DYNAMIC HW PRIORITY QUEUE BASED SCHEDULERS FOR EMBEDDED SYSTEM

Dinesh G Harkut[1] and Dr. M. S. Ali[2]

[1]Department of Computer Science & Engineering, Prof Ram Meghe College of Engineering & Management, Badnera-Amravati. 444 701,M.S. (INDIA).

[2]Principal, Prof Ram Meghe College of Engineering & Management, Badnera- Amravati. 444 701, M.S. (INDIA).

ABSTRACT

A real-time operating system (RTOs) is often used in embedded system, to structure the application code and to ensure that the deadlines are met by reacting on events by executing the functions within precise time. Most embedded systems are bound to real-time constraints with determinism and latency as a critical metrics. RTOs are generally implemented in software, increases computational overheads, jitter and memory footprint. Modern FPGA technology, enables the implementation of a full featured and flexible hardware based RTOs, which helps in reducing to greater extent these overheads even if not remove completely. Scheduling algorithms play an important role in the design of real-time systems. An Adaptive Fuzzy Inference System (FIS) based scheduler framework proposed in this article is based on the study and conclusion drawn from the research over the years in HW SW co-design domain. The proposed novel two phase FIS based adaptive hardware task scheduler minimizes the processor time for scheduling activity which uses fuzzy logic to model the uncertainty at first stage along with adaptive framework that uses feedback which allows processors share of task running on multiprocessor to be controlled dynamically at runtime. This Fuzzy logic based adaptive hardware scheduler breakthroughs the limit of the number of total task and thus improves efficiency of the entire real-time system. The increased computation overheads resulted from proposed two phase FIS scheduler can be compensated by utilising the basic characteristics of parallelism of the hardware as scheduler being migrated to FPGA.

KEYWORDS

Hardware scheduler, Job priority, Real-time operating system, Reconfigurable computing, Scheduling Algorithms, ANFIS, HW/SW co design.

1.INTRODUCTION

Technology innovations drive the today's consumer market. Many technologies that were not available a few years ago are quickly being adopted into common use. The technological advances of microelectronics have radically changed, among many others, the scenario of modern embedded real-time systems. Embedded devices are special purpose processor often designed to serve their unique purpose. These processor finds application in almost variety of products within different technical areas such as industrial automation, consumer electronics, automotive industry and communications and multimedia systems. Product ranging from train and airplanes to microwave ovens and washing machines are controlled by embedded systems. Sharp decrease in semiconductor prices and coupled with increase in performance, catalyst the rapid increase in the complexity of embedded applications. Furthermore, the high level of integration of silicon technology opens up many interesting possibilities also for the world of re-programmable hardware platforms. In embedded systems a real-time operating system (RTOS) is often used in order to structure the application code and ensure that deadlines are met. Moreover, RTOS also reduces the development time of complex embedded applications through hardware abstraction and multitasking. Further, intensified market pressure to rapidly develop cheaper product can also be address by using RTOS in embedded system product development but at the cost of several forms of overheads.

One of the constant challenges for real-time system designers is building a platform that can meet the timeliness requirements of the system where mere logical accuracy of its computation will not serve its purpose [1]. Thus the primary design goal of the RTOses is to minimizing the overheads, reduce latency and maximizing the determinism This paper is organized as follows. Section 2 is an overview of the Hardware/Software co design approaches. Section 3 describes related work of other research projects, proposed model is discussed in section 4 and section 5 is summary and conclusion from mainly previous work and related work.

2. HARDWARE SOFTWARE CO-DESIGN ARCHITECTURE

In embedded systems, RTOs is often used in order to structure the application code and ensure that deadlines are met. The notions of best-effort and real-time processing have fractured into a spectrum of processing classes with different timeliness requirements including desktop multimedia, soft real-time, firm real-time, adaptive soft real-time, rate-based and traditional hard real-time [2-3]. Real Time systems may be Hard-real time where missing deadline is catastrophic or Soft-real time where, occasional violation of deadline may not result in useless execution of the application but decreases utilization [4].

Traditionally RTOS's are implemented in software, but major drawbacks of standard software based RTOS's is that they suffer from computational overheads, jitter and often a large memory footprint. RTOS computational overheads is caused mainly by tick interrupt management, which get even worse with more task and high tick frequencies, but also task scheduling , resource allocation and de-allocation, deadlock detection and various other OS/API functions take execution time from the task running on the CPU.

The overhead due to the software kernel contributes to the degradation the overall performance of RTOS based embedded systems at the cost of high flexibility. On the other hand, task implemented as hardware modules placed in Hardware have the characteristics of high performance along with low flexibility and high cost.

The trend of utilizing reconfigurable FPGA's, which can be programmed absolutely infinite number of times, is drastically increasing. FPGA supports for high fabrication density, parallel computing and low cost as compared to ASIC made it possible to implement established software algorithms i.e. real-time kernel activity like scheduling, inter-process communications, interrupt management, resource management, synchronization and time management controls in hardware and thus consequently decreases system overhead, improve predictability and increases response time.

As a tradeoffs reconfigurable and hardware/software co-design approaches that offer real time capabilities and enhanced predictability of hardware and flexibility of software to support increasing complex systems become more feasible solution. (Figure 1).

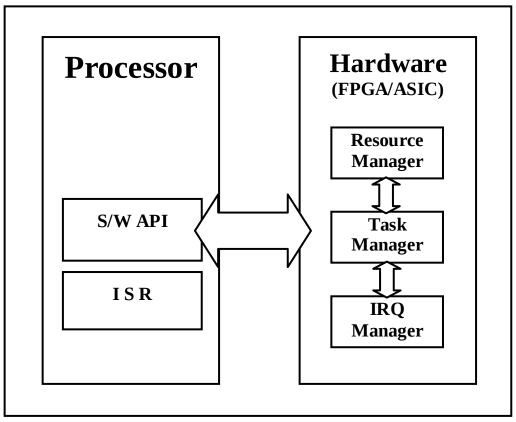

Figure 1 – Hw/Sw System Architecture

Recent advancements in FPGA technology have made it more economically feasible to explore migrating functionality across the hardware/software boundary. The flexibility of the FPGA fabric and availability of configurable soft IP components has opened the potential to rapidly and economically investigate different hardware/software partitions. This approach reach a level of maturity that are allowing system designers to perform hardware/software co-design of operating system core functionality such as time management and task scheduling that allow the advantages of higher level program development while achieving the performance potential offered by executions of these functions in parallel hardware circuits.

3. LITERATURE REVIEW

The main source of indeterminism in real time systems are speculative components like varying instruction cycle time caused by pipeline, branch predictions, varying execution time of RTOs kernel functions, external asynchronous interrupts etc. Migrating major real time kernel functionality from software to hardware, it is possible to remove jitter, lessen CPU overhead and improve the predictability of the system. Over the period of time, various attempts have been made to design the models and systems to overcome these problem and some of them were discussed in remaining section.

Lennart Lindh *et al.* [5-6] proposed a system FASTCHART; consist of hardware based RT kernel capable of handling 64 task with 8 different priorities. FASTCHART is a RISC based uniprocessor system which uses ID's of tasks with various queues to make efficient implementation feasible.

POLIS – A model proposed by F. Balarin, G. Berry, F. Boussinot *et al.* [7-8], is a Co-Design Finite State Machine (CSFM) synthesis model, supports globally asynchronous and locally synchronous computation. The POLIS generates C-code for processor and the optimized hardware. Entire implementation is splits between Software and ASICs. This model does not support large and complex design as large and complexity of processors makes the static estimation difficult. However, Co-simulation is provided using Ptolemy environment [9].

FASTHARD - a hardware based general purpose processors, proposed by Lindh *et al.* [10], is extension to his earlier work FASTCHART. Like FASTCHART, system is limited in supports for customization and scalability but supports features like rendezvous, external interrupts, periodic start and termination of task without CPU interference.

The COSYMA system is HW/SW co-design fine or coarse grained partitioning model design proposed by R. Ernst *et al.* [11-12], aims speedup software executions to meet timing constraints. It uses simulated annealing for partitioning along with profiling and symbolic approaches are used to calculate timing information. Though it does not support burst-mode communication, uses list and path based are used to estimate execution time of hardware.

RTU (Real Time Unit), a multi-processor system proposed by J. Adomat *et al.* [13] uses single interrupt input of each CPU to control and context switching. Lindh *et al.* [14] also proposes extensible multiprocessor system - SARA, which can be used together with RTU to remove the all scheduling and tick processing overheads.

T. Samuelsson *et al.* [15] benchmark the RTU, SARA on Rhealstone. The result seems to be more deterministic. J. Lee *et al.* [16] further integrate the RTU with δ-framework for co-design. In [17] S. Nordstrom *et al.* adapt µC/OS II RTOS with uniprocessor to boost the performance. S. Nordstrom *et al.* [18] configurability is added and commercialize by Prevas AB. RTU suffers from some of the limitations as it lacs support for counting semaphore, mutex, deadline detection/preventions & dynamic priorities which limits its practical usefulness.

STRON system, which basically based on and extension to µTRON project is proposed by T. Nakano *et al.* [19]. In this system basic system calls and functionality has been implemented in hardware kernel which results in increasing speedup and reducing jitter. Some of the features not supported by hardware are implemented in a small micro kernel. Unbounded priority inversion and lack of support for high granularity tick frequency are the basic limitation of STRON.

In [20-21], R. Gupta *et al.* developed VULCAN - Hardware/Software partitioning tool, which runs in polynomial time, uses heuristic graph partitioning algorithm. The ultimate goal is to minimize hardware cost while maintaining timing constraints. The original description was in Hardware-C [22], which is mapped to fine grained Control-Data Flow Graph. Test results are missing in the paper.

P. Chou *et al.* [23-24], proposes automated interface synthesis hardware software co-design framework for embedded system- CHINOOK. It ensures timing constraints [25] and also supports mapping of an embedded system model to one (or more) processor and peripherals.

H. De. Man *et al.* proposed heterogeneous hardware/software DSP system COWARE in [26], which is originally based on [27] and is basis of commercial CoWare N2C. COWARE system allows co-specification using existing languages VHDL, DFL, Sliage & C but imposes increased demands on generation of exhaustive library elements. This model supports the re-use and encapsulation of hardware and software by a clear separation between functional and communication behaviour of a system components.

In [28] Bjorn B. Brandenburg *et al.* discuss the LITMUS^{RT} project is a soft real-time extension of the Linux kernel with focus on multiprocessor real-time scheduling and synchronization. The primary focus is to provide a useful experimental platform for applied real-time systems research. It supports the sporadic task model with both partitioned and global scheduling [29]. LITMUS^{RT} is not yet stable interfaces.

A.Parisoto *et al.* [30] suggested F-Timer framework which is FPGA based task scheduler targeted at general purpose processor, capable of managing 32 tasks with 64 different priorities. System does not have any hardware support for task synchronization and resource handling. Paper also does not discussed about scheduling algorithm employed.

J. Stankovic *et al.* [31] takes a radically different approach task scheduling than normal RTOS's. Spring kernel is basically designed for large and complex multiprocessor based RTOS. Task management is based on dynamic and speculative planning based scheduling implement through heuristic algorithm and tree search which makes it capable to handle fine granularity of task deadlines. However large amount of pre-calculation overheads become the major bottleneck for overall speedup of the system.

J. Hildebrandt *et al.* describes hardware implementation of dynamic scheduling coprocessor in [32]. It is basically hardware scheduling accelerator which can be configured for several different algorithms along with the most advanced Enhanced Least Laxity First (ELLF). This system has increases the overall determinism but at the cost of higher complex logic and could no address trashing of task.

In [33-34], I. Mooney *et al.* presents hardware/software co-design RTOs framework, δ-Framework which supports 30 different processors. This framework generates all HDL code which can be implemented in FPGA. The system is cost effective as far as overall speedup and hardware area (number of gates) is concerned. More work on SOC was conducted [35] to integrate priority inheritance and deadlock avoidance mechanism.

V.Mooney *et al.* [36], design and developed configurable hardware scheduler to improve response time, interrupt latencies and CPU utilization and supports high tick frequency. The interrupt controller in scheduler supports 8 external interrupts each can be configured for dispatching a specific task and supports three different algorithms which can be change at run time.

Issues of extension to OS and flexibility arises out of moving entire OS to hardware can be overcome in model propose by Z. M. Wirthlin *et al.* in [37]. The nano-processor provides upgradability, flexibility and also enhancing the execution time by moving selected inefficient OS services in hardware to save on power consumption to a great extent as shown in several studies. Leveraging the potential of hardware parallelism, Paul Kohot *et al.* in [38], developed Real-Time Manager (RTM). Routine housekeeping tasks are implemented in hardware and thus relieve the processor for critical functions which boosts the overall performance. RTM supports static priority scheduling and handles task, time and event management. The author claims RTM decreases RTOS overheads by 90% decreases response latency by 81%.

In [39] M.Vetromille *et al.* describes how low tick granularity can cause jitter and result in deadline misses. HaRTS supports high tick frequency and thus reduce jitter without lower CPU available time for task to process. It requires less chip area and uses less power than additional processor but more complex to implement.

Communication speed between RTOS and hardware overshadowed the speed gain by hardware scheduler. This problem has been overcome by intelligent design proposed by S. Chandra *et al.* in [40]. The Hardware RTOS implemented for accelerating eCos, HW-eCos is interfaced to an ARM processor requires less gates to implement and provides better speedup. Paper does not discuss the number of tasks and resources supported by this system.

Z. Murtaza, S. Khan *et al.* describe SRTOS in [41] is aim at real-time DSP application targeted on AVZ21 DSP processor. However paper doesn't provide any experimental test result but system supports additional instruction for fast resource allocation and context switching.

H-Kernel system describe by M. Song *et al.* [42], is an outcome of through use of FPGA and thoughtful HW/SW co-design for specific application. Nevertheless H-Kernel is suitable for system with small numbers of task and increases performance in the tune of 50-60%, system become more complex and bulky as number of task increases.

ARPA-MT system proposed by A. S. R. Oliveira *et al.* in [43] presents multi threading processor with five stage pipeline. It supports heterogeneous task and context switches without hampering the processor performance.

Luis Almeida *et al.* in [44-45] describe the details about OReK_CoP, Hardware implementation of OReK Real-Time Kernel. All kernel functions execute in absolute time and almost in parallel, without interfering CPU – improves determinism and improve resource utilization. Better and more direct connections between CPU and coprocessor would have removed quiet large latency introduced due to PLB bus interface in this system.

Many HW SW based solutions have been proposed to reduce the context switching time [46]. Xaingrong Zhou, Peter Petrov *et al.* presented model by converging compiler, micro-architecture and OS kernel to reduce the context switching cost and improve overall responsiveness. In this system context switching may be deferred until next switch point to limit the number of context registers required to hold state. This arrangement may result in more deadline miss which can be avoided by more complex and good RTOS kernel design.

N. Maruyama *et al.* in [47] proposed architecture ARTESSO, where RTOS, checksum calculation, memory copying and TCP header rearrangement are ported to hardware. It uses novel virtual queue instead of FIFO based queues used in RTU and STRON, which are logic expensive. The author claims that this system is 6-9 times faster than STRON and 7 times more energy efficient than its software counterpart.

Numbers of research projects have been targeted to approach the task of designing OS for FPGA based reconfigurable computers (RC). Hayden Kwok-Hay *et al.* [48-49] describe BORPH, an operating system designed for FPGA-based RC which provides native kernel support for FPGA hardware and offers a homogeneous UNIX interface for both software and hardware processes. All the services from the software kernels are inherited in hardware processes. Performance of real-time wireless digital signal processing system based on BORPH will be presented.

HartOS proposed by Lange A.B. *et al.* is Hardware implemented Real-Time Operating System) in [50] is designed to be very flexible and support most of the features normally found in a standard software RTOS directly in hardware without sacrificing flexibility. HartOS has been compared to the commercial Sierra kernel, its performance outperforms that of the Sierra Kernel [51]. The HartOS the ability to let the kernel run at a higher clock frequency than the microprocessor, which allows more tasks to be processed serially at the same tick frequency, and speed up the part of the API functions executed in the kernel.

Scheduling algorithm plays as important role in the design of real-time systems which involves allocation of resources and time to jobs in such way that certain performance requirements are met. Most of the model discussed and reviewed are mainly focused on to improve the performance by migrating some of the house keeping routine jobs from software to hardware with a aim to leverage the potential of parallel processing of hardware which can further be improved to a greater extent if more realistic scheduling algorithm is devise and migrate it on hardware to assist processor and RTOs so as to increase the overall performance without increasing memory footprint and power consumptions.

Comparative study of various methodologies/models reviewed in the literature is given in the Table1[52].

4. PROPOSED ANFIS BASED TASK SCHEDULER

We are proposing ANFIS based hardware scheduler framework which is discussed in subsequent paragraph basically consist: Global Fuzzy scheduler – Long term scheduler and Local Adaptive scheduler – Short term scheduler.

Both of these scheduler work in cascade and are migrated on hardware which will work in synchronous with processor and RTOs to fulfil the overall systems objectives as illustrated in figure 2.

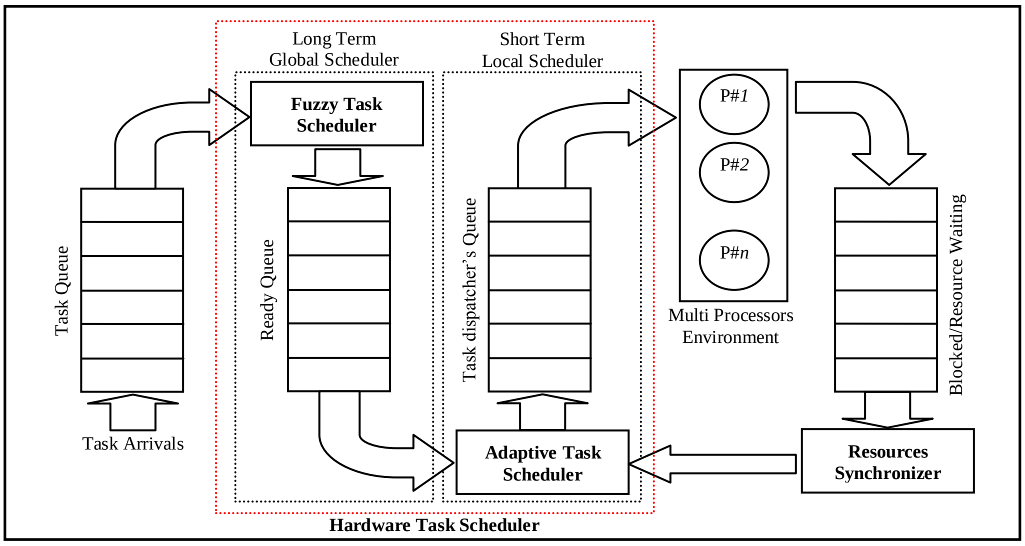

Figure 2 Proposed ANFIS Task Scheduler

In this framework, task queue is maintained in FCFS basis which is feed to Fuzzy Task scheduler which acts as long term global scheduler. To build a fuzzy system, inputs and output(s) to it must be first selected and partitioned into appropriate conceptual categories which actually represent a fuzzy set on a given input or output domain. Job priority, deadline and CPU time are selected as input to the Fuzzy Inference System (FIS) [53-54], which consist of five stages:

1. Fuzzification of inputs variables
2. Applying fuzzy operators
3. Applying implication methods
4. Aggregating outputs variables
5. Defuzzification of output variables

Here Madani's Fuzzy inference method or TSK or simply Sugeno method of fuzzy inference may be used [55]. Output is single value which is treated as Job Processing Priority (JPP) and maintained in job ready queue as per the newly calculated JPP which in turn feed to Adaptive Task Scheduler a second stage scheduler.

Table 1 - Comparative study of various methodologies/models

Methodology/ Model	Category	Main Objectives	Architecture Used	Claims by Authors	Shortcoming
FASTCHART [6] (1991)	Hybrid	To improve deterministic & remove jitter.	RISC based processor with Load Store architecture	Migrated full kernel to Hardware	Lacks interrupts capabilities, pipelining, cache & supports single processor.
POLIS [7] (1991)	Hybrid	To design dashboard controller with a mixed implementation of SW & ASICs.	Co-design Finite State Machine (CFSM)	Flexibility to evaluate HW/SW partitioning, architecture & scheduler.	Static estimation is difficult (complexity of processors). No support for large design.
FASTHARD [10] (1992)	Hybrid	To develop H/W based RT kernel for general purpose processor.	Memory mapped address/data bus scheme.	Support external interrupts & rendezvous.	Limited scalability and customization supported. No test result available.
COSYMA [11] (1994)	Hybrid	To speeding up S/W execution to meet timing constraints	Memory mapped address/data bus scheme.	Uses novel list & path-based scheduling to estimate HW execution time	Does not support burst mode communications.
RTU [13] (1994-2001)	HW	To reduce system overhead and improve predictability	Memory mapped VME bus scheme.	Supports multiple task, binary semaphores, event flags, watchdogs.	Does not supports dynamic priority & bus latency time slows the response time.
Silicon TRON [19]1995	Hybrid	To increase timing predictions & reduce/remove jitter	Memory mapped address/data bus scheme.	Supports task mgt., flags, semaphores, timers & external interrupt.	No mechanism to prevent unbounded priority inversion.
VULCAN [20] 1995	Hybrid	To reducing ASIC hardware cost	Control-Data Flow Graph based fine grained mapping.	Hardware/software partitioning results in reducing the overall cost.	Limited scope as support bus-oriented fixed architecture only.
CHINOOK [23]1996	Hybrid	To provide automated interface synthesis	Distributed architecture.	Supports mapping of processor & peripherals with strict timing constraints	Protocol of selected port incompatible & hence inflexible.
COWARE [26] 1996	Hybrid	To provide a design environment for heterogeneous HW/SW DSP systems	Memory mapped address/data bus scheme.	Supports re-use, encapsulation of HW & SW by separation of functional behavior.	Issues like multiple threads execution, deadlock detection needs to address.
LITMUSRT [28] 1996	S/W	To provide testbed to evaluate different RT scheduler on multicore platforms	Push/pull approach as std Linux RT scheduler.	Supports G-EDF based scheduling with private queue for each processor.	Maintainability & portability is major issues.
F-Timer [30] 1997	Hybrid	To reduce RTOS overheads and improve determinism	Memory mapped address/data bus scheme.	Supports external interrupts	No HW support for task synchronization or resource handling.
Spring Coprocessor [31] 1999	Hybrid	To design systems with guaranteed scheduling without blocking resources.	Memory mapped address/data bus scheme.	Supports fine granularity of task deadlines & multiprocessors.	Small overall speed as long time is used pre-calculations on EAT of all resources.
ELLF Scheduling Coprocessor [32] 2000	H/W	To developed scheduling accelerator by exploiting parallelism in HW.	Memory mapped address/data bus scheme.	Supports ELLF algorithm with dynamic priority calculation.	Relatively complex and high logic consumption.
The δ-Framework [33]2002	S/W	To provide framework targeted for HW/SW co design.	Memory mapped address/data bus scheme.	Uses less nos. of gates for equivalent HW area.	It just provides framework, design details missing.
Mooney [33] 2003	Hybrid	To design & develop configurable scheduler.	Both memory mapped & instruction set acceleration.	Supports Priority based, Rate monotonic & EDF algorithms & high tick rate.	Inflexible and benchmark results are not available.
Nano-processor [37] 2003	Hybrid	To increasing flexibility & compatibility with range of hardware.	Memory mapped address/data bus scheme.	Provides flexibility of choosing services to perform in HW with faster execution.	Needs some optimization to reduce power consumption.
Real-Time Task Manager	Hybrid	To improve performance by migrating routine task to HW.	Memory mapped address/data bus scheme.	Supports static priority & handles task, time & event mgt.	Hardware scales linearly with number of records.
Real-Time Task Manager [38] 2003	Hybrid	To improve performance by migrating routine task to HW.	Memory mapped address/data bus scheme.	Supports static priority & handles task, time & event mgt. with same tree.	Hardware scales linearly with number of records.
HaRTS [39] 2006	Hybrid	To reducing jitter with increase in granularity.	OPB bus scheme.	Requires less power, less chip area and supports high tick frequency.	Design is very complex to implement.
HW- eCos [40] 2006	Hybrid	To improve system performance & reduce the code size of the RTOS.	Memory mapped address/data bus scheme.	Removes context switching overheads through interrupt line to CPU.	Chip-to Chip slow communication speed become bottleneck in overall performance.
Silicon RTOS [41] 2006	Hybrid	To design efficient real time DSP applications.	Memory mapped address/data bus scheme.	Supports external interrupt management & uses priority based scheduling	Use of register file for each task limits the usage for system with many tasks.
H-Kernel [42] 2007	Hybrid	To gain large performance through thoughtful HW/SW co design.	Memory mapped address/data bus scheme.	Supports priority based task, interrupt, event & time mgt through H-kernel.	Support small number of task. Complexity increases with increase in task numbers
ARPA-MT [43] 2009	Hybrid	To produce specialized predictable & customizable processor.	Stack based priority ceiling.	Supports heterogeneous task & schedules using RM or EDF protocol.	Comparatively difficult to achieve claimed determinism & resource efficiency.
OReK_CoP [45] 2009	Hybrid	To port OReK kernel to HW coprocessor to improve performance.	PLB bus interfacing with stack based priority ceiling.	Supports asynchronous interrupt handling which improve determinism	Comparatively higher latency time and supports only binary semaphores.
Xiangrong et al [46] 2010	S/W	To improve responsiveness by reducing the context-switching time.	Micro-architecture & OS kernel.	Uses micro-architecture to lower context switching.	Design is very complex and consumes more power.
ARTESSO [41] 2010	Hybrid	To improve throughput by moving TCP header calculation & checksum to HW.	TCP/IP Protocol.	Supports priority based FCFS scheduler & uses novel virtual queue structure.	Comparatively less reliable and scalable.
BORPH [49] 2010	S/W	To compatible with FPGA based reconfigurable hardware.	Unix based OS uses virtual file system.	Reduces context switching drastically by exploiting the benefits of parallelism.	Hardware is more complex to design and maintain.
HartOS [50-51] 2011	Hybrid	To reduce jitter, computational overheads & memory footprint.	Coprocessor/stream & FSL-AXI stream interface.	Interrupt handled as task & mutex are protected by stack based priority ceiling.	Support for deadlock preventions and flag management missing.

The accuracy of schedulability analysis of system is largely depends on the accuracy with witch the worst-case execution time (WCET) have been derived which is nothing but the maximum amount of time each job can take to execute. The disadvantage of using WCETs under traditional task model like periodic, sporadic etc., is that system may be deemed un-schedulable even if they would function correctly most of the time when deployed. This drawback can be overcome by making our scheduler adaptive to the runtime varying conditions, to allocate per-task processors time share, instead of always using constant share allocation based on constant WCET and readjusting the priority of task. Overall quality-of-service (QoS) can be improved by ignoring the transient overload conditions. Dispatcher will dispatch the task from task dispatcher queue to processors bank to get serve. Further resource synchronisation is used to optimise the block on resource queue. Processors share allocations are adjusted using feedback and resource synchronisation techniques [56].

This proposed fuzzy-adaptive feedback based scheduling algorithm based framework in capable of handling multiprocessor environment. As this entire scheduling task is proposed to be port on hardware (FPGA), increased computation requirement can be compensated by parallelism of hardware with additional advantage of increase in determinism and lessening burdening of the processor, which results in increase in overall performance of the system.

5. CONCLUSION

The conclusion from a comprehensive literature review of the publication throughout the last three decades, is that the major drawback from software based RTO's can be removed by implementing the entire/ partial kernel of a real-time operating system in hardware. High fabrication density, parallel computing capability, flexibility and low developmental cost results in drastic increase in utilization of reconfigurable FPGA for implementation of a full featured and flexible hardware based RTOs.

Implementing a real-time kernel in hardware makes it possible to draw benefits from hardware characteristics such as parallelism and determinism. The execution time of real-time functions gets deterministic and task switch can be performed without any CPU time delay [6,30]. When real-time kernels are implemented in software, one of the disadvantages is that the execution time for the service calls will have a minimum and a maximum time. The time gap can be big and the worst-case time is one of the factors that will decide the utilization factor of the system. Moreover, speculative component along with varying number of tasks, complex scheduling algorithm further decrease the determinism. In hardware, the time gap can be designed to be close to 0, which leads to predictable time behaviour, simpler timing analysis of the system and almost no overhead. It is also easier to debug tasks since different protection modes are not required [57-60].

A hardware kernel executes in parallel to the CPU, which relieves pressure from the CPU which gets almost 100% execution time for the application tasks. There is less software code in memory since the functionality is implemented in hardware instead [30].

A software OS will generate a clock tick interrupt to the CPU when either it is executed or the lists of tasks (queues) are worked at or new periodic delay times are calculated for the tasks. With the hardware kernel in the system, it checks all queues concurrently and only generates an interrupt to the CPU when there is to be a task switch [57,61]. Another advantage of having the kernel in hardware is the possibility to use complex scheduling algorithms, unlimited of different queue types without any performance loss. Also there is an improved understandability and complexity reduction when the system is divided into parts [57-58]. ANFIS based hardware task scheduler will definitely results in increasing the overall performance of the system. The simulation results showed that the implementation of task management by hardware kept the correctness of system call, at the same time reduced the execution time of system calls and the overhead of processor. Implementation details and result analysis is beyond the scope of this paper.

REFERENCES

[1] D.Stewart, “Introduction to Real Time”, Embedded systems programming, CMP Media, November 2001.

[2] Z.Deng, J.W. Liu and S. Sun, “Dynamic scheduling of hard real-time application in open system environment”, Tech. Rep., University of Illinois at Urbana-Champaign 1996.

[3] S.M.Petters, “Bounding the execution time of real-time task on modern processors”, in Proceeding of 7th International Conference Real-Time Computing Systems and Applications, Cheju Island, pp. 498-502, 2000.

[4] J.Zhu, T.G. Lewis, W. Jackson and R.L. Wilson, “Scheduling in hard real-time applications”, IEEE software, Volume 12, pp. 54-63, 1995.

[5] L.Lindh, F. Stanischewski, “FASTCHART - Performance, Benefits and Disadvantages of the Architecture”, in Proceeding of 5th Euromicro Workshop on Real-Time Systems, 1993.

[6] L.Lindh, and F. Stanischewski, “FASTCHART- Idea and Implementation”, in IEEE International Conference on Computer Design (ICCD), Boston, USA, Oct.1991.

[7] F.Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki and B. Tabbara. “Hardware-Software Co-Design of Embedded Systems: The POLIS Approach”, Kluwer Academic Publishers, 1997.

[8] G.Berry and G. Gonthier. “The Esterel Synchronous Programming Language: Design, Semantics, Implementations”, Journal Science of Computer Programming archive, Elsevier North-Holland, Inc. Amsterdam, The Netherlands, Volume 19 Issue 2, pp. 87-152, Nov. 1992.

[9] J.T.Buck, S. Ha, E.A. Lee and D.G. Messerschmitt. “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems”, International Journal of Computer Simulation, special issue on “Simulation Software Development”, pp.155-182, April 1994.

[10] L.Lindh, “FASTHARD - a fast time deterministic hardware based real-time kernel”, in Proceedings of Real-Time Systems, 4th Euromicro workshop, pp. 21-25, June 1992.

[11] R Ernst, D Herrman, J. Henkel, Th. Benner, W. Ye, U. Holtmann and M. Trawny. “The COSYMA environment for hardware software co-synthesis of small embedded systems”, IEEE Micro, pp.159-166, 1996.

[12] R.Ernst, D. Herman, A. Osterling, T. Benner, T. Scholz and W. Ye. “The COSYMA system”, in Hardware Software Co-Design: Principles and Practice , Kluwer Academic Publishers,1997.

[13] J.Adomat, J. Furunas, L. Lindh, and J. Starner, “Real-time kernel in hardware RTU: a step towards deterministic and high-performance real-time systems”, in Proceedings of the 8th Euromicro Workshop on Real-Time Systems, L'Aquila, pp. 164-168, Jun. 1996.

[14] L.Lindh, T. Klevin, L. L. T. Klevin, and J. Furunäs, “Scalable architecture for real-time applications sara”, in CAD & CG’99, pp. 208-211, 1999.

[15] T.Samuelsson, M. Åkerholm, P. Nygren, J. Starner, and L. Lindh, “A comparison of multiprocessor real-time operating systems implemented in hardware and software”, in International Workshop on Systems Implemented in Hardware and Software, Advanced Real-Time Operating System Services (ARTOSS), 2003.

[16] J.Lee, I. Mooney, V.J., A. Daleby, K. Ingstrom, T. Klevin, and L. Lindh, “A comparison of the RTU hardware RTOS with a hardware/software RTOS”, in Proceeding ASP-DAC '03 Design Automation Conference Asia and South Pacific, ACM New York, USA, pp.683-688, Jan. 2003.

[17] S.Nordstrom, L. Lindh, L. Johansson, and T. Skoglund, “Application specific real-time microkernel in hardware”, in proceedings of 14th IEEE-NPSS Real Time Conference, pp. 4-9, Jun. 2005.

[18] S.Nordstrom and L. Asplund, “Configurable hardware/software support for single processor real-time kernels”, pp. 1-4, Nov. 2007.

[19] T.Nakano, A. Utama, M. Itabashi, A. Shiomi, and M. Imai, “Hardware implementation of a real-time operating system”, in proceeding of IEEE International Symposium of 12th TRON project, Tokoy, Japan, pp. 34-42, Nov. 1995.

[20] R.Gupta. “Co-Synthesis of Hardware and Software for Digital Embedded Systems”, the Springer International Series in Engineering and Computer Science, Volume 329, 1995.

[21] G.DeMicheli, R. Gupta, D. C. Ku, F. Mailhot and T. Truong. “The Olympus Synthesis System for Digital Design”, Design & Test of Computers, IEEE Volume 7, Issue 5, pp. 37-53, Oct. 1990.

[22] D.C.Ku and G. DeMicheli. “HardwareC – a language for hardware design Ver 2.0” CSL Technical Report CSL-TR-90-419, Stanford, April 1990.

[23] P.Chou, Ross Ortega and Gaetano Borriello. “The Chinook Hardware Software Co-Synthesis System”, in Proceedings of the International Symphosium on System Synthesis, pp. 22-27, Sept. 1995.

[24] P.Chou, E. Walkup and G. Borriello. “Scheduling for Reactive Real-Time Systems”. IEEE Micro archive Journal, IEEE Computer Society Press Los Alamitos, CA, USA. Volume 14, Issue 4, pp. 37-47, August 1994.

[25] D.Ku and G.De Micheli. “High-level Synthesis of ASICs under Timing and Synchronization Constraints”. Kluwer Academic Publishers, Norwell, MA, USA , ISBN:0-7923-9244-2, 1992.

[26] H.De Man, D. Verkest, K. Van Rompary and I. Bolsens. “Coware – A Design Environment for Heterogeneous Hardware Software Systems”, Design Automation of Embedded Systems, pp.357-386, Oct. 1996.

[27] H.De Man, Karl Van Rompaey, Diederik Verkest and Ivo Bolsens. “Coware – A design environment for heterogeneous hardware software systems”, in Proceedings of the European Design Automation Conference, pp. 252-257, Sept. 1996.

[28] B.Brandenburg , J. Calandrino, H. Leontyev, A. Block, U. Devi, and J. Anderson, "LITMUSRT: A Test-bed for Empirically Comparing Real-Time Multiprocessor Schedulers", in Proceedings of the 27th IEEE Real-Time Systems Symposium, pp. 111-123, December 2006.

[29] F.Cerqueira and B. Brandenburg, "A Comparison of Scheduling Latency in Linux, PREEMPT-RT, and LITMUSRT", in Proceedings of the 9th Annual Workshop on Operating Systems Platforms for Embedded Real-Time applications, pp. 19-29, July 2013.

[30] A.Parisoto, J. Souza, A., L. Carro, M. Pontremoli, C. Pereira, and A. Suzim, “F-timer: Dedicated FPGS to real-time systems design support”, in proceeding of 9th Euromicro Workshop on RTS, Toledo, Spain, pp. 35-40, Jun.1997.

[31] J.Stankovic, W. Burleson, J. Ko, D. Niehaus, K. Ramamritham, G. Wallace and C. Weems, “The spring scheduling coprocessor: a scheduling accelerator”, in IEEE Transactions on Very Large Scale Integration Systems, Volume 7, pp. 38-47, Mar. 1999.

[32] J.Hildebrandt, F. Golatowski, and D. Timmermann, “Scheduling coprocessor for enhanced least-laxity-first scheduling in hard real-time systems”, in Proceedings of the 11th Euromicro Conference on Real-Time Systems, pp. 208-215, 1999.

[33] V.Mooney and D. Blough, “A Hardware/Software Real-Time Operating System Framework for SOCs”, Design & Test of Computers, IEEE, Volume 19, Issue 6, USA, pp.44-51, Nov. 2002.

[34] V.Mooney and J. Lee. “Hardware/Software Partitioning of Operating Systems: Focus on Deadlock Detection and Avoidance”, in IEEE Proceeding, Computer and Digital Techniques, UK, pp. 167-182, July 2005.

[35] V.Mooney and J. Lee, , “RTPOS: A Novel Deadlock Avoidance Algorithm and Its Hardware Implementation”, in CODES+ISSS '04 Proceedings of the international conference on Hardware/Software Codesign and System Synthesis, IEEE Computer Society Washington, DC, pp. 200-205, 2004.

[36] V.Mooney III, P. Kuacharoen and M. A. Shalan, “A configurable hardware scheduler for real-time systems”, in Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, CSREA Press , pp. 96-101, 2003.

[37] M.Wirthlin, B. Hutchings, and K. Gilson. “The Nano Processor: a Low Resource Reconfigurable Processor” in IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, pp.23-30, April 1994.

[38] P.Kohout, B. Ganesh, and B. Jacob, “Hardware support for real-time operating systems”, in Proceeding of First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2003), Newport Beach CA, pp. 45-51, Oct. 2003.

[39] M.Vetromille, L. Ost, C. Marcon, C. Reif, and F. Hessel, “RTOS scheduler implementation in hardware and software for real time applications”, in 17th IEEE International Workshop on Rapid System Prototyping, pp. 163-168, Jun. 2006.

[40] S.Chandra, F. Regazzoni, and M. Lajolo, “Hardware/software partitioning of operating systems: a behavioral synthesis approach”, in GLSVLSI ’06 Proceedings of the 16th ACM Great Lakes symposium on VLSI, (NY, USA), pp. 324-329, ACM, 2006.

[41] Z.Murtaza, S. Khan, A. Rafique, K. Bajwa, and U. Zaman, “Silicon real time operating system for embedded DSPs”, in ICET’ 06: Proceedings of International Conference on Emerging Technologies, (Peshwar), IEEE, pp. 188-191, Nov. 2006.

[42] M.Song, S. H. Hong, and Y. Chung, “Reducing the overhead of real-time operating system through reconfigurable hardware”, in proceedings of 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, pp. 311-316, Aug. 2007.

[43] A.S.R.Oliveira, L. Almeida, and A. B. Ferrari, “The ARPA-MT embedded SMT processor and its RTOS hardware accelerator”, Industrial Electronics, IEEE Transactions on, Volume 58, No. 3, pp. 890-904, March 2011.

[44] Luis Almeida, A. S. R. Oliveira and Antonio B. Ferrari. “A specialized and predictable processor for real-time systems”, in Workshop on Application Specific Processors, pp. 32-38, Nov. 2009.

[45] Luis Almeida, Nelson Silva, Arnaldo Oliveira and Rui Santos. "The OReK real-time micro kernel for FPGA-based systems-on-chip", in proceedings of 6th Workshop on Embedded Systems for Real-time Multimedia, (ESTImedia 2008), IEEE Xplore, Atlanta Georgia, pp. 75-80, Oct. 2008.

[46] Xiangrong Zhou, Peter Petrov “Rapid and low-cost context-switch through embedded processor customization for real-time and control applications” DAC San Francisco, CA, pp. 352-357, July 2006.

[47] N. Maruyama, T. Ishihara, and H. Yasuura, “An RTOS in hardware for energy efficient software-based TCP/IP processing”, in IEEE Symposium on Application Specific Processors, pp. 58-63, June 2010.

[48] Hayden Kwok-Hay So, Xun Changqing, Wen Mei, Wu Nan, Zhang Chunyuan. “Extending BORPH for shared memory reconfigurable computers Field Programmable Logic and Applications (FPL)” in 22nd International Conference on IEEE Improving Usability of FPGA-Based Reconfigurable Computers Through Operating System Support, Oslo. pp. 563-566, Aug. 2012.

[49] Hyden Kwok-Hay So, Robert W. Broderson “BORPH: An Operating System for FPGA-Based Reconfigurable Computers” DAC University of California, Berkeley, Technical Report No. UCB/EECS, pp. 92-96, July 2007.

[50] Lange, A.B. “Hardware RTOS for FPGA based embedded systems”, Master's thesis, University of Southern Denmark. http://www.hartos.dk/publications/thesis/hartos.pdf accessed on Nov.2015.

[51] Kohout, P., Ganesh B. and Jacob B. “Hardware support for Real-Time Operating Systems”, in Proceedings of the First IEEE/ACM/IFIP International Conference on HW/SW co-design and System Synthesis, pp.45-51, 2003.

[52] D. G. Harkut & M.S.Ali, “Hardware Support for Adaptive Task Scheduler in RTOS”, Intelligent Systems Technologies & Applications, Volume 384, Springer, UK, pp. 227-245, 2015.

[53] Sabeghi M., Naghibzadeh M., Taghavi T., “Scheduling Non-Preemptive Periodic Task in Soft Real-time Systems using fuzzy Inference”, 9th IEEE International Symposium on Object and component-oriented Real-Time distributed Computing(ISORC), April 2006.

[54] Mahdi Hamzeh, Sied Mehdi Fakhraie, and Caro Lucas, "Soft real-time fuzzy task scheduling for multiprocessor systems", International journal of intelligent technology Vol. 2 No. 4, pp. 211-216, 2007.

[55] Mamdami E. H., Assilian S., “An experiment in linguistic synthesis with a fuzzy logic controller ”, in International Journal of Man-Machine Studies, Vol.7,No.1, pp. 1-13, 1975.

[56] Jang,J.S.R., “ANFIS: Adaptive-Network-based Fuzzy Inference Systems”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, pp. 665-685,1993.

[57] Lindh,L., Stärner, J. and Furunäs, J. “From Single to Multiprocessor Real-Time Kernels in Hardware”, in IEEE Real Time Technology and Applications Symposium. Chicago, May 1995.

[58] Furunäs,J., Adomat, J., Lindh, L., Stärner, J. and Vörös, P. “A Prototype for Interprocess Communication Support, in Hardware in Real-Time Systems”, in Proceedings of EUROMICRO Workshop, Toledo, Spain, pp. 18-24, June 1997.

[59]Lindh, L. “A Real-Time Kernel implemented in one chip”, in IEEE press, Real-Time Workshop , Oulu, June 1993.

[60] Adomat,J., Furunäs, J., Lindh, L. and Stärner, J. “Real-Time Kernel in Hardware RTU: A Step Towards Deterministic and High-Performance Real-Time Systems”, in Proceedings of the Euromicro Workshop on Real-Time Systems, L’Aquila, Italy, June 1996.

[61] Lindh,L. “Utilization of Hardware Parallelism in Realizing Real Time Kernels”, Doctoral Thesis, TRITA – TDE 1994:1, ISSN 0280-4506, ISRN KTH/TDE/FR-94/1-SE, Department of Electronics, Royal Institute of technology, Stockholm, Sweden, 1994, accessed on Nov.2015.

AUTHORS

Dinesh G Harkut received B.E.(Computer Science & Engineering) & M.E. (Computer Science & Engineering) from SGB Amravati University in 1991 and 1998 respectively. He completed his masters in Business Management and obtained his Ph.D. from SGB Amravati University in Business Management in 2013 while serving as a full-time faculty in the Dept. of Computer Science & Engineering at Prof Ram Meghe College of Engineering & Management, Badnera – Amravati. His research interests are Embedded Systems and RTOS.

Dr. M. S. Ali is a Professor and Principal of Prof Ram Meghe College of Engineering & Management, Badnera – Amravati. He obtained his B.E.(Electronics & Power) and M.Tech.(Power Electronics) from Nagpur University and I.I.T. Powai, Mumbai in 1981 & 1984 respectively He obtained his Ph.D. from SGB Amravati University in 2006. He has been on the SGB University's various body like Board of Studies, Faculty of Engineering & Technology and Academic Council since last fifteen years. He is Hon'ble Chancellors nominee on the senate of RTM Nagpur University. His research interests are Operating Systems, Artificial Intelligence and Java Technologies.