International Journal of Scientific Research & Engineering Trends
Volume 5, Issue 2, Mar-Apr-2019, ISSN (Online): 2395-566X

A Review Article Development of Embedded System For Functional Electrical Simulator

Anila Sheikh **Mr. Kshitij Singhal**
Dept. of Electronic & Communication
Moradabad Institute of Technology
Moradabad, India

Abstract- **A virtual prototyping system is constructed by replacing real processing components with component simulators running concurrently. The performance of such a distributed simulation decreases drastically as the number of component simulators increases. Thus, we propose a novel parallel simulation technique to boost up the simulation speed. In the proposed technique, a simulator wrapper performs time synchronization with the simulation backplane on behalf of the associated component simulator itself. Component simulators send null messages periodically to the backplane to enable parallel simulation without any causality problems. Since excessive communication may degrade the simulation performance, we also propose a novel performance analysis technique to determine an optimal period of null message transfer, considering both the characteristics of a target application and the configurations of the simulation host. Through intensive experiments, we show that the proposed parallel simulation achieves almost linear speedup to the number of processor cores if the frequency of null message transfer is optimally decided. The proposed analysis technique could predict the simulation performance with more than 90% accuracy in the worst case for various target applications and simulation environments we have used for experiments.**

Keywords- **Distributed simulation, embedded system, multiprocessor, parallel simulation.**

I. INTRODUCTION

Over the years, compilers have grown to large and complex software systems. Many individual code optimizations have been implemented and are often individually callable by the user. This places a large burden on the developer, who has to verify that all optimizations work correctly and that they actually bring benefit. Indeed, having good compiler performance is crucial to fully use the processing capabilities of the target processor, and therefore to save energy. Debugging compilers is known to be notoriously hard. Furthermore, selecting the optimal sequence of optimizations and their associated parameters is far from trivial. To address this problem, iterative compilation techniques have been proposed that iteratively compile a program with different optimization sequences and measure the performance of each sequence. To come to a solution to these problems, we propose to embed functional simulators in compilers for two reasons:

- To facilitate debugging of compiler optimization passes
- To facilitate fast turn-around times on iterative compilation.

Optimization passes can be debugged by running the original code and the transformed code through a functional simulator. The optimization pass is correct if both code versions produce equivalent results, otherwise failure is indicated to the developer. This approach allows to run test cases very quickly during the compilation process. While it does not replace running extensive test suites, it gives the advantage that a compiler bug can be pinpointed very quickly. It allows the immediate identification of the failing compiler pass as well as the code region that triggers the bug. Embedding simulators in compilers can speed-up iterative compilation as it allows to fine-tune the optimization of individual code regions.

Focusing the optimization process to individual code regions has several benefits such as selecting a different optimization sequence for each code region and iterating longer over hot code regions. Furthermore, by executing the code regions in a functional simulator, it is possible to measure code properties that allow to steer the compilation sequence in the right direction. A potential limitation is that performance may depend also on the interaction between code regions, which is not measured by this model. Hereto, full program iterative compilation would be necessary as well.

The main motivation for this work is the complexity of compiling for communication-exposed architectures such as ADRES, Trips and Wave Scalar. They aim to increase instruction-level parallelism (ILP) by executing instruction blocks on grids of ALUs where

© 2019 IJSRET

communication between the ALUs is exposed to the compiler. Those architectures address embedded systems applications, in which a good use of available computing capabilities is crucial for power management. Compiling for such architectures consists of mapping data flow graphs on the grid of ALUs in such a way that concurrency (ILP) is maximized while communication overheads are minimized. We have found the transformation of sequential code to ISAs for communication-exposed architectures very error-prone. Furthermore, compilation for these architectures uses iterative approaches such as place-and-route algorithms and simulated annealing.

Simulation and virtual systems because reality sucks. Ironically, while simulation is almost universally implemented as software on computers, the use of simulation to develop computer software itself is still quite rare. Here, we will look specifically how custom embedded computer systems can be simulated for the benefit of embedded software development. We will not address the use of simulation to do hardware-software code sign, system-on-chip design, and other hardware-oriented tasks.

Note that as embedded systems start using multiple processors and multi core processors, using simulation and "virtual prototypes" for software development is one of the best ways to get a grip on the complex debugging and diagnostics issues arising from the concurrent execution of multiple threads of control on multiple processors. Finally, we should acknowledge that the idea of using simulated computers for software development is not new. Looking at the first issue of Embedded Systems Programming from 1988 they have both advertisements and a product review of a simulator for embedded systems. The arguments put forward are the same as those of today. Thanks to Jack Ganssle, www.ganssle.com, I can show you a picture of an ad from 1988.

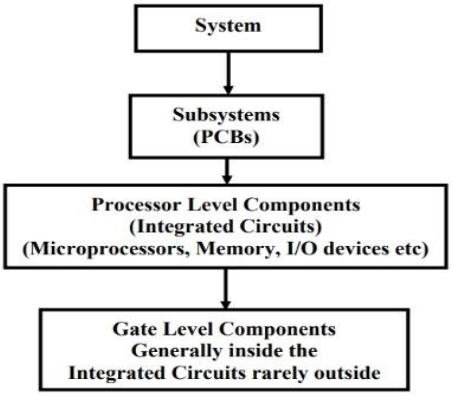

Fig.1 The Hierarchical Components.

II. RELATED WORK

Most research on distributed event-driven simulation for multi core embedded systems has been focused on performance improvement by raising the abstraction level of the simulation model and/or reducing the synchronization overhead between the component simulators. We first review the previous studies from the time synchronization perspective. A. Time Synchronization A lock-step method synchronizes processor simulators every cycle, which is commonly used due to its simplicity and high accuracy.

Even though this method assures causality that no past event arrives at the simulator, it degrades the simulation performance significantly because of excessive IPC overhead. There are two kinds of approaches to reduce the synchronization frequency: conservative and optimistic. A conservative approach guarantees that no past event will occur by advancing the local clock of a simulator only up to the minimum timestamp of the possible input events. An optimized conservative approach [8] was proposed assuming that a simulator notifies globally of the next earliest event, which is usually not feasible. In, time synchronization with equidistant interval has been suggested for HW/SW simulation.

The communication overhead between HW and SW components can be reduced by increasing the interval. However, the functionality of target application may be corrupted if the synchronization interval is not carefully chosen considering the characteristics of target application. For instance, shared memory consistency will not be preserved. In, the time synchronization point is predicted based on a static analysis of application software running on each processor. The predicted minimum time, called look ahead, should be large in order to get more gain. A similar approach to is employed in where the look ahead is obtained by dynamic execution path prediction and hardware prediction.

The performance of these approaches depends on the accuracy of prediction. An optimistic approach, on the other hand, allows each simulator to advance its local clock optimistically assuming that no past event will arrive. Once the assumption is broken, current simulation should roll back to the latest checkpoint to handle the events safely without any causality problem. If a processor simulator does not support a roll back mechanism, as is the usual case, this approach is not applicable. Recently, a technique, called virtual synchronization, has been proposed where each processor simulator does not synchronize its own local clock to the global clock. The central idea is to separate functional simulation and timing simulation.

© 2019 IJSRET

III. STRUCTURE OF AN EMBEDDED SYSTEM

The typical structure of an embedded system. This can be compared with that of a Desktop Computer. Normally in an embedded system the primary memory, central processing unit and many peripheral components including analog-to digital converters are housed on a single chip. These single chips are called as Microcontrollers. This is shown by dotted lines .On the other hand a desktop computer may contain all these units on a single Power Circuit Board (PCB) called as the Mother Board. Since these computers handle much larger dimension of data as compared to the embedded systems there has to be elaborate arrangements for storage and faster data transfer between the CPU and memory, CPU and input/output devices and memory and input/output devices.

The storage is accomplished by cheaper secondary memories like Hard Disks and CDROM drives. The data transfer process is improved by incorporating multi-level cache and direct memory access methods. Generally no such arrangements are necessary for embedded systems. Because of the number of heterogeneous components in a desktop computer the power supply is required at multiple voltage-levels (typically ±12, ± 5, ± 3, 25 volts). On the other hand an Embedded Systems chip may just need one level DC power supply (typically +5V).

In a desktop computer various units operate at different speeds. Even the units inside a typical CPU such as Pentium-IV may operate at different speeds. The timing and control units are complex and provide multi-phase clock signal to the CPU and other peripherals at different voltage levels. The timing and control unit for an embedded system may be much simpler.

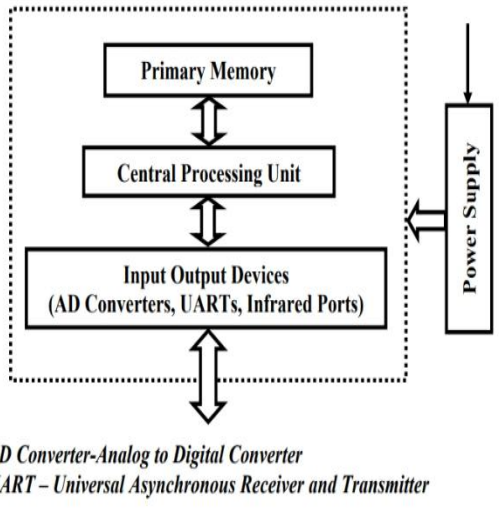

Fig .2 The typical structure of an Embedded System.

IV. LITERATURE REVIEW

Vittorio Arcidiacono, Roberto Menis, 2007 IEEE: Power quality is a key issue in island, shipboard Integrated Electric Power Systems (IEPS). However, it is not of easy accomplishment, due to the complexity of shipboard power station (generators are many and differ by sizes, prime movers, control systems, etc.) and to the intrinsic weakness of the shipboard grid. This paper presents a new approach for controlling voltage and reactive power generated in the power station of an All Electric Ship (AES). A shipboard Voltage and VAR Integrated Regulator (WIRE) is proposed.

The WIRE is aimed at controlling alternators voltage, main busbar voltage and jointly optimizing the reactive power generated by each alternator. WIRE project reunions into an integrated device the most advanced control functions presently implemented in smartest land power stations. WIRE is also endowed with an embedded power-station/IEPS real-time simulator, suitable for performing factory functional, diagnostic and Hardware-In-theLoop (HIL) tests. The paper will show how WIRE can improve power quality into an AES, while respecting marine constraints like: reduced room availability, redundancy, reliability, optimal and safe utilization of generators, fast commissioning, and diagnostics.

Benoît Chretien, LydieNouvelière, 2013 IEEE: The boom in the number of active safety functions and driver assistance systems during the last several years has made vehicle design more and more difficult. To support the early development process, particularly the design of electronic and electric architectures, the capability to simulate and compute complete vehicle behavior is critically important and useful. This paper presents such a simulator, which consists of key components such as the drivetrain and controller subsystems. Then, after having presented the generic structure of a driver assistance system, concrete use cases of this simulator are given, which assess vehicle performance according to different functional and hardware architectures.

Hsin-hung Lin, Chih-Wen Hsueh, 2006 IEEE: As the increasing of system performance and computing power, embedded systems are more complicated and interactive. Therefore, operating system (OS) plays a more important role in embedded systems to utilize various hardware and software resources. However, no OS can meet all requirements of various embedded systems. Due to the advance of reconfigurable processors, system requirements can be more dynamic. Embedded OS has becoming more critical in the development of embedded systems and thus there is a strong need of configurable embedded OS to better and faster build up the target system. In this paper, we propose a configurable OS, called COS, based on SOA (Service-Oriented

© 2019 IJSRET

Architecture) for embedded SoC (System on a Chip) systems to build an application specific OS according to the system requirements. COS can be easily configured to better utilize the resources of the target embedded platform and have better support to the embedded application. Moreover, the COS can easily extend new features or functionalities of other OSes and even be adapted to the designs of other OSes. We implement COS on an ARM platform to prove its configurability and also evaluate its overhead. We believe that COS can be configured to meet the various requirements in embedded SoC systems and help to speed up the embedded system development process.

V. CONCLUSION

In this paper, we proposed a novel parallel simulation technique that shows scalable performance as the number of component simulators increases.We proposed a novel estimation technique to predict simulation performance with a set of parameters to configure the parallel simulation. The proposed technique formulates the simulation performance as a function of notification period based on the memory access behavior of a target application. Through intensive experiments, we confirmed that the proposed estimation technique is capable of accurately predicting simulation performance with the wide variation of simulation configurations. The estimation error does not exceed 10% in the worst case in all our experiments. Therefore, the proposed estimation technique can be used to configure the parallel simulation environment in an optimal way for a given target application.

REFERENCES

[1] H. Kim, D. Yun, and S. Ha, "Scalable and retarget able simulation techniques for multiprocessor systems," in Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synthesis, Oct. 2009, pp. 89–98.

[2] R. M. Fujimoto, "Parallel and distributed simulation systems," in Proc. Winter Simul. Conf., Dec. 2001, pp. 147–157.

[3] K. M. Chandy and J. Misra, "Asynchronous distributed simulation via a sequence of parallel computations," Commun. ACM, vol. 24, no. 4, pp. 198–205, Apr. 1981.

[4] D. R. Jefferson, "Virtual time," ACM Trans. Programming Languages Syst., vol. 7, no. 3, pp. 404–425, Jul. 1985.

[5] J. Jung, S. Yoo, and K. Choi, "Performance improvement of multiprocessor systems co simulation based on SW analysis," in Proc. Des. Autom. Test Eur., Mar. 2001, pp. 749–753.

[6] M. Chung and C. Kyung, "Enhancing performance of HW/SW co simulation and co emulation by reducing communication overhead," IEEE Trans. Comput., vol. 55, no. 2, pp. 125–136, Feb. 2006.

[7] S. Mukherjee, S. Reinhardt, B. Falsafi, M. Litzkow, S. Huss-Lederman, M. Hill, J. Larus, and D. Wood, "Fast and portable parallel architecture simulators: Wisconsin wind tunnel II," IEEE Concurrency, vol. 8, no. 4, pp. 12–20, Oct.–Dec. 2000.

[8] W. Sung and S. Ha, "Efficient and flexible cosimulation environment for DSP applications," IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol. E81-A, no. 12, pp. 2605–2611, Dec. 1998.

[9] F. Fummi, M. Loghi, S. Martini, M. Monguzzi, G. Perbellini, and M. Poncino, "Virtual hardware prototyping through timed hardware software cosimulation," in Proc. Des. Autom. Test Eur., Mar. 2005, pp. 798–803.

[10] M.C.Ndinechi, O.A.Ogungbenro, K.C. Okafor, Digital Metering System: A Better Alternative For Electromechanical Energy Meter In Nigeria, International Journal Of Academic Research Vol. 3. No. 5. September, 2011, I Part

[11] Oluwole Oyetoke and Adedayo Adedapo, A Microcontroller Based Embedded System Design for Device Automation and Control in Intelligent Buildings, International Journal of Research (IJR) e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 12, December 2015

© 2019 IJSRET