Application-Aware NoC Design for Efficient SDRAM Access

Wooyoung Jang and David Z. Pan
Department of Electrical and Computer Engineering
University of Texas at Austin
wyjang@cerc.utexas.edu, dpan@ece.utexas.edu

ABSTRACT
In this paper, we propose an application-aware networks-on-chip (NoC) design for efficient SDRAM access. In order to provide short latency for priority memory requests with few penalties, a packet is split into several short packets which then are scheduled by the proposed flow controller in a router. Moreover, our NoC design further improves memory performance by matching application access granularity to SDRAM access granularity. Experimental results show that our application-aware NoC design improves on average 32.7% memory latency for latency-sensitive cores and on average 3.4% memory utilization compared to [1].

Categories and Subject Descriptors
C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design - Packet-switching networks.

General Terms
Algorithms, Performance, Design.

Keywords
NoC, on-chip communication, flow control, router, memory, QoS

1. INTRODUCTION
In many-core processors based on networks-on-chip (NoC), memory quality-of-service (QoS) becomes one of the most important issues since both memory and on-chip network are critical shared resources. However, the improvement of memory performance aided by a memory subsystem independently working with an on-chip network is severely limited. Therefore, memory-aware NoC design has attracted great attentions.

Many researchers have developed various QoS approaches [2-5] for NoC. However, they are not optimized for memory requests that cause the longest latency. Recently, the responsibility for high memory performance has been shared not only with memory subsystems but also with on-chip networks. In [1], each NoC router instead of memory subsystems schedules memory requests for a best-effort memory service. As a result, since memory requests arrive at a memory subsystem in the order more friendly to SDRAM operations, average memory latency and utilization (defined as the number of clock cycles used for data transfer divided by the number of total clock cycles) greatly improve with lower NoC design cost. However, as different applications generate their specific memory requests with various latency constraints and packet lengths, [1] needs to support various priority services and match application access granularity to SDRAM access granularity.

In this paper, we propose an application-aware NoC design for efficient SDRAM access. Our key motivations are two-fold. First, different applications request various SDRAM latencies. For example, demand memory requests generated by a microprocessor are commonly served as a priority packet since the microprocessor may halt until the demand memory request is served. However, since the priority packet is served first by network routers which do not consider SDRAM operations, there is strong possibility to meet bank conflict and data contention which makes memory performance worse. In addition, a long best-effort packet prevents a priority packet being served fast. Therefore, a priority service which is efficient in accessing SDRAM should be provided and long best-effort packets should be split to short packets and then served. Second, different applications request various lengths of SDRAM data whereas DDR I/II SDRAM always generates fixed-length data. Even if DDR III SDRAM can generate variable-length data, it has few advantages due to long t_{CCD} (CAS to CAS delay time) [6]. As a result, if the length of data requested by applications is neither the same as the length of data served by SDRAM nor a multiple of the length of data served by SDRAM, unnecessary data may be accessed and then thrown away. Therefore, the access granularity mismatch problem should be considered. Based on these motivations, the major novelty and contribution of this paper include the following.

- A guaranteed SDRAM service (GSS) flow controller is proposed for applications sensitive to SDRAM latency, which provides various priority services with few penalties.
- An SDRAM access granularity matching (SAGM) NoC design is proposed. Based on SDRAM access granularity, a packet is split to short fixed-length packets and then scheduled by our GSS flow controller. In addition, our memory subsystem uses a partially open-page and an auto-precharge (AP) mode.
- We show that our application-aware NoC design significantly improves not only total memory utilization but also memory latency for a priority packet.

2. PROBLEM DESCRIPTION
2.1 Priority SDRAM Service in NoC
A microprocessor commonly generates a demand request and a prefetch request. A demand request should be served as soon as possible since a microprocessor may stall until a service of the demand request is received. On the other hand, a prefetch request does not need to be served with a priority since it may be not promptly used. Memory requests of multimedia processors and peripherals are also handled similarly to a prefetch request.

Fig. 1(b) and (c) show two different approaches as to how to treat a priority request with respect to others, where two demand requests, two prefetch requests and two requests by a video processor are filled in input buffers of an NoC router as shown in Fig. 1(a). BA means a bank address and all requests are read operations. In addition, RAs (Row Addresses) of all requests are different except for prefetch 2 and request 2.

A memory scheduler in Fig. 1(b) regards a priority memory request to have the same priority as others and then schedules memory requests to avoid bank conflict and data contention and to encourage row-buffer hit and bank interleaving [1]. As a result, whereas all memory requests are successively executed with no bank conflict, the execution of demand 2 is considerably delayed. On the other hand, in Fig. 1(c), demand memory requests are executed with a priority. However, since demand 2 accesses the

DAC'10, June 13-18, 2010, Anaheim, California, USA

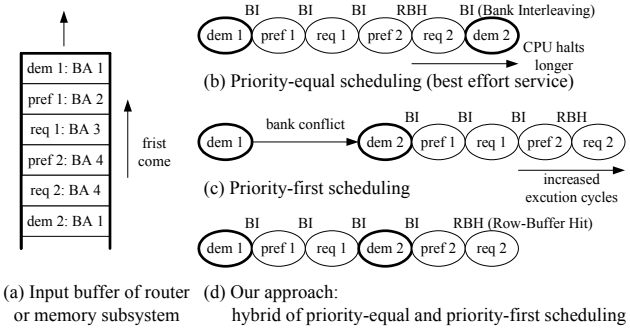

Figure 1: Examples of scheduling memory requests

same bank as demand 1 access with different RA, bank conflict happens. It makes any data not delivered while a row buffer of bank 1 becomes idle and is filled with data of demand 2. Consequently, total execution time of six requests is longer.

We propose a hybrid scheduling algorithm which achieves the same memory utilization as priority-equal scheduling and the same memory latency for demand requests as priority-first scheduling as shown in Fig. 1(d). The scheduling is performed by a flow controller in each NoC router which is similar to [1].

Moreover, we consider a long best-effort packet. In winner-take-all bandwidth allocation, it causes a priority packet to be severely delayed. The reason is that if a long best-effort packet is already scheduled, a priority packet should wait until the long best-effort packet finishes being delivered. To solve this problem, packets are split to short fixed-length packets and then scheduled. A length of short packets is determined by SDRAM access granularity. Consequently, priority packets can get more opportunities to get a channel.

2.2 SDRAM Access Granularity Mismatch

SDRAMs transfer or receive fixed-length data (= number of data bit × burst length) per read/write. DDR I SDRAM has burst length (BL) 2, BL4 and BL8 modes and DDR II/III SDRAM has BL4 and BL8 modes. Especially, DDR III SDRAM has an additional selectable BL4/BL8 on-the-fly (OTF) mode. For example, if SDRAM with 16-bit data bus is set to a BL8 mode, it always generates 16 bytes per read/write. On the other hand, applications or cores request various data lengths to SDRAM. For example, H.264 [7] decoders request 4, 8 or 16 bytes per row for motion compensation to SDRAM. If H.264 decoder reads 4 or 8 bytes, the rest of data are thrown away.

Simple solutions are to reduce the number of data bit or to use short BL. If the number of data bit is changed to 4 bits, there is no useless data. However, the entire system does not have enough memory bandwidth. Therefore, we use short BL to match access granularity. That is, SDRAM is set to a short BL mode and a packet is split to short packets based on the BL, which is also related to Section 2.1. To support a short BL mode, our memory subsystem operates with a partially open-page and AP mode.

3. APPLICATION-AWARE NOC DESIGN

3.1 GSS Flow Controller

In this section, we propose a GSS flow controller providing various priority services and achieving similar memory utilization to a best-effort scheduler. Let *h(n)* be a packet, which is already allocated a channel by our GSS flow control at the *n*th arbitration. Let $h_i(n+1)$ be any packet *i* of all completing packets, *H(n+1)* that may be allocated the same channel as *h(n)* by our flow controller at the *(n+1)*th scheduling. The packets, *h(n)* and $h_i(n+1)$ contain an address and a command to access SDRAM, denoted by (RA_n, BA_n, R/W_n) and ($RA_{n+1,i}$, $BA_{n+1,i}$, $R/W_{n+1,i}$), respectively, where the notations are (row address, bank address, read/write command). Thus, bank conflict, bank interleaving and row-buffer hit are defined as (BA_n=$BA_{n+1,i}$ and RA_n≠$RA_{n+1,i}$), (BA_n≠$BA_{n+1,i}$) and (BA_n= $BA_{n+1,i}$ and RA_n=$RA_{n+1,i}$), respectively. Based on these notations and definitions, algorithm 1 shows how our flow controller works for the GSS, which consists of two parts.

Algorithm 1 Flow Control for GSS

```
1:  if new packet h_k(n+1) comes in router then
2:    for h_i(n+1) ∈ H(n+1) do
3:      t_i ← t_i+1;
4:      if h_k(n+1) is priory packet, h_i(n+1) is best-effort packet and
        BA_{n+1,i}=BA_{n+1,k} then
5:        h_i(n+1) is except from H(n+1);
6:    if h_k(n+1) is priority packet then
7:      t_k ← 2 to 5; // PCT for priority packet
8:    else
9:      t_k ← 1; // for best-effort packet
10: if h(n) is done then
11:   for h_i(n+1) ∈ H(n+1) do
12:     T_i(t_i) in Fig. 2 ← h_i(n+1);
13:     T_i(0) in Fig. 2 ← h_i(n+1);
14:   if SP_PCT = ∅ then
15:     for h_i(n+1) ∈ H(n+1) do
16:       t_i ← t_i+1;
17:     go to line 11;
```

First, any packet (*i*) is given a token (t_i) depending on its input order and priority (line 1-9). Let a new packet, $h_k(n+1)$ come in a router. All old packets are given to one additional token (line 3). Then, if the new packet has a priority, old best-effort packets accessing the same bank as the priority packet are except from *H(n+1)*. It means that best-effort packets which access the same bank as that of the priority packet are not scheduled until the priority packet is scheduled. Then, the new packet gets an initial token. If it is a best-effort packet, one token is given (line 9). Otherwise, any two to five tokens are given (line 9), called PCT (Priority Control Token). If one token is given to the priority packet, it is a priority-equal scheduler and if five tokens are given to the priority packet, it is a priority-first scheduler.

Second, when *h(n)* finishes being delivered, competing packets, $h_i(n+1)$ in a router are scheduled (line 10-17). They are input to Fig. 2 according to the number of token a packet has. That is, if any packet has 1, 2, 3, 4 or 5 tokens, the packet is input to $T_i(1)$, $T_i(2)$, $T_i(3)$, $T_i(4)$ or $T_i(5)$, respectively (line 12). All packets are also input to $T_i(0)$ in line 15. If there is no packet passing the filter (line 14), all packets are given one additional token (line 16) and then filtered again (line 17). Finally, if there are some packets passing the filter, one among the packets is output to SP_{PCT}(Scheduled Packet). If PCT is *n* in line 7, SP_n is used in Fig. 2 where $T_o(t_i)$ is the filtered output of $T_i(t_i)$. SP_n=A?B?C means A is chosen if A is not 0. If A is 0 and B is not 0, B is selected. Finally, if both A and B are 0 and C is not 0, C is chosen.

In our NoC design, normal packets are not scheduled by our GSS flow controller. That is, our GSS flow controller for memory packets and a conventional flow controller for other packets are parallelly performed. Then, two resulting packets are scheduled by a 2-input conventional flow controller. Therefore, other normal packets are not delayed by our flow controller.

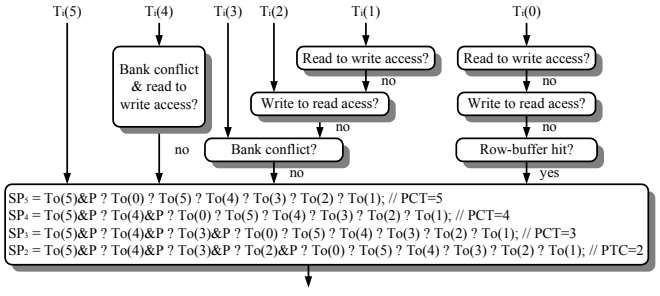

Figure 2: Filtering packets for SDRAM scheduling

3.2 SAGM NoC Design

In NoC designs, it is useful to split a packet into shorter packets since on-chip network resources can be more efficiently reserved. In the newest video system, a length of packets requested by a video encoder/decoder gets shorter whereas a length of packets requested by a video enhancer gets longer. A long best-effort packet causes a priority packet to be further delayed since a priority packet wait until the long best-effort packet finishes transferring. On the other hand, a short packet causes SDRAM utilization to be deteriorated since most SDRAMs always generates longer fixed-length data than the short packet. Thus, the optimal packet length can greatly improve memory performance.

We split a packet to shorter packets considering an SDRAM access granularity. Since our routers communicate through OCP (Open Core Protocol) [8] or AMBA protocol [9] widely used, packets consist of only body flits. Instead, information in head and tail flits is included in additional controls and address buses. Therefore, network loads do not increase whereas the number of a core or a different router interconnected to each router is limited.

DDR I/II SDRAM always transfers/receives fixed-length data per read/write operation. Most memory subsystems prefer a BL8 mode in DDR I/II SDRAM since BL2 and BL4 modes can cause command efficiency to be worse critically. As shown in Fig. 3, let a PRE command for BA1 and a CAS command for BA2 issued to SDRAM at the same time. In Fig. 3(a), the PRE command is performed earlier than the CAS command. Consequently, data of Packet 2 are written with some delays. In Fig. 3(b), the CAS command is performed earlier than PRE command. Consequently, the bank 1 gets idle with some delays. Fortunately, SDRAM can omit a PRE command if a CAS command is executed with AP. Consequently, the PRE command and the CAS command are not delayed as shown in Fig. 3(c).

Under this operation, it is useful that a BL (granularity) of best-effort packets is 2 and a BL mode in DDR I/II SDRAM is set to 4. Now that DDR III SDRAM has a selectable BL4/BL8 OTF mode, it is useful that a BL (granularity) of packet is 4. For example, if a BL of any packet is 9, it is split to five packets which BLs are 2, 2, 2, 2 and 1 for DDR I/II SDRAM and three packets which BLs are 4, 4 and 1 for DDR III SDRAM. It is efficient not only to match the access granularity but also to serve a priority packet faster in winner-take-all bandwidth allocation. If the length of best-effort packet is 9, a priority packet waits until all 9 BLs of the packet are transferred. On the other hand, if it is split to the short packets, a priority packet just waits until 2, 2 and 4 BLs of the short packet are transferred in DDR I, II and III SDRAM, respectively.

To implement this idea, we make a core generate short packets and the last packet has a tag to execute AP. Since the relation of split packets is row-buffer hit, there is no loss of memory utilization. Our router proposed in Section 3.1 prefers row-buffer hit to bank interleaving even if both cause no loss of memory utilization. Therefore, if best-effort packets split do not meet any priority packet, they are delivered successively.

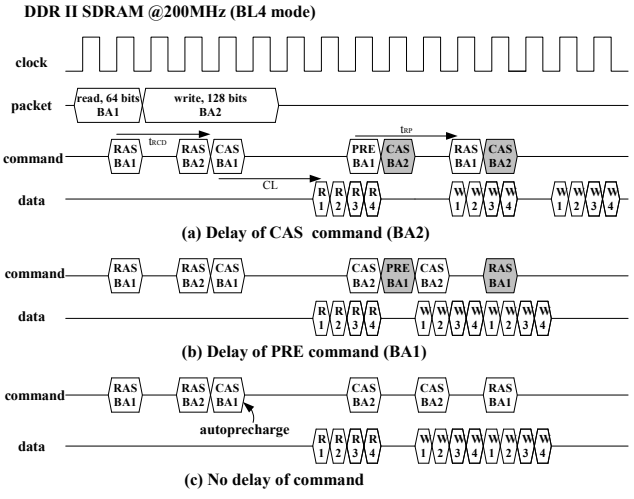

Figure 3: SDRAM Operations when BL=4

Fig. 4 is our memory subsystem only with a memory command generator. It makes DDR SDRAM work for a partially open-page mode. A bank keeps activating (open-page) after an access of packets without any tag indicating the last packet. However, if a bank is accessed by a packet with a tag, the bank is deactivated (closed-page) by AP. In addition, when a priority packet is bank conflict with a previous best-effort packet, the bank may be closed even if the previous best-effort packet has no tag.

A packet that is input to our SDRAM command generator is decoded to extract SDRAM access information such as BA, RA, CA (Column Address), a length of data and a type of command. Then, they are stored in a PRE buffer. A PRE buffer issues a PRE command only when a priority packet meets bank conflict with an open bank accessed. Then, the packet is stored to a RAS buffer. A RAS buffer issues a RAS command if a packet does not have the relation of a row-buffer hit with an open bank accessed. Then, the packet is stored in a CAS buffer. A CAS buffer always issues a read/write command. If a tag is attached to a packet, a command is executed with AP. Finally, an SDRAM command controller schedules all PRE, RAS and CAS commands and generates SDRAM interface signals.

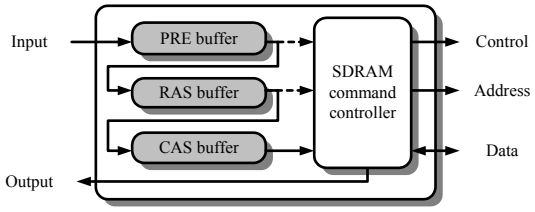

Figure 4: SDRAM Command Generator

4. EXPERIMENTAL RESULTS

Our application-aware NoC design is implemented in a Verilog hardware description language, where DDR I/II SDRAM are set to a BL4 mode and DDR III SDRAM is set to a selectable BL4/ BL8 OTF mode. It is compared to [1] and conventional NoC including a round-robin based NoC router and a full memory subsystem, called CONV. A full memory subsystem employs a design concept from Sonics' MemMax [10] and Denali's Databahn [11] which are an SDRAM scheduler and an SDRAM command generator, respectively. Moreover, a conventional NoC design and [1] use a BL8 mode in a memory subsystem.

All NoC designs are applied to a Blu-ray [7], a single DTV [1] and a dual DTV model, which consist of 9, 9 and 16 subsystems, respectively. They are mapped to a 3×3, 3×3 and 4×4 mesh grid by [12], respectively. All simulations run for one million cycles.

4.1 No Priority Memory Request

Our application-aware NoC design is experimented when there is no priority packet. Our NoC design is implemented to two versions. One is that a GSS flow controller is just employed and the other is that both GSS flow controller and SAGM NoC design are employed, called GSS and GSS+SAGM, respectively. Table 1 shows their memory performance.

Our GSS flow controller achieves slightly better average memory utilization and latency than [1]. On the other hand, the GSS flow controller shows slightly worse latency of demand packets compared to [1]. However, the latency of demand packets is not important since demand packets are not assigned to a priority packet. Our NoC design employing a GSS flow controller and an SAGM design further improves memory performance.

Table 1: Comparison on Benchmarks without Priority Memory Request

Bench mark	DDR DRAM	Memory utilization				Avg. latency of all packets (cycle)				Avg. latency of demand packets (cycle)			
		CONV	[1]	GSS	GSS+ SAGM	CONV	[1]	GSS	GSS+ SAGM	CONV	[1]	GSS	GSS+ SAGM
Blu-ray	133MHz[a]	0.755	0.763	0.771	0.774	121	81	74	69	111	63	65	60
	266MHz[b]	0.651	0.691	0.717	0.761	157	109	101	86	153	91	89	74
	533MHz[c]	0.505	0.592	0.600	0.619	216	134	140	131	216	113	124	113
Single DTV	166MHz[a]	0.717	0.737	0.766	0.776	144	101	86	71	140	80	74	61
	333MHz[b]	0.625	0.673	0.715	0.756	173	120	108	91	171	96	94	77
	667MHz[c]	0.463	0.554	0.577	0.596	244	154	143	140	248	126	127	119
Dual DTV	200MHz[a]	0.696	0.707	0.708	0.712	154	104	89	80	128	73	67	57
	400MHz[b]	0.555	0.627	0.627	0.682	246	149	141	115	196	107	104	85
	800MHz[c]	0.426	0.559	0.531	0.547	364	191	195	184	266	133	144	128
Average		0.599	0.656	0.668	0.691	202	127	120	107	181	98	99	86
Ratio[d]		0.914	1.000	1.018	1.054	1.591	1.000	0.942	0.846	1.847	1.000	1.007	0.878

Table 2: Comparison on Benchmarks with Priority Memory Request

Bench mark	DDR DRAM	Memory utilization				Avg. latency of all packets (cycle)				Avg. latency of demand packets (cycle)			
		CONV +PFS	[1]+ PFS	GSS	GSS+ SAGM	CONV +PFS	[1]+ PFS	GSS	GSS+ SAGM	CONV +PFS	[1]+ PFS	GSS	GSS+ SAGM
Blu-ray	133MHz[a]	0.729	0.742	0.77	0.774	141	106	77	72	97	59	42	38
	266MHz[b]	0.612	0.621	0.699	0.75	176	134	112	96	123	73	72	60
	533MHz[c]	0.454	0.517	0.561	0.608	248	166	151	138	179	88	98	90
Single DTV	166MHz[a]	0.676	0.699	0.755	0.779	163	124	96	76	105	64	57	41
	333MHz[b]	0.58	0.613	0.684	0.738	192	143	116	107	128	74	72	66
	667MHz[c]	0.387	0.489	0.534	0.559	309	182	158	151	213	94	98	95
Dual DTV	200MHz[a]	0.655	0.675	0.7	0.709	183	124	103	80	131	62	55	36
	400MHz[b]	0.521	0.577	0.608	0.657	280	178	153	127	156	81	78	68
	800MHz[c]	0.405	0.481	0.518	0.53	389	252	210	207	198	104	101	99
Average		0.558	0.602	0.648	0.678	231	157	131	117	148	78	75	66
Ratio[d]		0.85	0.917	0.987	1.034	1.821	1.233	1.029	0.922	1.508	0.793	0.763	0.672

[a] DDR I SDRAM [b] DDR II SDRAM [c] DDR III SDRAM [d]Ratio is based on the SDRAM-aware NoC design [1]

Our NoC design is synthesized by DesignVision from Synopsys with an industrial process technology library. The gate count of our NoC design is about 6.2% smaller than [1] in a 3×3 mesh platform. The reason is that a PRE buffer in our SDRAM command generator is smaller than [1] and eight counters per router needed in [1] is not used.

4.2 Priority Memory Request

Our application-aware NoC design is tested when a demand packet is assigned to a priority packet. We implement a conventional NoC design and [1] with a priority-first scheduler, called CONV+PFS and [1]+PFS, respectively. Table 2 shows their memory performance, where the performance ratio is based on [1].

Our application-aware NoC design proves more merits when there is a priority packet on NoC. [1]+PFS improves on average 20.7% latency of priority packets compared to [1]. However, total memory utilization and latency are 8.3% and 23.3% worse than [1]. On the other hand, our GSS flow controller improves on average 23.7% latency of priority packets compared to [1]. Total memory utilization and latency are just 1.7% and 2.9% worse than [1]. Therefore, our GSS flow controller has fewer penalties to support a priority service.

Our (GSS+SAGM)-algorithm improves not only 32.7% latency of priority packets but also 3.4% memory utilization and 7.8% memory latency compared to [1]. It also improves 12.7% memory utilization, 25.2% latency of all packets and 15.2% latency of priority packet on average compared to [1]+PFS.

5. CONCLUSION

We propose an application-aware NoC design for efficient SDRAM access, which includes a flow controller for GSS and an NoC design for SAGM. It greatly improves latency of priority memory requests, memory utilization and latency of all packets in several industrial video systems. In conclusion, our NoC design provides more opportunity for bandwidth-hungry SoC designs with a guaranteed priority service.

6. REFERENCES

[1] W. Jang and D. Z. Pan, "An SDRAM-aware router for networks-on-chip," *Proc. Design Automation Conference*, 2009.

[2] K. Goossens, J. Dielissen and A. Rădulescu, "Æthereal network on chip: concepts, architectures and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sep. 2005.

[3] M. Millberg, E. Nilsson, R. Thid and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," *Proc. DATE*, 2004.

[4] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," *Proc. Design, Automation and Test in Europe*, 2005.

[5] Y.-C. Lan, S.-H. Lo, Y.-C. Lin, Y.-H. Hu and S.-J. Chen, "BiNoC: a bidirectional NoC architecture with dynamic self-reconfigurable channel," Proc. *International Symposium on Networks on chip*, 2009.

[6] "Device operations & timing diagram," *Samsung Electronics*, http://www.samsung.com/global/business/semiconductor.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and System for Video Technology*, vol. 13, no. 7, July 2003.

[8] Open Core Protocol Specification, Release 2.0, 2003, http://www.ocpip.org.

[9] "AMBA open specifications," *ARM*, http://www.arm.com.

[10] "MemMax Scheduler," *Sonics Inc*., http://www.sonicsinc.com.

[11] "Databahn DRAM Memory Controller IP," *Denali Software Inc.*, http://www.denali.com.

[12] W. Jang and D. Z. Pan, "A3MAP: Architecture-Aware Analytic Mapping for Networks-on-Chip," *Proc. ASPDAC*, 2010.