Evaluation of Application-Specific Multiprocessor Mobile System

Abu Asaduzzaman & Imad Mahgoub
Department of Computer Science & Engineering
Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431, USA
Abu/MCLab: (561) 297-2802 & Imad/Office: (561) 297-3458
aasaduzz@fau.edu & imad@cse.fau.edu

Keywords
Application specific systems, MPEG4, multiprocessors, cache size, performance evaluation

ABSTRACT
The popularity of application-specific computing systems is on the rise and many systems have been developed in fields such as multimedia, high-speed networks, information systems, signal and image processing. Real-time application-specific systems are more complex and require more time to develop. In this paper, we explore the architecture of a multiprocessor mobile system running MPEG4 application. We develop a simulation program using VisualSim to evaluate the system performance in terms of utilization, delay, and total number of transactions processed by the different system components for various cache sizes and task rates.

1. INTRODUCTION

A large number of computing devices are built every year for different purposes. Examples include cellular phones, video conferencing systems, home security systems, digital cameras, and portable media players (PMP). Any one of these systems normally executes a specific program repeatedly and can therefore be called application specific system. Most of these systems are normally embedded within larger electronic devices and suitable for mobile environment. For time critical applications, these systems must react to system changes and must compute certain results in real time [1].

Design metrics of application-specific mobile systems include performance (such as utilization, delay, and transaction), power, cost, time-to-market, and safety [1, 2]. Key components for such a system include hardware – CPU, memory, bus, cache, and software – operating system (OS) and application as shown in Figure 2. The application is broken into tasks and given to different processors. A processor is designed to perform the specific task assigned to it. For example, CPU-1 runs on any open OS, has only first level cache (CL1), and performs Task-1. On the other hand, CPU-N needs second level cache (CL2), third level cache (CL2), and real time operating system (RTOS) to perform Task-N. All CPUs share memory through single shared bus.

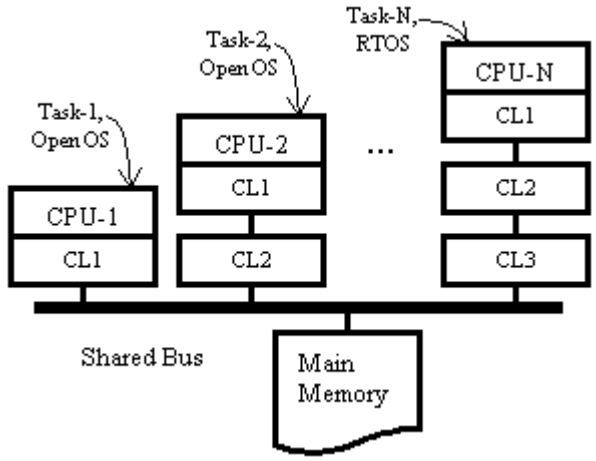

Figure 1: A shared bus multi-processor architecture

CPU is always faster than main memory [Figure 2 (a)]. In this example, CPU wastes at least 3 cycles if it needs something from main memory. A small, fast, and

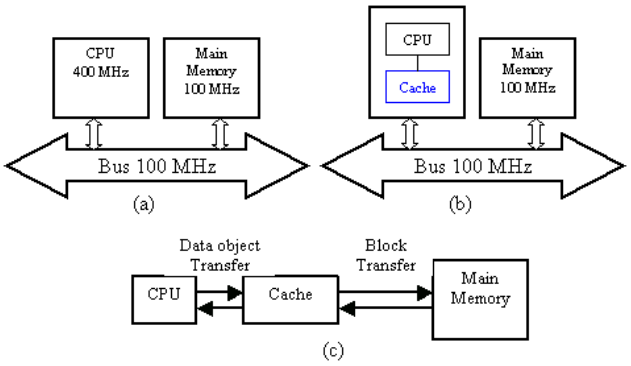

Figure 2. (a) CPU, main memory, and bus; (b) Processor cache; (c) Data transfer between CPU and cache and between cache and main memory

 ISBN: 1-56555-284-9

expensive memory, called cache, is used between CPU and main memory to improve performance by reducing the data access time as shown is Figure 2 (b). Data between CPU and cache is transferred as data object and between cache and main memory as block as shown in Figure 2 (c). In our simulation, we assume a block size to be equal to a frame size

It is a difficult challenge to implement an application-specific system that fulfills desired functionality, and at the same time optimize various design metrics. The difficulty is increased by the time-to-market pressure. The average time-to-market is only 8 months for contemporary application-specific mobile devices. Delayed entry to market causes serious negative effect on revenues. For example, for a lifetime of 52 weeks, a delay of 4 weeks results in a revenue loss of 22% and a delay of 10 weeks results in a loss of 50% [1, 3].

Simulation helps avoid delayed entry of products to the market and improves efficiency and productivity [4]. In this work, we focus on architecture exploration of an application-specific mobile system by evaluating the system performance via simulation.

This paper is organized as follows: Section 2 presents the architecture we used for our simulation. In Section 3, we explain the application to be supported by this proposed architecture. Simulation analysis is done in Section 4. In Section 5, we discuss the simulation results. Finally, we conclude our work in Section 6. At the end, VisualSim simulation block diagram and simulation cockpit is attached as Appendix A.

2. ARCHITECTURE

2.1 Simulated Architecture

Our focus is on architecture exploration of application-specific multiprocessor mobile system by evaluating the system performance via simulation. We simulate a simplified architecture with two processors as shown in Figure 3. This architecture is designed to support video communication applications. In this particular case, two processors are considered. Digital signal processor (DSP) decodes the encoded video streams and application processor (AP) plays it back. Both DSP and AP have first level caches (CL1). We are interested to investigate the impacts of various CL1 sizes on system performance. DSP, also, has its local memory that works as a buffer. DSP and AP use inter-processor communication (IPC) with a register-based messaging unit (MU) and a shared memory system. Here, main memory is being shared by DSP and AP and they are connected via a shared bus.

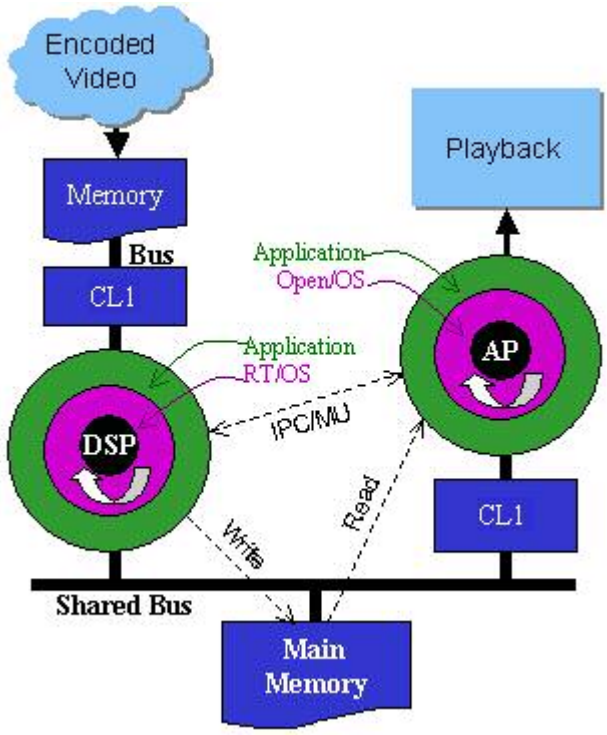

Figure 3: Simulated architecture of application specific multiprocessor mobile systems

DSP writes the decoded video streams into the main memory and sends a message to AP. AP reads the video streams and plays them back. For simplicity, we consider DSP dedicated memory and shared main memory are unlimited in size. For larger memory, delay associated with memory read and write can be ignored [5].

2.2 Processors

Our architecture consists of two processors. We use digital signal processor (DSP) to indicate a microprocessor specifically designed to perform digital signal processing and application processor (AP) to indicate a microprocessor specifically designed to perform application processing.

2.3 Caches

In our architecture, both DSP and AP processor have first level caches (CL1). Cache sizes of 32 KB, 64 KB, and 128 MB already have been used. We vary CL1 sizes from 384 KB to 1024 KB to see the overall system performance.

2.4 Memory

DSP, in our architecture, has a dedicated local memory which functions as a buffer. DSP may perform read from and write into this local memory. Main memory is shared by both DSP and AP. However, DSP only writes

into main memory and AP only reads from it. Both memory sizes are considered unlimited for the sake of simplicity.

2.5 Buses

We use a dedicated bus to connect DSP and its local memory and a shared bus to connect DSP, main memory, and AP. For simplicity, we consider similar properties (such as speed and queue size) for both buses.

3. APPLICATIONS

3.1 Video Communications

We consider video communication applications like video conferencing. At its most basic level, compression is performed when an input video stream is analyzed and information that is less significant to the viewer is discarded. Each event is then assigned a code - commonly occurring events are assigned few bits and rare events will have more bits. The transmitter encodes and transmits the encoded video streams, the receiver decodes the encoded video streams and play them back as shown in Figure 4.

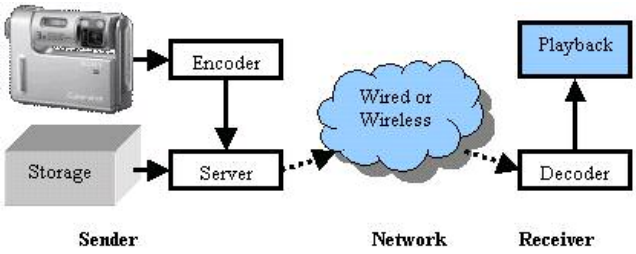

Figure 4: Video communications

Video compression standards include Motion Picture Experts Group 1 (MPEG-1) (up to 1.5 Mbits/sec, CD-ROM video applications), MPEG-2 (between 1.5 and 15 Mbits/sec, digital television applications), MPEG-4 (multimedia and web applications). We use Common Intermediate Format (CIF YUV 4:2:0, 30 fps) in our simulation [6].

We consider MPEG4 application. Specifically, the video file is formatted with Common Intermediate Format (CIF), YUV 4:2:0, width 352 pixels, height 288 lines, and 30 frames per wall-clock second. All three pictures types, namely intra-coded (I frame), predictive picture (P frame), and bi-directionally predictive picture (B frame), are considered. For our simulation we use a group of picture (GOP) that has 7 picture frames as shown in Figure 5.

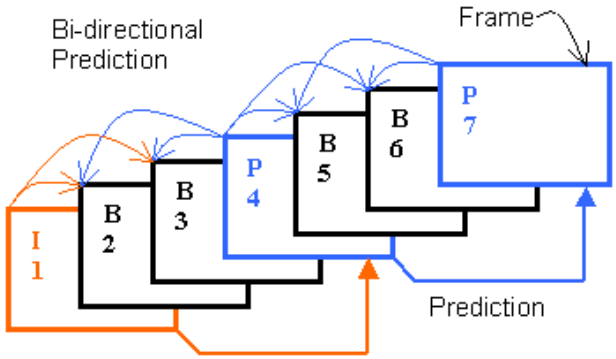

Figure 5: Sample MPEG4 picture frames

I frame (frame 1) is intra coded, that means, it does not have any predictive coding. As a result, random access is supported for I frame. Frame 4 (P) is predicted from 1 (I). Frames 2 (B) and 3 (B) are predicted from 1 (previous I frame) and frame 4 (next P frame). Frame 4 (P) is decoded before 2 (B) and 3 (B). Frame 4 (P) is predicted from 1(I) and frame 7 (P) is predicted from 4 (P). Frames 5 (B) and 6 (B) are predicted from 4 (P) and 7 (P).

P frames can be decoded from previous I (or P) frame. Both previous I (or P) and next P frames are needed to decode any B frame. It is important that for a GOP the encoding, transmission, and decoding order is the same. The playback order is in the natural sequence (1, 2, 3, 4,) and different from decoding order.

Decoding order of the frames = 1, 4, 2, 3, 7, 5, 6

Playback order = 1, 2, 3, 4, 5, 6, 7

Structure (at the encoder) is usually specified using two parameters, M and N. An I frame is decoded every N frames and a P frame every M frames, the rest are B frames. In the simulation, we select N = 7 and M = 3 (as shown in Figure 5) with the consideration that the prediction error does not exceed a certain threshold [7].

3.2 Representative Workload

The workload defines all possible scenarios and environmental conditions that the system-under-study will be operating under [4, 8, 9]. The quality of the workload used in the simulation is important for the accuracy and completeness of the simulation results. In our simulation, we use cache hit ratio and miss ratio to model the system. The hit ratio and miss ratio are calculated based on a decoded trace file of an MPEG4 application. The decoded trace file we generate using the Microsoft MPEG-4 Video Reference Software does not have memory references available to be used. Hence, for this study, we stay on the

conservative side by assuming that the data transfer rate between CPU and cache is in frames. As an extension for this work, we plan to generate the memory references by using the Microsoft MPEG-4 Video Reference Software.

For DSP, the hit ratio and miss ratio are calculated based on the cache sizes and the MPEG4 decoding algorithm. Cache size is finite and fixed for a specific architecture. Frame size is also finite and fixed for a specific algorithm. When DSP decodes a P frame, it looks for I frame into the cache. If cache has that I frame then it is considered a hit. If cache is big enough to fit 2 frames at the same time, then it will always be a hit to decode a P frame. To decode a B frame is different. In the case of a B frame, if cache can fit only two frames, then there will be at least one miss. But if cache is big enough to fit 3 or more frames, then B frame can be decoded from cache. Also, it is assumed that the reference information in both P and B frames result in negligible misses.

The AP cache is assumed to use a pre-fetching scheme that in the event of a miss will pre-fetch the maximum number of frames that the cache can fit. So, if the AP cache size is N, then there is one miss per N frames and the hit ratio is (N-1)/N and the miss ratio is 1/N.

Total number of bytes (B) in a MPEG4 CIF YUV 4:2:0 encoded file is N*3*width*height/2. Here N is the total number of frames. So, for CIF YUV 4:2:0 352 pixels by 288 lines encoded video stream,

Frame Size = 3*352*288/2 bytes □152 KB

Hits and misses are shown in Table 1 – 6.

Table 1: DSP cache hit/miss for cache size of 384 KB. Cache can hold 2 frames at most (384/152 □2).

Frame seq. number	1	4	2	3	7	5	6
Decoding order	I	P	B	B	P	B	B
Hit or Miss (H/M)	M	H	M	M	H	M	M

So, DSP cache hit ratio = 2/7 □28.0% and miss ratio = 5/7 □72.0%

Table 2: AP cache hit/miss for cache size of 512 KB. Cache can hold (384/152 □2) i.e., 2 frames at most.

Frame seq. number	1	2	3	4	5	6	7
Playback order	I	B	B	P	B	B	P
Hit or Miss (H/M)	M	H	M	H	M	H	…

For large number of frames, 1 miss per 2 frames. So, hit ratio = 1/2 = 50.0% and miss ratio = 1/2 = 50.0%

Table 3: DSP cache hit/miss for cache size of 512 KB. Cache can hold 3 frames at most (512/152 □3).

Frame seq. number	1	4	2	3	7	5	6
Decoding order	I	P	B	B	P	B	B
Hit or Miss (h/m)	M	H	H	H	H	H	H

So, DSP cache hit ratio = 6/7 □86.0% and miss ratio = 1/7 □14.0%

Table 4: AP cache hit/miss for cache size of 512 KB. Cache can hold 3 frames at most (512/152 □3).

Frame seq. number	1	2	3	4	5	6	7
Playback order	I	B	B	P	B	B	P
Hit (h) or Miss (m)	M	H	H	M	H	H	…

For large number of frames, 1 miss per 3 frames. So, hit ratio = 2/3 □67.0% and miss ratio = 1/3 □33.0%

Table 5: DSP cache hit/miss for cache size of 1024 KB. Cache can hold 6 frames at most (1024/152 □6).

Frame seq. number	1	4	2	3	7	5	6
Decoding order	I	P	B	B	P	B	B
Hit or Miss (h/m)	M	H	H	H	H	H	H

So, DSP cache hit ratio = 6/7 □86.0% and miss ratio = 1/7 □14.0%

Table 6: AP cache hit/miss for cache size of 1024 KB. Cache can hold 6 frames at most (1024/152 □6).

Frame seq. number	1	2	3	4	5	6	7
Playback order	I	B	B	P	B	B	P
Hit (h) or Miss (m)	M	H	H	H	H	H	…

For large number of frames, 1 miss per 6 frames. So, hit ratio = 5/6 □84.0% and miss ratio = 1/6 □16.0%

Table 7 summarizes the hit ratio and miss ratio of different CL1 sizes for both DSP and AP.

Table 7: Hit ratio and miss ratio for DSP and AP caches.

Cache size (KB)	**Max. Frames the cache can hold**	**DSP**		**AP**	
		Hit (%)	**Miss (%)**	**Hit (%)**	**Miss (%)**
384	2	28	72	50	50
512	3	86	14	67	33
1024	6	86	14	84	16

 ISBN: 1-56555-284-9

As shown in Table 7, DSP does not take significant advantage of increasing cache size from 512 KB to 1024 KB, so the hit ratio and miss ratio remain unchanged. But AP cache hit ratio increases from 67 to 84. This is because to decode a B frame, DSP may need access to at most 2 other frames (previous I (or P) frame and next P frame). So having more than 3 frames into the cache does not improve hit ratio over having exactly 3 frames into the cache.

4. SIMULATION

4.1 Assumptions

The following assumptions are made for the simulation model.

1. Even though AP runs under general purpose operating system, it is capable of handling the traffic in sync with DSP (runs under real-time operating system).
2. Sizes of both DSP local memory (buffer) and shared main memory are unlimited. That means, there is no delay involved with the actions of reading from buffer, writing into main memory, and reading from main memory.
3. Data transfer between CPU and cache is in frames and cache block size is equal to a frame size.
4. DSP and AP use IPC/MU to communicate with each other without any delay.
5. CIF YUV 4:2:0 formatted (352 pixels by 288 lines, 30 fps) video stream file has been used as a representative MPEG4 application.

4.2 Performance Metrics

We measure the following three performance metrics – utilization, mean delay, and transactions.

Utilization: The CPU utilization is defined as the ratio of the time that CPU spent computing to the time that CPU spent transferring bits and performing un-tarring and tarring functions [1, 4]. The CPU utilization ranges from 0% to 100%, in real system from 40% (lightly loaded) to 90% (heavily loaded). As a rule of thumb, a utilization of 50% is considered acceptable [10].

Mean delay: Mean delay is the average delay of all the tasks. Delay (or latency) is the time between the start of execution of a task and the end. Delay is measured in terms of simulation time units [1, 4, 10].

Transactions: Total number of transactions processed is the total number of tasks performed (entered and existed) by a component [1, 4].

4.3 Input Parameters

We vary the following parameters as input to the simulation model – cache size and task rate.

Task rate is the total number of tasks completed per simulation time unit. Task time is the time that one task needs to be processed. Considering 30 frames per wall-clock second playback speed, one frame should take 1/30 wall-clock second. Table 8 lists all the parameters used in the simulation.

Table 8: System parameters

Number of processors	2
Simulation time	10000.0 simulation time units
Task time	10.0 simulation time units
Task rate	Task time * (1.0 to 0.2)
CPU time	Task time * 0.6
Memory time	Task time * 0.4
Bus time	Memory time * 0.4
Cache time	Memory time * 0.6
Cache sizes	384 to 1024 KB
Cache hit ratio	28% to 86%
Bus queue length	300
Block size = frame size	152 KB

To increase comparison visibility, we make the assumption that 10.0 simulation time units = 1/30 wall-clock seconds. Task time has been distributed among CPU time, (main) memory time, bus time, and cache time proportionally as listed in Table 8 [4].

4.4 Simulation Model

We use VisualSim, a simulation tool from Mirabilis Design, Inc., to simulate our architecture. Detailed simulation block diagram and simulation cockpit are shown in Appendix A.

Block diagram is drawn for system components (such as DSP, AP, cache, bus, and memory) using VisualSim blocks. We use parameters for blocks as shown in Table 2. Proper connections are made to simulate the architecture. Random numbers are generated to represent tasks. A generated number is filtered based on the hit ratio to simulate the workload. As an example, say hit ratio is 80% and randomly generated number is between 1 and 100. If the random number is between 1 and 80, it represents a hit; otherwise it is a miss. VisualSim simulation cockpit provides functionalities to run the model and to collect simulation results.

5. RESULTS AND DISCUSSION

In this research work, we investigate the impacts of various CL1 sizes and task rates on system performance in terms of utilization, mean delay, and total number of transactions processed. We use MPEG4 CIF YUV 4:2:0 (352 pixels by 288 lines) formatted video stream file at 30 fps. We vary cache size from 384 KB to 512 KB and from 512 KB to 1024 KB, task rate from 1 task per 10 simulation time units to 1 task per 4 simulation time units, DSP cache hit ratio from 28% to 86%, and AP cache hit ratio from 50% to 84%. Total simulation time is 10000.0 simulation time units and bus queue length is 300.

5.1 Cache size variation

We keep the system with bus queue size of 300 and task rate of 1 task per 10 simulation time units. We change the cache sizes to see the impacts on utilization, mean delay, and total number of transactions processed for bus, memory, DSP, and AP.

First, we consider cache size versus utilization as shown in Figure 6. For cache size of 384 KB, we see that only two frames can be kept into the cache at a certain point of time. We know B frames are predicted based on both previous I (or P) frame and next P frame. As a result there would be more cache misses. When cache size was increased to 512 KB, three frames fitted into the caches at the same time and the miss rate was reduced significantly. Then we increase cache size to 1024 KB. It is noticeable that utilization of bus, memory, and AP dropped considerably. But DSP utilization did not change significantly. This is because, even though DSP cache can have 6 frames at the same time, it can use at most three.

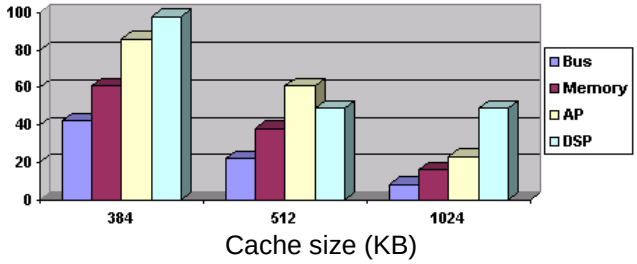

Figure 6. Utilization (%) versus Cache Size with task rate 0.1 simulation time units

Then we run our simulation with increased task rate. Figure 7 shows the results for 1 task per 6 simulation time units. Results for cache size of 384 KB is not shown, because DSP utilization goes beyond 100% that is not possible. As a result simulation fails. For cache size of 512 KB, both utilizations become extremely high. At this task rate, if cache size is increased to 1024 KB, AP utilization reduces significantly, but DSP utilization remains unchanged due to the fact that DSP does not take advantage of having more than 3 frames into the cache at the same time.

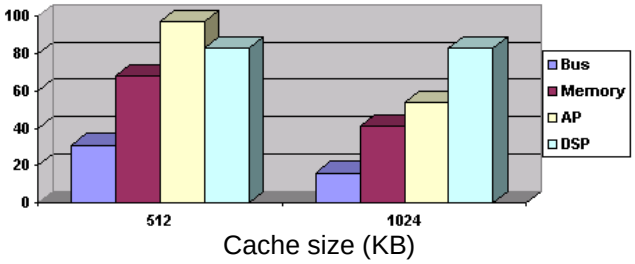

Figure 7. Utilization (%) versus Cache Size with task rate 0.2 simulation time units

Second, we consider cache size versus mean delay (simulation time units) as shown in Figure 8. For DSP and AP delay is significant for cache size of 384 KB. As cache sizes increase, delay decreases. In our simulation, bus and memory delay did not change, since each has a constant transaction processing speed.

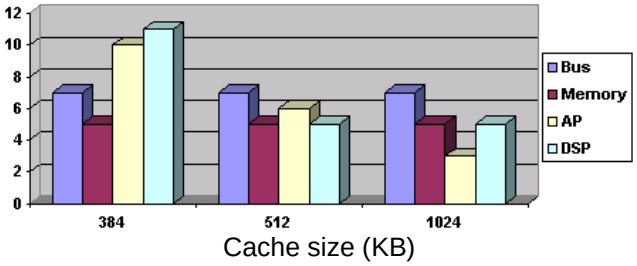

Figure 8. Mean-delay versus Cache Size (KB)

Finally, we investigate the impact of various cache sizes on the total number of transactions processed (Figure 9). We measure transaction as the total number of tasks entered into plus tasks exited from the component during the whole period of simulation period [4]. For DSP and AP, the total number of transactions processed remains unchanged with the variation of cache sizes. Total number of transactions processed by DSP = Total number of transactions processed by AP = 2 * total number of tasks generated.

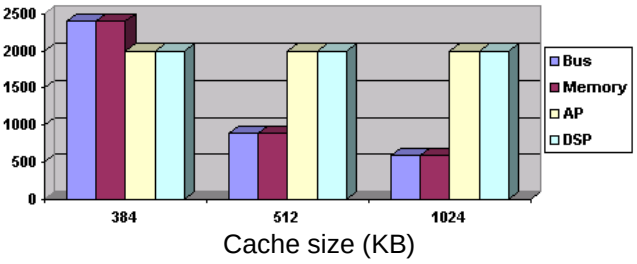

Figure 9. Transactions (tasks entered and exited) versus Cache Size

On the other hand, for bus and memory, cache size has significant effect on the total number of transactions processed. As shown in Figure 9, the total number of transactions processed decrease with increase of cache size for bus and memory.

From Figures 6, 8, and 9, we observe that for cache size of 384 KB, DSP and AP utilization and delay and bus and memory transactions are very high. On the other hand, for cache size of 1 MB, DSP and AP utilization and delay and bus and memory transactions are very low. So for our architecture cache size of 512 KB is optimal. According to Figure 7, when task rate increases utilization also increases. After certain point, utilization goes close to 100%. Increased cache size from 512 KB to 1024 KB reduces utilization for AP but not for DSP.

5.2 Task rate variation

We keep the system with cache size of 512 for both DSP and AP and bus queue length 300.

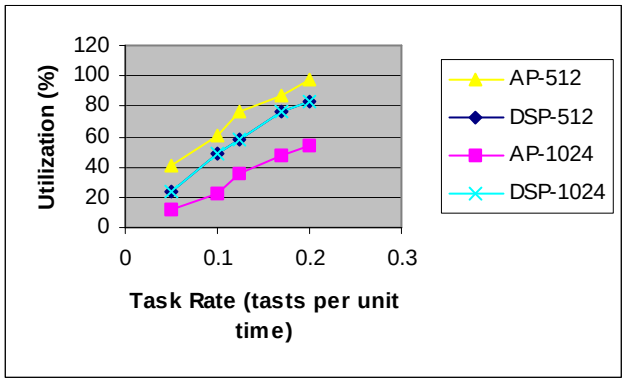

Figure 10. Utilization Vs Task Rate

We increase task rate from 1 task per 10 simulation time units to 1 task per 4 simulation time units. When task rate is increased, utilization increases. At task rate of 1 task per 5 simulation time units (0.2), AP utilization is almost 99% as shown in Figure 10. Further increase in task rate causes the system to break down. AP cannot process the tasks, so task queue starts dropping the tasks. A bigger cache size (for example, from 512 KB to 1024 KB) and/or a bigger bus queue length (for example, 300 to 500) should reduce the utilization for AP. DSP utilization increases with increased task rate but stay the same when cache size increases from 512 KB to 1024 KB.

6. CONCLUSION

In this paper, we explore the architecture of a multiprocessor mobile system running MPEG4 application. We develop a simulation program using VisualSim to evaluate the system performance in terms of utilization, delay, and total number of transactions processed by the different system components for various cache sizes and task rates. The simulation program helps optimize the cache size for a given task rate.

Including a second level cache will certainly improve the system performance, something that we will investigate in the future. Also, as an extension of this work, using the Microsoft MPEG-4 Video Reference Software, we plan to generate memory reference traces to drive the simulation program.

7. REFERENCES

[1] F. Vahid and T. Givargis, *Embedded System Design – A Unified Hardware/Software Introduction*, John Wiley & Sons, New York, NY, 2002

[2] A.S. Tanenbaum, *Structured Computer Organization*, Prentice Hall PTR, Upper Saddle River, NJ, 4th edition, 1999

[3] D. Chiou, P. Jain, L. Rudolph, and S. Devadas, *"Application-Specific Memory Management for Embedded Systems Using Software-Controlled Caches"*, ACM 1-58113-188-7/00/0006, DAC 2000, Los Angeles, CA, USA

[4] VisualSim: Mirabilis Design, Inc. http://www.mirabilisdesign.com/

[5] G. Brar, S. Kundu, P. Worah, S. Biswas, A. Mukhopadhyay, and A. Basu, *"OaSis: An Application Specific Operating System for an Embedded Environment"*, Department of Computer Science and Engineering and Department of Mathematics, IIT Kharagpur, India http://www.mla.iitkgp.ernet.in/papers/oasis.pdf

[6] R. Schaphorst, *Videoconferencing and Videotelephony – Techonology and Standards*, Artech House, Norwood, MA, 2nd edition, 1999

[7] S.R. Ely, "MPEG video coding - A simple introduction", EBU Technical Review Winter 1995

[8] A. Maxiaguine, S. Kunzli, and L. Thiele, "Workload Characterization Model for Tasks with Variable Execution Demand", Project supported in part by KTI/CTI, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland

[9] A. Avritzer, J. Kondek, D. Liu, and E.J. Weyuker, "Software Performance Testing Based on Workload Characterization", WOSP '02, July 24-26, 2002 Rome, Italy, AT&T Labs, ACM ISBN 1-1-58113-563-7 02/07, 2002

[10] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts*, Sixth Edition, John Wiley & Sons, Inc. ISBN 0-471-25060-0

Appendix A: VisualSim simulation block diagram and simulation cockpit

Simulation tool used is VisualSim from Mirabilis Design, Inc. [http://www.mirabilisdesign.com/].

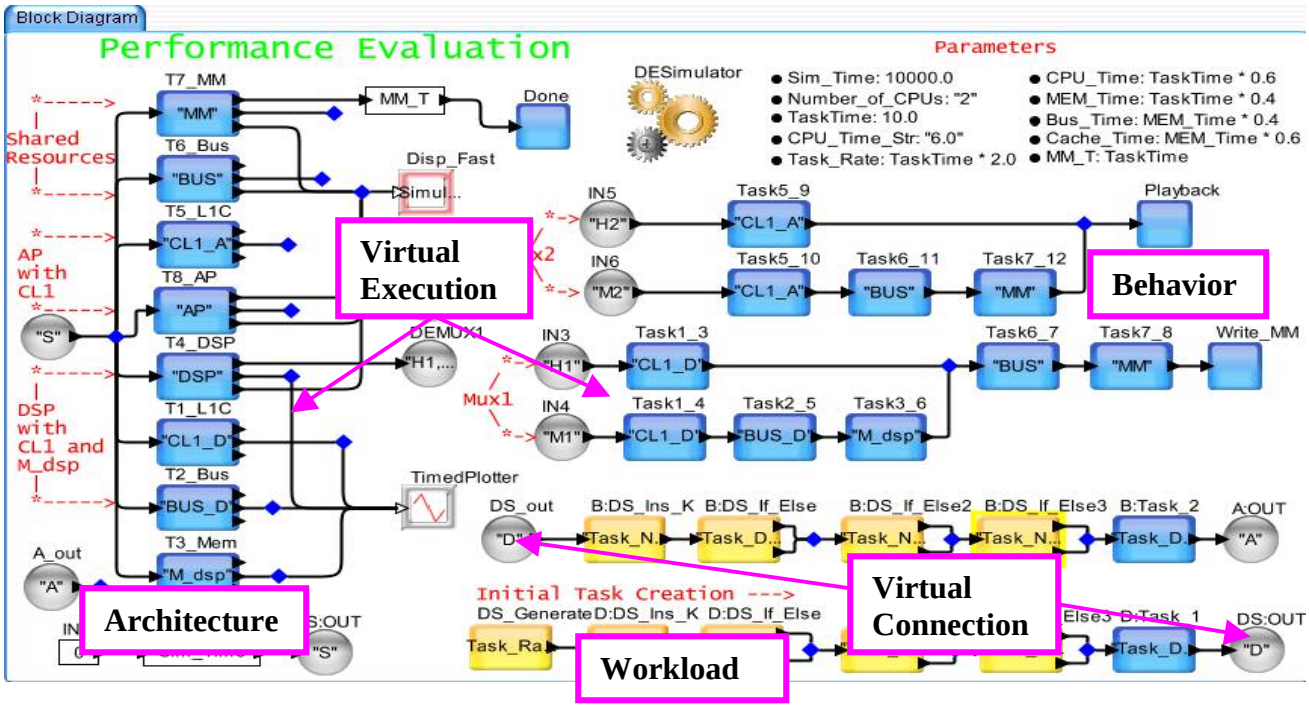

Figure A. VisualSim Block Diagram. The system to be evaluated can be described in three parts – Architecture, Behavior, and Workload. Architecture: Elements such as CPU, cache, bus, memory, and RTOS are specified here. Behavior: This describes the actions performed on the system. Examples include network traffic shaping. Workload: Transactions that traverse the system such as network traffic. Mapping between behavior and architecture is performed using Virtual Execution. Connection can be made using dedicated and/or Virtual Connections. The virtual execution capability makes re-mapping from hardware to software to ASIC by just changing a parameter. The output of a block can be displayed or plotted. In Figure B, output of shared bus (T6_Bus) is shown in the right-top window and the output of DSP memory (M_dsp) is shown in the right-bottom window.

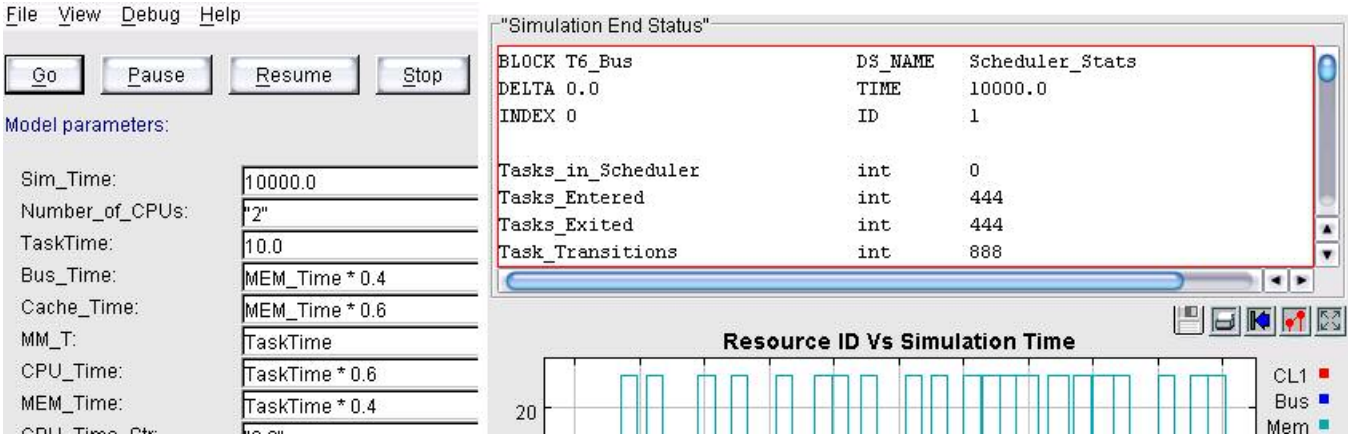

Figure B. VisualSim Simulation Cockpit. The Simulation Cockpit provides functionalities (left window) to run the model (block diagram) and to collect simulation results (right window). Parameters can be changed before running the simulation without modifying the block diagram. The final results can be saved into a file and/or printed for further analysis.