Performance Analysis for MPEG-4 Video Codec Based on On-Chip Network

June-Young Chang, Won-Jong Kim, Young-Hwan Bae,
Jin Ho Han, Han-Jin Cho and Hee-Bum Jung

In this paper, we present a performance analysis for an MPEG-4 video codec based on the on-chip network communication architecture. The existing on-chip buses of system-on-a-chip (SoC) have some limitation on data traffic bandwidth since a large number of silicon IPs share the bus. An on-chip network is introduced to solve the problem of on-chip buses, in which the concept of a computer network is applied to the communication architecture of SoC. We compared the performance of the MPEG-4 video codec based on the on-chip network and Advanced Micro-controller Bus Architecture (AMBA) on-chip bus. Experimental results show that the performance of the MPEG-4 video codec based on the on-chip network is improved over 50% compared to the design based on a multi-layer AMBA bus.

Keywords: SoC design, platform-based design, on-chip network, MPEG-4 codec, on-chip bus.

I. Introduction

As system-on-a-chip (SoC) grows in design complexity, data traffic of IP cores becomes more and more important. Particularly in multimedia SoCs designs, such as video phones, teleconference systems, 3G-324M, MPEG-4, H.264, and HDTV, a considerable amount of data traffic is required. To accommodate all modules with sufficient data traffic bandwidth, an SoC designer should pay attention to on-chip interconnect design [1], [2]. In a platform-based design, forecasting beforehand SoC's data traffic and designing suitable data communication architectures are important.

Several factors such as bus speed, bus width, and bus architectures have a great deal of influence on the performance of the on-chip bus (OCB) architecture. Various types of bus architectures for SoCs are introduced: Advanced Micro-controller Bus Architecture (AMBA) [3], CoreConnect [4], CoreFrame [5], and SiliconBackPlane [6] support the connection of multiple buses in arbitrary topologies. AMBA is classified into two groups: single-layer advanced system bus (ASB) or advanced high-performance bus (AHB)/ advanced peripheral bus (APB), and a multi-layer AHB/APB architecture.

All of the OCB architectures mentioned above, while flexible and relatively inexpensive to implement, appear to have limited scalability due to the arbitrated, non-pipelined nature of their interconnection buses. OCB architectures are suitable for a relatively small number of IP cores on SoC. As the number of IP cores on SoC increases, bus bandwidth will degrade due to the bus collision from multi-masters. The introduction of multiple buses to improve bus bandwidth leads to the power inefficiency [7].

The on-chip network (OCN) is the new communication architecture of SoC design that overcomes the limits of the OCB architecture by providing higher data traffic bandwidth and higher scalability [7]. The OCN architecture provides parallel communication among existing IP cores to improve data traffic bandwidth.

Also the OCN's direct connection feature between IP cores eliminates the need for different interface implementations for different bus widths, which improves scalability of the communication architecture. But one major disadvantage of the OCN is the silicon cost. The complexity of the router and the number of OCN components, FIFOs (first in, first out), switches, and arbiters, increases silicon cost compared to the OCB architectures.

To select the communication architecture for SoC design, a performance analysis is essential. Many works have been done theoretically on OCN [7], [8]. However, not enough researches have yet been carried out on the performance analysis by applying an actual OCN architecture on SoC design and the studies on an OCN-based SoC platform.

This paper presents the results of the performance analysis when applying an MPEG-4 codec on OCN architectures and compares them with those of single/multi-layer AMBA-based architectures. Section II describes the architectures of single/multi-layer AMBA and OCN. The performance analysis on each architecture and experimental results are presented in sections III and IV, respectively.

II. SoC Bus Architectures

In this section, we describe various bus architectures such as a single-layer ASB, multi-layer AHB, and OCN for an MPEG-4 codec design, a typical multimedia application requiring a large amount of data traffic. For a performance analysis of the SoC bus architecture, we used MoVa [9], which is an MPEG-4

Table 1. Specifications of MPEG-4 video codec, MoVa.

Standard	MPEG-4 simple profile @ level2
Performance	Codec: CIF 7.5fps/QCIF 30 fps Decoder: CIF 15fps/QCIP 30 fps
Bit rate	128/133 kbps
Video format	SQCIF/QCIF(176×144)/CIF(352×288)
Technology	0.35 μm 4-metal
Gate count	1,700,000 gates
Chip size	110.25 mm^2
Op. frequency	27 MHz

codec system implemented by a single-layer ASB/APB. Its design specifications are shown in Table 1.

1. Single-Layer Bus Architecture

The block diagram of the MPEG-4 codec implemented on the single-layer bus (SLB) architecture is depicted in Fig. 1. The bus master ARM7TDMI processor can be programmed to process various video algorithms, for example, MPEG-4. It also fetches instructions from an on-chip memory (IntMem), executes them, and sets the control register for slave. The ASB bus consists of an arbiter, a decoder, and a bridge. Each element controls the bus master arbitration, the module selecting signal generation by address decoding, and the connection between two modules in sequence.

The arbiter determines access rights of masters to the bus: an ARM or a direct memory access controller (DMAC). The decoder runs on a centralized address-decoding module, which generates a selection signal for each slave on the ASB bus. A bridge is the only bus master on the APB bus for peripheral slaves. The MPEG-4 codec hardware module comprises a codec hardware module (core module) and video input/output hardware module (I/O module).
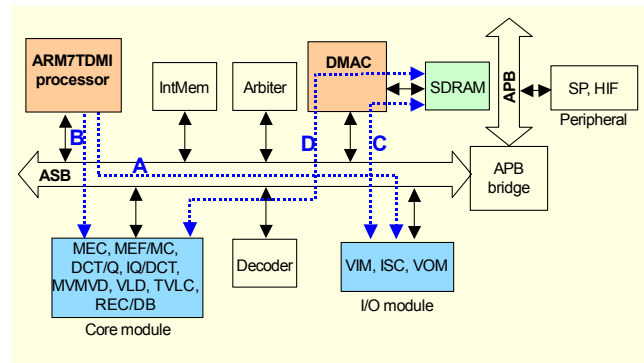

Fig. 1. Single-layer bus architecture.

In the SLB architecture, the basic bus operations of two bus masters, ARM processor and DMAC, are classified as follows:

[A operation]: initialization of I/O modules by ARM processor.

[B operation]: initialization of core modules by ARM processor.

[C operation]: data transfer operation between SDRAM and I/O modules by the DMAC.

[D operation]: data transfer operation between SDRAM and core modules by the DMAC.

In the SLB architecture, while one master uses a physical bus the other master cannot use it. While the ARM processor initializes the control register of the hardware module, the

DMAC is supposed to wait for access to the bus, which results in performance degradation of the bus.

2. Multi-layer Bus Architecture

The multi-layer bus (MLB) architecture, based on the AHB protocol, consists of multiple physical buses and enables parallel access paths between multiple masters and slaves. This gives us the benefit of increased bandwidth on overall buses. Each master has its own AHB layer and is connected to the slave by an interconnection matrix (BusMatrix). Since each AHB layer has only one master, master-to-slave muxing is required.

In Fig. 2, the MLB architecture consists of a 3-layer AHB bus: system AHB, core AHB, and I/O AHB, each having its own master, an ARM processor, core DMAC and I/O DMAC. The system AHB layer is connected to the ARM processor, arbiter, decoder, and bridge of the AHB. The ARM processor controls the core and I/O modules by initializing the control register of the core and I/O modules.

The arbiter manages bus master arbitration. The decoder generates a slave module selection signal by address decoding. The I/O DMAC transfers data from SDRAM to the I/O module, and vice versa, on an I/O AHB layer, as does the core DMAC between the SDRAM and core module on the core AHB layer.

In the MLB architecture, while the ARM processor sets the control register of the slave hardware module, the DMAC enables a data transfer between the SDRAM and slave module. In the case of setting the control registers of the core module, the I/O DMAC enables a data transfer between the SDRAM and I/O module through the I/O AHB layer. Also, during the ARM processor's setting of the control registers of the I/O
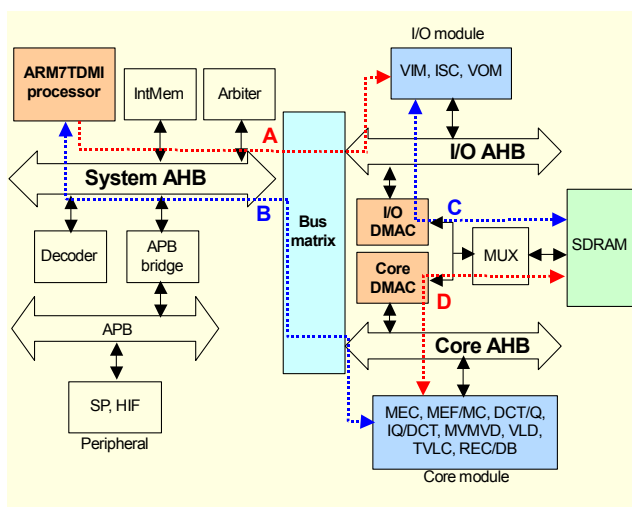
module on the I/O AHB layer, the core DMAC transfers data between the SDRAM and core module on the core AHB layer. Therefore, the MLB architecture increases both the parallel operation and the utilization of the bus, and consequently improves bandwidth of the bus compared to that of the SLB architecture.

3. On-Chip Network Architecture

The OCN architecture is composed of a master network interface (MNI), slave network interface (SNI), and crossbar switch. The MNI generates packets using control data from the ARM processor or DMAC and transfers them to a non-blocking crossbar switch.

The SNI transfers packets from the master to the slave module. The crossbar switch routes packets and works a channel function for a data transfer between the master and slave modules. The OCN architecture in Fig. 3 illustrates a design for improving bandwidth of the bus by increasing the parallel operation of the MLB. The bus master, like the ARM processor or DMAC, can elevate system performance by transferring data without latency. In an OCN, while the ARM processor sets the control registers of the slave module, the DMAC can transfer data to/from slave modules without latency. This is a featured structure of an OCN, which was hard to provide in the MLB.

In the OCN architecture, we can increase the number of parallel operations through separating clusters from the OCN-cluster (OCN-C) split architecture. As shown in Fig. 4, if we split the local buffer from the hardware module of the MPEG-4 codec, we can enhance parallel operations. This results in an improvement of bus bandwidth by the following procedure: first, separating the operation of setting the control register of

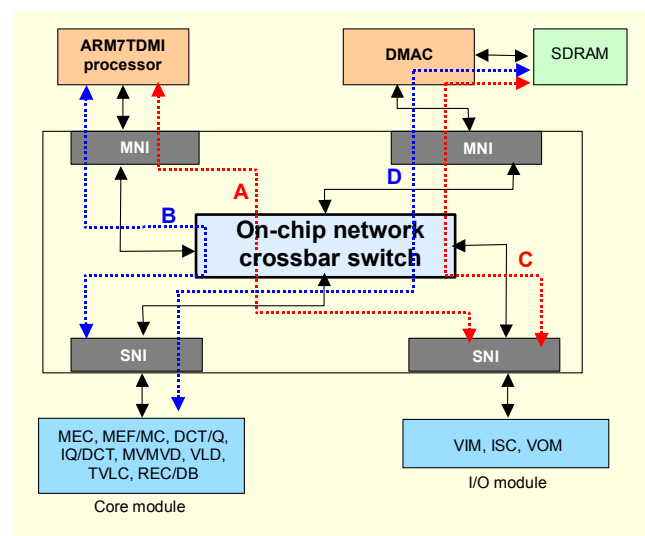Fig. 2. Multi-layer bus architecture.

Fig. 3. On-chip network architecture.

the slave module (A) and data transfer through the DMAC (B), and second, executing a parallel operation.

In the case of the MPEG-4 encoder, slave modules that transfer data to the SDRAM through the DMAC can be a video input module (VIM), motion estimation course (MEC), motion estimation fine/motion compensation (MEF/MC), and reconstruction (REC) modules. We split the local buffer of the hardware modules related to those and assign them to a local buffer cluster. This technique makes parallel operations more possible and improves system throughput.
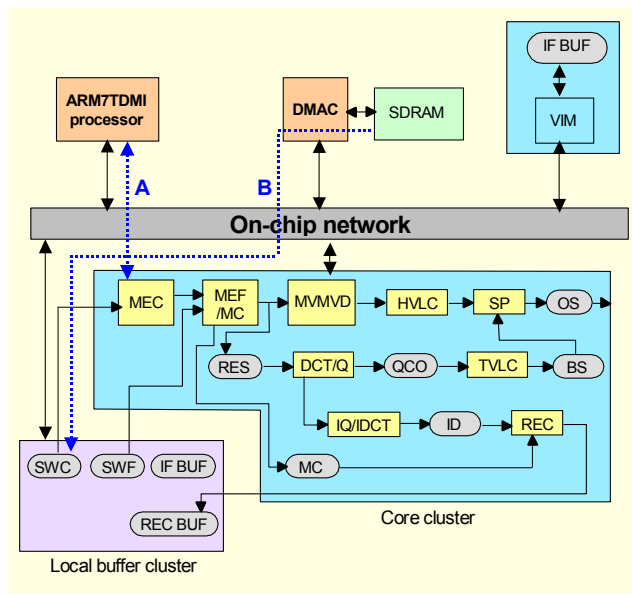

Fig. 4. OCN-cluster split architecture.

III. Performance Analysis

In this chapter, we describe the method and results of our performance analysis on the MPEG-4 codec according to respective architectures of data communication. We use bus speed, bus width, bus architecture, and parallel operation based on bus architecture as parameters for the analysis. In the basic operation of the OCN architecture as shown in Fig. 3, the possible cases for parallel operation are {A,C}, {A,D}, {B,C},

Table 2. Parallel operation table of bus architectures.

Parallel operation	SLB	MLB	OCN
{A,C}	X	X	O
{A,D}	X	O	O
{B,C}	X	O	O
{B,D}	X	X	O

and {B,D}.

Table 2 summarizes the possible combinations for parallel operation according to respective bus architectures. Since a large number of masters share a common bus in the OCB, parallel operations such as {A,C} and {B,D} are impossible. In these cases, while one master is using the bus, the others have to wait for its termination. This consequently degrades bus bandwidth. A large number of masters transmit packet type data through a crossbar switch spread as a network in an OCN. In this case, parallel operations such as {A,C} and {B,D} are possible. The more parallel operations increase, the better bus bandwidth improvement can be achieved.

For the performance analysis, operation cycles, dependence, the number of pipeline stages, and possible parallel operations on bus architectures are input variables. We can compute a maximum cycle count for processing a macro block (MB) by an MB-based pipeline scheduler. Then, we analyze the performance of the bus architecture based on the maximum number of cycle counts.

1. Performance Analysis of SLB Architecture

Table 3 shows the cycles of each operation obtained from the simulation of MoVa [9], which is an MPEG-4 codec system implemented with an SLB architecture. The hardware cycle of SLB is the maximum cycle of the MPEG-4 hardware module. None of the SLB parallel operations {A,C}, {A,D}, {B,C}, or {B,D} are possible. In the case of {A,C} of the SLB architecture in Fig. 1, an I/O module cannot transfer data to the SDRAM through the DMAC, while the ARM processor uses buses for setting the control register of the I/O module. Since the DMAC has to wait during the ARM processor's setting of the control register of the I/O module, bus bandwidth degrades. The SLB scheduling results for processing the MB show that the encoder runs 4,545 cycles with a 4-stage pipeline and that the decoder does 3,519 cycles with a 3-stage pipeline.

2. Performance Analysis of MLB Architecture

The MLB architecture uses a multi-layer AHB/APB that has a 32-bit data width and system clock extendible to 54 MHz. As bus width is extended to 32-bit, the operation cycle for a data transfer by the DMAC shrinks to 1/2. For the hardware cycle applied to the MLB in Table 3, we use an average execution time of cycles utilized when MoVa processes three video frames. Since the BusMatrix splits the I/O AHB and core AHB in an MLB architecture as shown in Fig. 2, parallel operations such as {A,D} and {B,C} are possible.

Therefore, we can reduce the number of operations for

Table 3. Operation cycles of bus architectures.

MPEG-4 encoder						MPEG-4 decoder					
		SLB	MLB	OCN	OCN-C			SLB	MLB	OCN	OCN-C
Operation	Type	Number of cycles				Operation	Type	Number of cycles			
SWC0Init	FW	39	39	39	0	PSBUFReadInit	FW	46	46	46	0
SWC0	DMA	115	58	58	58	PSBUFRead	DMA	324	162	162	162
SWC1InitPre	SW	83	83	83	83	VLDInit	FW	67	67	67	0
SWC1Init	FW	50	50	50	0	VLD	HW	2,240	173	173	173
SWC1	DMA	299	150	150	150	SWF1YInit	FW	130	130	130	0
IRDecision	SW	30	30	30	30	SWF1YI	DMA	109	55	55	55
MECInit	FW	96	96	96	0	SWF1Y2InitPre	SW	85	85	85	85
MEC	HW	2,600	2,156	2,156	2,156	SWF1Y2Init	FW	50	50	50	0
IFWriteInit	FW	157	0	0	0	SWF1Y2	DMA	109	55	55	55
IFWrite	DMA	545	273	273	0	SWF1Y3InitPre	SW	80	80	80	80
ISWriteInit	FW	46	0	0	0	SWF1Y3Init	FW	50	50	50	0
ISWrite	DMA	60	30	30	0	SWF1Y3	DMA	109	55	55	55
MECPost	FW	34	34	34	0	SWF1Y4InitPre	SW	85	85	85	85
SWF0Init	FW	78	78	78	0	SWF1Y4Init	FW	50	50	50	0
SWF0	DMA	349	175	175	175	SWF1Y4	DMA	109	55	55	55
SWF1YInitPre	SW	70	70	70	70	SWF1UInitPre	SW	75	75	75	75
SWF1YInit	FW	50	50	50	0	SWF1UInit	FW	50	50	50	0
SWF1Y	DMA	299	150	150	150	SWF1U	DMA	109	55	55	55
SWF1UInitPre	SW	77	77	77	77	SWF1VInitPre	SW	79	79	79	79
SWF1UInit	FW	50	50	50	0	SWF1V	DMA	109	55	55	55
SWF1U	DMA	109	55	55	55	SWF1VInit	FW	50	50	50	0
SWF1VInitPre	SW	77	77	77	77	ISWriteinit	FW	46	0	0	0
SWF1VInit	FW	50	50	50	0	ISWrite	DMA	60	30	30	0
SWF1V	DMA	109	55	55	55	VLDPost	FW	77	77	77	0
MEFMCInit	FW	109	109	109	0	IQIDCTQInit	FW	72	72	72	72
MEFMC	HW	2,500	1,805	1,805	1,805	IQIDCTQ	HW	1,200	997	997	997
MEFMCPost	FW	74	74	74	0	MVMVDInit	FW	40	40	40	0
MVMVDInit	FW	50	50	50	0	MVMVD	HW	150	76	76	76
MVMVD	HW	192	76	76	76	MCInit	FW	140	140	140	140
HVLC	SW	130	130	130	130	MC	HW	700	700	700	700
PreRC	SW	300	300	300	300	RECInit	FW	35	35	35	0
DCTQInit	FW	72	72	72	72	REC	HW	400	572	572	572
DCTQ	HW	1,200	1,051	1,051	1,051	OFWriteInit	FW	179	0	0	0
IDCTQ	HW	1,200	997	997	997	OFWrite	DMA	434	217	217	217
TVLCInit	FW	34	34	34	34	RECWriteInit	FW	83	83	83	0
TVLC	HW	1,300	231	231	231	RECWrite	DMA	342	171	171	171
TVLCPost	FW	24	24	24	24	DBReadInit	FW	43	43	43	0
PostRC	SW	340	340	340	340	DBRead	DMA	79	40	40	40
SPInit	FW	103	103	103	103	DBInit	FW	33	33	33	0
SP	HW	114	64	64	64	DB	HW	1,200	1,200	1,200	1,200
RECInit	FW	35	35	35	0	DBWriteInit	FW	84	84	84	84
REC	HW	400	572	572	572	DBWrite	DMA	293	147	147	147
RECWriteInit	FW	83	83	83	0	PostDB	HW	200	200	200	200
RECWrite	DMA	342	171	171	171						

setting the control register of the I/O module and cycles related to data transfer by the I/O DMAC. From the scheduling with features of an {A,D} and {B,C} parallel operation, we find that the encoder runs on 3,510 cycles with a 4-stage pipeline and that the decoder does 2,100 cycles with a 3-stage pipeline.

3. Performance Analysis of OCN Architecture

In the OCN architecture depicted in Fig. 3, the DMAC can transfer data to/from slave modules without bus latency while the ARM processor sets the control registers of the slave module. That is, parallel operations {A,C} and {B,D}, which were impossible in the MLB, are able to work. All the parallel operations such as {A,C}, {A,D}, {B,C}, and {B,D} are possible. Since the operation of setting the control registers of the slave module and a data transfer through the DMAC operate simultaneously, we can reduce overall cycle time and get results showing that the encoder runs on 2,800 cycles with a 4-stage pipeline, while the decoder does 1,900 cycles with a 3-stage pipeline.

By splitting the hardware module and local buffer with an OCN-cluster split architecture, it is possible to reduce the operation cycle setting the control registers of the slave module, and consequently to enhance bus bandwidth. For OCN-C scheduling when processing an MB, the encoder runs on 2,300 cycles with a 4-stage pipeline and the decoder does 1,300 cycles with a 3-stage pipeline

IV. Experimental Results

Table 4 shows a performance comparison of bus architectures. The number of frames per second (fps) on an identical frequency is used for performance comparison.

We find that 7.6 frames are processed in 27 MHz with the codec mode in an SLB. In the case of an MLB architecture, the performance increases up to 22 fps since we use AHB operating at 54 MHz. If we execute the SLB and MLB architectures with the same 27 MHz, we can get a 31.8% higher performance in the MLB than in the SLB architecture. We can process 34.4 fps with 54 MHz in the OCN-C architecture, which is a 56.4% improvement in performance compared to the MLB.

The OCN-C architecture enhances the performance 31.3% higher than the OCN architecture. This implies that the manner of splitting clusters operating in parallel in OCN greatly affects the resulting performance. The ascending order of performance in the MPEG-4 codec based on the number of frames possible to process per second is as follows: SLB, MLB, OCN, and OCN-C. Figure 5 summarizes the performance results for them.

Table 4. Operation cycle of bus architectures.

Bus type	Coding mode	Maximun cycle/MB	Execution time/frame	Frame rate (fps)	
				27 MHz	54 MHz
SLB	Encoder	4,545	70.0 ms	14.3	28.6
	Decoder	3,519	61.5 ms	16.3	32.6
	Codec		131.5 ms	7.6	15.2
MLB	Encoder	3,510	54.0 ms	18.5	37
	Decoder	2,100	36.7 ms	27.3	54.6
	Codec		90.7 ms	11	22
OCN	Encoder	2,800	43.1 ms	23.2	46.4
	Decoder	1,900	33.2 ms	30.1	60.2
	Codec		76.3 ms	13.1	26.2
OCN-C	Encoder	2,300	35.4 ms	28.2	56.4
	Decoder	1,300	22.7 ms	44	88
	Codec		58.1 ms	17.2	34.4
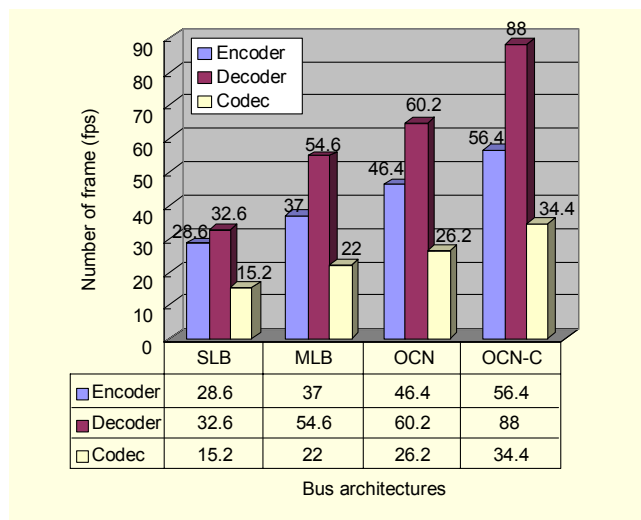
Fig. 5. Performance comparisons of MPEG-4 video codec.

	SLB	MLB	OCN	OCN-C
Encoder	28.6	37	46.4	56.4
Decoder	32.6	54.6	60.2	88
Codec	15.2	22	26.2	34.4

V. Conclusion

This paper describes a performance analysis for applying an MPEG-4 codec on an on-chip network and compares it with the performance of a single/multi-layer AMBA. In the MPEG-4 codec, the OCN-cluster split architecture gets the highest performance over the single/multi-layer AMBA and OCN architectures. In particular, the OCN-cluster split architecture enhances the performance by 56.4 and 31.3% compared to multi-layer AMBA and OCN architectures, respectively. This implies that the manner of splitting clusters operating in parallel in an OCN greatly affects the resulting performance. The OCN architecture is highly recommended in multimedia applications

that require large amounts of data traffic and high communication complexity such as HDTV and digital multimedia broadcasting [10].

References

[1] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L.Todd, *Surviving the SOC Revolution: A Guide to Platform-Based Design*, Kluwer Academic Publishers, ARM Ltd, Nov. 1999.

[2] Kyeong Keol Ryu, Eung Shin, Mooney, and V.J., "A Comparison of Five Different Multiprocessor SoC Bus Architectures," *Proc. of the EUROMICRO Symposium on Digital. Systems Design*, Sept. 2001, pp. 202-209.

[3] *AMBA Specification Rev 2.0m*, Document Number ARM IHI 0011A.

[4] CoreConnect Bus Architecture, http://www.chips.ibm.com/products/coreconnect.

[5] B. Cordan, "An Efficient Bus Architecture for System-on-Chip Design," *Proc. IEEE 1999 Custom Integrated Circuits Conf.*, May 1999, pp. 623–626.

[6] SiliconBackplane Bus Architecture, http://www.sonicsinc.com/sonics/products/siliconbackplaneIII.

[7] L. Benini and G. Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computers*, Jan. 2002, pp. 70-78.

[8] Se-Joong Lee et al., "An 800MHz Star-Connected On-Chip Network for Application to System on a Chip," *IEEE ISSCC Dig. Tech. Papers*, Feb. 2003, pp. 468-469.

[9] Seong-Min Kim, Ju-Hyun Park, Seong-Mo Park, Bon-Tae Koo, Kyoung-Seon Shin, Ki-Bum Suh, Ig-Kyun Kim, Nak-Woong Eum, and Kyung-Soo Kim, "Hardware-Software Implementation of MPEG-4 Video Codec," *ETRI J.*, vol.25, no.6, Dec. 2003, pp.489-502.

[10] Bong-Ho Lee, Kyu-Tae Yang, Young Kwon Hahm, Soo In Lee, and Chieteuk Ahn, "A Framework for MPEG-4 Contents Delivery over DMB," *ETRI J.*, vol.26, no.2, Apr. 2004, pp.112-121.

June-Young Chang received the BS degree in computer science from Chonnam National University in Gwangju, Korea, in 1985, the MS degree in computer science from Chungang University in Seoul, Korea, in 1987 and the PhD degree in computer science from Chonnam National University in 1996. He joined ETRI in 1999 as a Senior Member. His current research interests include VLSI/CAD, SoC design methodology and multimedia SoC design.

Won-Jong Kim received the BS degree in electronics engineering from Chonnam National University in 1989. He received the MS and PhD degrees in electronics engineering from Hanyang University in 1992 and 1999. He joined ETRI in 2000 as a Senior Member. His research interests include CAD for VLSI, SOC design methodology, and multimedia SOC design.

Young-Hwan Bae was born in Seoul, Korea on October 29, 1962. He received the BS and MS degrees in electronic engineering from Hanyang University in 1985 and 1987. He joined ETRI in 1987, where he works in developing CAD tools and design methodology for SOC.

Jin Ho Han was born in Korea on Feb 8, 1977. He received the BS and MS degrees in electronic engineering from Korea Advanced Institute of Science and Technology in 1998 and 2001. He joined ETRI in 2001, where he currently works in low power embedded processor design and SOC design methodology development as a project member.

Han-Jin Cho was born in Seoul, Korea on July 8, 1960. He received the BS degree in electronic engineering from Hanyang University in 1982. He received the MS and PhD degrees in electrical engineering from the New Jersey Institute of Technology in 1987 and the University of Florida in 1992, respectively. He joined ETRI in 1992, where he currently works in SOC design methodology development and wireless multimedia SOC design as a project manager.

Hee-Bum Jung received the BS degree in electronics engineering from Sogang University, Seoul, Korea in 1981, the MS degree in electrical engineering from Korea Advanced Institute of Science and Technology in 1983, and the PhD degree in electrical engineering from Columbia University, New York, NY, USA in 1992. From 1983 to 1987 he was with KIET (formerly ETRI), Gumi, Korea where he was involved in the development of custom integrated circuit design and modeling. While in Columbia University, he was also a student intern (1990 to 1991) of AT&T Bell Laboratory, Murray Hill, NJ, U.S.A. with the research topic of HBT modeling. He rejoined ETRI in 1993 to pursue VLSI circuit design, and is currently directing the SoC design research department. His interests include platform-based SoC design methodology and low power SoC design for portable IT equipment.