \Theta_{req}^{j} + D_{proc} + \sum_{j=1}^{m'} \Theta_{resp}^{j} + \frac{L_{resp}}{C} \quad (4)$$

$$\begin{aligned} D_{total} &\leq \left\lceil \frac{x}{n \cdot L_{req}} \right\rceil \cdot D_1 \\ &\quad + \left(\left\lceil \frac{x}{L_{req}} \right\rceil - n \cdot \left(\left\lceil \frac{x}{n \cdot L_{req}} \right\rceil - 1 \right) - 1 \right) \cdot \frac{L_{resp}}{\rho_{resp}} \end{aligned} \quad (5)$$

A maximum of n request packets of L_{req} $words$ (i.e. $n \cdot L_{req}$ $words$) can be outstanding. Then, there are $\lceil \frac{x}{n \cdot L_{req}} \rceil$ periods of D_1. In total, $\lceil \frac{x}{L_{req}} \rceil$ request packets have to be processed by Subsystem 2. After $\lceil \frac{x}{n \cdot L_{req}} \rceil$ periods of D_1, at least $n \cdot (\lceil \frac{x}{n \cdot L_{req}} \rceil - 1) + 1$ response packets have been received by Subsystem 1. Then, at most $(\lceil \frac{x}{L_{req}} \rceil - n \cdot (\lceil \frac{x}{n \cdot L_{req}} \rceil - 1) - 1)$ response packets still have to be received. This takes at most $\frac{L_{resp}}{\rho_{resp}}$ s for each response packet.

The backlog of the LR servers can still be determined by the original equation of Stiliadis (Equation (2)). σ_i is now lower bounded by [1] [7]

$$\sigma_i \geq n_i \cdot L_i \cdot \left(1 - \frac{\rho_i}{C}\right) \quad (6)$$

4.4 DRAM memory model

A DRAM memory system has some specific characteristics that are important for the performance analysis of SoC architectures. The method of Stiliadis assumes that the processing delay of a packet is proportional to the size of the packet. In case of a memory system, this proportionality does not hold [8]. This can be taken care of by a so-called packet stretcher as described in [6]. Using the model of a DRAM memory system as described in [6], the DRAM controller can be modeled as an LR server.

4.5 TDMA

A useful SoC arbitration policy is TDMA. TDMA uses a periodic schedule. In each round, the input streams of an LR server using TDMA are served in a Round Robin fashion. The maximum number of packets of stream i (i.e. w_i) that the LR server can serve in a round is determined in advance. Let assume that V streams share an LR server with TDMA as arbitration policy and that the packets of stream i have a size L_i. Then, the maximum amount of service stream i receives from the LR server (i.e. ϕ_i) is $\phi_i = w_i \cdot L_i \ words$.

The total amount of service the streams receive in one round from the LR server is called a "frame". The size of a frame (i.e. F) is $F = \sum_{i=1}^{V} \phi_i \ words$.

The latency of a packet is the maximum amount of time between the moment the first word of that packet arrives in the LR server and the moment that the last word of the packet has left the LR server. Assume the LR server has a maximum rate of $C \ \frac{words}{s}$. In the worst case scenario the packet has to wait $\frac{F-\phi_i}{C} \ s$ before the stream is serviced. The packet itself is serviced in $\frac{L_i}{C} \ s$. Therefore, the latency of stream i for TMDA is [7]

$$\Theta_i^{TDMA} = \frac{F - \phi_i + L_i}{C} \tag{7}$$

The LR server with TDMA allocates a maximum rate of $\frac{\phi_i}{F} \cdot C$ to stream i.

In a similar way the Θ_i can be derived for other arbitration policies (see [7]).

5. Multi-channel DVB-T set-top box case study

A multi-channel DVB-T set-top box case study is used to illustrate the power of the performance analysis method. We analyze the performance of several schedule and interconnect variants.

5.1 System description

Digital video signals are received by an antenna (see Figure 4) and sent via the Radio Front-End (RF) to the DVB-T Channel Decoders (CDs). Four DVB-T CDs are used to decode four different channels. Two decoded channels are stored, two decoded channels are processed by H.264 decoders. The output of the H.264 decoders can be used for dual screen or dual window functionality. The scope of this case study is indicated by the dotted shape in Figure 4.

A possible implementation is shown in Figure 5. The four DVB-T CDs are combined on one subsystem. To temporarily store and exchange data between subsystems, an external

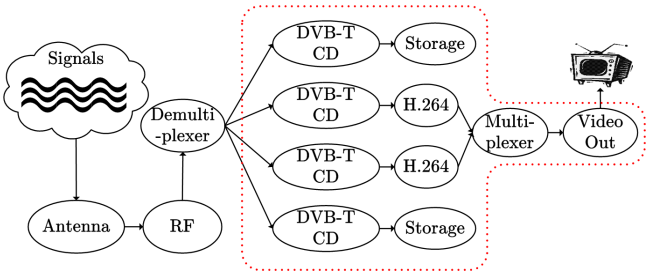

Fig. 4. Multi-channel DVB-T set-top box functional overview

DRAM memory system (indicated by DRAM controller and DRAM) is used. The DRAM controller is modeled as an LR server, using TDMA as arbitration policy. Each subsystem is connected via a private link to the DRAM controller. The output of the DRAM is sent back to the subsystems via a bus. The burstiness of the input and the output of the DRAM controller is constrained by using regulators (indicated by "R") (see [9]).

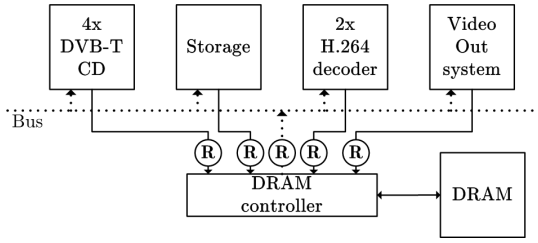

Fig. 5. Private links interconnect

5.2 Schedule variants

The first step is to characterize all traffic streams between the subsystems and the memory system. One of the traffic streams is the traffic of the DVB-T CD to perform the decoding. The four DVB-T CDs are mapped onto one processor and are active alternately. A DVB-T CD outputs an interleaved stream of so-called $OFDM$-symbols of 20 kB. In each active period of a DVB-T CD, two $OFDM$-symbols have to be loaded from the DRAM into on-chip buffers (a_i and b_i). During an active period, a new $OFDM$-symbol is created in an on-chip buffer (c_i) which has to be written to the DRAM.

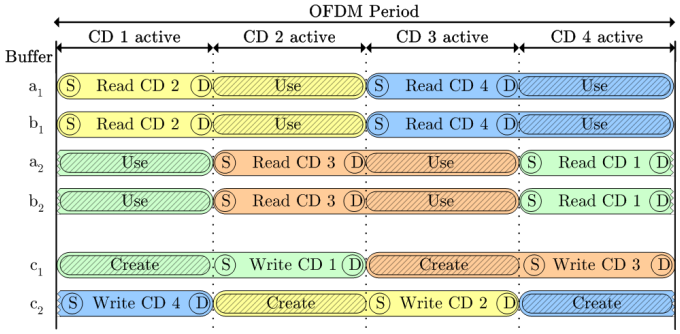

Fig. 6. Repeating pattern of Schedule 1, "S" represents sent read or write request, "D" represents deadline of request, the read buffers (a_1, a_2, b_1 and b_2) are used during "Use" and the write buffers (c_1 and c_2) are filled during "Create"

One option is to load the $OFDM$-symbols when the previous DVB-T CD is active (see Figure 6). This means

that 40 kB have to be loaded within $\frac{1}{4}$ $OFDM$-period (i.e. $\frac{1}{4} \cdot 896\ \mu s$). When the DVB-T CD becomes active, the required $OFDM$-symbols are available. When the next DVB-T CD becomes active, the produced $OFDM$-symbol is written to the DRAM. This variant is called Schedule 1. Assume that $L_{req} = 8\ bytes$ and $L_{resp} = 128\ bytes$. ρ_{req} represents the rate of the read requests

$$\rho_{req} \geq \left\lceil \frac{2 \cdot 20\ kB}{128\ B} \right\rceil\ requests\ per\ \frac{1}{4}\ OFDM\ period \quad (8)$$

The latency constraint is 2 blocks of 20 kB, i.e. 313 packets of 128 $bytes$, within $\frac{1}{4}$ $OFDM$-period. Hence, the lower bound for the request rate ρ_{req} is $313 \cdot 8/(\frac{1}{4} \cdot 896\ \mu s) = 11.2\ MBps$. The other traffic streams are determined in a similar way.

An alternative schedule variant is to divide the active period of a DVB-T CD into two equal parts. In the first part, the two $OFDM$-symbols are loaded from the DRAM, in the second part the produced $OFDM$-symbol is written to the DRAM. Now, both operations have a deadline of $\frac{1}{8}$ $OFDM$-period. Therefore, the rates of this traffic are higher compared to Schedule 1. This variant is called Schedule 2.

An overview of the traffic streams can be found in Table 1. All traffic is to / from the DRAM. The σ of a traffic stream can be calculated by Equation (6). Note that H.264 also needs DRAM access for accessing private data.

Table 1. Overview of the characteristics of the traffic streams, "T" represents an $OFDM$-period, "M" represents the DRAM

Schedule	Between	ρ_{req} $(MBps)$	ρ_{resp} $(MBps)$	Deadline $(\frac{\#requests}{s})$
1	CD - Read	11.2	179	313 / $\frac{1}{4} \cdot T$
1	CD - Write	89.7		157 / $\frac{1}{4} \cdot T$
2	CD - Read	22.4	358	313 / $\frac{1}{8} \cdot T$
2	CD - Write	179		157 / $\frac{1}{8} \cdot T$
-	CD - Write	16.0		28 / $\frac{1}{4} \cdot T$
-	H.264 - Read	0.500	8.00	28 / $\frac{1}{2} \cdot T$
-	Storage - Read	0.500	8.00	28 / $\frac{1}{2} \cdot T$
-	H.264 - Read	18.7	299	38880 / $\frac{1}{60}$
-	H.264 - Write	41.5		6480 / $\frac{1}{50}$
-	Video - Read	2.59	41.5	6480 / $\frac{1}{50}$
-	M - Refresh	1.02		$1 / 7.81 \cdot 10^{-6}$

A Mathematica model has been created to execute the performance analysis. The input consists of values of σ, ρ, L and n for each traffic stream and a model of the communication infrastructure using LR servers. Then, Equations (2), (3), (4) and (5) are used to calculate the maximum delay per traffic stream and the required queue size per LR server. Different TDMA-wheels, DRAM frequencies and pipeline degrees are analyzed. Only results where the maximum delay for each traffic stream meets the deadline are used. Per memory frequency and per maximum pipeline degree the best solution (i.e. the smallest queue size required for the LR server and the regulators) is selected. The results are shown for Schedule 1 and 2 using the private links interconnect in Figures 7 and 8. Each pipeline degree has a unique color. Analyzing the results of Schedule 1, we observe the following.

- For frequencies smaller than 150 MHz, no solution has been found that guarantees all deadlines.

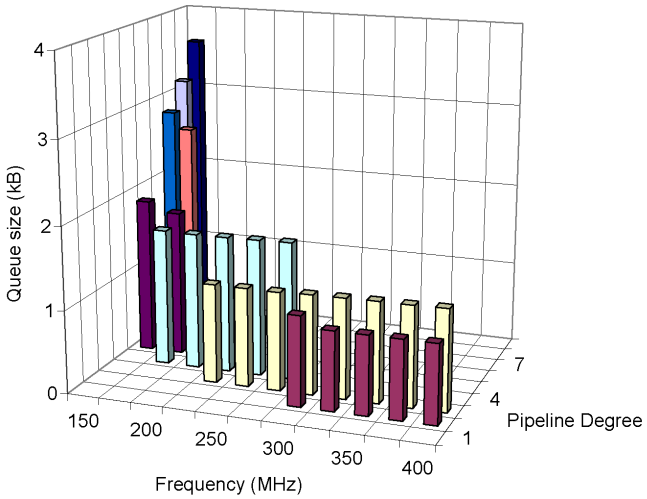

Fig. 7. Results of private links interconnect using Schedule 1

- By increasing the frequency, it is possible to meet all deadlines with a lower pipeline degree and smaller queues. If packets are processed faster, fewer packets have to be in flight simultaneously.
- The queue size is monotonic in the pipeline degree and the frequency.
- If the frequency gets higher, there is less benefit from reducing the pipeline degree.

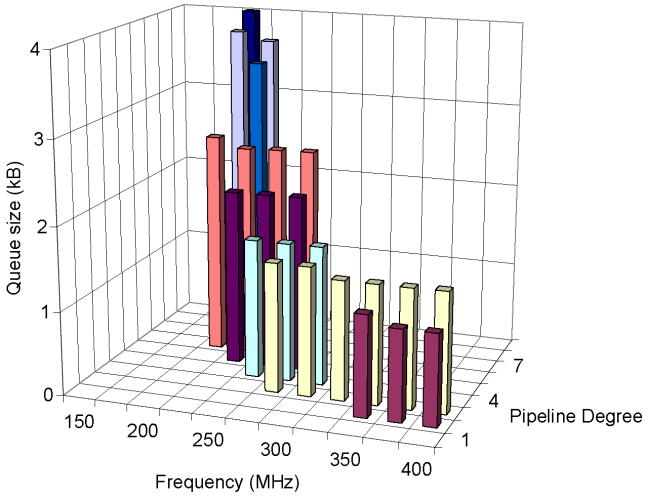

Fig. 8. Results of private links interconnect using Schedule 2

The deadlines for reading and writing the $OFDM$-symbols are more strict for Schedule 2 compared to Schedule 1. Therefore, a higher frequency and a higher pipeline degree are required to meet all deadlines (see Figure 8).

5.3 Bus interconnect variant

An alternative interconnect is analyzed in combination with Schedule 1. In a bus based interconnect variant (see Figure 9), all traffic to and from the memory system is transmitted via one bus. Before transmitting data over the bus, a request has to be sent to the bus arbiter. The requests are regulated to constrain their burstiness. The bus arbiter (modeled as an LR server with TDMA) determines the order on the bus. The

requests arriving at the DRAM controller are processed in a First-Come First-Served fashion.

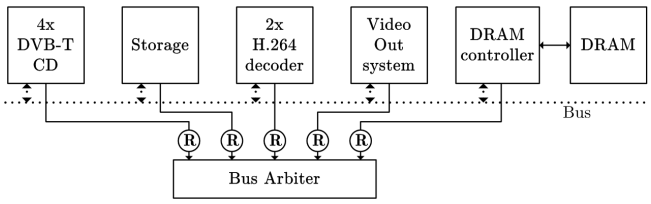

Fig. 9. Bus based interconnect

Using this interconnect variant, we get the results of Figure 10. With this bus based interconnect the traffic to and from the memory system shares the same wires. Therefore, the latencies of the traffic streams increase. To compensate this, the frequency and the pipeline degree have to be significantly higher compared to the architecture with private links to meet all deadlines.

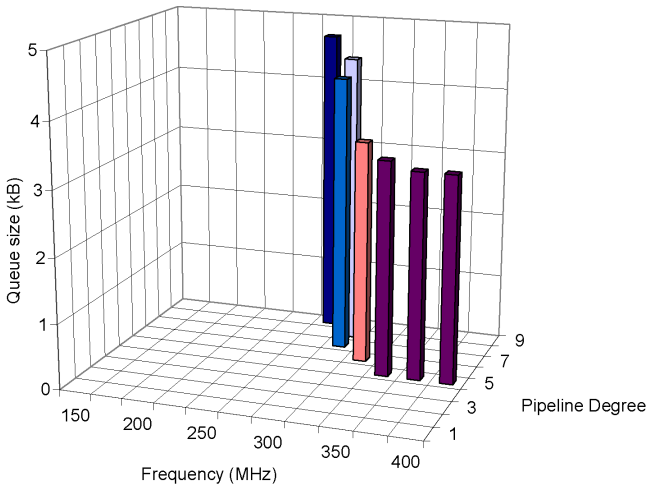

Fig. 10. Results of bus based interconnect using Schedule 1

5.4 Packet size

Until now, the read and write actions of the memory system use blocks of 128 $bytes$. An alternative is to use blocks of 64 $bytes$. Then, the read and write actions have more overhead in the memory system. The results are shown in Figure 11, using private links interconnect and Schedule 1.

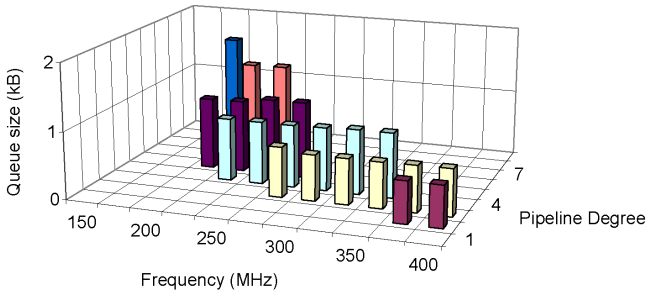

Fig. 11. Results of private links interconnect using Schedule 1 and packets of 64 $bytes$

Due to the extra overhead in the memory system, a higher frequency and a higher pipeline degree are required compared to using blocks of 128 $bytes$ to meet all deadlines. In cases where all deadlines are met for a particular frequency and pipeline degree, it is better to use blocks of 64 $bytes$ than of 128 $bytes$, because substantially smaller queues are required.

6. Conclusion

We have shown that an extended form of the Latency-Rate Server network calculus can be applied for fast exploration of SoC architectures. Performance requirements of subsystems are captured as a set of traffic flows, using a powerful and elegant traffic model, and associated latency constraints. After modeling the SoC infrastructure as a set of LR servers, it can be verified whether the worst-case delays incurred by the SoC infrastructure satisfy the latency constraints of the traffic flows. Several extensions, such as request-response streams, pipeline degree, and TDMA scheduling, were used to make the network calculus support SoC performance analysis.

With a multi-channel DVB-T set-top box case study we demonstrated that models of SoC architectures can be built and evaluated quickly. The modelling time of a variant of the case study was a couple of hours and the execution time of the model is less than one second on a standard PC. The case study illustrated that key architecture choices, such as schedule or interconnect variant, can be varied easily to support exploration of architecture options. The impact on the required frequency, queue size and pipeline degree could be evaluated, showing e.g. that the cheaper bus interconnect requires a higher frequency than the private links interconnect. We therefore conclude that the proposed method for static performance analysis can support a SoC architect in gaining early insight into the design problem at hand and in quickly identifying the most promising design options.

References

[1] R. L. Cruz, "A calculus for network delay, part i: Network elements in isolation and part ii: Network analysis." *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–141, 1991.

[2] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 611–624, 1998.

[3] D. Stiliadis, "Traffic scheduling in packet-switched networks: analysis, design, and implementation," Ph.D. dissertation, 1996.

[4] S. Chakraborty and L. Thiele, "A new task model for streaming applications and its schedulability analysis," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 486–491.

[5] K. Richter, M. Jersak, and R. Ernst, "Performance analysis for complex embedded applications," *International Journal of Embedded Systems, Special Issue on Codesign for SoC*, vol. 45, no. 1/2, pp. 33–49, 2005.

[6] T. Henriksson, P. van der Wolf, A. Jantsch, and A. Bruce, "Network calculus applied to verification of memory access performance in SoCs," *ESTIMedia 2007, Oct 4-5 2007, Salzburg*.

[7] J. P. Vink, "Performance Analysis of SoC Architectures based on Network Calculi," Master's thesis, Eindhoven University of Technology, 2007.

[8] "Mobile DDR SDRAM MT46H32M16LF," Micron Technology, Inc., 2004.

[9] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.