I.J. Education and Management Engineering **2011, 5, 10-16**
Published Online November 2011 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2011.05.02

Available online at http://www.mecs-press.net/ijeme

Design of High-Performance Real-Time Bus in Parallel Processing System

Cheng Xin, Zhou Yunfei

State Key Lab of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, P.R. China

Abstract

Rapid increases in the complexity of algorithms for real-time signal processing applications have made multi-processors parallel processing technology needed. This paper proposes a design of high-performance real-time bus (RTB), based on which distributed shared memory (DSM) mechanism is established to implement data exchange among multiple processors. Adopting DSM mechanism can reduce the software overhead and improve data processing performance significantly. Definition and implementation details of RTB and data transmission model are discussed. Experimental results show the stable data transmission bandwidth is achieved with performance not affected by the increasing number of processors.

Index Terms: Real-time bus, parallel processing, data exchange, distributed memory

1. Introduction

Rapid increases in the complexity of algorithms for real-time signal processing applications have led to performance requirements exceeding the capabilities of conventional digital signal processor (DSP) architectures [1]. Many applications targeted for DSPs, such as motion control system for ultra-precision multi-dimension stage in lithography device, need to adopt multi-DSP parallel processing architecture.

The purpose of parallel processing is to shorten the execution time of computing tasks by adopting multiple processors to process incoming signals simultaneously [2]. Therefore, speed-up ratio is closely related to task parallelism, which is the bottleneck of parallel processing performance. Efficient distribution of parallel tasks will inevitably involve the requirements of data transmission among processors. As a result, it needs to establish high-speed data transmission channel among processors to complete real-time data exchange, which ensures the parallel processing performance.

C6000 series DSP in TI has the advantage of high cost-effective, so it is widely used in signal processing systems. However, there is no multi-processor interface and link port in this series, which makes it difficult to

* Corresponding author.
E-mail address: Amo.hust@foxmail.com

construct high-speed data transmission channel to form multi-DSP parallel mechanism, and serious solutions aimed to this problem are developed. Multi-DSP parallel system based on standard bus, e.g. PCI, VXI and VME, etc. has the advantages of modularization and scalability. [3] investigated typical applications of multi-DSP systems based on standard bus, but schemes adopted in those applications are complex and adopts shared bus or shared memory, which makes systematic overall performance restricted by the bus bandwidth [4].

Multi-channel buffered serial port (McBSP) has the data limited transmission bandwidth, which is not suitable for high-speed data transmission. With dual-port RAM (DPRAM) widely used in data exchange between processors, [5] and [6] introduced a multi-DSP system based on DPRAM and first in first out (FIFO) respectively. Yan and Zhang [7], Pan and Zhao [8] adopt same thinking to design a shared memory structure to realize the high-speed data transmission among 3 and 4 DSPs respectively but avoid the performance bottleneck - shared bus bandwidth. However, this shared memory structure will be extremely complex if more DSPs exist. Furthermore, for the fixed interconnection structure among DSPs, parallel system has poor performance in reconfiguration and scalability. Fixed connection is sometimes not needed - no doubt a waste of resources.

DSM systems allow relatively easy modification and efficient execution of existing shared-memory system applications, which preserves software investments while maximizing the resulting performance. In addition, the scalability and cost-effectiveness of underlying distributed memory systems are also inherited [13-15]. This paper proposes a design of high-performance real-time bus (RTB), based on which distributed shared memory (DSM) mechanism is established to implement data exchange among multiple processors. Bus topology and broadcast transmission are adopted in RTB to achieved stable high performance. Communication protocol is implemented in FPGA with no software overhead needed.

The rest of the paper is organized as follows. Section 2 introduces the function of RTB in parallel processing architecture, while section 3 discusses the implementation details and key points. Section 4 provides the data transmission model and analysis, and section 5 describes the experimental results. Finally, section 6 concludes the paper.

2. Real-time bus in parallel processing

As shown in Fig.1, RTB aims to implement data exchange of distrRTButed local memories to establish DSM mechanism serving processors involved in parallel architecture. Adopting this DSM mechanism can reduce the software overhead and improve data processing performance significantly. Advanced communication architecture makes the data transmission performance not affected by the increasing number of DSPs, meanwhile improves the scalability and reconfiguration function.

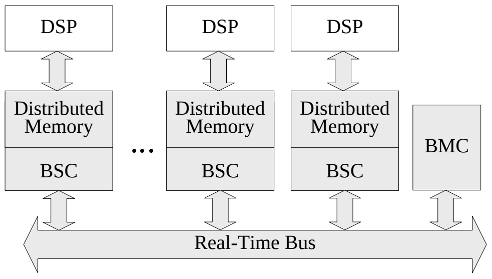


Figure 1. Real-time bus in parallel processing

Every DSP has distrRTButed local memory connected with it. BMC（Bus Master Controller）generates RTB timing and provides control signals. BSC connects with distrRTButed memory, and determine whether the BSC is a sender or receiver in the current cycle, and then switch the direction of data bus and execute data sending or storage according RTB protocol in every data exchange process.

Definition of distrRTButed memory is the same and the responsRTBility of RTB data transmission is to realize the data exchange between different memories. What DSP does is only to write the to-be-exchanged data into its private memory block. DSP can access to its local memory freely without taking part in data exchange.

DistrRTButed memory is composed of DPRAM with one port connected to DSP and another port connected to BSC. Conflict caused by simultaneous memory location access can not be ignored. The common solution is “busy flag arbitration” provided by DPRAM, which can hold the later access until the end of the previous access.

3. RTB protocol and implementation

A. Definition of RTB protocol

RTB provides a real-time and fast medium for transferring data among BSC. It consists of a 16-bit address bus, 32-bit data bus and 2 control lines. The relevant definition is described in table I.

4. Definition of RTB signals

RTB is a non-multiplexed synchronous broadcast bus, and its protocol is given below. (1) although there is no synchronous clock signal, ADD and AEn signals are both triggered by the same internal global synchronous clock (DATA and DTACK are provided by BSC), and no handshaking is involved in data transmission; (2) unlike shared address / data signal lines of multiplexed bus, the Address and data are given at the same time on separate address and data signal lines. (3) BMC provides address and control signals, and the selected BSC provides the data while all other BSCs stores the data. Considering the requirement of strict data transmission latency, it is impossible to adapt complex handshaking protocol to determine which BSC is the data sender. ADD and AEn signals are used to address the data sender (a certain BSC). Once the ADD signals are valid (AEn valid), the identities of sender and receiver are determined.

signals	BSC		BMC	Description
	Output Side	*Input Side*		
ADD	In	In	Out	Driven by BMC. BSC uses these address lines to determine whether the cycle is to be a read or a write cycle
AEn	In	In	Out	Driven by BMC. Indicating that ADD valid for low level;
DATA	Out	In	Not care	During each RTB cycle these lines are driven by exactly one BSC and all the other BSCs store this data;
DTACK	Out	In	Not care	Indicating that DATA are valid for low level;

B. Implementation of data exchange

In every RTB cycle, BMC drives ADD and AEn signals and BSC monitors those signals. If ADD [15:12] matches the number of slot in which BSC is located, the current BSC is selected. Obviously, during each cycle, there is always exactly one BSC for sender, which puts data on the data lines, and then drives DTACK signal to confirm the validity of data. BMC is the only RTB master, while the quantity of BSC is the same as that of DSP. Figure 4 gives an example of a RTB cycle.

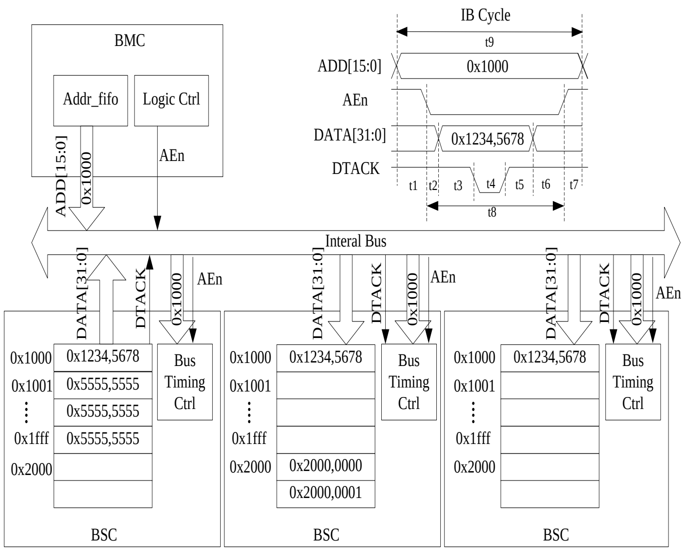


Figure 2. One RTB data transmission cycle and timing

In each parallel computation cycle, a data transmission sequence is to realize data exchange among different memories. Data transmission sequence is composed of several RTB cycles, and 32-bit data transmission is implemented in each RTB cycle. As discussed above, the ADD signals in each RTB cycle contain the definition of sender and receiver, so that it can set the data transmission sequence by setting RTB address. The setting of the sequence depends on the actual data requirement among DSPs, and can be reconfigured by setting the relevant registers in BMC. It is very important to downsize the sequence because it can significantly reduce the data transmission time, which can improve the efficiency of data transmission.

5. Data transmission model and analysis Template

The model described in this section is only for data transmission on RTB where only one transmission mode is involved, as shown in Fig.3 (a). DSP1 is the only source node, while other DSPs are destination nodes, and broadcast mode with single host and multi-slave are adopted. Fig. 3 (b) illustrates the analysis of data transmission model.

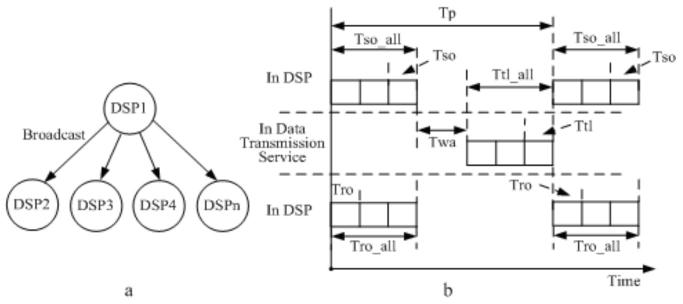


Figure 3. Data transmission model

TABLE I. PARAMETERS LIST IN DATA TRANSMISSION MODEL

Parameter	Description
T_p	The length of time for one parallel computing cycle.
T_{so_all}	The length of time the sending processor is engaged in the data transmission in a parallel computing cycle; during this time the processor cannot perform other operations.
T_{so}	The length of time the sending processor is engaged in the transmission of a 32-bit data.
T_{wa}	The length of time spent in waiting execution of data transmission sequence.
T_{tl}	The length of time spent in the transmission of a 32-bit data from sending to the receiving node.
T_{tl_all}	The length of total time spent in the transmission network from sending to the receiving node in a parallel computing cycle.
T_{ro}	The length of time the receiving processor is engaged in the transmission of a 32-bit data.
T_{ro_all}	The length of time the receiving processor is engaged in the reception of data in a parallel computing cycle; during this time the processor cannot perform other operations.
L_d	The size of data in transmission sequence, in word (32-bit), transmitted in a parallel computing cycle.
K	The ratio between T_{tl_all} and T_p , which indicates the efficiency of data transmission function.
N	The number of processors in the parallel architecture.

$$T_{tl\_all} = L_d \times T_{tl} \qquad (1)$$

$$\begin{aligned} T_{total} &= T_{so\_all} + T_{tl\_all} + T_{ro\_all} + T_{wa} \\ &= L_d(T_{so} + T_{tl} + T_{ro}) + T_{wa} \end{aligned} \qquad (2)$$

$$K = L_d \times T_{tl} / T_p \qquad (3)$$

In formula 1, T_{tl} is the transmission latency of single datum, in word (32 bit), and its stability only depends on the network transmission performance. Therefore, T_{tl_all} solely depends on L_d and can be calculated accurately. In formula 2, T_{so} and T_{ro} are determined by EMIF Settings, and T_{wa} indicates the total time spent on waiting for data transmission service. In formula 3, K indicates the efficiency of data transmission service, and the smaller it is, the less time spent on data transmission during the parallel computing cycle. The uncertainty of T_{total} is mainly caused by DSP software plan (T_{wa}), but not T_{tl_all} . In practical parallel servo computing, T_{tl_all} must be accurate and to small proportion of T_p for the reason that partition of data-insensitive and data-sensitive computing is based on it. On the contrary, there is no high requirement for T_{total} accuracy.

6.Experiment and result

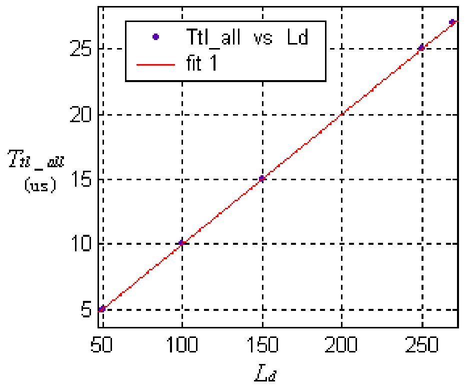
The time parameters of RTB timing mentioned in Fig. 2 are listed in table III.

TABLE II. TIME PARAMETERS OF RTB TIMING

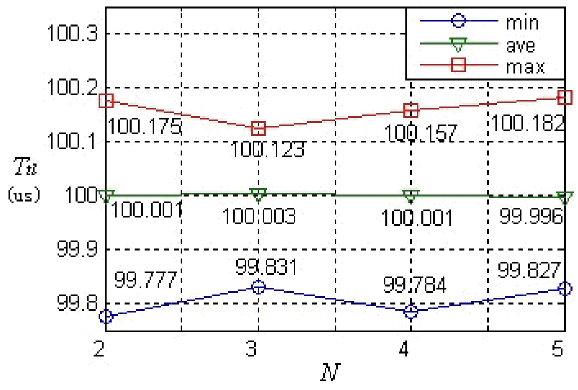
Parameter	Value (ns)			Description
	Min	*Tpy*	*Max*	
t1	-	20	-	ADD[15:0] valid before AEn valid
t2	0	10	20	AEn valid to DATA[31:0] valid
t3	10	10	20	DATA[31:0] valid to DTACK valid
t4	20	20	30	DTACK low pluse
t5	-	10	-	DTACK invalid to DATA[31:0] high impedance
t6	0	20	20	DATA[31:0] high impedance to AEn remains valid
t7	10	10	30	ADD[15:0] remain valid after AEn invalid
t8	-	70	80	AEn low pulse
t9	-	100	-	Total IB cycle time

Figure 4 (a) plots T_{tl_all} with respect to L_d using results from experiment with (N =6). Indicators of the goodness of the fit are: SSE（sum of squared error）=1.19e-005, RMSE（root mean square error）=0.001992, R-square= 1. From the plot, it is not difficult to see that the linear dependency of T_{tl_all} on L_d is almost perfect.

Figure 4 (b) plots the value of T_{tl} obtained from experiment. The error is of sub-nanosecond scale, and the major part of it is caused by non-uniformity of signal transmission characteristics in RTB backplane and internal reference clock jitter in BMC. According to the experimental results, T_{tl} is accurate enough, so that the T_{tl_all} and K can be both calculated accurately using the value of L_d .



(a) T_{tl_all} vs L_d , with N =6;



（b） T_{tl} vs N

Figure 4. Performance of RTB data transmission

7. Conclusion

This paper proposes a design of high-performance RTB, several conclusions can be drawn from preceding discussion and analysis of experimental result.

(1) Data transmission performance on RTB is not affected by the increasing number of DSPs.

(2) Stable achieved bandwidth in communication is 10M *32bit/S, and it can be increased significantly by improving signal transmission characteristics in RTB backplane.

(3) In practical application, time spent on data transmission is only 6.575% in every parallel computing cycle.

References

[1]. James Kohout, Alan D. George, A high-performance communication service for parallel computing on distrRTButed DSP systems, Parallel Computing, 2003, 29: 851-878.

[2]. Murphy C W , Harvey D M , Nicolson L J . Low Cost TMS320C4O/ XC6200 Based Reconfigurable Parallel Image Processing Architecture[C]. Proc. of IEE Colloquium on Reconfigurable Systems, Glasgow, U K. 1999: 91-95.

[3]. Model 4293 Qctal TMS320C6203 Processor VME Board, http://www.pentek.com.

[4]. Aleksandar Milenkovic, Veljko Milutinovic. A performance evaluation of cache injection in bus-based shared memory multiprocessors. Microprocessors and Microsystems, 2002.2(26): 51-61

[5]. Ercan Fikret M, Fung Yu-fai, Suleyman Demokan M. Communication in a multi-layer MIMD system for computer vision[J].Journal of Systems Architecture, 2000 , 46 :1349 - 1364.

[6]. TMS320C6000 EMIF to External FIFO Interface. Spra 543, 1999

[7]. Yan Luxin, Zhang Tianxu, et al. A DSP/FPGA-based parallel architecture for real-time Image processing. Proc .of 6th World Congress on Control and Automation，Dalian, China, 2006: 10022 - 10025.

[8]. Pan Fangsheng, Zhao Feng, et al. Implementation of Parallel Signal Processing System Based on FPGA and Multi-DSP. Computer Engineering, 2006, 32(23): 247-249. （in Chinese）

[9]. Li Qun, Xie Li, Sun Zhongxin. The techniques and implementation of a distributed shared memory. Computer research & development.1997.5(34) 327-331

[10]. Jelica Protit, Milo Tomasevit, and Veljko Milutinovit. Distributed Shared Memory: Concepts and Systems. IEEE Parallel & Distributed Technology, 1996.2(4):63-79

[11]. Zhen Fang, Lixin Zhang, John B. Carter, Liqun Cheng, Michael Parker. Fast synchronization on shared-memory multiprocessors: An architectural approach. J. Parallel Distrib. Comput. 65 (2005) 1158 – 1170