

The Impact of Communication on the Scalability of the Data-parallel Video Encoder on MPSoC

Erno Salminen, Tero Kangas, and Timo D. Hämäläinen
Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland
email:erno.salminen@tut.fi

Abstract—This paper presents design space exploration of a data-parallel video encoder running on Multiprocessor System-on-Chip (MPSoC). The impact of communication on the scalability is analyzed. The exploration is carried out with the abstract models for application and architecture by using Transaction Generator (TG) which enables rapid modeling and simulation of complex applications regarding the dependencies between application tasks. The application model is based on profiled execution times and data sizes from real application code. Results show that chosen parallelization method keeps communication between processing elements at low level. It allows good scalability in terms of communication bandwidth, image size, and number of processing elements but unequal computational load in some cases restricts the scalability. Moreover, the impact of overlapping the communication and computation is analyzed.

I. INTRODUCTION

Parallel processing can be utilized to meet the strict computation requirements of modern applications, such as real-time video encoding [5]. Parallelization allows lower clock frequency and lower operating voltage compared to fast uniprocessor system. Both factors decrease the power consumption, frequency linearly and voltage quadratically, and more than compensate the increased number of switching nodes [9].

Many studies consider how changes in application affect the speedup assuming fixed computation platform. This study analyzes how the changes in communication mechanisms affect the speedup assuming fixed application. Efficient communication between processing elements as well as load balancing are crucial to obtain good scalability. The communication requirements are proportional to the number of processing elements and, in general, inversely proportional to their size. Advances in silicon processing technology allow building multiprocessor System-on-Chip (MPSoC) devices to perform parallel computation within a single chip.

In this paper, a design space exploration for data-parallel video encoder MPSoC [6] is presented. The processing elements (PEs) are interconnected with a single shared bus to analyze its feasibility. Various architectures are explored by changing the number of PEs, their frequency, and bus frequency. Furthermore, the upper bound for performance resulting from ideal network and the impact of direct memory access (DMA) are determined. Data-parallel video encoder application is modeled with Transaction Generator [7].

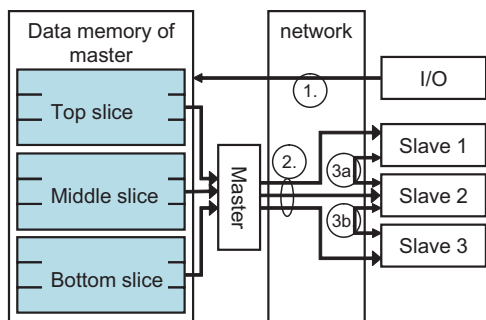
II. RELATED WORK

Communication networks are often compared by defining their load-latency curves [3]. When offered load increases beyond the saturation point, the latency increases very rapidly approaching infinity. In a shared bus, the available bandwidth per PE, i.e. saturation point, is the total bandwidth divided by the number of PEs which clearly restricts the scalability. In addition, capacitive loading and the wire lengths increase with system size. However, parallel computers utilizing shared bus with up to 70 processors have been built [2]. Network-on-chip (NoC) topologies, such as mesh or fat-tree, utilize shorter wires and offer substantially larger maximum bandwidth but also higher average latency with small load [1] [8].

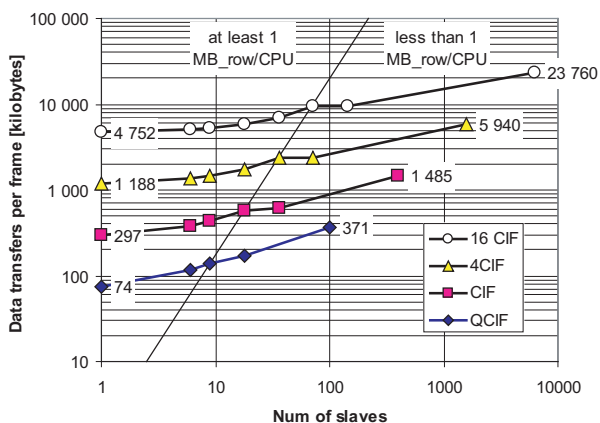
Uniform traffic pattern is widely utilized in network studies since it is easy to generate and obtain bounds for latency and maximum bandwidth analytically. However, traffic tends to be localized in practice [8] and there are many dependencies between application processes mapped to different PEs. Therefore, the PEs do not access the network continuously (offered traffic is not constant) and traffic is said to be self-throttling [3]. Therefore, realistic traffic loads from actual SoC applications should be identified. This study considers the dependencies between the tasks in addition to spatial distribution and the data rate.

The runtime is the sum of computation, communication, and synchronization. More precisely, only the non-overlapped portions of each in the critical path are considered [2]. For example, PE_1 transfers data to busy PE_2 which cannot process the data before it has finished its current task. As long as data arrives to destination prior to the end of the current task, the communication is fully overlapped with computation and has no impact on performance. In contrast, the network impacts runtime directly when a (single-scalar) processor is stalled after a cache miss. Moreover, load imbalance and synchronization due to dependencies can cause idle periods in PEs even in the presence of an ideal network.

With DMA controller, a PE can overlap the processing and transferring of data. The impact depends strongly on the relative lengths of computation and communication. When communication requirements are very low or extremely high, DMA offers no significant speedup. The biggest gain can be achieved when computation and communication times are roughly equal and PE always has tasks that are ready for execution after the previous is completed.



(a) Horizontal data-parallelization of QCIF video to 3 slaves.



(b) Data transfer requirements per frame.

Fig. 1. Data parallel video encoding.

III. DATA-PARALLEL VIDEO ENCODER TEST CASE

A detailed presentation of the video encoder can be found in [6]. The video encoding processing is done in master-slave configuration, in which all processing elements have local data and instruction memories. All the slaves execute the same encoding functions but operate on different parts of the image. Each PE processes one slice consisting of one or more horizontal macroblock rows as shown in Fig. 1(a). At first, the master PE reads one frame of raw video from I/O unit. Second, it divides the frame into slices and sends them to the slaves. Hence, a raw frame is transferred twice across the communication network at the beginning of each frame. Third, the slave PEs must exchange rows with their neighbors (if any) to carry out motion estimation. Finally, the master merges their results (opposite to transfer 2) into a final bit-stream and sends that to I/O (opposite to transfer 1). Data amount depends on the number of PEs in phase 3 only.

The data transfer requirements per frame are illustrated in Fig. 1(b). Four different frame sizes are included from QCIF (176 x 144 pixels) up to 16CIF (1408 x 1152). Note that both axes are logarithmic. Here, only the transfers of video data are considered, since they account approximately 99 per cent of all traffic. The graph is divided into two parts: left side depicting coarse, row-wise parallelization and right side depicting fine-

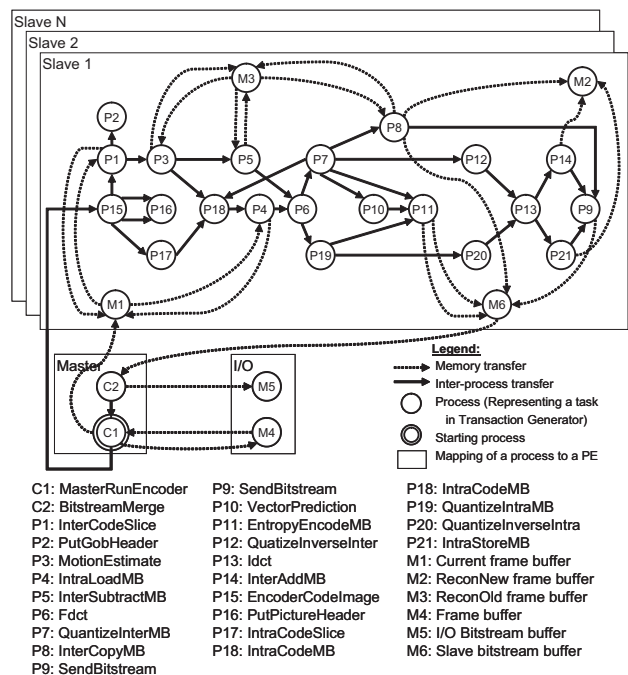


Fig. 2. Task graph and the example mapping of the video encoder application. Each slave has identical set of encoding tasks that are initiated by master's task C1.

grained parallelization. The division happens when the number of slaves equals the number of macroblock rows (MB_{row}). The largest possible system has equal number of slaves and macroblocks.

With one slave, the transferred data amount per frame equals 2 raw image frames and increases linearly to roughly 4 frames when the slice size is one row. At the most parallel configuration, i.e. one macroblock per PE, the total data amount is roughly 10 frames. At the same time, the required amount of local data memory per slave decreases. For 25 frames per second, the worst case total data rates, i.e. with maximum number of slaves, are approximately 10 MB/s (QCIF), 37 MB/s (CIF), 146 MB/s (4CIF), and 581 MB/s (16CIF). The data rates to the network from both master and IO are equal and vary between 1 MB/s (QCIF) and 60 MB/s (16CIF). The slave data rates are at maximum 0.2 MB/s (QCIF) and 1.6 MB/s (16CIF). As a conclusion, the data rates increases only modestly with the number of PEs. The above values account only the data transfers. Artifactual communication, for example cache misses, would increase the traffic but is not considered here. Note that number of PEs affects the problem size only in master's tasks which constitute only very small fraction of all computation.

IV. MODELING WITH TRANSACTION GENERATOR

In this work, Transaction Generator (TG) [7] is utilized with transfer-dependent application model. It injects and reads data to/from a network according to external behavior of the application tasks. The data transfers between tasks that are mapped to separate PEs are forwarded to the network.

The task runtime and transfer sizes are profiled from a real application in ARM7 instruction set simulator. The impact of hardware accelerator can be easily analyzed by profiling them and updating the corresponding task runtimes in the TG model. Only the data transfers are profiled since all PEs are assumed to have local instruction memories.

The application model is illustrated in Fig. 2. Both master and I/O unit execute two tasks. All slaves have the same program code and, hence, similar task graph consisting of 21 computation tasks and 4 memory tasks. A shared memory system (omitted here for brevity) could be modeled by mapping the memory tasks into a shared memory instead of a slave PE. The structure of the task graph of one slave does not depend on the frame size whereas the transfer sizes and computation times do. Adding more slaves, increases the total task count from 29 with one slave to 1520 tasks with 72 slaves. The accuracy of the model has been evaluated against cycle-accurate HW/SW co-simulation for QCIF frames and 1 to 9 PEs. The estimation error for runtime is about 10% on average. The co-simulation for 9 slaves took nearly 16 hours and less than 10 minutes in TG. On a basic parallelization scheme [6], there is at least one row per PE, but more parallel cases can be easily estimated with TG. However, the minimum slice size is set to 9 macroblocks for CIF and 18 macroblocks for 4CIF.

V. EXPLORATION RESULTS

All the encoding times are measured for INTER frames. Unless otherwise specified, the PEs are operating at 100 MHz. All cases are run with DMA switched on and off. Since the transfer requirements are rather modest, this study concentrates on single shared bus that is modeled in transaction level. HIBI network [10] is used and configured as single bus. The transaction level model allows estimating ideal network by transmitting the whole message in one clock cycle. A message may contain the whole macroblock or even the whole frame. This way, it is possible to determine the minimum runtime for certain application by considering only the computation times and idle times due to transfer dependencies. In other cases, the width of the bus is 32 bits, i.e. 4 bytes, and bus frequency is either 1, 10, or 100 MHz.

The simulated speedups for CIF-sized video encoding are shown in Fig. 3. The straight line depicts ideal speedup which equals the number of slaves. Ideal network nearly achieves that estimate when the load is balanced and a 100 MHz bus with DMA is very close. The actual speedup is not linear due to load imbalance causing plateau areas in the curves. At the plateau, increasing the number of slaves does not decrease the largest slice size but only increases the traffic. A 100 MHz bus handles the increased traffic well but the frame rate drops slightly with smaller bus frequencies. The improving steps occur when the number of macroblock rows is an integer multiple of the number of slaves.

As shown, the basic video encoder does not stress the communication network heavily due to modest PE frequency. Furthermore, the C codes could be optimized or some parts could be executed with hardware accelerators. Fig. 4 shows the

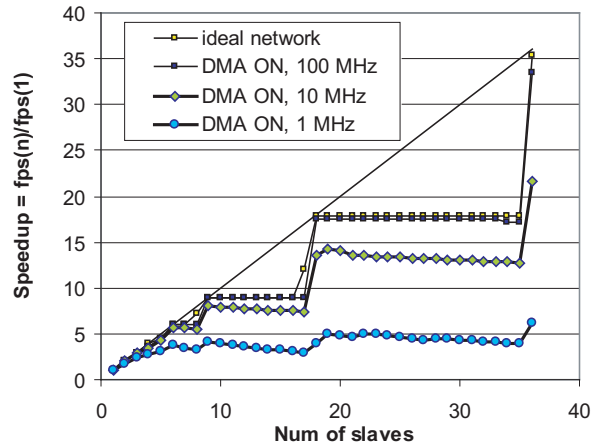


Fig. 3. Speedup for CIF-sized frames with different bus frequencies. DMA is ON in all cases.

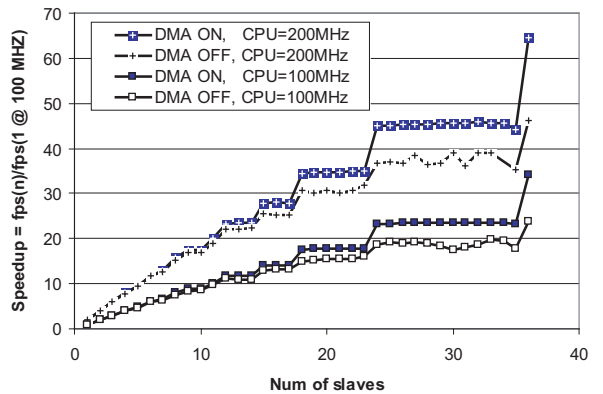


Fig. 4. Speedup for 4CIF with PE frequency is 100 or 200 MHz, and bus frequency is 100 MHz. Results are given with and without DMA.

speedup when the frequency of the PEs is doubled and bus frequency is 100 MHz. Speedup is given relative to system with 1 CPU at 100 MHz and 100 MHz bus as in Fig. 3. This clearly demonstrates how the video encoding is dominated by computation times instead of communication. Compared to Fig. 3 the shape of the curve is smoother due to larger frame size and, hence, load is more balanced. Furthermore, the impact of DMA, due to overlapping of computation as communication, is illustrated. Interestingly, a 10 MHz with DMA gives similar performance as 100 MHz bus without DMA. DMA behavior is similar in all evaluated frames sizes. The additional speedup due to DMA depends linearly on the system size irrespective of the bus frequency. Such speedups were at maximum 1.9x (CIF) and 1.55x (4CIF). Note that the speedup would be larger if increased data rate of DMA compared to a processor-controlled transfers was included.

So far, the bus frequency was assumed constant irrespective of the system size. In reality, the delay of a wire depends on its length. Repeater insertion changes the delay dependence from

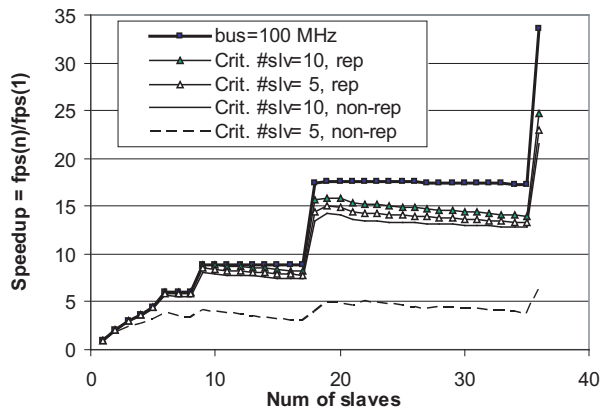


Fig. 5. Speedup for CIF when bus frequency depends on the number of slaves. Results are shown for repeated and non-repeated wires. DMA is ON in all cases.

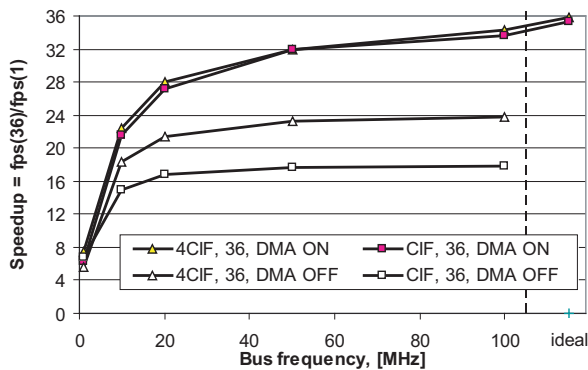


Fig. 6. Speedup for CIF and 4CIF as a function of bus frequency. System size is 36 slave PEs. Results are shown with and without DMA.

quadratic to linear but increases the power dissipation and area. The examples have shown that already a bus frequency of 100 MHz (same as PE frequency) provides enough capacity for real-time video processing and allows good scalability. More realistic frame rates can be interpolated from previous results, when the maximum bus frequency is determined. These are shown in Fig. 5 along with constant 100 MHz case.

Since the frequency depends on the utilized silicon technology and final layout, a simple frequency estimation is utilized here. Cases with different critical system size, i.e. the largest possible system that can operate at PE frequency, are shown. For example, when system size is two times the critical size, the bus length has doubled. Consequently, the frequency of non-repeated bus has dropped to 25% of the original in case (and to 50% with repeaters). Cases with critical sizes 5 and 10 PEs are shown. For repeated global wires, the reachable distance per clock cycle is, for example, 28 mm (180 nm process, 700 MHz) and 10 mm (70 nm process, 2.5 GHz) [4]. Hierarchical networks have shorter links and, hence, their operating frequency is less dependent (in optimal case independent) of the system size. They need to be adopted

in large parallel systems - not necessarily due to shared bandwidth of the bus, but due to prohibitively long delays in long wires. Considering the area costs of the bus are often lower than those of NoCs [10], a bus-based network is still a good candidate for many systems. Simulations were run also with 2-level hierarchical bus. The runtimes are (practically) identical whereas the maximum wire lengths are halved.

Fig. 6 shows the correlation between bus frequency and obtained speedup in a system with 36 PEs. The speedup achieved with ideal network is also shown (DMA has no effect then). The speedup is the same for both frame sizes when DMA used. However, larger frame size, 4CIF, achieves greater speedup when DMA is not used.

VI. CONCLUSIONS

This paper analyzed the parallelization of video encoder in bus-based SoC. The analysis of transfer requirements is followed by extensive simulations with varying system size and frame size. This application has rather low bandwidth demand even if large frames are processed with large number of processing elements. Therefore, a single, shared 100 MHz bus practically achieves the theoretical maximum performance and the shared bandwidth of the bus is not a problem in this case. However, the delay and power consumption caused by long wires possesses more serious threats for scalability.

DMA allows overlapping computation and communication and the resulting speedup can be over 1.5x in this case. The utilized data-parallel processing scheme keeps communication at low level, can offer good scalability but suffers from load imbalance with certain numbers of slave PEs. In the future, the fine-grained parallelization methods and functional pipelining must be thoroughly evaluated.

REFERENCES

- [1] A. Andriahantainaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "SPIN: a scalable, packet switched, on-chip micro-network," in *DATE*, Munich, Germany, March 2003, pp. 70–73.
- [2] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture - A Hardware/Software Approach*. Morgan Kaufmann, 1998.
- [3] W. J. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann Publishers, 2004.
- [4] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, 2001.
- [5] A. A. Jerraya, "Long term trends for embedded system design," in *Euromicro Symposium on Digital System Design (DSD)*, Rennes, France, September 2004, pp. 20–26.
- [6] T. Kangas, K. Kuusilinna, and T. D. Hämäläinen, "Scalable architecture for SoC video encoders," *accepted to Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, 2006.
- [7] T. Kangas, J. Riihimäki, E. Salminen, K. Kuusilinna, and T. D. Hämäläinen, "Using a communication generator in SoC architecture exploration," in *Symposium on System-on-Chip*, Tampere, Finland, November 2003, pp. 105–108.
- [8] P. Pande, C. Grecu, A. I. M. Jones, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, August 2005.
- [9] C. Rowen, *Engineering the Complex SoC*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.
- [10] E. Salminen, T. Kangas, J. Riihimäki, V. Lahtinen, K. Kuusilinna, and T. D. Hämäläinen, "Hibi communication network for system-on-chip," *Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, vol. 43, no. 2, May 2006.