MPEG-based Performance Comparison between Network-on-Chip and AMBA MPSoC

Rishad A. Shafik, Paul Rosinger, Bashir M. Al-Hashimi
School of ECS, University of Southampton, Southampton, UK, SO17 1BJ
Email: *ras06r@ecs.soton.ac.uk, pmr@ecs.soton.ac.uk, bmah@ecs.soton.ac.uk*

***Abstract*—Using analytical and simulation results, this paper presents comparative analyses between network on chip and shared-bus AMBA using real application traffic with MPEG-2 video decoder in cycle-accurate realistic simulation environment. We show that despite higher channel latency, NoC has higher bandwidth advantage and outperforms shared-bus AMBA, requiring lower frequency in order to decode the video bitstream at given frame rate.**

I. Introduction

Network-on-chip (NoC) is emerging as a viable on-chip communication infrastructure for multi-processor system on chip (MPSoC) [1]. Video streaming is seen as a key feature of future MPSoC and MPEG video decoding is a major component of these systems [2]. To date there has been good progress in developing flexible NoC architectures (such as [3]) and efficient routing algorithms in terms of contention avoidance and energy consumption (such as [4]). For the NoC methodology to gain further maturity, comparative studies between shared and segmented-bus topologies and NoC need to be performed with the aim to identify the benefits and shortcomings of each approach. To this end, a number of such studies have recently been reported. An MPEG-4 based performance analyses of single core on-chip networks are presented in [5]. Analytical cost analyses involving area, power, frequency, throughput, latency and energy of NoC and bus-based architectures are presented in [6], [7]. Review of guiding principles towards the evolution of NoC as an emerging SoC architecture is presented in [8]. A comparative evaluation between P2P and NoC with MPEG-2 video encoder is carried out in [2] considering area, power, data parallelism, MPEG frame rate and scalability. Analytical comparisons involving shared-bus, P2P and NoC architectures were reported in [9], [10] considering power and energy consumption, and overall design effort.

Most of the comparisons reported between NoCs and shared bus SoCs, as in [7], [6], use analytical and synthetic traffic patterns. The comparisons presented in [11] are based on real application but it does not consider application performance. Using analytical and simulation results, this paper presents comparative analyses between NoC and shared-bus AMBA for multi-processor application using *real application traffic* with MPEG-2 video decoder in cycle-accurate realistic simulation environment. The aim of this comparison is not to demonstrate which bus-based or NoC architecture or mapping performs best, rather this paper investigates the application-specific performance comparison between popular and generic shared-bus AMBA and NoC MPSoC architectures.

The rest of the paper organisation follows. Section II describes the design of NoC and AMBA using MPEG-2 video decoder cores. Section III defines the comparison metrics and evaluate them through cycle accurate realistic simulations. Finally, Section IV concludes the paper.

II. System Design

A. MPEG-2 Video Decoder Cores

In this work, we have developed a MPEG-2 video decoder that employs five cores. The application partitioning is done in line with [12] and no attempt has been made to optimise the partitioning. A block diagram of the multi-processor MPEG-2 video decoder used in this work is shown in Fig. 1. The input

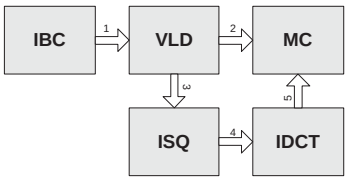

Fig. 1. Block diagram of MPEG-2 video decoder used in this work

buffer controller (IBC) core stores and forwards the original video bitstream to the variable length decoder (VLD) core, which organises the bitstream into two sequences: header and video sequence. The quantisation matrices and macroblocks (MBs) are sent to inverse scanner and quantiser (ISQ) core, while header and motion specific information are sent to the motion compensator (MC) core. The core ISQ sends the DCT coefficients to the inverse discrete cosine transformer (IDCT) core, which transforms them into actual time domain format through in a lossy manner. The picture-ready blocks from IDCT are sent to MC core, which forms predictions, organises and stores decoded video.

Each IP core has a dedicated local memory of $32768\ (1024 \times 32)$ bits, which is large enough to contain processed DTUs of the previous core until it is processed. The memory is directly interfaced with the input port by memory access controllers. Output port is connected to the processor. Optional control and credit signals are also connected for compatibility. A simplified block diagram of an IP core is shown in Fig. 2.

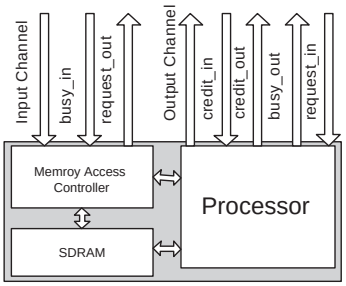

Fig. 2. Simplified block diagram of an IP core

B. Shared-bus AMBA

The AMBA protocol [13] is an open standard, on-chip bus specification. Three distinct buses are defined in AMBA: advanced high-performance bus (AHB), advanced system bus (ASB), and advanced peripheral bus (APB). Due to its high performance and high clock-frequency, AHB has been a dominant shared-bus technology [8] and is chosen in this work. AHB has been used in similar comparisons in [8], [5].

Employing the AMBA AHB, a single-layer central multiplexor configuration with pipelined transaction is used in this work. MPEG-2 video decoder cores are configured by using the input port as slave port and output port as master port (Fig. 3). Single burst transfer without waiting states and 32-bit payload of each data transaction unit (DTU) are used. The cores share access to the bus in the sequence of IBC, VLD, ISQ and IDCT (MC reads from local memory and processes only). The duration for which a master core can hold the interconnect access in AMBA depends on the types of data being processed, for example 4 clock cycles for header processing, 8 clock cycles for encoded macroblocks, 4 clock cycles for skipped macroblocks.

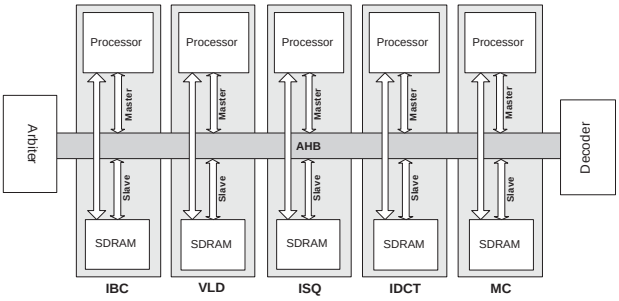

Fig. 3. Block diagram of AMBA with MPEG-2 decoder cores

C. Network on Chip

Even though application-specific NoCs outperform general purpose NoCs, the designs of such NoCs vary depending on the application, partitioning among multiple cores and resource allocations [14]. Our aim, in this work, is to compare the performance without restricting the architecture to the application itself. Hence, a general purpose architecture for NoCs is preferred. Due to their simplicity, performance and scalability [15], mesh-based topology with deterministic XY routing are considered with shortest path mapping between communicating cores. Each NoC packet has 32-bit payload. Single-flit-packet *wormhole* routing [14] is considered due to its popularity [5].

The basic topological element of the NoC structure is the tile, which has three major parts: processing element (PE), network interface (NI) and switch. The MPEG-2 decoder cores are plugged into the PE (Fig. 2) and responsible for computation. The switches carry out the communication tasks. Every switch in an NoC lays out five input and five output ports and credit information to and from each port. A generic switch architecture is shown in Fig. 4. Each transaction is

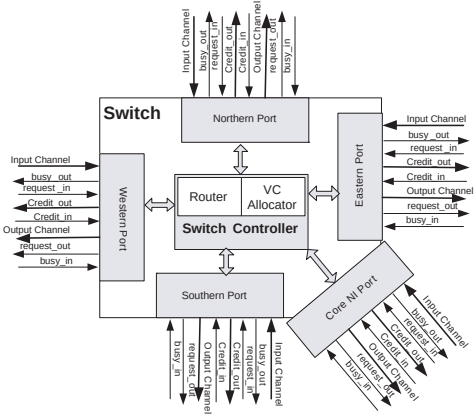

Fig. 4. Block diagram of switch for mesh-topology NoC

initiated with handshake signals: *busy* and *request*, followed by credit signals with current status of the virtual channel (VC) buffers. Input channel has buffers for eight incoming flits (flow control units) as they arrive and hence no congestion takes place for single flit injection per clock cycle. Flits in different virtual channel-buffers are served by VC allocator in a round-robin fashion to give fair waiting times for flits in different input channel ports. The router decides for the outgoing port for a packet depending on header information. NI decouples between communication and computation by incorporating necessary translations that need to take place in order to make the core architecture compatible for packet-based communication in NoC.

A simplified block diagram of general purpose (3×3) NoC employing shortest path mapping of MPEG-2 cores and pipelined 32-bit packet based NI architecture is shown in Fig. 5.

III. Simulations and Comparative Analyses

A. Simulation Setup and Test Cases

To compare the MPEG-based performance between NoC and AMBA, two SystemC simulators were employed. The basic simulation setup for NoC and AMBA used to perform the comparisons are briefly described below:

1) The MPEG cores (Fig. 1) are configured for NoC using NI and for AMBA using port configurations (Fig. 3).

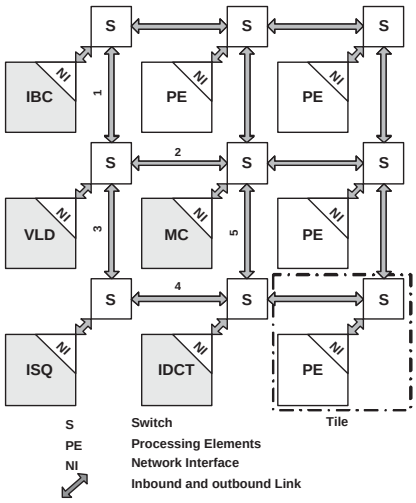

Fig. 5. Block diagram of (3×3) tile-based NoC with MPEG cores

2) Transaction level modeling is used in both NoC and AMBA for data communication, while behavioural modeling is used for MPEG-based data computation/processing.
3) To track simulation and performance parameters, separate monitor modules were implemented with each core.

The MPEG-based performance comparisons between NoC and AMBA are carried out using five different MPEG-2 video bitstreams for our cycle accurate simulation model as shown in Table I. The bitstreams are chosen with increasing number of frames and encoded macroblocks, giving different compressions for the encoded video bitstreams, corresponding processing times involved between encoded and skipped macroblocks.

TABLE I
TEST VIDEO BITSTREAMS

Bitstream	No. of Frames	Encoded MBs	Picture Size (Pixels)	Frame Rate (fps)
test1.m2v	55	1570	252x288	25
test2.m2v	63	1791	352x240	25
test3.m2v	68	1817	320x288	25
test4.m2v	211	5362	352x288	30
test5.m2v	323	7841	480x480	30

B. Performance Metrics

To better understand the underlying application performance, we use different metrics that contribute to the performance analysis of the MPEG-2 decoder to compare between NoC and shared bus AMBA. To understand multi-core performance, we define core concurrency and core efficiency and to understand interconnect performance, we define channel latency and bandwidth. Later, the performance of the application in NoC and AMBA are compared.

1) Concurrency: Concurrency defines the number of cores that are able to execute computation at the same time and is dependent on the way IP cores communicate with each other. Higher degree of concurrency effectively reduces the total application time through overlap of executions among cores. The fact that the MPEG cores in NoC are connected through dedicated physical links (Fig. 5), providing it spatial multiplexing of different channels, it is expected to exhibit higher level of concurrency in video decoding among cores, unlike AMBA, where all the cores share a common physical link in time-multiplexed way (Fig. 3). In order to investigate the actual concurrency from the simulations, we define the average degree of concurrency, D, as

$$D = \frac{\sum_{t=1}^{T_A} N(t)}{T_A}, \tag{1}$$

where T_A is the total application time (in clock cycles) and $N(t)$ is the number of cores executing the computation at t-th clock cycle. Given, $N(t) \leq N_{max}$, where N_{max} is the total number of cores, it can be shown that, $D_{max} = N_{max}$.

In Table II, a tabular comparison of core concurrency between NoC and AMBA is shown recorded from the simulation logs. As shown in the Table II (Column 3), for video bitstream $test1.m2v$, $T_A = 4093102$ clock cycles, during which the application is executed with $N = 1$ core for 492746 clock cycles, with $N = 2$ cores for 481524 clock cycles, with $N = 3$ cores for 436943 clock cycles and with $N = 4$ cores for 214020 clock cycles. Hence, T_A in AMBA is made up mostly with $N = 1$, 2 and 3 cores. With the application times and concurrent execution times obtained from Table II and using the Equation 1, D_{AMBA} for $test1.m2v$ is given as 0.89. On the other hand, due to dedicated links among cores, NoC is able to exploit the concurrency at a higher level. For video bitstream $test1.m2v$, while only 12 clock cycles are executed with single core, $N = 2$, 3, 4 and 5 cores execute for 839, 337763, 388561 and 210696 clock cycles (mostly made up of $N = 3$, 4 and 5 concurrent core executions), respectively (Table II, Column 2). Thus, NoC allows more overlap among core execution and reduces T_A to only 1198080 clock cycles and gives NoC a higher D_{NoC} of 3.05 according to Equation 1.

Referring to Table II, with higher number of macroblocks in other video bitstreams, concurrent execution times for different N increase but the degree of concurrency almost remains same due to inter-dependent processing of macroblocks among VLD, ISQ, IDCT and MC (Section II-A). The degree of concurrency are found by simulated concurrent execution times in Table II and using the Equation 1 for bitstreams $test2.m2v$, $test3.m2v$, $test4.m2v$ and $test5.m2v$ as 3.04, 3.05, 3.09 and 3.05, respectively for NoC and as 0.88, 0.88, 0.88 and 0.88, respectively for AMBA. NoC maintains a higher average degree of concurrency at $D_{NoC} = 3.05$, compared to $D_{AMBA} = 0.88$. Due to overlap of core execution, on average T_A for NoC is reduced to approximately 29% of T_A for AMBA. Hence, it is evident that NoC suits MPSoC architectures, where concurrent processing is desirable.

2) Core Efficiency: Core efficiency defines how efficiently the cores can utilise the execution cycles within the application

TABLE II
CORE CONCURRENCY OF NOC AND AMBA FOR DIFFERENT VIDEO BITSTREAMS

	test1.m2v		test2.m2v		test3.m2v		test4.m2v		test5.m2v	
N	NoC	AMBA	NoC	AMBA	NoC	AMBA	NoC	AMBA	NoC	AMBA
1	12	492746	12	573553	12	565186	12	1181715	12	1547034
2	839	481524	2278	571108	3964	580423	4371	1783051	9431	2519739
3	337763	436943	357264	516620	329721	519778	1113725	1585839	1205932	2284629
4	388561	214020	385174	254628	377258	255051	1101221	797316	1214726	1261325
5	210696	0	333417	0	359887	0	987957	0	1998025	0
T_A	1198080	4093102	1407726	4850733	1413258	4881477	4113964	14389936	6070048	21053545
D	3.02	0.89	3.04	0.88	3.05	0.88	3.09	0.88	3.05	0.88

time. Total execution time for an IP core is given as

$$T_E = T_P + T_{NP} \leq T_A, \tag{2}$$

where T_E is the execution time, T_P is the total processing time and T_{NP} is the total non-processing time, all in clock cycles. Non-processing time, T_{NP} is defined as

$$T_{NP} = T_R + T_W + T_I, \tag{3}$$

where T_R is the reading time of the input data at the IP core and T_W is the time required for output data to be written at the output ports and are both 0 clock cycle due to local memory and pipelined output writing, T_I is the idle time and is the major contributor to non-processing time. The idle time, T_I, is caused by i) not having interconnect access due to mutually exclusive sharing even though processed data is available for writing, or ii) not having enough data to process. Due to dedicated links among cores in NoC, there is no waiting time for interconnect unavailability, unlike AMBA. In order to investigate quantitatively how effectively the computation cycles are being utilised within execution time of a core from simulations, we define core efficiency, ρ, as

$$\rho = \frac{T_P}{T_E} = \frac{T_E - T_{NP}}{T_E}, \tag{4}$$

where T_E, T_P and T_{NP} are total core execution, computation and non-computation cycles defined by Equation 2 and 3. The execution times and non-processing times of each core for five different videos recorded from the simulation logs are shown in Table III. As shown in Table III, the core IBC in NoC has execution time of 305391 clock cycles and non-processing times of 0 clock cycle and the core VLD has 6 clock cycles (waiting time for data being available) non-processing time for waiting in 1198074 clock cycles of execution time for bitstream $test1.m2v$. Both cores IBC and VLD, have maximum 100% core efficiency in NoC. The cores ISQ, IDCT and MC in NoC have non-processing times due to waiting for DTUs to arrive for processing as ISQ receives DTUs from VLD, IDCT receives DTUs from ISQ and MC receives DTUs from VLD and IDCT. The average core efficiencies of ISQ, IDCT and MC are found by using execution times and non-processing times obtained from Table III and using Equation 4 as 73.14%, 70.55% and 80.43%, respectively. On the other hand, due to shared interconnect access, re-arbitration times make up a major component of the non-processing times of all the cores in AMBA and hence, the application times increase (Section III-B1). Hence, it has higher idle times and lower core efficiencies of 57%, 22%, 24.6%, 28.6% and 22% for the cores IBC, VLD, ISQ, IDCT and MC, respectively (Table III). A graphical comparison of the average core efficiencies of NoC and AMBA with given core resources (Fig. 2) are shown in Fig. 6. Due to dedicated interconnects, cores in NoC utilise the execution cycles more efficiently compared to AMBA.

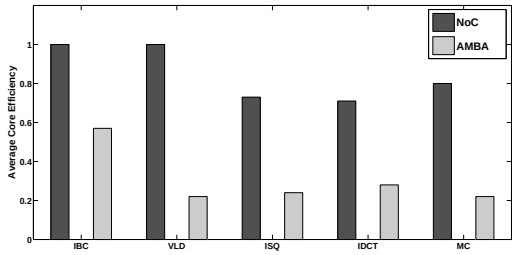

Fig. 6. Average Core Efficiencies in NoC and AMBA

3) Channel Latency: Latency (in time units) represents an important performance parameter for embedded systems. Per data transaction unit (DTU: packet with 32-bits payload for NoC, each 32-bit transaction data for AMBA) channel latency is a measure of how fast per DTU is routed over the channel from output of a core processor to input of the destination core. For both NoC and AMBA, the pipelined transaction of data takes place in multiple hops starting from initiator core to destination core. Considering no waiting states in NoC and AMBA, we define the average per DTU channel latency, L_{ch} as

$$L_{ch} = \sum_{n=1}^{N} \frac{\left[\tau^{S}_{c-in}(n) + \tau^{S-D}_{in-in}(n) + \tau^{D}_{in-c}(n)\right]}{N}, \tag{5}$$

where $\tau^{S}_{c-in}(n)$ is the time elapsed for data to travel from source output port to source interconnect port, $\tau^{S-D}_{in-in}(n)$ is the time elapsed for data to travel from source interconnect port to destination interconnect port and $\tau^{D}_{in-c}(n)$ is the time elapsed for data to travel from destination port to the destination memory, all for n-th DTU out of total N DTUs.

For AMBA, $\tau^{S}_{c-in}(n) = 1$ clock cycle after bus access is granted and locked. During $\tau^{S-D}_{in-in}(n) = 1$ clock cycle the arbiter does the necessary routing of the data and notifies the slave port. Due to direct memory interface, $\tau_{D_{in-c}}(n) = 0$ clock cycle. Minimum channel latency (without waiting states)

TABLE III
EXECUTION AND NON-PROCESSING TIMES OF CORES IN NOC AND AMBA FOR DIFFERENT VIDEO BITSTREAMS

Bitstream		IBC		VLD		ISQ		IDCT		MC	
	arch.	T_E	T_{NP}	T_E	T_{NP}	T_E	T_{NP}	T_E	T_{NP}	T_E	T_{NP}
test1.m2v	NoC	305391	0	1198074	6	877594	234756	818408	259353	1140704	223353
	AMBA	907154	364325	4089720	3198694	2690195	2041280	2352726	1703817	4089684	3198664
test2.m2v	NoC	349709	0	1407720	6	1024332	278977	991954	292974	1343984	261601
	AMBA	1079919	405208	4803141	3777805	3047764	2284885	2715453	1952580	4849387	3777775
test3.m2v	NoC	349993	0	1413252	6	1033501	277712	995838	294131	1348900	264065
	AMBA	1074657	463781	4873569	3797452	3135133	2363899	2703375	1932147	4873533	3797422
test4.m2v	NoC	1024130	0	4113958	6	3061612	783160	2955636	832189	3922459	767842
	AMBA	3214424	1413330	14372668	11199831	9213171	6939252	8199424	5925511	14372632	11199801
test5.m2v	NoC	1484309	0	6070042	6	4377452	1220626	4350221	1232548	5794595	1137740
	AMBA	4740479	2117758	21015438	16395196	14047999	10736748	11196332	7885087	21015402	16395166

per DTU for AMBA is then given by Equation 5 as $L_{ch} = 2$ clock cycles.

Due to symmetric nature of NoC channels, $\tau_{D_{c-in}}(n) = \tau_{S_{in-c}}(n) = 2$ clock cycles involving intermediate NI packetising and de-packetising. The delay, $\tau^{S-D}_{in-in}(n)$, in Equation 5 involves communication over an array of switches for each DTU that travels through the channel and depends on the number of intermediate switches travelled. Independent of routing algorithm or path travelled, $\tau^{S-D}_{in-in}(n)$ is given by

$$\tau^{S-D}_{in-in}(n) = \sum_{m=1}^{M-1} [\tau^m_{ic-r}(n) + \tau^m_r(n) + \tau^m_{r-oc}(n) + \tau^{m-(m+1)}_{oc-ic}(n)]. \quad (6)$$

Equation 6 is a result of multi-hop communication through M intermediate switches. The time required for the n-th packet to travel from input channel to the router of the m-th switch, $\tau^m_{ic-r}(n)$ is 1 clock cycle. Also, the time required for routing decision on the m-th switch for n-th packet, $\tau^m_r(n)$ is 1 clock cycle. The n-th packet travels from router to the output channel of the m-th switch immediately in our implementation and hence $\tau^m_{r-oc}(n) = 0$ clock cycle. Finally, the time required for the n-th packet to travel from output channel of m-th switch to input channel of the $(m+1)$-th switch, $\tau^{m-(m+1)}_{oc-ic}(n)$ is 1 clock cycle. According to Equation 6, L_{ch} for NoC has a variable delay with a minimum of 7 clock cycles (when $M = 2$ for shortest path mapping and XY routing).

Our simulations also verify that channel latency for NoC and AMBA are 7 clock cycles and 2 clock cycles, respectively. For same traffic pattern, channel latency sets up the major difference between the architectures. However, when pipelined transactions are used for both architectures, NoCs can employ concurrent and overlapped execution of cores (Section III-B1).

4) Bandwidth: Bandwidth (in bits per second) is a measure of the amount of data that can be passed through the interconnect in a given period of time. According to [6], the maximum available bandwidth for any node in any architecture is given by

$$BW_{arch_{MAX}} = \frac{\sum_{l=1}^{L_{arch}} w_{arch}(l) f_{arch}(l)}{H_{arch}},\ bits/cycle. \quad (7)$$

where $BW_{arch_{MAX}}$ is the maximum bandwidth for the architecture concerned, L_{arch} is the number of outgoing links being used, $w_{arch}(n)$ is the size of the l-th link in data bits only (or number of data wires), $f_{arch}(l)$ is the frequency of l-th link of the architecture being considered and H_{arch} is the number of hops (in clock cycles) required for node-to-node communication. As shown in Section II-C, due to spatial multiplexing of outgoing channels in NoC switch, $L_{NoC} = 4$ links (Fig. 5) and $H_{NoC} = 7$ clock cycles, as compared to $L_{AMBA} = 1$ link (assuming the aggregate usage scenario) and $H_{AMBA} = 2$ clock cycles. Considering single-packet per clock cycle injection rate in NoC and AMBA and time-sharing of bandwidth among 4 communicating cores in AMBA in 7, Equations 8 and 9 show that NoC has 14.2% bandwidth advantage over AMBA.

$$BW_{NoC} = \frac{(32 \times f_{NoC})}{7}\ bits/cycle \quad (8)$$

$$BW_{AMBA} = \frac{(32 \times f_{AMBA})}{(2 \times 4)}\ bits/cycle \quad (9)$$

In practice, the actual switching frequency will also depend on capacitive loading. According to [7], due to capacitive loading and global wire lengths in AMBA, considering $f_{NoC} = 3 \times f_{AMBA}$, the bandwidth definitions in Equations 8 and 9 give NoC a 2.428 times higher bandwidth advantage.

C. Comparative Application Performance

1) Per Macroblock Decoding Time: Given a video bitstream, the efficiency of an MPEG-2 decoder defines how quickly the bitstreams in the video can be decoded. Due to the fact that display sizes does not necessarily translate to the frame size of the videos, it is often convenient to first understand the application efficiency in terms of average per macroblock (MB) decoding time T_{MB}, in clock cycles [5]. Full-sized $4:1:1$ picture of each frame requires 30 rows with 45 MBs per row, i.e. total of 1350 MBs per frame. Dividing T_A obtained from Table II by the number of encoded MBs obtained from Table I, average per MB decoding time can be found and are shown in Table IV.

Referring to Table IV, for the video bitstream $test1.m2v$, $T^{AMBA}_{MB} = 2607$ clock cycles, which is 2.42 times higher than $T^{NoC}_{MB} = 763$ clock cycles (due to higher concurrency and overlap described in Section III-B1). For larger videos, depending on their core efficiency and concurrency (Sections III-B2 and III-B1), T_{MB} scales accordingly for AMBA and NoC. Due to low concurrency and core efficiency

TABLE IV
PER MB DECODING TIME AND REQD. FREQUENCY OF NOC AND AMBA

Bitstream	T_{MB}^{NoC}, clock cycles	T_{MB}^{AMBA}, clock cycles	f_{NoC} at standard fps, MHz	f_{AMBA} at standard fps, MHz	T_A, ms
test1.m2v	763	2607	25.8	77.9	46.53
test2.m2v	786	2708	26.5	80.7	53.07
test3.m2v	778	2687	26.3	80	53.82
test4.m2v	767	2684	31.1	95.9	132.44
test5.m2v	774	2685	31.3	95.5	193.64

(Sections III-B1 andIII-B2), AMBA has large non-processing time and hence higher application time and T_{MB}^{AMBA} (on average 2.46 times higher than T_{MB}^{NoC}).

2) Operating Clock Frequency: Clock frequency is an important parameter as processors with high clock rates dissipate power proportional to operating frequency [7]. The operating clock frequency required to give standard frame rate for the video bitstreams shown in Table I, can be found as,

$$f = (T_{MB} \times MBs\ per\ frame \times fps), \qquad (10)$$

where fps is the standard frames per second shown in Table I. The approximate operating clock frequencies found using Equation 10 are shown in Table IV. Due to lower T_{MB} in NoC, it can operate at lower clock frequency, f_{NoC}, than operating clock frequency for shared-bus AMBA, f_{AMBA} to decode the video bitstreams in similar application times shown in Table IV. For bitstream $test2.m2v$, NoC requires only 26.52MHz as against 80.7MHz for AMBA to achieve the frame rate of 25 fps (on average f_{NoC} is 29% of f_{AMBA}). A graphical comparison of the required clock frequencies of the five different bitstreams at their standard frame rate is shown in Fig. 7. The application times (in ms) (Table IV) are found

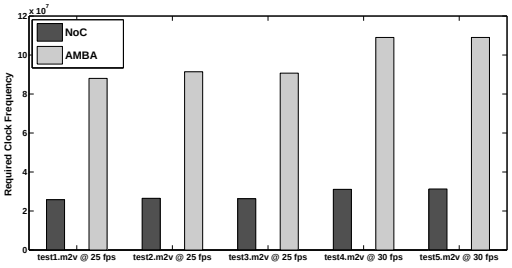

Fig. 7. Required clock frequency of NoC and AMBA for different MPEG bitstream decoding

by dividing corresponding T_A (in clock cycles) from Table II by the associated frequencies (Table IV).

It is evident that as the number of cores increases, the operating frequency can scale up for AMBA since core efficiencies would be lower, making T_{MB} much higher as a result. However, the fact that high clock frequencies in AMBA can be inhibited by capacitive loading as described in Section III-B4, this can restrict some multi-core applications with tens of cores to operate on AMBA.

IV. CONCLUSIONS

In this paper, we have performed comparative analyses between shared-bus AMBA and NoC using realistic MPEG-2 video traffic in cycle-accurate realistic simulation environment. Supported by analytical and simulation results, it has been shown that the NoC reduces decoding time by a factor of 2.46 on average compared to AMBA (Section III-C) for the given mapping (Fig. 5). Despite higher channel latency, NoCs have higher core efficiency, concurrency and bandwidth advantage over AMBA and can operate at lower frequency (approximately 29%) than AMBA for same decoding bitstream (Section III-C). Our comparisons focus on performance aspects between NoC and AMBA using a real application, while it supports the comparisons involving power, area and scalability in [9]. It is hoped that the findings in this paper would contribute towards the current research efforts in identifying appropriate on-chip communication architecture for emerging multimedia MPSoC applications.

ACKNOWLEDGEMENT

The authors would like to thank the EPSRC (UK) for funding this research in part under grant EP/C512804/1 and EP/E035965/1.

REFERENCES

[1] B. Al-Hashimi, Ed., *System-on-Chip: Next Generation Electronics*. IEE Press, May 2006, chapter 17.
[2] H. Lee, U. Y. Ogras, R. Marculescu, and N. Chang, "Design space exploration and prototyping for on-chip multimedia applications," in *Proceedings of the DAC*. California, USA, July 24-28 2006.
[3] E. Rijpkema, K. Goossens, and P. Wielage, "A router architecture for networks on silicon," in *Proceedings of the 2nd Workshop on Embedded Systems*, 2001, pp. 181–188.
[4] J. Hu and R. Marculescu, "Dyad - smart routing for network-on-chip," in *Proceedings of DAC*. ACM Press, 2004, pp. 260–263.
[5] J. Chang, W. Kim, Y. Bae, J. Han, H. Cho, and H. Jung, "Performance analysis for mpeg-4 video codec based on on-chip network," *ETRI Journal, Korea*, vol. 27, no. 5, pp. 497–503, October 2005.
[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Special issue on Networks on chip and Reconfigurable Fabrics of the VLSI Journal of Integration, Elsevier Science Publishers B. V.*, vol. 38, no. 1, pp. 19–42, October 2004.
[7] Arteris, "A comparison of network on chip and buses," http://www.arteris.com/noc_whitepaper.pdf.
[8] L. Benini and D. Bertozzi, "Network-on-chip architectures and design methods," *IEE Proceedings Computers and Digital Techniques*, vol. 152, no. 2, pp. 261–272, March 2005.
[9] H. Lee, N. Chang, U. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 3, pp. 1–20, August 2007.
[10] E. Salminen, A. Kulmala, and T. D. Hamalainen, "On the network-on-chip comparison," in *Proceedings of DSD*, 2007, pp. 503–510.
[11] D. Puschini and F. Clermidy, "A comparison between noc and bus architectures based on a real-application," in *Proceedings of ReCoSoC, 2006*, Montelier, France, July 2006, pp. 194–200.
[12] ISO, "ISO/IEC 13818-2: 1995 (E)," http://www.iso.org.
[13] ARM, "Advanced microprocessor bus architecture (AMBA) specification, v2.0, 1999," http://www.arm.com.
[14] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A design methodology for application-specific networks-on-chip," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 2, pp. 263–280, 2006.
[15] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J. Soininen, M. Forsell, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 105–112.