Information Systems and Signal Processing Journal (2016) 1: 24-30
Clausius Scientific Press, Canada

An Improved Method of Signal Processing Platform Component Model

Liang Shaoxun[1,a], Sun Hongsheng[1,b], Hu Zeming[1,c]

[1]Department of Electronic Engineering, Zhengzhou Institute of Information Science and Zhengzhou, Henan, China
[a]273827405@qq.com,[b]1255579072@qq.com,[c]zz_hz@163.com

Keywords: Signal Processing Platform, Data-flow-driven, Component

Abstract: Data-flow-driven (DHDF) has been widely used in information processing, and the event triggered concurrent dataflow model (ECDF) improve the timeliness and efficiency of information processing system by introducing a priority threads and event trigger mechanism. However, when applied to an information processing platform, the model cannot provide parameter support for component execution order. Base on the Data-flow-driven and the event triggered concurrent dataflow model, an event-triggered dataflow driven mechanism is designed for signal processing platform. The simulation results show that the model greatly enhanced the execution efficiency and response-speed of the platform.

1. Introduction

Since the software radio is proposed, as an important research direction of high-speed real-time information processing system platform design and development towards the components, platform direction[1], in general high-performance hardware processing platforms, such as ATCA, VPX Platform on the use of standardized, standardized general-purpose software architecture, information processing applications to achieve component development. Independent modular development of component functions makes it reusable. Polar shortens the development cycle of information processing applications and reduces R & D costs[2]. At the same time, the use of standardized software platform architecture shields the underlying hardware platform differences, R & D personnel can transfer the main work to the information processing component function code implementation, reducing the development threshold.

Traditional information processing systems use von Neumann-based software model, the use of control flow driven mechanism, the program execution sequence by programming the instruction stream to determine, in essence, is a serial implementation. In order to further enhance the efficiency of the implementation of the program, the current information processing platform are used data flow-driven mechanism, the purpose is to make full use of information processing between the parallelism of the components, making the entire system as efficient as possible high-speed work. The nature of the data flow-driven model makes the response to the real-time event is not strong, the system resource utilization is not high.

Event-triggered concurrent data flow model In order to make full use of the parallelism among the components in the information processing process and improve the efficiency of information processing by the platform, the thread nodes are added on the traditional data flow model and the priorities of different thread nodes are given priority. The deployment of system resources to

implement the algorithm complexity of the thread, thereby enhancing the overall information processing applications operating efficiency. However, on the distributed heterogeneous multi-core processor platform, this model can not provide elements such as container and component priority scheduling strategy, which seriously affect the performance of the platform.

In order to improve the scheduling efficiency of the information processing platform and improve the real-time information processing of the information processing platform, a new data flow driving model suitable for the information processing platform is urgently needed to effectively improve the operating efficiency of the system. DHDF and ECDF[3] are the scheduling models already in practical application, and ECDF has a more complete model framework, so this paper based on ECDF to consider the practical application of the scheduling strategy for components, container priority, cache, A new data stream driving model suitable for information processing platform is realized.

2. Component and data flow models

2.1 Component model.

A component is a binary executable or piece of data that has the ability to perform a specific intended work independently, and can provide services to applications, operating systems, and other components. The most important part of component technology is the component model, which describes the structure of the component, how it operates on the component, and the interaction between the components. At present, the main embedded component models are Koala model, Pecos model, ECOS model, PBO model, etc. In China, there are EZCOM model and other embedded component models, but there are still many shortcomings, such as lack of non-functional Koala model Constraints, ECOS model is only in the source code reuse phrase[4].

Traditional component model In order to adapt to the general scheduling framework, choose fewer scheduling attributes to provide the basis for more scheduling strategies, each component is described with a five-tuple:N $= (P_c, P_Q, T_B, T_E, T_L)$.

P_c:The relative priority of each server is determined by the importance of the processor on which the component resides;

P_Q:The relative priority between components;

T_B:The earliest expected start time for the component;

T_E:The latest expected end time of the component;

T_L:The expected execution time of the component request, which is the runtime cost of the function of the server component, is determined by the server component.

Although the model is the most widely used and supports most scheduling strategies, the model simplifies the structure of the component to the maximum possible number of scheduling algorithms. It only provides component priority and Such as the start and finish time of the request, the general component model lacks the component functions which are necessary for the scheduling algorithm when it is applied to the information processing platform with high real-time requirement. Obviously, the scheduling algorithm can not satisfy the requirement of component attribute Claim.

Therefore, in order to further improve the efficiency of the information processing application, we must reestablish a component model according to the scheduling strategy.

2.2 Data flow driven model.

Data flow program diagram calculation model is different from the von Neumann-based model of another calculation model, referred to as data flow model. Data flow driven model is generally described by the data flow diagram, as shown in Figure 1. The data flow graph is a special directed

graph composed of several nodes and edges. The flow of the data flows in the data flow graph by the process of the data token flowing on the edge. When a data token is present on all input edges of a node, the function of that node is activated.

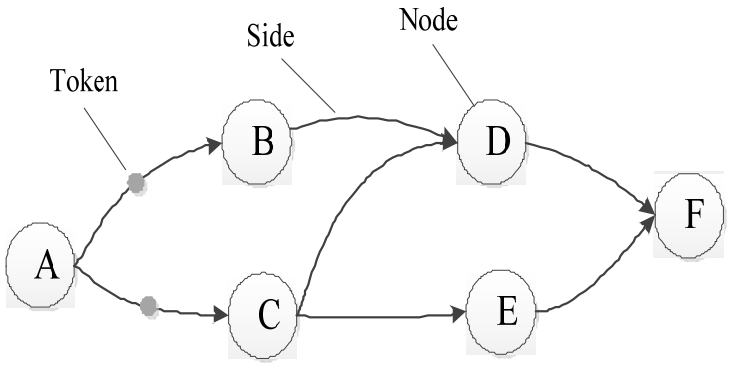

Figure 1 Data flow diagram

In a pure data flow model, a data flow diagram can be represented as a collection of nodes that process data and a set of edges for data transfer between nodes, so that the data flow graph can be represented as a pair (N, E) .Where N is the set of all the nodes in the program that finish the data processing function. Each node contains a set of input ports and a set of output ports. E is the set of logical connection relations among the nodes in the program. , Each side representing a data flow path. By computing nodes and transmitting data to achieve specific computing edge.

Each compute node may be represented as a six-tuple: $N = (F_A, F_E, T_I, T_O, P, S)$.

F_A Represents the activation function of the node,the return value indicates whether the node is activated. This function checks the input port set of the node and returns the true value if all input port data has been updated.

F_E Represents the calculation function of the node. When the activation function F_A of the node returns the true value, the function is executed. This function completes the function of the node and is the basic unit of complex calculation.

T_I Represents the set of all input ports of a node, and can be an empty set.

T_O Represents a collection of all output ports of a node, and may be an empty set.

P The attribute set of the node is determined according to the purpose of the node.

S Indicates the state set of a node, which indicates the activation and execution of the node.

In data-flow-driven mode, data is transferred directly between instruction nodes in the form of data tokens, which are driven by the availability of data tokens[5]. According to the different ways of data token processing, the data flow model can be divided into static data flow model and dynamic data flow model.

Static data flow model works as follows: Only when the data token of all the input edges of a certain node is available, and there is no token on any output edge of the node, the node can be activated and perform the corresponding operation[6].

The working mechanism of the dynamic data flow model is similar to that of the static data flow model. The most important feature is that the data token is marked so that at any given moment, any one edge of the data flow graph is allowed to have multiple Data token. The token of the data token is accompanied by a label that identifies the token's time-to-previous correspondence. As long as a data token with the same token is present on each input edge of the node, the node function is activated to perform the appropriate action. Marked tokens avoid the confusion of data and improve the accuracy of information processing. At present, this kind of dynamic data flow model with mark

is the most studied data flow model in the world[7].

3. A New Data Flow Model for Information Processing Platform

Components in the information processing platform are divided into two basic elements: containers and components. The component part provides a template for creating actual components, which is the basis for component creation and utilization. The container section defines a way to group components together into useful structures, providing a locale for arranging components and their interaction with other components. Components are deployed to the processor placed in the same or different containers, mutual communication between the container through the shared cache to achieve shared cache can effectively improve system resource utilization. When the data satisfies the activation condition at the same time, the scheduling algorithm needs to decide the activation function of the component according to the component priority.

3.1 Component Data Interface Model and Management.

As shown in Figure 2, the platform components include input / output ports, shared buffers, connectors, component function functions, computational buffers, and other available byte functions.

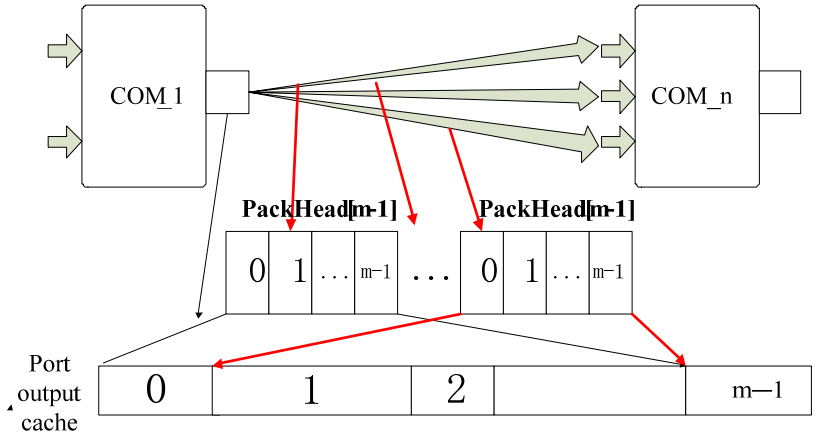

Figure 2 Component model

Input / output port is the port of the data in and out of the component, at the same time the input port data is ready to be one of the flags that the component can be called. The connector takes the source and destination components from the connector function and concatenates the output port of the source component with the input port of the target component so that it shares a cache. Component Function The function is used to implement the specific function of the component.

Taking into account the information processing platform components and the correlation between the high-speed data communication between each other, the cache interface component model as the most important part. In order to solve the problem of data communication between different clock domains in the system, the first-in-first-out queue (FIFO) is introduced to parallelize the read and write operations. The output ports correspond to multiple connectors and share the same output buffer. In order to facilitate the management of shared cache, the shared cache is divided into several blocks, each block size can be different[8], each connector has m block. Data processing to complete the data read and write a cache block read and write is completed once the data distribution function is called, and then the next block of data cache operation.

Due to the limitation of component port cache size, when data processing is carried out quickly,

it is inevitable that data overflow occurs due to insufficient buffer. This requires that the data processing speed of the component be regulated when the buffer overflows. The number of available port bytes in the interface function can be used as a reference for the component's data processing speed control, which has been demonstrated in the component model.

3.2 New Data Flow Driving Model.

In order to meet the needs of the information processing platform scheduling strategy, a new data stream driving model for information processing platform is proposed after further research on components and containers.

The data flow model can be described as a four-tuple (N, C) , Where N is the set of all components used in the data processing flow, each containing a set of input ports and an output port set; C is a collection of component storage containers in the data processing flow, each container providing One or more components of the operating environment, but also contains an input port set and an output port set, the priority of the container set by the user;

A component can be represented as an combination : $\mathrm{N} = (F_N, P_N, T_B, T_I, T_O, \mathrm{H}, \mathrm{E}, \mathrm{S})$,among them:

F: A function that represents a functional component, a component that is activated, executes the function, and performs a processing function for the component;

P_N: The component priority, component priority by the component itself or user-specified attributes;

T_B: Indicates the total execution time of the component request, that is, the execution time of the interface function of the server component, as determined by the server component.

T_I: That a node all the input port of the collection, can be empty set;

T_O: A collection of all output ports of a node, it can be an empty set;

H: Indicates the number of available bytes in the input and output ports of the component, and the minimum available bytes in both input and output buffers.

E: That the current component of the source component and target components;

S: Indicates the component's state set, which indicates the activation and execution of the component.

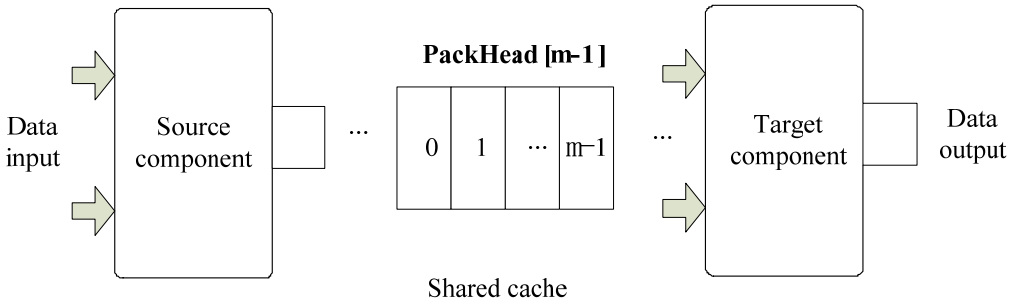

Figure 3 to deal with the information processing platform based on event-triggered multi-threaded data flow model data processing

In the new event-triggered multi-threaded dataflow model platform model, after the data enters the processing flow, the source component calls the component function to complete the data processing, enters the port write function, and writes the processed data to the output port FIFO Buffer, when the buffer of a data block is filled, call the data distribution unit. According to the component connector information, the data distribution unit searches the component target component handle, and then finds the handle of the container where the target component is located, and activates the task of the container where the target component is located by the write event. The

number of available bytes in the buffer of the output port during data processing can provide a reference for the scheduling algorithm to adjust the operating speed of the component.

For the receiving end of the data. When the container for the target component is activated, it loops through the data state of all the components in the container. When all of the input ports of the target component are ready for data, the container activates the target component handler for data processing. Data processing on the call to read the port function, from the component input port FIFO buffer to read data for processing, while changing the input port FIFO are pointer location.

Data This process is repeated between components and components until the entire data processing flow is complete.

4. Experimental results

In this paper, the ECDF model and the new data flow model (NDFD) are evaluated from the system real-time and the system resource interest rate using the task graph shown in Fig 4.

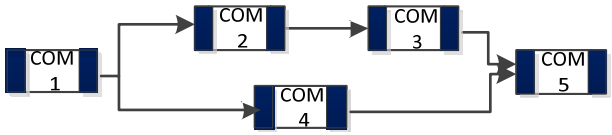

Figure 4 Process flow

Test Experiment 1: In the ECDF model, the data read component reads 4096 signed integer values at a time. When the data on all input sides of the component is ready, the component is activated and performs the corresponding operation. The NDFD model also reads 4096 signed integer numbers. When more than one component satisfies the activation condition, the corresponding actions are activated and executed in the component priority order. The average time for component execution and the average time to complete a data transaction are shown in Table 1.

Table 1 Scheduling results based on ECDF model and NDFD model

Node	COM1	COM2	COM3	COM4	COM5	Total time
ECDF	17	3598	5780	4652	2025	17032
NDFD	19	3601	5800	4600	2100	15069

Test Experiment 2: Each component in the ECDF model needs a fixed-size buffer. All components of the NDFD model need only one shared buffer. The cache sizes required for the two models are shown in Table 2.

Table 2 Buffer size based on ECDF model and NDFD model

Node	COM1	COM2	COM3	COM4	COM5	Total cache
ECDF	1024	1024	1024	1024	1024	5120
NDFD	0	0	0	0	0	4096

From the experimental results, it can be seen that the use of the new data stream driver model can significantly reduce the processing time of the system application. The cache size of the two models is also clear that the new data flow model can meet the application requirements in the case of smaller system cache, and improve the system resource utilization.

5. Conclusions

The traditional data flow model has been widely used in software development, system scheduling and so on. But when applied to the information processing platform, it can not support the scheduling strategy which is more demanding in real-time, mainly existing data flow The driver

model can not support the key data such as component priority for the scheduling algorithm. Event-triggered data flow model is a good choice, but with the information processing platform of the poor match. In order to solve this problem, this paper improves the information processing platform, increases the number of available bytes in the port buffer and the component priority, and proposes a new data flow driver model for the component information processing platform. Experiments show that this model is effective and feasible, and it has laid a solid foundation for future scheduling strategy with strong real-time.

References

[1] Bezati E, Thavot R, Roquier G, et al. High-level dataflow design of signal processing systems for reconfigurable and multicore heterogeneous platforms. Journal of real-time image processing. Forum Vol., 251-262 (2014).

[2] Liu Long. COM component programming design method [D]. Harbin Engineering University, in December 2006.

[3] LI Hai.Dataflow-driven general radar simulation framework [J] .Beijing Institute of Technology, September 2005, 25 (9): 823-830.

[4] Dong Yuchi. COM component implementation strategy design and development [D]. Chengdu: University of Electronic Science and Technology, 2014 July.

[5] Yazar T. Design of Dataflow[J]. Nexus Network Journal, 2015, 17(1): 311-325.

[6] Ha S, Oh H. Decidable dataflow models for signal processing: Synchronous dataflow and its extensions[M]//Handbook of Signal Processing Systems. Springer New York, 2013: 1083-1109.

[7] Lu Junjiang. A Radar Information Processing Multi-task Scheduling Algorithm [J]. Information and Electronic Engineering, August 2012, 10 (4): 460-464.

[8] HAN Dong. Modeling and Design of Data Cache in Transparent Transparent Interface [J]. Electronic Design & Engineering, October 2014, 22 (20): 190-193.