A Case Study in Hardware-Software Codesign of Distributed Systems — Vehicle Parking Management System†

Trong-Yen Lee, *Pao-Ann Hsiung*, and *Sao-Jie Chen*

Department of Electrical Engineering, National Taiwan University, Taipei, TAIWAN, R.O.C.

*Institute of Information Science, Academia Sinica, Taipei, TAIWAN, R.O.C.

Abstract

Hardware-software codesign of distributed systems is a more challenging task than that of centralized embedded systems. Each phase of codesign, such as copartitioning, cosynthesis, cosimulation, and coverification, must take into account physical restrictions imposed by the distributed nature of such systems. Although codesign of distributed systems is more complex, yet many common parts of the systems can be reused for codesign. For the above two reasons, we have adopted an object-oriented (*OO*) *codesign approach, which allows a natural structural restriction and object design reuse. A parallel approach to hardware-software partitioning is also presented. We illustrate our techniques through a case study on a Vehicle Parking Management System* (*VPMS*). *The study shows the benefits of OO codesign and parallel partitioning. Validation is accomplished through prototyping and emulation.*

Keywords: *object-oriented codesign, parallel copartitioning, emulation, distributed embedded systems, case-study*.

1. Introduction

Distributed systems, such as remote teaching facilities, vehicle parking systems, auditorium air-conditioning, coal-mine signal systems, and others abound in our everyday life and are almost all computerized, that is, they all contain some sort of integrated chips or processors running some software. These systems are difficult to design due to their physical restrictions and varied behavior, which is a result of inherent concurrencies. But, at the same time, we also observe that there are some sorts of symmetricities, however small they may be, among distributed systems. For example, remote teaching facilities may have several *similar* tracking systems that work concurrently to sense and capture classroom activities for remote displaying. Such existing similarities are one aspect of distributed systems that we can take advantage of for codesign.

We define a distributed system as follows. When a system consists of more than one part that must be located at or embedded in different physical locations, it is called a *distributed system*. The design of distributed systems has always been a challenging task. Codesign of distributed systems must solve not only hardware-software communication issues within a single embedded unit, but also the communication issues between different parts of a distributed system (which may consists of either hardware, or software, or both). Hardware-software copartitioning must take into

† This work was supported by the National Science Council, R.O.C., under grant NSC 88-2215-E-002-037

consideration the physical restrictions of a distributed system. Besides timing and cost constraints, part modularity and physical restrictions are also important factors that must be taken into account in the codesign of distributed systems. Interface must be synthesized not only for the hardware and the software within a system part, but also for the different parts of a distributed system.

Previous work related to hardware-software codesign [1] have assumed that physical restrictions or interconnections are arbitrary, but such assumptions are not realistic and most systems do have a definite physical distribution. Further, previous work have also not considered how design parts may be reused in a distributed system. There have been some related work on distributed embedded systems in the literature.

Prakash and Parker [2] formulated heterogeneous multiprocessor system synthesis as a mixed integer linear program in SOS, a formal synthesis approach developed at University of Southern California. Further, based on Prakash and Parker's formulation, Wolf [3], [4] developed a heuristic algorithm for architectural co-synthesis of distributed embedded computing systems.

Object-oriented techniques have been applied to system-level synthesis such as in PSM [5] and ICOS [6]. Wolf applied OO techniques to co-synthesis [7], but it was limited to analysis of the methods contained in each object class.

2. Object-Oriented Codesign and Parallel Partitioning

For ease of structural constraint satisfaction and design reuse, we have adopted an object-oriented codesign approach. A distributed system is modeled using *Object Modeling Technique* (OMT) [8] which includes three models, namely, *Object Model*, *Dynamic Model*, and *Functional Model*. A distributed system specified using these three models is then partitioned into hardware and software using a parallel partitioning technique.

The parallel partitioning technique starts from two initial partitionings, namely, the two extreme design alternatives: all hardware and all software. Next, along each iteration the partitioning algorithm moves one step forward starting from the two initial solutions in parallel. Moving a step forward means shifting some part of an all-hardware solution into software and shifting some part of an all-software solution into hardware. The partitioning algorithm stops when a partition is reached that satisfies all time and cost constraints and has a minimum cost. Although both threads of moving forward are greedy, but this parallel algorithm achieves better results than one single greedy method.

3. Case Study Description

The case study presented in this article is a project for designing a vehicle parking system, we call this system *Vehicle Parking Management System* (VPMS). VPMS consists of three parts: an entry, an exit, and a display. Entry and exit are similar in most respects. Both of them allow vehicles to pass through them one by one. Display shows the current number of free parking space available in a parking lot or garage.

An entry (or an exit) gate consists of three parts: a ticket facility, a gate controlled by a gate-motor, and a pair of sensors. The ticket facility at the entry stamps the current date and time and gives a new ticket to an in-coming vehicle. The ticket facility at the exit checks whether the ticket (parking) fees have been paid and the current time is within 15 minutes of the ticket fee payment. After a positive response is received from the ticket facility, a gate controller opens the entry (exit) gate to allow a vehicle to drive in (out). A pair of sensors are located after the gate (in the direction of the vehicle, that is, further in for the entry and further out for the exit). The sensors then send a signal to the gate controller to close the gate after a vehicle has passed by. At the same time, the sensors also send a signal to the display for updating the displayed number of parking vacancies.

Constraints for the VPMS system include: a maximum cost of $1,300 and a minimum entry (exit) gate response time of 1 μs. The minimum gate response time implies that the gate should not be closing or opening too slowly. A slow gate open would make the VPMS user unhappy, while

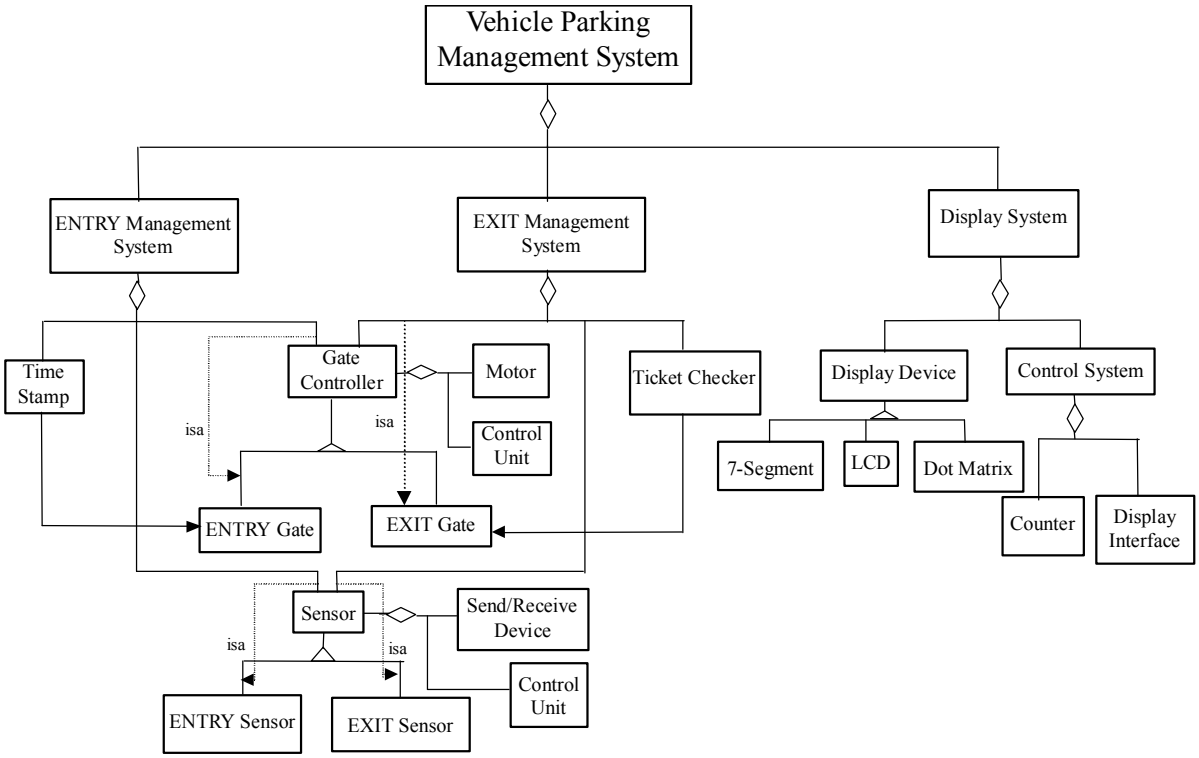

Fig. 1 Object Model of VPMS

a slow gate close would allow more than one vehicle to pass through the gate for each entry/exit transaction and would make the VPMS boss unhappy. Hence, gate response time is specified for correct execution of the system.

The above description clearly shows that VPMS is a distributed system having some common parts. We can thus model VPMS using OO techniques and then codesign it.

4. Codesign of the Vehicle Parking Management System

In this section, through the above VPMS, we will show how to apply *Object Modeling Technique* (OMT) [8] to hardware-software codesign and illustrate parallel partitioning. There are two reasons for choosing OO design techniques in the codesign of distributed systems: (1) object-oriented model allows explicit *modularity* in correspondence to structural restrictions in physically distributed systems, and (2) distributed systems often have parts that are common in many respects, thus object design *reuse* becomes a useful technique.

The object model of VPMS is shown in Fig. 1. VPMS includes three parts: ENTRY management system, EXIT management system, and DISPLAY system. The ENTRY management system includes time-stamp, gate controller, and a pair of sensors (send and receive devices). The EXIT management system is similar to the ENTRY management system, which includes ticket checker, gate controller, and a pair of sensors. The gate controller and sensor object models in ENTRY and EXIT management systems have many parts in common, therefore, their OMT models can be reused. The display system consists of control system (counter and display interface) and display device such as 7-segment display, LCD, or dot matrix LED display. The counter value (count) indicates the number of available parking vacancies.

The dynamic model of VPMS describes those aspects of a system concerned with time and the sequencing of operations. The dynamic

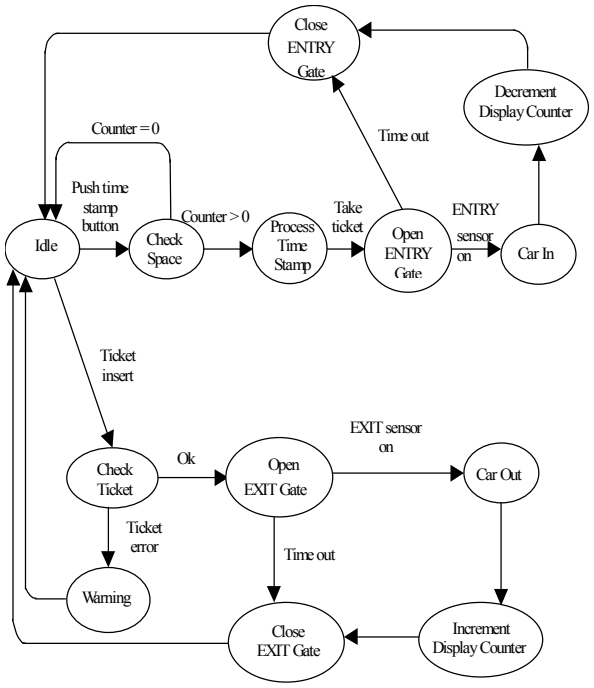

Fig. 2 Dynamic Model of VPMS

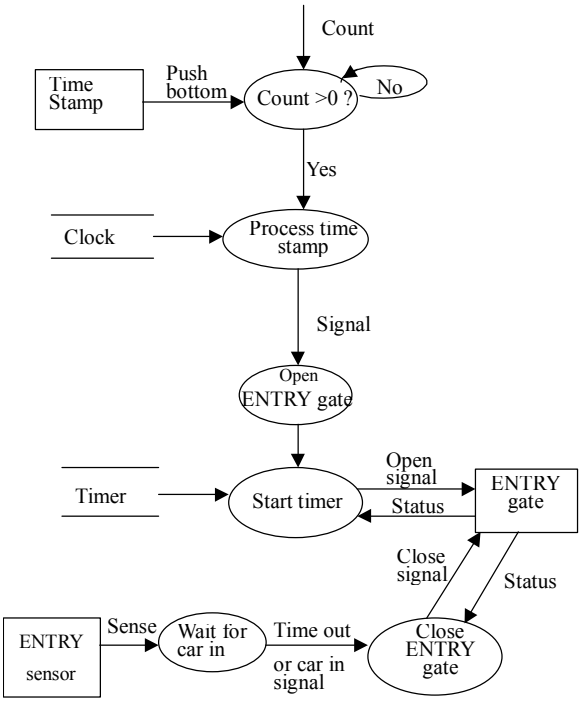

Fig. 3 Functional Model of ENTRY Management System

model of VPMS is represented graphically with a state diagram as shown in Fig. 2. The system is idle when there is no car entry or exit. There are two sequences to process car entry and exit.

The functional model of VPMS describes those aspects of VPMS such as: transformations of value-function, mappings, constraints, and functional dependencies. The functional model captures what VPMS system does, without regard for how or when it is done. The functional model is represented with data flow diagrams. Functions are invoked as actions in the dynamic model and are shown as operations on objects in the object model. The functional models of VPMS are shown in Figs. 3 and 4. In the ENTRY management system, if the number of available space (count) is positive, then an open signal is sent to the ENTRY gate for allowing car entry. The gate is closed when the ENTRY sensor senses car entry or the timer indicates time out. The functional model of EXIT management system as shown in Fig. 4 is similar to the ENTRY management system.

Given the three OMT models for VPMS (Figures 1, 2, 3, and 4) along with cost and performance constraints, we start designing VPMS. We observe that there are three parts that could be designed as software or hardware, namely, the counter of display, the driver in the control unit of motor, and the driver in the control unit of sensor. Thus, there are eight different kinds of possible partitions into hardware and software.

In the following, we describe how partitioning and emulation results in the final VPMS design. This final result has the lowest cost while satisfying the given performance constraint on the gate response time.

Partitioning. We use parallel partitioning technique (as discussed in Section 2) for VPMS

Table 1. Codesign Alternatives in VPMS

A	B	C	D	E	F	G	H
H_C, H_S, H_M	S_C, H_S, H_M	H_C, S_S, H_M	H_C, H_S, S_M	S_C, S_S, H_M	S_C, H_S, S_M	H_C, S_S, S_M	S_C, S_S, S_M

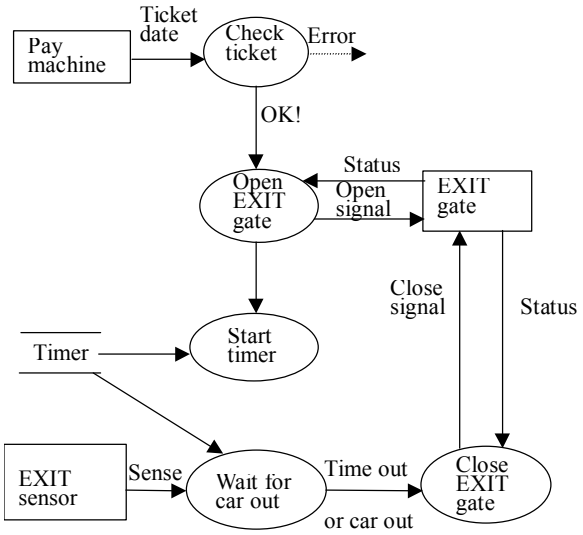

Fig. 4 Functional Model of EXIT Management System

codesign. We start from the two extreme design alternatives: all-hardware and all-software. For the all-hardware alternative, based on performance constraint satisfaction we try to reduce the cost. For the all-software alternative, based on cost constraint satisfaction, we try to increase performance. There are eight design alternatives as shown in Table 1, where H_C is a counter designed by hardware, S_C is a count function designed by software, H_S is a driving circuit of sensor designed by hardware, S_S is a driving function of sensor designed by software, H_M is a driving circuit of step motor designed by hardware, and S_M is a software driving function of step motor

Starting from all-hardware alternative **A** and all-software alternative **H**, using the above method, we move from **A** to **D** and from **H** to **F**, in parallel. These two parallel steps have the following meanings:

- **A→D**: **D** has a smaller cost than **A** while satisfying the 1 μs gate response time constraint,
- **H→F**: **F** satisfies the 1 μs gate response time constraint, while **H** did not.

Partitioning stops after just one parallel step because **F** is a locally optimal solution that satisfies the gate response time as well as its cost cannot be reduced further. Parallelism has thus saved some efforts in partitioning because for the all-hardware initial solution approach, we would require two steps to reach **F** from **A** (**A** → **D** → **F**).

Codesign: According to the three OMT models (as shown in Figs 1, 2, 3 and 4), we designed the hardware and software components of VPMS. The object model helped us to map physical hardware components and software modules to the objects in VPMS. The dynamic model helped us to organize the behavior of each mapped component. The functional model helped us to actually interconnect the hardware and software modules through interface construction.

Emulation: We chose a single chip integrated circuit 8051 [9] for software design which has 4K bytes of on-chip programmable memory for software programming, four input/output ports for data or control interface, two 16-bit timer/counters for time calculation or counting, and one series port for data transmission. Based on the object, dynamic, and functional models,

Table 2 Comparison of eight design alternatives of VPMS

	A	**B**	**C**	**D**	**E**	**F***	**G**	**H**
Cost ($)	1448.25	1415.25	1420.25	1278.25	1297.25	1155.25	1160.25	1028.50
(%)	100.00	97.72	98.07	88.26	89.57	79.77	80.11	71.02
Area (mm^2)	58,425	52,838	59,129	61,330	48,787	50,988	57,279	42,182
(%)	100.00	90.44	101.21	104.97	83.50	87.27	98.04	72.22
Power Consumption (W)	3.15	2.90	3.22	3.26	2.67	2.70	3.03	2.47
Response time(μs) (sensor to display)	180	13,000	180	180	13,000	13,000	180	13,000
Response time(μs) (sensor to gate)	0.22	0.22	1.06	0.22	1.06	0.22	1.06	1.06

we designed the emulator for each part of hardware and coded the software in VPMS. The constraints for VPMS include: a maximum cost of $1,300 and a minimum gate response time of 1 μs. On applying parallel partitioning technique, we found that design alternatives **D** and **F** (see Tables 1 and 2), satisfied the given constraints. Therefore, we emulated only design alternatives **D** and **F**. The cost and performance analysis results are shown in Table 2. From the emulation results, we conclude that the design alternative **F** is the best in terms of having the lowest cost ($1155.25) while satisfying the performance constraints.

5. Conclusion

The codesign of distributed systems was demonstrated through a case study on *Vehicle Parking Management System* (VPMS). We have shown how OMT can be used to model and help in identifying which parts can be implemented as hardware and which as software. We have also proposed a parallel partitioning technique that can reach locally optimal solution much faster. Future work will consist of applying this technique to other industrial examples and extending it to CMAPS [10].

References

[1] T.-Y. Yen and W. Wolf, *Hardware-software co-synthesis of distributed embedded systems*, 1996, Kluwer Academic Publishers, The Netherlands.

[2] S. Prakash and A. C. Parker, "SOS: Synthesis of application-specific heterogenous multiprocessor systems," *Journal of Parallel and Distrbuted Computing*, Vol. 16, No. 4, pp. 338-351, December 1992.

[3] W. Wolf, "Guest editor's introduction: hardware-software codesign," *IEEE Design & Test of Computers*, Vol. 10, No. 3, pp. 5, September 1993.

[4] W. Wolf, "Hardware-software codesign of embedded systems," *Proceedings of the IEEE*, Vol. 82, No. 7, pp. 967-989, July 1994.

[5] P.-A. Hsiung, S.-J. Chen, T.-C. Hu, and S.-C. Wang, "PSM: An object-oriented synthesis approach to multiprocessor system design," *IEEE Transactions on VLSI Systems*, Vol. 4, No. 1, pp. 83-97, March 1996.

[6] P.-A. Hsiung, C.-H. Chen, T.-Y. Lee, and S.-J. Chen, "ICOS: An intelligent concurrent object-oriented synthesis methodology for multiprocessor systems," *ACM Transactions On Design Automation of Electronic Systems*, Vol. 3, No. 2, pp. 109-135, April 1998.

[7] W. Wolf, "Object-oriented cosynthesis of distributed embedded systems," *ACM Transactions On Design Automation of Electronic Systems*, Vol. 1, No. 3, pp. 301-314, July 1996.

[8] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-oriented modeling and design*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

[9] Intel, *Embedded microcontrollers and processors*, Vol. 1, Intel Corporation, 1993.

[10] P.-A. Hsiung, "CMAPS: A cosynthesis methodology for application-oriented parallel systems," *ACM Transactions On Design Automation of Electronic Systems*, Vol. 5, No. 1, Jan. 2000 (to appear).