ISSN:2229-6107

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY

E-mail :
editor.ijpast@gmail.com
editor@ijpast.in

www.ijpast.in

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 12, Issue 2 May 2022

Hardware/software partitioning in reconfigurable embedded systems: a new approach

Mr. N.SRINIVAS RAO[1],Dr.A.VENKATESWARLU[2],Ms.Y.ANITHA[3], J. Rajyalaxmi[4],Ms.K.Sunitha[5],

Abstract—

The separation of hardware and software is a vital aspect in developing a flexible embedded system. In order to reach the high performance of dedicated hardware, computer architectures that can change their hardware to each application are being designed, and reconfigurable computing is a potential way to resolving the conventional trade-off between flexibility and performance. In this research, we first review and describe existing hardware and software partitioning techniques before proposing a novel approach for task division and scheduling that takes use of the dynamic reconfiguration and delay of reconfigurable hardware. The suggested method divides a massive programme into smaller, more manageable jobs, each of which is related to the others through constraints. And based on the sequence in which the activities were carried out, a directed acyclic graph (DAG) was created to illustrate the connections between them. Then, a method called GATS, which combines the Genetic Algorithm and the Tabu Search algorithm, is used to map the particular application described in the DAG to the hardware and software platform. Priority-based scheduling allows for the quickest possible assignment and execution sequence of tasks. The testing results demonstrate the method's strong performance and its ability to transfer the application task to the reconfigurable system.

Key words :

Task scheduling; Hardware/software isolation; Genetic algorithm; Tabu search method; Reconfigurable embedded system

INTRODUCTION

These days, hardware and software are inseparable components of every electronic system worth its salt. Some sort of computer hardware and software working together to carry out a certain task defines an embedded system. It's used in a wide variety of vehicles, networks, smart homes, hospitals, clinics, and even military applications. In contrast to the physical components, software is simpler and more quickly to create and update. As a result, the time and money required to create software is much lower. However, the performance gains from hardware are substantial. A designer of an embedded system should aim to reduce the total time, space, and energy needed to run the system. Embedded systems may be built in one of two primary ways. Both hardware and software are considered. The hardware approach relies on the construction of specialised hardware logic circuits to complete system functionality, whereas the software approach uses microprocessor software to do the same. Assigning system functions to the desired structure in the software and hardware domain while still satisfying design restrictions is the primary objective of hardware/software partitioning, which is essentially a combination optimization issue. It consists of three parts: allocating processing units (choosing the right software and hardware for the job), assigning tasks (making sure they get done in the right order and at the right time), and scheduling them so that they run as efficiently as possible and within budget. When considering solution quality and solution time, the problem is challenging since the solution space is large and disjoint in many dimensions. The complexity of the issue may be reduced by simplifying the model of the goal structure, and this is done by focusing on the execution time, cost, power, and other important overhead while evaluating the hardware and software partition.

Associate Professor[2,], Assistant Professor,[1,3,4,5,]
Mail Id : normula09@gmail.com,Mail Id : wenkateswarlu@gmail.com, Mail id : verasangianitha478@gmail.com,
Mail Id : rajyalakshmivarshini@gmail.com, Mail Id : sunitha.sunitha250@gmail.com ,
Department of ECE, Swarna Bharati Institute of Science and Technology (SBIT),
Pakabanda Street,Khammam TS, India-507002.

ISSN 2229-6107 www.ijpast.in
Vol 12, Issue 2 May 2022

RELATED PAPERS

Smaller, lighter, less power-hungry, more complicated, and so on are all trends in the embedded system industry thanks to advancements in integrated circuit technology. The progress of embedded system development has been stymied by the persistence of the traditional design methodology. There must be tight integration and coordination between the software and hardware design phases. As a result, the concept of "hardware and software co-design" has emerged. Changes in Hardware and Software Segmentation .Research into hardware/software co-design started in the early 1990s, with the concept being publicly suggested at the inaugural International Workshop on Hardware/Software Codesign (CODES) in 1992. Then, several prestigious institutions began doing theory and research on software and hardware co-design for embedded systems. In addition, certain EDA suppliers have released instruments that facilitate the joint development of hardware and software. Prakash and Parker's SOS system [1] is the first hardware and software co-design system. They created it at the University of Southern California. It's possible to schedule work over many processors, but the technology is too sluggish to be used in large-scale applications. The German Technical University in Braunschweig's COSYMA (Co-synthesis for Embedded Architecture) [2] system is limited to using a single CPU and a single ASIC. Software may make use of the partitioning approach to optimise calculations using co-processors. The lack of simultaneous processing between the CPU and coprocessor is the biggest problem with COSYMA [3]. Frank Sloke's Corsair system [4] is a multi-processor and multi-ASIC-friendly embedded system design environment. The tabu search technique is used to construct the system model. However, the system model evaluation is static, therefore it cannot evaluate dynamic systems. In 1997, Else suggested using simulated annealing and the tabu search technique to separate hardware and software. In order to build the scheduling table and provide a foundation for the choice of software and hardware, he outlined a model called the condition task graph that makes use of a list scheduling algorithm. The tabu search technique outperforms simulated annealing in hardware and software partitioning, according to experimental results [5]. A hardware/software separation technique for IP-based low-power embedded devices was first presented by Henkel in 1999. The goal is to minimise sleep and standby times to cut down on energy use [6]. TherapodTiangong and colleagues examined three heuristic techniques for hardware and software co-design in 2002 [7], and concluded that the tabu search strategy is better in hardware and software partition over the genetic algorithm and the simulated annealing algorithm. A hybrid reconfigurable system was suggested by Michalis D. Galanos in 2006.

Studies on Time Management for Organizing Tasks

Many areas of computer science and telecommunications make use of the scheduling issue, which falls under the category of combinational optimization problems. Algorithm design and complexity theory have many commonalities. Liu and Leyland [9] initially suggested research on task scheduling, however their proposal glosses over certain technical issues. Many models for real-time tasks are based on this concept, and it even goes as far as the processor's environment, where it may be used for scheduling and feasibility analysis during algorithm development. A non-dynamic scheduling approach for periodic activities was presented in reference [10], and it involves suitable grouping. Recent years have seen a rise in platform resource and processor usage as a result of the employment of grouping technique and suitable scheduling policy. China Taiwan National University researcher Hsu Heng Rue studied the dynamic voltage scheduling issue for scheduling periodic tasks in real time under energy limitations [11]. Research into parallel job scheduling on multiprocessors, often represented as directed acyclic graphs (DAGs), has progressed fast during the last 20 years. Scheduling tasks using a directed acyclic graph (DAG) is a method of coordinating the allocation of resources and distributing work across available processors. Overall, the task's execution time, power consumption, area, and other indicators are optimal provided that limitations are met. The issue is intractable in NP-completeness. Since Becchi and Crowley believe that task management is the key to improving the computing performance of a multi-processor platform, they designed a run-time monitoring software to record the process's dynamic behaviour as it moves from one processor to another. Experiments demonstrate that the overall system performance may be greatly enhanced by using the dynamic process allocation technique on a heterogeneous multi-processor platform [12]. Genetic algorithm for multi-processor job scheduling [13] is one example of a novel approach that has been used in recent years to handle the multiprocessor scheduling challenges. Using this new way of computation increases the precision of the final result. However, the algorithm's performance might be enhanced. The

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 12, Issue 2 May 2022

dynamic placement of hardware jobs on the FPGA is a key area of study for task scheduling on CPU + FPGA structures. However, task allocation, task migration, and other difficulties of mixed task scheduling have received far less attention in the literature [14].

FPGA-BASED RECONFIGURABLE SYSTEM

Fast progress in reconfigurable technology for embedded applications may be traced back to the introduction of programmable devices, most notably the field-programmable gate array (FPGA). Because of advancements in reconfigurable technology, the line between hardware and software is becoming more porous. In a software-controlled information processing system, the term "reconfigurable" refers to the system's ability to be transformed into a new information processing system with the use of reusable resources in order to meet the needs of a variety of applications. If just marginally extra resources are required, the system may be achieved in software and hardware via the use of reconfigurable technology. One option for completing the computation is to create a specialised hardware circuit on FPGA, analogous to an ASIC. However, FPGA circuits may be tailored to specific jobs for optimal performance. Using the properties of a large-scale programmable device (FPGA) that can be repeatedly programmed and configured, reconfigurable systems execute circuitry reconfiguration in real time. While an electronic system is operating in real time, its circuitry may undergo dynamic changes. The core concept is to make full or partial utilisation of the inherent FPGA logic resources via time-sharing reuse. It allows for the sequential operation of discrete-time logic circuits on a single FPGA. Transformable, Adaptable, and Dynamic Hardware and Software

In 2005 [16], a specialised research group called RAW coined the following definition of dynamic reconfigurable: Dynamically reconfigurable hardware architectures and devices are those that may rapidly alter (during system operation) their functionalities and connections. According to Figure 1, the hardware platform, the mapping from particular application to the hardware platform, and the controls required during the running of the system are the three most important topics for the study of dynamic reconfigurable systems.

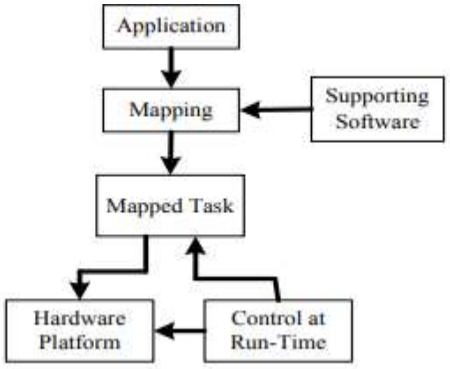

Figure1. Research contained in dynamic reconfiguration system

The hardware that enables dynamic reconfiguration may be broken down into two categories: context configure devices and part reconfigurable configure device [17]. The complicated controls and data structures are often implemented in software, while hardware is used for the more variable and time-consuming aspects of the system [18].

RESULTS OF EXPERIMENTS

The following software simulation is performed to test the hardware/software partitioning technique provided here and the efficiency of the configuration scheduling approach. To begin, we construct a random task flow graph with 30, 40, 50, 60, 70, or 80 nodes using the TGFF tool (Windows Version). A variety of data, including the time and space required for reconfiguration and the expenses associated with implementing new hardware and software, are stored in each node. It is estimated that for each work on the processor, the average execution time of reconfigurable hardware is 10 times quicker than the average execution time of a microprocessor[23]. This means that a reconfigurable hardware implementation may complete the same computing activities 10 times faster. Hardware and software testing simulations use an Inter 1GHz CPU, 512MB RAM, Linux, and the GNU compiler. A single CPU and a Vertex II series xc2v1000 FPGA with 1280 CLBs constitute the assumed target system architecture. The fitness value may be calculated in one of three ways, and the results of each method are compared in Table1.

Table1. The best fitness value comparison of GA, TS and GATS

	30	40	50	60	70	80
TS	0.853600	0.852139	0.853721	0.860228	0.863175	0.864006
GA	0.848012	0.848369	0.842773	0.846343	0.847218	0.845983
GATS	0.922487	0.924224	0.920106	0.914286	0.928246	0.949677

Figure5 shows the correlation between fitness levels. GATS clearly produces better results than

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 12, Issue 2 May 2022

both the genetic algorithm and the tab search algorithm. It demonstrates the GATS algorithm's strengths, including its ability to climb steep hills and several starting points. The GATS method takes more time to execute than the GA, TS algorithm, but it produces more accurate results. For this reason, the GATS algorithm may be used in contexts where precision is of paramount importance.

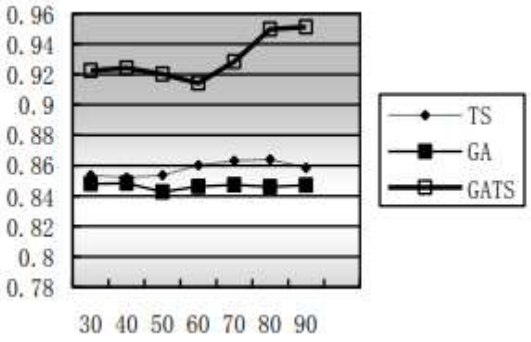

Figure 2. Fitness Value/Nodes Curve

Table2 shows the results using these three algorithms in different scale applications, in which S and NS stand for the scheduling strategies with and without configuration prefetch respectively.

Table2 Data comparison of GA,TS and GATS

Task	GA		TS		GATS	
Node	NS	S	NS	S	NS	S
30	2071	1644	2058	1950	2056	1538
40	2845	2241	2835	2743	2809	2076
50	3680	3087	3573	3402	3362	2458
60	4316	3085	4311	4001	4310	2770
70	4944	3324	4806	4522	4776	3014

TASK SCHEDULING ALGORITHM

Traditional operating systems rely heavily on task scheduling technology. Particularly in large-scale applications, the hardware job cannot be set to the reconfigurable device at once. This is especially true in dynamically reconfigurable systems. As a result, scheduling takes on more significance, with the method used for scheduling having a direct bearing on system performance. There are two basic goals to arranging your tasks. First, optimising the device setup procedure such that all available resources are used effectively. Tasks that may be completed sequentially or concurrently at the same time should be scheduled to the device at the same time. If the setup procedure takes too long, slowing down the system may be a bottleneck, but that impact can be mitigated by optimising the configuration sequence. The scheduling of reconfigurable hardware is likewise a restricted layout challenge. Finding the schedule's start time is just half the battle; you must also determine the layout location of jobs in reconfigurable hardware given a set of limitations and a certain number of available resources. In this last section, we'll use the genetic search algorithm and the tabu search algorithm to determine the optimal task execution order and the lowest possible assignment time for the complete task flow diagram, given the partitioning result. Dynamic Acyclic Graph SchedulingIn reality, the heuristics technique is the best option for handling DAG scheduling issues when the weight values of the nodes and edges are completely arbitrary. In conclusion, the present DAG scheduling technique may be broken down into four distinct types: List Scheduling, Clustering, Task-Duplication-Based, and Random Search. The list scheduling algorithm's central tenet is to choose the highest priority job from a list and perform it using otherwise idle computer resources. This is accomplished by first sorting the priority of nodes in order to produce a scheduling list in which already tasks appear. Clustering scheduling algorithm works on the premise that, given an infinite number of processors, the nodes of the DAG task graph should be considered a cluster when initiating scheduling. Subsequent scheduling iterations should then merge all of these clusters without increasing the overall task completion time. In order to reduce the wait time between tasks, the scheduling technique based on task duplication involves doing a replica of the precursor mission while the processor is idle. In order to find a solution to a problem, the random search approach mostly on chance. Although it yields superior search results than competing algorithms, its scheduling complexity means it is seldom used. Algorithms for Scheduling Configuration Prefetches. One of the defining characteristics of a dynamically reconfigurable hardware architecture or device is the ease with which it may switch between different configurations and use cases. The processing time of systems that use dynamic reconfiguration may be broken down into two distinct phases: the actual work being done, and the time spent getting ready for that work to begin. When the time it takes for each module and for modules to communicate with one another is factored in, the total effective operating time increases. The delay in switching functions that results from setting up the appropriate configuration is called "preparation time." Now, the time it takes to set up a dynamically changeable system is a major limitation. Hide the crucial software setup time and boost the performance of reconfigurable devices to reduce the time needed for preparation. When a node in a DAG is scheduled for execution, the primary notion is to setup the successor node in advance. In addition, a configuration wait queue is used to store nodes that need to be configured but

ISSN 2229-6107 www.ijpast.in
Vol 12, Issue 2 May 2022

cannot immediately begin work because the FPGA configuration port is already in use.

CONCLUSIONS

Technologies for separating hardware and software in programmable embedded systems are the topic of this study. In-depth examinations of the features of reconfigurable systems and the major problems of dynamic reconfigurable technology are presented. There is presented a model for reconfigurable hardware architecture that includes a microprocessor, a configuration controller, reconfigurable hardware (FPGA), memory, and configuration file memory. In addition, a directed acyclic graph (DAG) is provided, which may be used to simulate embedded systems that can undergo configuration changes. After weighing the pros and cons of both GA and TS, the GATS algorithm was developed to mitigate the worst aspects of both. The GATS method was effective because it combined the advantages of genetic algorithms with the tabu search algorithm. For hybrid CPU+FPGA architectures, we offer DAG-based scheduling techniques such configuration prefetch and priority-based scheduling algorithm. The GATS algorithm's system partition is analysed with the help of scheduling algorithms. According to the findings, it cuts down significantly on both reconfiguration and total application execution times.

REFERENCES

[1] Parkash S, Parker A C. SOS: Synthesis of Application-Specific Heterogeneous Multiprocessor Systems. Journal of Parallel and Distributed Computing, Vol.16, 1992, pp: 338-351.

[2] Henkel and Ernst. High-Level Estimation Techniques for Usage in Hardware/Software Co-Design. In IEEE/ACM Proc of Asia and South Pacific Design Automation Conference, Yokohama, Japan, February 1998: 353-360.

[3]ERNST, R, J HENKEL, and T. BENNER. Hardware-Software Co-Synthesis for Micro-Controllers[J], IEEE Design & Test of Computer, 1993, 10(4): 64-75.

[4] Frank Slokar, Matthias Dorel Ralf Munzenberger, Richard Hofmann. Hardware/Software Codesign and Rapid Prototyping of Embedded Systems. IEEE Design & Test of Computers, 2000, 17(2): 28-38.

[5] P ELES, Z PENG, K KUCHCINSKI, et al. System Level Hardware/Software Partitioning Based on Simulated Annealing and Tabu Search [J]. Design Automation for Embedded Systems, 1997, 2(5): 5-32.

[6] HENKEL J. A Low Power Hardware/Software Partitioning Approach for Core-Based Embedded System[C]. In: Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference, 1999, 122-127.

[7] THEERAYOD WIANGTONG, PETER Y.K CHEUNG, WAYNE LUK. Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware-Software Codesign[J]. Design Automation for Embedded Systems, 2002(6): 425-449.

[8] MICHALIS D GALANIS, ATHANASIOS MILIDONIS, GEORGE THEODORIDIS. A Method for Partitioning Applications in Hybrid Reconfigurable Architectures[J]. Des Automation Embedded System, 2006(10): 27-47.

[9] Liu C , Layland J. Scheduling algorithms for multiprogramming in a hard real-time environment [J]. Journal of the ACM,1973,20(1):4-61.

[10] R. Gupta and G. De Micheli. System-level synthesis using re-programmable components. Proceedings of EDAC, IEEE Press, 1992, 2–7.

[11] Institute of Electrical and Electronics Engineers, New York, USA, IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1987), 1987.

[12] J. Iyoda. ParTS: A Support Tool to Hardware/Software Partitioning. Master thesis, Federal University of Pernambuco, Brazil, May 2000 (in Portuguese).

[13] L. Silva. An Algebraic Approach Hardware/Software Partitioning. PhD. thesis, Federal University of Pernambuco, Recife, Brazil, July 2000.

[14] C.Steiger, H.Walder and M.Platzner. Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-Time Tasks[J].IEEE Transactions on Computers, 2004, 53(11): 1393-1407.

[15] R.B. Hughes and G. Musgrave. The lambda approach to system verification. Hardware/Software Codesign, Kluwer Academic Publisher, 1996, 427–451.

[16] A.Dehon. Comparing computing machines.Configurable Computing: Technology and Applications of Proc.Of SPIE, 1998, 3526:124-133.

[17] J. Iyoda, A. Sampaio, and L. Silva. ParTS: A partitioning transformation system. Proceedings of FM99(World Congress on Formal Methods), Vol. 1709, Lecture Notes in Computer Science, 1999, 1400–1419.

[18] A. Kalavade and E. Lee. The extended partitioning problem: Hardware/software mapping, scheduling and implementation-bin selection. Design Automation for Embedded Systems, Vol. 2, No. 2, 1997, 125–163.

[19] J. Madsen, J. Groge, P.V. Knudsen, M.E. Petersen, and A. Haxthausen. Lycos: The lyngby co-synthesis system. Design Automation of Embedded Systems, Vol. 2, No. 2, 1997, 195–235.

[20] R. Niemann and P. Marwedel. An algorithm for hardware/software partitioning using mixed integer linearProgramming. Design Automation of Embedded Systems, Vol. 2, 1997, No. 2, 165–193.

[21] D. Saha, R..S. Mitra, and A. Basu. Hardware/software partitioning using genetic algorithm. Proceedings. of 10th International Conference on VLSI Design, India, 1997, 155–160.