

# Instruction Memory Architecture Evaluation on Multiprocessor FPGA MPEG-4 Encoder

Ari Kulmala, Erno Salminen, and Timo D. Hämäläinen  
Tampere University of Technology, Institute of Digital and Computer Systems,  
P.O. Box 553, Korkeakoulunkatu 1, FI-33101 Tampere, Finland  
ari.kulmala@tut.fi

**Abstract**—Memory is a significant performance limiting factor of the multiprocessor systems especially when shared. In FPGAs, the memory amount of the device is fixed and thus, optimal memory usage is essential. This paper analyses how the fixed amount of memory should be divided between instruction memories and instruction caches for multiprocessor systems and compromised with the number of processors. The measurements are done with a SPMD (Single Program Multiple Data) multiprocessor system of up to 14 soft core processors running a MPEG-4 video encoder on FPGA. The instruction memory count is ranged between one and seven. It is shown that the traditional distributed memory architecture is outperformed by shared instruction memories with sufficient cache sizes. The number of processors is in general the most significant single factor once the sufficient cache size is reached. The best performance was obtained with only one shared instruction memory, 8 KB cache and 13 processors.

## I. INTRODUCTION

The ever-increasing need for more computational power in embedded devices has driven digital system designers to use multiprocessor systems instead of just one processor core. Obviously, using more processors requires also more memory. However, memory resources may be scarce and limited, as in FPGAs or PLDs, or the chip cost must be minimized by using as low amount of memory as possible.

Shared data memories are common, but shared instruction memory has gained only little attention due to typical sequential programming paradigm. However, different parallelization styles, like Single Program Multiple Data (SPMD) paradigm, can benefit from shared instruction memories. Shared instruction memory is used in a multiprocessor system utilizing Nios processors for parallel LU optimization in [1]. The memory is accessed with a special prefetch unit that feeds the caches of the processors to reduce the memory contention. The effect of the shared instruction memory alone is not considered, but they have noticed that cache configuration can make a 20% difference on the performance.

The general effect of cache size is explained in [2] that shows that most of the misses (>60%) are *capacity misses*, i.e. due to cache size. This can be addressed only by increasing the cache size. We have conducted a study on the shared instruction memory effect on performance in a four processor system with fixed instruction cache (icache) size of

8 KB [3]. The results show that the shared instruction memory causes 4% runtime overhead with six processors, and is still sufficient option with ten processors (overhead 13%).

In this paper, we analyze how a fixed amount of memory of the architecture should be divided between instruction memory banks and icaches to maximize performance. Having fixed amount of memory, increased icache sizes or several instruction memories result in reduced processor count in the architecture, since all the processors in our system need memory also for e.g. local data memories. Therefore, a compromise between processor count and memory distribution is required.

The study results are best suited for implementations where processors can share identical memories, e.g. SPMD, on FPGA. To our knowledge, our study is first of its kind; therefore the amount of related work is rather small.

This paper is divided into following sections. First, Section II discusses the hardware architecture and the MPEG-4 encoding application. Section III shows the different memory configurations used and Section IV discusses the synthesis and measurement details. The results are shown in Section V and Section VI concludes the paper.

## II. MPSOC ARCHITECTURE AND MPEG-4 APPLICATION

Our architecture consists of one master processor and parametrizable amount of identical slave processors. The architecture is depicted in Fig. 1. The figure shows the architecture with single shared memory. The instruction memory is accessed via Avalon external bridge and arbitrator.

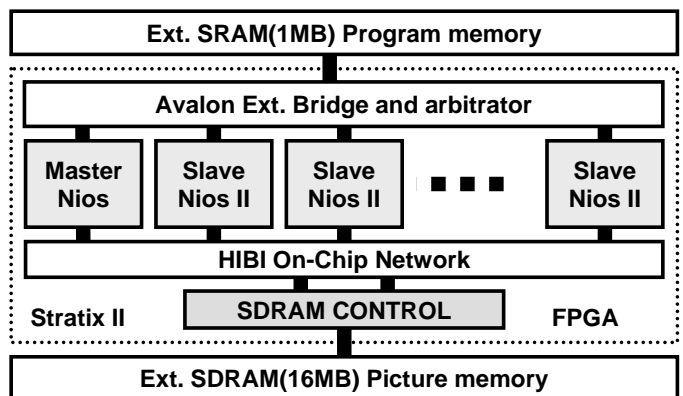


Fig. 1. MPSoC Architecture with single instruction memory.

The architecture also contains Heterogeneous IP Block Interconnection (HIBI) network that connects the processors and shared external picture memory controller (SDRAM ctrl) together. HIBI is used solely to data transmissions between CPUs and the picture memory, and for CPU-to-CPU communication. Each processor has an instruction cache, 64 KB (kilobyte, 1024 bytes) local scratch-pad data memory (image data is fetched from the shared data memory in portions), and roughly 2 KB ROM memory for boot program. Data cache is not utilized.

The MPEG-4 video encoder is based on horizontal spatial parallelization and achieves very high scalability with support of arbitrary number of processors, and hence, in practice parametrizable performance [4]. This makes it ideal for embedded, rapidly upgradeable video encoding engine.

The master processor controls the encoding flow. It divides the video image to equally sized slices for each slave processor. Slave processors encode the image slices in parallel and store the result to the shared data memory. When all the slaves have finished the encoding, the master processor finishes the encoding and starts the processing of the next frame. The slaves perform encoding and thus, access the instruction memory most of time. Slaves are identical and follow SPMD paradigm. That allows them to have also identical program memories, thus only one copy of the program in the memory is required. The master has its own program code that is different from the code of the slaves. The master processor mostly waits for the slaves to finish during the encoding phase.

The architecture is run on Altera DSP Development Kit, Stratix II Professional edition. It contains Stratix II S180 FPGA, 1 MB external SRAM, 32 MB external SDRAM, and I/Os such as UART and Ethernet.

### III. MEMORY CONFIGURATIONS

The encoding application of the slaves fits to 128 KB memory. Several different instruction memory configurations are used. They are shown in Table 1 that also presents on-chip memory consumption, which will be explained shortly.

The master is always connected to the external SRAM. The FPGA on-chip memory is dual-ported so a single physical memory instance can be considered as two logical instruction memories. Therefore, we have three shared instruction memories when 128 KB of the on-chip memory is used to form one instruction memory on-chip as illustrated in Fig. 2. The instruction memory is only read so there are no conflicts between accesses from separate ports.

Fig. 3 depicts a configuration with two 128 KB on-chip instruction memories and one external memory, resulting in logically five instruction memories.

The last case includes three physically and therefore 6 logically shared on-chip memories and one external memory as depicted in Fig. 4. The depicted configuration is essentially a distributed memory configuration since each processor has

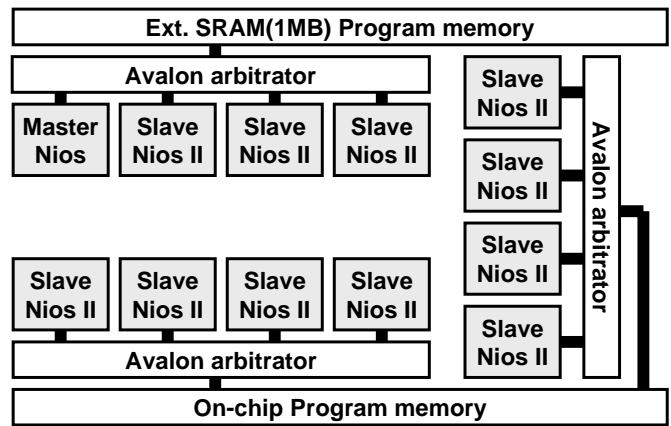


Fig. 2. Two physically, three logically shared program memories.

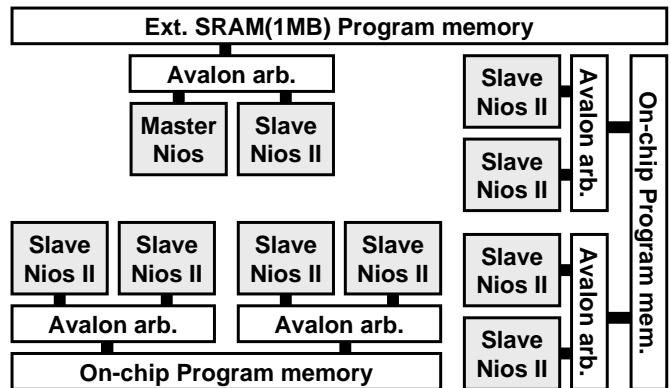


Fig. 3. Three physically, five logically shared program memories.

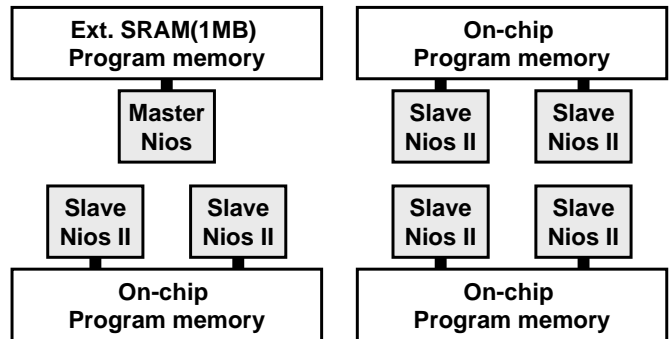


Fig. 4. Four physically, seven logically shared program memories.

own port to the shared memory. The Nios II instruction caches are direct mapped and contain eight 32-bit words per line [5].

In addition, instruction cache sizes for the slave processors are varied, which affects the execution significantly when shared memory is used. Five different sizes are compared: 512 bytes and 8, 16, 32, and 64 KB. Nios II requires at least 512 bytes icache. In our case the number of processors is bounded by the available local memory per processor and not by logic elements. Using more memory for icaches decreases the number of Nios processors. The master processor has a

fixed 8 KB icache, which is not varied because only slaves form the basis of the encoder and the video encoding speed is mostly dependent on slaves.

#### IV. ARCHITECTURE SYNTHESIS AND MEASUREMENTS

An Adaptive Logic Module (ALM) is the basic building block of Stratix II architecture, but the area is usually (e.g. by synthesis tools) reported in Adaptive Look-Up Tables (ALUTs). One ALM contains basically two flip-flops and 8-input LUT and a single ALM can be divided into 2 ALUTs. The used Stratix 2S180 FPGA includes 143 520 ALUTs for hardware logic [6]. One slave processor requires about 4700 ALUTs. The whole architecture with one external instruction memory and 13 CPUs (icache 8 KB) requires 86 000 ALUTs (60% of the device capacity). Therefore, the logic resources of the FPGA are not restricting the amount of processors.

The maximum number of processors can be initially estimated by counting the amount of memory required for processors and comparing that with available memory. However, the on-chip memory of the Stratix II FPGA is divided to several different block sizes: M512, M4K, and M-RAM. All the memory blocks include also parity bits, which can also be used for e.g. data storing. For example, M512 memory includes 512b for pure data + 64 bits for parity, totaling 576 bits actually usable (for example in a configuration of 64x9b). M4K has 4096+512 and M-RAM 524 288+65 536 bits. However, the parity bits cannot be used as storage in every configuration, resulting in overhead in memory usage. The total amount of memory, including parity bits, in the used Stratix 2S180 FPGA is 9 383 040 bits (1145 KB) [6].

As a consequence, the amount of bits used for the memories can be significantly lower than actual amount of consumed memory bits of the device. Typically, of the total amount of the consumed bits, 88-89% is actually used and the overhead is 11-12% (1 parity bit per 8 data bits). Table 1 shows the required memory bits for each configuration, actual memory consumption, and how much of the actual

consumption is used for the storage.

Table 1 also contains the memory consumption of the processors. The master always takes 80.8 KB. The "Slave CPU req. Mem [KB]" reports the amount of memory each slave requires in total, containing for example local data memory and icache. It should be noted that increasing icache size also increases the total memory due to e.g. cache tag tables. For example, changing the icache size from 16 KB to 32 KB with one instruction memory results in total memory increase from 83.4 KB to 100.2 KB. The increase is 16.8 KB instead of 16 KB. The memory requirement of a single slave ranges from 67 KB to 134 KB.

The total memory utilization amount ranges from 87% to 98% of the whole device capacity. Unfortunately, the configurations with over 140 KB of free memory capacity could not be synthesized with more slave processors. The reported processor counts are maximum obtainable. Table 1 also shows the amount of overhead memory consumption in KB. The overhead stays relatively constant within a range of 119-133 KB.

In this work, only push-button synthesis is used and no special synthesis options have been utilized. Altera Quartus II 6.0 is used as the synthesis tool.

The system is monitored with a custom hardware component that is fully independent of the memory configuration. For example, if the measurements were done with the processor timers, there would be some configuration-specific error due to varying instruction memory access times. Therefore, the hardware monitor gives the best accuracy. Unfortunately, the Nios II core is encrypted and the cache behavior (as of particular interest, miss rate) cannot be monitored.

The MPEG-4 simple profile application is run for 20 times and 50 frames per iteration. The results are averages of these runs. The used video sequence is standard Foreman and the picture format is CIF (352x288 pixels). The processors have been run on 50 MHz to allow fair comparison between instruction memories (external SRAM is slower) and the

TABLE 1. THE MEMORY CONFIGURATIONS AND ARCHITECTURE ON-CHIP MEMORY CONSUMPTION.

# Instruction Memories	#CPUs	ICache [KB]	Master CPU req. Mem. [KB]	Slave CPU req. Mem [KB]	On-chip Instruction Memory [KB]	Required memory total [KB]	Req. % of the device capacity	Total consumed memory [KB]	Consumed % of the device capacity	Required % of Consumed	Free capacity [KB]	Overhead [KB]
1	14	0.5	80.8	67.1	0.0	953.0	83 %	1 086.0	95 %	88 %	59.4	132.9
	13	8	80.8	75.0	0.0	981.1	86 %	1 100.3	96 %	89 %	45.1	119.2
	12	16	80.8	83.4	0.0	998.6	87 %	1 120.8	98 %	89 %	24.6	122.2
	10	32	80.8	100.2	0.0	982.4	86 %	1 102.8	96 %	89 %	42.6	120.3
	7	64	80.8	133.6	0.0	882.1	77 %	1 004.5	88 %	88 %	140.9	122.4
3	12	0.5	80.8	67.1	128.0	946.8	83 %	1 080.5	94 %	88 %	64.9	133.6
	11	8	80.8	75.0	128.0	959.1	84 %	1 084.6	95 %	88 %	60.8	125.6
	10	16	80.8	83.4	128.0	959.7	84 %	1 084.6	95 %	88 %	60.8	124.9
	8	32	80.8	100.2	128.0	910.1	79 %	1 033.0	90 %	88 %	112.4	122.9
	6	64	80.8	133.6	128.0	876.6	77 %	996.3	87 %	88 %	149.1	119.7
5	10	0.5	80.8	67.1	256.0	940.6	82 %	1 070.8	93 %	88 %	74.6	130.1
	9	8	80.8	75.0	256.0	937.0	82 %	1 061.4	93 %	88 %	84.0	124.4
	8	16	80.8	83.4	256.0	920.8	80 %	1 043.1	91 %	88 %	102.3	122.3
	7	32	80.8	100.2	256.0	937.9	82 %	1 063.0	93 %	88 %	82.4	125.1
7	8	8	80.8	75.0	384.0	990.0	86 %	1 110.9	97 %	89 %	34.5	120.9
	7	16	80.8	83.4	384.0	965.4	84 %	1 085.8	95 %	89 %	59.6	120.5

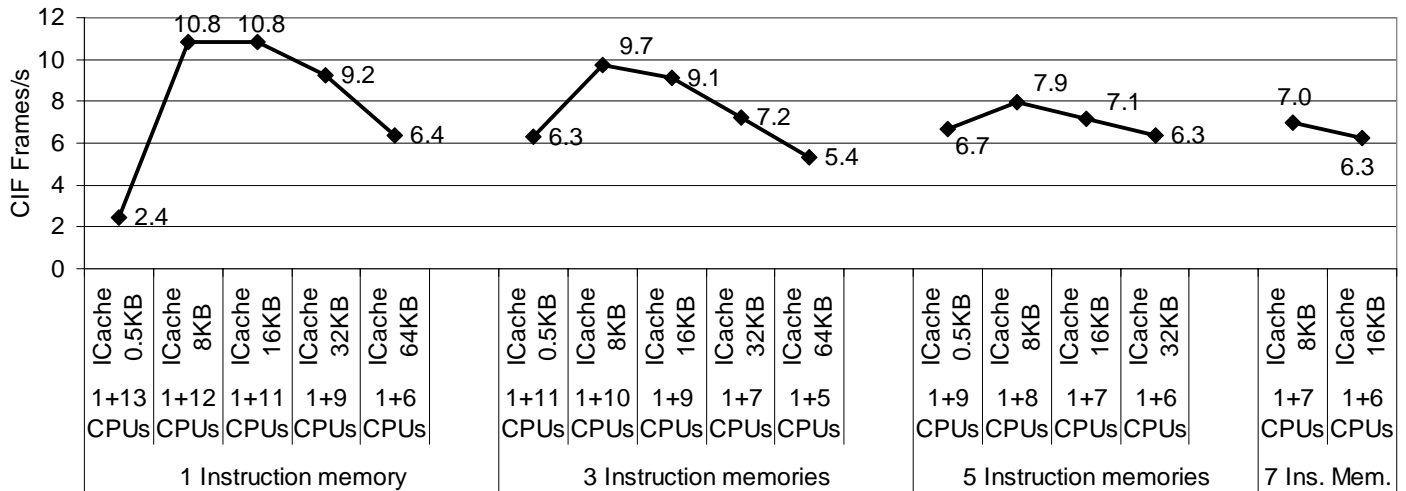


Fig. 5. Performance with varying number of instruction memories, different icache sizes, and number of processors.

memories have the same read latency (2 clock cycles, on-chip memories could have latency of 1). The other components of the architecture are run on 100 MHz to minimize their impact on the performance. For example, the HIBI on-chip network, configured as a single shared bus, utilization is only 4%.

## V. RESULTS

We have benchmarked configurations of one and three logical instruction memories with all the icache sizes (0.5, 8, 16, 32, and 64KB). The configuration having five instruction memories is benchmarked with 0.5, 8, 16, and 32 KB icaches because we could have only 5 processors with 64 KB icaches, which is a distributed memory configuration. We can have a better distributed memory configuration with 16 KB icaches, seven processors, and seven instruction memories, which is measured. Largest icache sizes were not evaluated for five and seven instruction memories since those allowed fewer processors than memories. The architecture with seven instruction memories and 0.5 KB icaches could have only 8 processors, the same as with 8 KB icaches, so it was not included.

### A. Performance comparison of the configurations

Fig. 5 shows the performance (frames/s, FPS) of the different configurations. The number of CPUs is marked as 1 + N<sub>s</sub>, meaning that there is one master CPU and N<sub>s</sub> slave CPUs. The figure shows clearly that the most dominating factor is the number of processors. The best configurations are therefore found with only one shared instruction memory. The configuration of 1+12 CPUs with 8 KB icache was the best with a slight margin, 1+11 CPUs with 16KB icache had practically the same performance. As the icache size further increases to 32 KB, the amount of processors drops to 1+9 and the overall performance weakens. It shows that the 8KB and 16KB icaches are good enough so that the memory contention does not play a significant role anymore.

Therefore, 32 KB icaches, neither 64 KB icaches, cannot improve the performance by reducing the contention.

The same effect can be seen with each memory configuration. Typically, configurations with 8KB icache outperform the others due to larger number of processors, and 16 KB icache configurations are close second. The last configuration, 1+6 with 7 instruction memories is practically the traditional *distributed memory architecture*. It is one of the worst configurations in the performance sense due to low number of processors. On the other hand, the configuration with most processors with one shared memory and only 0.5 KB icache, the performance of the architecture is the worst. Naturally, this is due to high amount of contention to the instruction memory caused by the highest number of processors and lowest amount of icache.

The average performance of each slave processor in the configuration is shown in Fig. 6. The figure shows the relative performance of a slave CPU with different memory

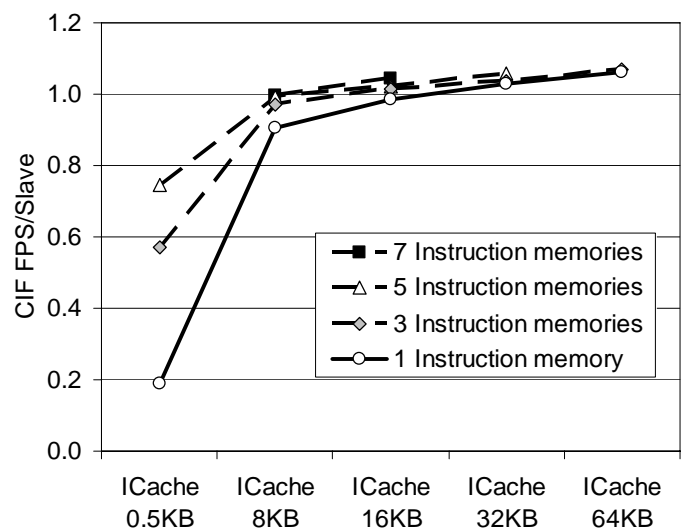


Fig. 6. Average performance of each slave processor.

configurations. The differences in the curves are due to the instruction memory contention. Clearly, the knee point of icache size is at about 8 KB. Still with 16 KB icache, we get a notable increase in performance/slave with one instruction memory. The Fig. 6 also proves that the performance increase by larger icaches than 16 KB are insignificant as the Fig. 5 suggested. Naturally, as the number of instruction memories decreases, the amount of icache memory matters the most. Thus, we gain less in performance when increasing icache memory from 0.5 KB to 8 KB with 3 instruction memories than with one.

The impact of icache to the performance is further depicted in the Fig. 7, which contains one shared instruction memory and all the possible processor counts benchmarked. The figure shows that with only 0.5 KB icache, the actual maximum performance (saturation point) is reached. There, the performance saturates with 1+5 processor configuration. When saturated, the performance does not increase with the number of processors due to the memory contention. Furthermore, Fig. 7 in conjunction with Fig. 6 shows that the performance is notably better with 16 KB icache than with 8KB, but because 8 KB architecture can have one more processor, the resulting encoding speed is practically the same.

Also, ideal line is drawn into the Fig. 7 showing how the system would scale ideally (so that performance with e.g. 1+3 processors would be 3x the performance of 1+1 processors). The gap between ideal and measured ones is due to parallelization overhead and memory contention. However, it can be expected that memory contention does not significantly decrease the performance with e.g. 32KB icaches, since doubling the icache size from 16KB does not yield a significant improvement in performance. Therefore, the gap is mainly due to parallelization overhead.

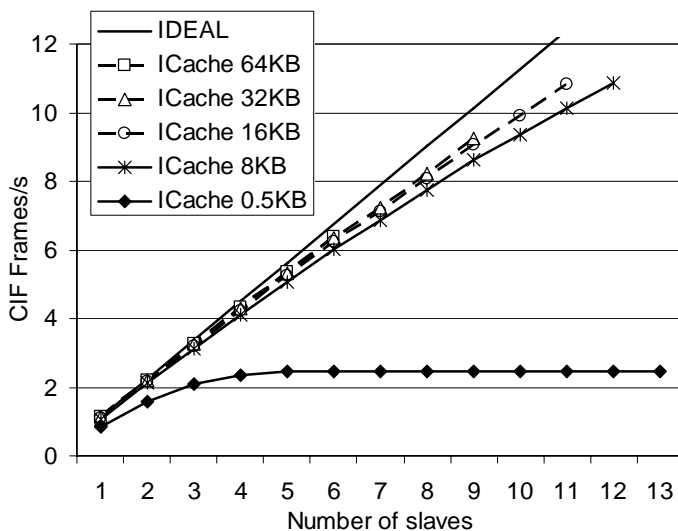


Fig. 7. Varying icache sizes and number of slave processors benchmarked in one instruction memory configuration.

## B. Discussion

In the measurements, the memories have been used at the same clock frequency as the CPUs. This is typical in FPGA devices, since the embedded on-chip SRAM is fast, reaching 350-370 MHz while, for example, a 16-to-1 multiplexer implemented in LEs can be run at 422 MHz [6]. Thus, the typical situation in e.g. high-end microprocessors that the CPU may be run on several times larger frequency than memory, is not present in FPGA soft multiprocessor systems.

Running the CPU faster than memory would increase the relative memory access times. In order to have some performance improvement, caches have to be run at the same speed as the CPU and the main instruction memory would be slower. As the access times increase, or more accurately, cache miss penalty increases, the cache size would play even more important role that it did in these measurements.

## VI. CONCLUSIONS

We have benchmarked 15 different shared instruction memory architectures and one distributed memory architecture with a multiprocessor SPMD MPEG-4 video encoder executed on FPGA. The fixed amount of memory of the FPGA compromises the amount of instruction memories, amount of processors, and maximum icache size.

In the measurements, the best configuration was the architecture with one instruction memory, 13 processors, and 8 KB icaches. In general, the most significant factor is the number of processors once the sufficient cache size have been determined. Even though the performance of a single processor with smaller icache can be lower than with larger icache, the possible extra processors make up for the performance.

Therefore, for the highest performance, the best configuration is to use a single shared instruction memory for all the processors, maximizing the number of processors. This is in contrast to typical implementations that favor distributed instruction memory architectures.

## REFERENCES

- [1] X. Wang and S.G. Ziavras, "Performance optimization of an FPGA-based configurable multiprocessor for matrix operations," Proceedings of the Field-Programmable Technology (FPT), IEEE, 15-17 December 2003, pp. 303-306.
- [2] J.L. Hennessy and D.A. Patterson, "Computer Architecture – A Quantitative Approach", Chapter 5, third edition, Morgan Kaufmann Publishers, San Francisco, USA, 2003.
- [3] A. Kulmala, E. Salminen, O. Lehtoranta, T. D. Hämäläinen, and M. Hännikäinen, "Impact of Shared Instruction Memory on Performance of FPGA-based MP-SoC Video Encoder", The 9th IEEE workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp. 59-64, Apr. 2006.
- [4] A. Kulmala, O. Lehtoranta, T.D. Hämäläinen, and Marko Hännikäinen, "Scalable MPEG-4 Encoder on FPGA Multiprocessor SoC," EURASIP Journal on Embedded Systems, Special issue on Field-Programmable Gate Arrays in Embedded Systems, Hindawi Publishing Corporation, 2006. Accepted. Available online: <http://www.hindawi.com/journals/es/volume-2006.html>.

- [5] Altera Corporation, "Nios II Processor Reference Handbook", Section 5, version NII5V1-6.1, November 2006.
- [6] Altera Corporation, "Stratix II Device Handbook, Volume 1," version SII5V1-1.1, July 2004.