Reducing Communication Overhead for Streaming Application by Memory-Aware Scheduling on Mpsoc

Ms. Niveditha. K
PG Scholar, Department of ECE
Hindusthan Institute of Technology,
Coimbatore, Tamilnadu, India

Mrs. Kalpana. K
Assistant Professor, Department of ECE
Hindusthan Institute of Technology,
Coimbatore, Tamilnadu, India

***Abstract*—Reducing communication overheads for streaming application by task scheduling. Communication overhead not exclusively increases the temporal property performance but in addition they decrease the memory usage for streaming applications. By minimizing communication overhead a smaller amount all often applied and system performance improved. Streaming application method step by step data are modeled as dependent tasks. For steaming application method massive buffers are in-between tasks. The target is to minimize communication overhead whereas minimizing the memory usage. To resolve the matter inside information dependencies remodel to in-between information dependencies to overlap the execution of computation and communication tasks. By doing this communication overhead all removed. Based on the schedulability analysis formulate the scheduling problem as an integer linear programming model and obtain an optimal solution. Based on the analysis obtain heuristic memory-aware optimal task scheduling to efficiently obtain a near optimal solution.**

Keywords—Streaming application,IntegerLinear programming, Task scheduling

I. INTRODUCTION

Multiprocessor systems-on-chip (MPSoC) with a large number of different processing cores are now common. MPSoC consists of multiple heterogeneous processing elements, memory hierarchies, and input/output components. All these components are linked to each other by an on-chip interconnect. This architecture meets the performance needs of Telecommunication, network security and other application. Streaming application is a form of on-demand software distribution. Only essential portions of an application code need to be installed on the computer. Application streaming is a related concept to application virtualization, where application are ran directly from a virtual machine on a central server that is completely separate from the local system. Communication overhead is one of the most critical design issues in embedded systems. In this paper, we focus schedulability analysis ILP model based Memory-Aware Optimal Task Scheduling (MAOTS) with objective is minimizing the overall memory usage and then propose a Heuristic Memory Aware Task Scheduling(HMATS) to obtain a near optimal solution. Experimental results show that the planned technique will achieve 25% and 30% reduction in memory usage and conjointly 16% reduction in optimal solution. Then it implements a simulator based on the processor model ARM7 MPcore processor.

II. PROPOSED SYSTEM

Streaming applications that method streams of information are typically shapely as periodic dependent tasks, within which streams of information are communicated from task to task. Communication overhead not exclusively increases the temporal property performance but in addition they decrease the memory usage for steaming applications. In existing system round robin scheduling is used, the disadvantages of round robin scheduling are larger waiting time and response time and also low throughput. In some cases length of the quantum and number of processes are very large at that time overhead may occur. The overhead in communication causes delay in process, jitter and distortion this may cause output delay. In this method increasing the memory usage. In this paper we reduce the communication overhead for application by another scheduling method. The existing system has increasing the communication overhead and increasing the memory usage. In this paper only reduce the inter-core communication overhead by pre-emptive scheduling. Pre-emptive scheduling is where the process is interrupted to stop it allow another process to run. Each process gets a time slice to run in; at the point of each context switch a timer will be reset and will deliver and interrupt when the time slice is over. The scheduler can decide which process to run next. Tasks are usually assigned with priorities. The proposed system has so many advantages pre-empts the currently executing process, all multitasking operating systems use pre-emptive scheduling and many multiprocessor systems also employ pre-emptive inter-task scheduling when they run parallel computations.

III. METHODOLOGY

Many real-time operating systems have been created computing systems. MPSoC in contrast generally require their core functions to be implemented in a very small amount of software both for performance and memory limitations. Real-time operating system processes in application from interfacing with one another by placing them in tasks. It assigns priorities to tasks according to their importance in the application and switches between them using a scheduling algorithm.

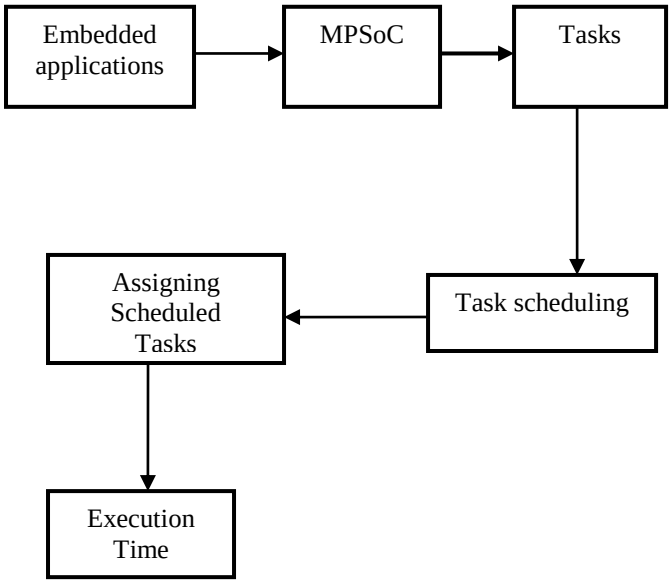

Figure 1 Overall Description

1. Embedded Applications

The embedded application is given to the MPSoC that consist of multiple processors. The application is then divided in to number of tasks. These tasks are scheduled among the processors. Finally the execution time has to be found. The embedded applications are telecommunication, image processing and MP3 players etc... Many embedded applications are commonly referred to as streaming applications. From Figure 1 Overall Description

2. MPSoC

The MPSoc architecture consists of M processor cores $\{P_1,P_2,\ldots..P_m\}$,a hierarchical backbone bus a bus arbiter and a shared memory. Every processor core is connected via a bus to a shared memory, and access requests from processor cores are managed by the bus arbiter. From Figure 2 This target architecture has been widely adopted in many embedded platforms. They are two types of MPSoC architectures heterogeneous MPSoC and homogeneous MPSoC.

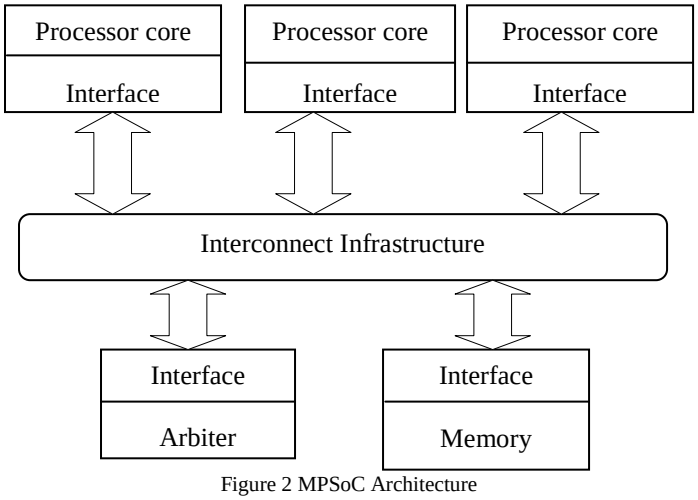

Figure 2 MPSoC Architecture

3.Tasks

A task is like a process or thread in an Operating System (OS). Task term used for the process in the embedded systems. A task consists of executable program, state of which is controlled by OS. The state during running of a task represented by information of process status process structure its data, objects and resources and task control block. From Figure 3 there are four tasks and the execution time of each task is two time units. There are four edges and each edge associates with one communication task.

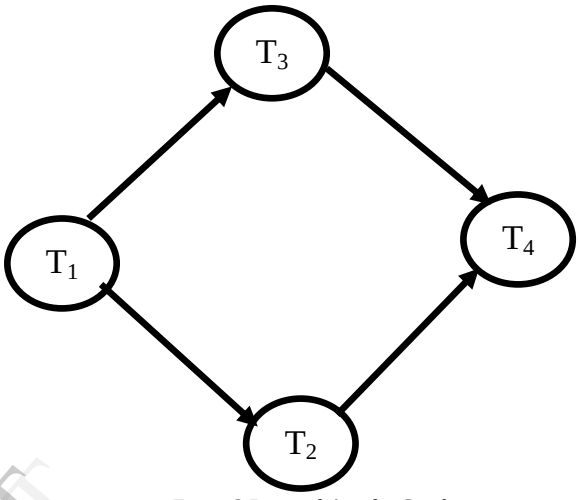

Figure 3 Directed Acyclic Graph

4. Task Scheduling

A running system has many processes, may be even into the hundreds. The part of the kernel that keeps track of all these processes is called the scheduler because it schedules which process should be run next. Pre-emptive scheduling is where the process is interrupted to stop it an allow another process to run. Each process gets a time slice to run in; at the point of each context switch a timer will reset and will deliver and interrupt when the time slice is over. The scheduler can decide which process to run next. Tasks are assigned usually assigned with priorities. To run a certain task that has a higher priority before another task although it is running. Then running task is interrupted for some time and resumed later when the priority task has finished its execution.

5. Execution Time

In this paper we consider only four tasks are allocated that execute in same system with two processor and shared memory. In each task set have different task priorities. And also set dependent of each tasks. Then each task execute in priority based. All tasks are completed by average waiting time about 25 microseconds.

IV. RESULTS AND DISCUSSION

Directed Acyclic Graph (DAG) shown in figure 3 is considered and implemented in Linux operating system. Consider there are four tasks and also each task assigned by priority based. DAG graph consist four edges and each edge associates with one communication task T_1, T_2, T_3 and T_4 are tasks.

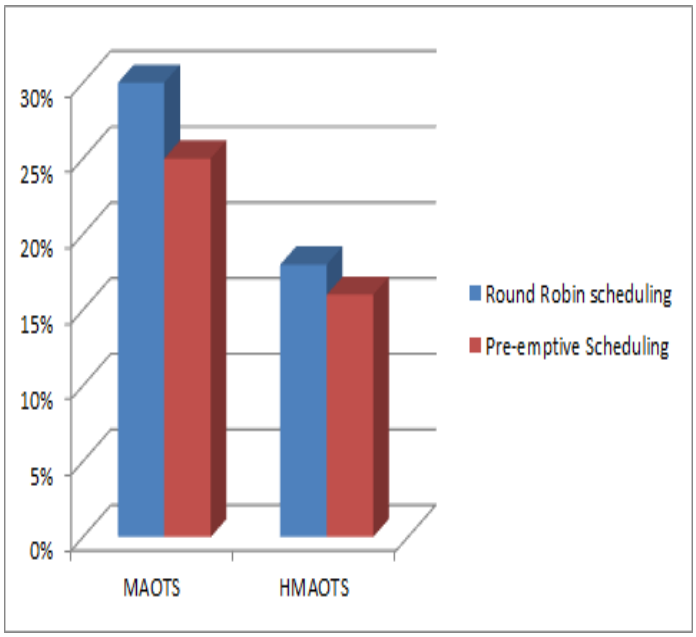

Figure 4 Comparison between two scheduling

The above graph figure 4 shows that comparison between two scheduling, the proposed technique can achieve 25% in memory usage, optimal solution achieve in proposed technique is 16%. The proposed heuristic approach can obtain near optimal memory usage with less time compared with the ILP-based optimal solution.

V. CONCLUSION

The task scheduling problem of communication overhead for streaming applications is totally removed by proposed technique. We experiments on a set of benchmarks from streaming application. The experimental results show that the proposed approach can significantly reduce the optimal solution and improve the memory usage. The results among different processors depend on the task schedule on each of those processors and vice versa.

ACKNOWLEDGMENT

Author is sincerely thankful to the department of Electronics and Communication Engineering in Hindusthan Institute of Technology for providing experimental setup.

REFERENCES

1. Y.Wang,D.Liu,Z.Qin and Z.Shao, "Memory-aware optimal scheduling with Communication overhead minimization for streaming applications on chip multiprocessors," in proceedings of the 31st IEEE Real-Time Systems Symposium(RTSS'10),2010,pp.350-359.
2. I.Assayad and S.Yovine, "A Scheduler synthesis methodology for jointHW/SW design exploration of SOC," Design Automation for Embedded Systems,Vol.14,no.2,pp.75-103,2010.
3. M.H.Wiggers,M.J.G.Bekooij,and G.J.M.Smit, "Buffer capacity computation for throughput constrained streaming applications with data dependent inter-task communication," in proceedings of the 14th IEEE Real time and Embedded Technology and Applications Symposium(RTSS'08),2008,pp.183-19.
4. K.S.Vallerio and N.K.Jha,"Task graph extraction for embedded system synthesis," in proceedings of the 16th international conference on VLSI Design (VLSID'03),2003,pp.480-486.
5. I.Issenin and N.Dutt," Data reuse driven enerey-aware MPSoC cosynthesis of memory and communication architecture for streaming applications," in proceedings of the 4th international conference on HW/SW codesign and system synthesis (CODES+ISSS'06),2006,pp.294-299.
6. Y.Zhang,X.S. and D.Z.Chen,"Task scheduling and voltage selection for energy minimization," in proceeding of the 39th Design Automation Confrence (DAC'02),2002,pp.183-188.
7. R.Xu,R.Mlhem, and D.Mosse,"Energy-aware scheduling for steaming applications on chip multiprocessors," in proceedings of the 28th IEEE international Real-time systems symposium(RTSS'07),2007,pp.25-38.
8. F.Sun,S.Ravi,A.Raghunanthan,and N.K.Jha,"Synthesis of application specific heterogeneous multiprocessor architectures using extensible processors" in proceedings of the 18th international conference on VLSIDesign(VLSID'05),2005,pp.551-556.
9. Y.Guo,D.Zhu and H.Aydin,"Reliability-aware power mamnagement for parallel real time applications with precedence constraints," in 2001 International Green Computing Conference (IGCC'11),July 2011,pp.1-8.
10. G.Varatkar and R.Marculescu,"communication –awre task scheduling and voltage selection for total systems energy minimization," in proceedings of the 2003 IEEE/ACM International Conference on Computeraided Design(ICCAD'03),2003,pp.510-517.
11. J.Hu and R.Marculescu,"Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints," in proceedings of the conference on Design,Automation and Test in Europe (DATE'04),2004,pp.234-239.
12. J.Zhu,I.Sander and A.Jantsh, "Energy efficient streaming application with gauranteed throughput on MPSoCs," in proceedings of the 8th ACM International Conference on Embedded Software (EMSOFT'08),2008,pp.119-128.