THE FACTORS INFLUENCING THE PERFORMANCE OF MULTIPLE PROCESSOR SYSTEMS FOR TELEMETRY DATA ACQUISITION AND PROCESSING

OUYANG Can
Principal Engineer
Beijing Research Institute of Telemetry
Beijing, P.R.China

SHI Chang-jie
Professor
Science and Technology Committee
Ministry of Aerospace Industry, P.R.China

ABSTRACT

Multiple processor architecture has been used in telemetry processing systems for many years. But how to evaluate the performance of such a system on telemetry data processing has not yet been adequately addressed. Utilizing a deterministic model, this paper tries to study the performance of multiple-processor telemetry data processing system. It demonstrates that the bus arbitration scheme, the ratio of processing time to communication time and the processing decomposition are the three important factors influencing the performance of such kind of systems. All the three factors are discussed. The results obtained have been applied to the design of the MSDDTS (Multi-Stream Data-Driven Telemetry System), a new generation multi-stream telemetry system in china, in order to gain the best cost-effectiveness.

INTRODUCTION

As aircraft and spacecraft technology advanced, the demand for real-time telemetry data acquisition and processing increases. However, higher bit rate and multi-stream increasingly multiply the difficulties. The traditional general-purpose host computer is not suitable for this requirement. Multiple processor system gives the way to solve the problem. Many multiple processor architectures and systems have been proposed and some of them have been actually used. Data-flow architecture is one of them. It has been widely applied to the real-time telemetry data processing. But how to evaluate the performance of this kind of systems has not yet been adequately addressed.

Analyzing the performance of multiple processor system is a very complex task, since many factors jointly determine system performance and the modification of some factors affect many others. So it is crucial to tune system parameters such that a whole system can achieve its peak performance at the minimum cost. In recent years, many models for predicting multiple processor system performance have been developed. This paper applies a model presented by Vrsalovic, et al.[1] to study the performance of telemetry processing system which has the data-flow architecture. The results obtained has been used in the design of MSDDTS [2], a new generation multi-stream telemetry system in china, to make the system approach its best cost-effectiveness.

The next section reviews the data-flow architecture used in telemetry processing systems. In the section following the next, a model for predicting the system performance is described, and the factors influencing system performance are discussed. Finally, some considerations in the design of the MSDDTS are presented.

SYSTEM ARCHITECTURE

The multiple-processor architecture used in MSDDTS is the data-flow architecture shown in figure 1. The m input devices (I1, 12,...,Im) receive m data streams, assign a unique tag value to each parameter for every stream and send the data-tag pair to the High Speed Data Bus independently. The n processors (P1, P2,...,Pn) get data-tag from High Speed Data Bus, execute the algorithms pointed by the tags and return the results to High Speed Data Bus. The data received by a processor may be a raw data from input devices or a processed result from processors. The k output devices (O1, O2, ..., Ok) can capture any data-tag appeared on the High Speed Data Bus and put them out. The High Speed Data Bus is the main path for data transfer among these function modules. The secondary computer bus in the system is primarily used for administrative tasks, e.g. modules setup and test etc., usually not for real-time parallel processing.

There is no shared memory in such system. The processors are loosen coupled. Each processor has a microprocessor, a program memory, a data memory, a tag match unit and the bus interface. The tagged data reached to a processor is firstly matched with tag map in the match unit. Only the needed data by the processor is to be buffered and to activate a processing sub-program associated with the tag. After the sub-program executed, the result is labeled with a new tag and put on to High Speed Data Bus again. This is the data-driven execution mode, and it implies that the computation is asynchronous on the sub-program level and purely functional.

On this architecture, processors can handle different telemetry parameters with different algorithms simultaneously. If a parameter needs more than one processing algorithms, a pipeline operation can be easily established by allocating these algorithms to different

processors for increasing real-time responsibility. Actually, any other combination type of algorithm chaining can be implemented.

THE FACTORS INFLUENCING SYSTEM PERFORMANCE

In this section, we will study the factors influencing system performance. The system has a data-flow architecture described above. The model for analyzing is based on a deterministic model presented in [1].

The speedup of a multiple-processor system can be defined as the ratio of the time required to perform all the tasks on a single processor to the time required to perform the same tasks on the n-processor system.

Let us assume that the real-time telemetry processing is a cyclic processing, that is to say, there is a time period T such that the number and the type of telemetry data needed to be processed are the same between time duration [t, t+T) and [t+T, t+2T). In a data-flow system, the time for treating one data sample is composed of three components: the bus access time, the algorithm activate time and the algorithm computing time. The first two components can be considered as the communication time between processors, and the last component is the processing time. For convenient, we denote Np as the number of computation units per period time T, tp as the time to complete one unit of computation. Similarly, denote Nc as the number of communication times per period time T, and tc as the time to transfer one massage without delays induced by multi-processors. So, if there need some kind of synchronization between processing time and communication time, the speedup of a data-flow system can be given by

$$S_{syn} = \frac{Np*tp}{Np*tp/Dp(n) + Nc*tc*A(n)/Dc(n)} \qquad (1)$$

Where, Dp(n) is the decomposition function for processing, which is defined as the processing time of single processor to the processing time of the n-processor for the same application. Dc(n) is the decomposition function for communication. While A(n) is the normalized communication time, which is the real communication time divided by the ideal communication time without any delay. Conversely, if no synchronization is required between communication and processing. the speedup is given by

$$S_{asy} = \min\{Dp(n), Np*tp*Dc(n)/(Nc*tc*A(n))\} \qquad (2)$$

Assume that each computation unit will produce one result, and the result is communicated between processors by bus, we can let Np=Nc. Furthermore, due to the total

communication time on bus will not decreases as n increases, the communication decomposition function Dc(n) will be equal to 1. If we define the ratio of processing-to-communication time x as tp/tc, the expressions (1) and (2) can be simplified to expressions (3) and (4).

$$S_{syn} = \frac{x}{A(n) + x/Dp(n)} \qquad (3)$$

$$S_{asy} = \min\{Dp(n), x/A(n)\} \qquad (4)$$

From (3) and (4), one can see that the speedup is completely determined by A(n), x and Dp(n). In the following paragraphs, we will discuss these three factors.

Arbitration Protocols

The normalized communication time, A(n), is a function of the number of processors in the system (n) and the traffic intensity of each processor (ρ). Different bus arbitration scheme has different A(n). The most commonly used protocols for system bus are equal-priority protocol, unequal-priority protocol and rotating-priority protocol. They are abbreviated here as E-P, U-P and R-P respectively.

Figure 2 describes the variation of A(n) with different n and ρ under these three protocols. For simplicity, we have assume that the processors has the same traffic intensity. It demonstrates that the R-P presents the smallest communication delay for all ρ and n among these three protocols. It also shows that when ρ and n is small, the E-P gives the worst performance, but under any other conditions, the U-P has the worst performance.

As expressions (3) and (4) expressed, the speedup is only dependent on A(n) when x and Dp(n) is fixed. Substituting A(n) with the three normalized communication time function, we can get the speedup S of these three protocols. Figure 3 shows S varying with n under synchronization mode and asynchronization mode. In this figure, Dp(n) is with the best case, i.e. Dp(n) = n, and x with the common case, x = 20. From these curves, the following judgment can be made:

1. there is a peak performance for all three protocols, and after the peak performance is reached the system performance will not increase as the number of processors increase;
2. asynchronization mode gives the better performance than synchronization mode for all protocols;
3. the asynchronization mode presents linear speedup before the peak performance reached;

4. the R-P arbitration protocol gives the greatest peak performance among these three protocols.

The ratio of processing time to communication time

From expression (4), one can see that the peak performance increases as x increase for the R-P bus arbitration scheme and the asynchronized processing-communication mode. It implies that we can decrease tc or increase tp to get better peak performance. To decrease tc means to increase the bus transfer rate and decrease the algorithm activate time. To increase tp is to say to increase the execute time of computation unit. This can be done by enlarging the grain size of parallel processing. For telemetry processing application, that means we should allocate the sub-programs about the same parameter into one processor once the processor has enough time to respond to each arrival data.

Processing Decomposition

The decomposition function for processing, Dp(n), is greatly dependent on the amount of parallelism inherent in the processing problem and the method applied to allocate sub-problems to the processors. For a good allocating algorithm, the loads between processors are completely balance, i.e. the numbers of computation units allocated to each processor are equal. But usually it is not the case, there may be some processors whose number of computation units is greater than the average. Let q denote the ratio of maximum difference exceed the average number to the average. The processing decomposition function is

$$Dp(n) = n/(1 + q) .$$

The q expresses the affect of different allocation scheme for a specific application. The best allocation algorithm will make q equal zero. But a worst allocation algorithm might result in great q, even greater than one. Figure 4 shows the decrease of system speedup with q increasing.

Summary of Discussion

The above discussion can be summarized as below:

1. There is a peak performance about the number of processors for the system with a fixed bandwidth of communication, and the system performance will decrease as the number of processor increase once the peak value is over;
2. The asynchronized mode between processing and communication presents a linear speedup before the peak performance reached, and has greater peak value than the synchronized mode;

3. The rotating-priority bus arbitration protocol gives the best speedup among the commonly used arbitration protocols;
4. Increasing the bus transfer rate and decreasing the algorithm activate time can increase the peak value of system speedup, and so does enlarging the grain size of parallelism;
5. The more uneven is the load of processors, the worse is the system performance; so the allocating scheme should balance the load of processors whenever possible.

DESIGN CONSIDERATION

The results obtained from the discussion above has been applied to the design of the Multi-Stream Data-Driven Telemetry System (MSDDTS). In order to gain the best cost-effectiveness, we made the following selections.

1. The arbitration scheme of the High Speed Data Bus employs the rotating-priority protocol;
2. The bus transfer rate is 10 MHZ;
3. The arbitration period and the data transfer period are overlapped;
4. The FIFO is used in the interface to High Speed Data Bus, in order to implement the asynchronized mode between processing and communication;
5. The address of processing algorithm is get directly by the match unit to obtain the smallest activate time;
6. An optimist allocation algorithm is utilized for automatically distributing the processing requirements, which is defined by user, to make the load of processors be balance as nearly as possible.

The MSDDTS, has a simple architecture and a high performance for small scale real-time parallel processing systems. But its performance will rapidly decrease when the number of processor exceed a certain value, e.g. 6 (for x=20). If a higher performance is demanded, the bandwidth of communication must be further extended. The best case is that the bandwidth linearly increase with the number of processor, i.e. $Dc(n) = n$, see expressions (1) and (2). However the cost will increase too. So a tradeoff would be made in an actual application project.

REFERENCES

1. D. Vrsalovic, D. Siewiorek, and E. Gehringer, “Performance prediction for Multi-processor system,” Proc. 13th International Conference on Parallel Processing, Aug. 21-24 1984, pp. 139-146.

2. C. Ouyang, and C. J. Shi, "Multi-stream Data-driven Telemetry System", Proc. 27th International Telemetry Conference, Nov. 1991, pp. 819-823.

3. F. El Guibaly, "Design and Analysis of Arbitration Protocols", IEEE Trans. on Computers, Vol. 38, No. 2, Feb. 1989, pp. 161-171.

Fig. 1 System Architecture

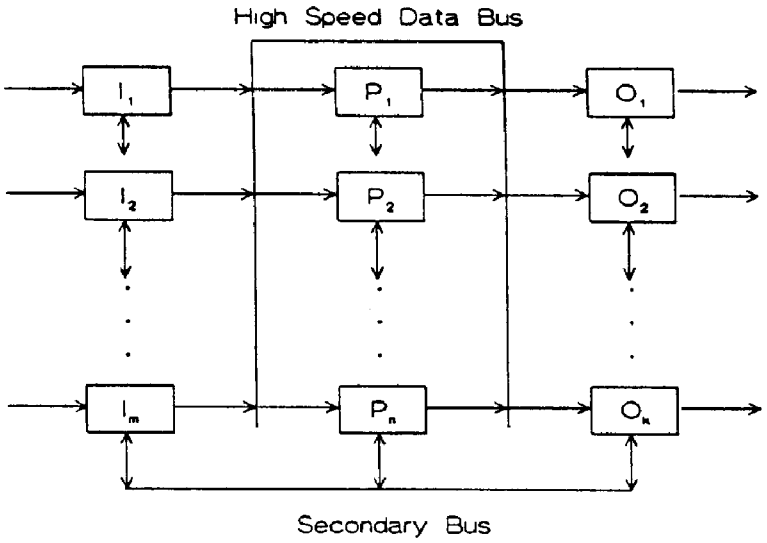

Fig. 2 The normalized communication time of different arbitration protocol

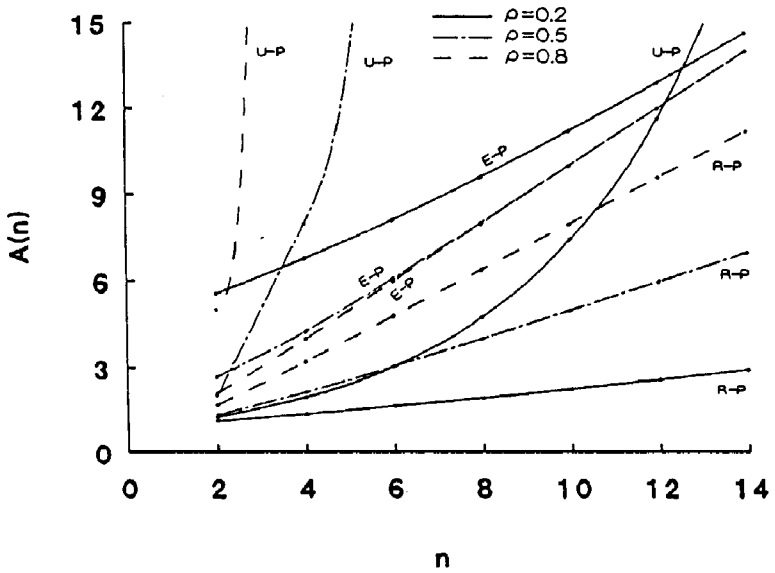

Fig. 3 The Speedup on different bus arbitration scheme

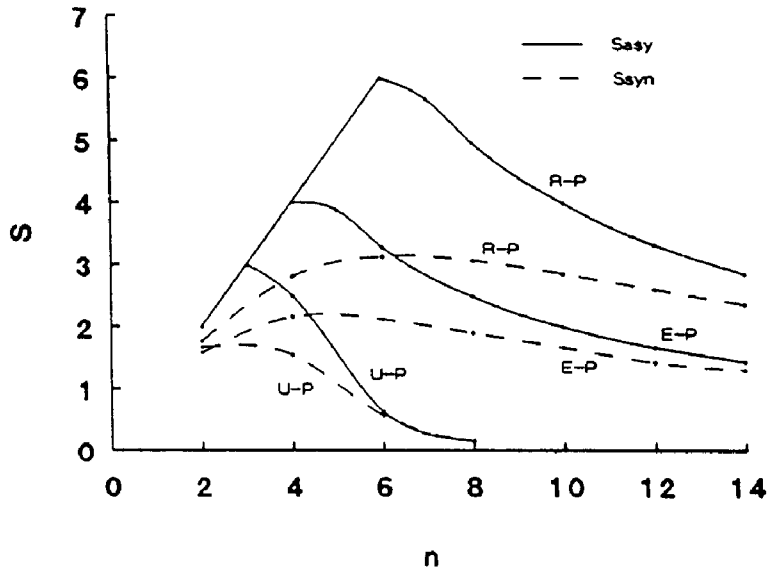

Fig. 4 The effect of unbalanced load to speedup