A Research of the Development of VME Bus in VxWorks

Yi-min ZHOU
Dept. of Computer
School of Optical-Electrical and Computer Engineering,
University of Shanghai for Science and Technology
Shanghai, China, 13901854513

JUN WANG
Dept. of Computer
School of Optical-Electrical and Computer Engineering,
University of Shanghai for Science and Technology
Shanghai, China, 15121043705

***Abstract*—Based on the principle of the VME bus and combined with lithography machine control system, This paper briefly introduced the development of VME bus under VxWorks operating system, including the bus address space allocation and interrupt setting method. VG5, VME bus master board, is GE Fanuc's high-performance PowerPC single board computer running in the system of VxWorks and it is suitable of standard 6U VME classis. We analysis and modify BSP to achieve interruption and access of board register, and ultimately implement communications between boards on the VME bus.**

Keywords-VxWorks operating system; VME bus; BSP; VG5

I. INTRODUCTION

With the rapid development of embedded systems, embedded operating system has been widely used in various areas, such as network communications, industrial control, communications, defense and aerospace. WindRiver's VxWorks operating system occupies an important position in the field of embedded applications due to the features of preemptive scheduling, interrupt latency and the system kernel can be cut.

VME (VersaModule Eurocard) bus was first introduced by Motorola and it is a 32 industrial open standard backplane bus, which later become the IEEE standard. After years of upgrading, the VME system has been developed very well. Products that developed around VME system throughout many high reliability required areas, such as industrial control, aerospace, transportation and medical.

II. VME BUS CHASSIS

A. Control System Composed of VME Chassis

Lithography machine control system implements communications between various functional boards through the VME bus. VME lithography machine control system is functionally divided into various subsystems to complete real-time functions. In general, the coupling functions are divided into the same subsystem and we will achieve the sub-system functions through a VME bus chassis.

An integral VME Chassis is composed of VME bus, VME bus control panel and VME function board. Controlling of each subsystem is implemented by subsystem control panel and its core software is solidified in the flash.

In the scheduling of VME bus control panel processor, different functional boards on the VME bus cooperated to achieve functions of the subsystems. Subsystem control panel completes the communication between the host computer and other subsystems via Ethernet and RS232.

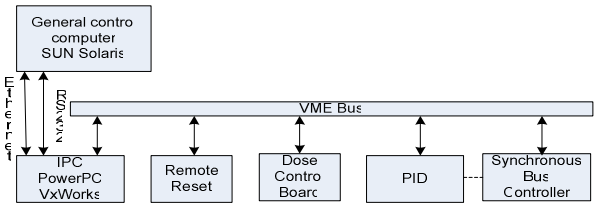

Figure 1. Lithography Lighting Projector Subsystems

B. Master- Slave Board on VME Chassis

A VME chassis card, sized for 6U, is a flexible, scalable and multi-processor supportable bus system, which using master-slave structure. As VME bus is one of the VME bus system shared resources, it only allows one master device to drive VME bus at the current time.

Master: launch the data transfer request. Multiple masters can reside in the VME bus.

Slave: response to requests from master. A board card can be both the master and the slave. Most I/O board card is slave and a small part is master. Generally, A master is also a slave.

Arbiter: give master the right of bus controlling. A system can only have one arbiter, which is usually played by the system control board located in the first slot.

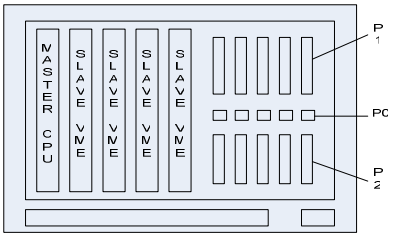

Figure 2. Architecture of VME Chassis

Pic2 shows a typical VME chassis. Master CPU board in the left slot acts as an arbiter, and it is also the Slave. VME sub-boards inserted in the 1-4 slots are slaves. Port P1 is a standard VME interface, which supports 24-bits address and 16-bits data. Port P2 is an expanded port that supports 32-bits

address and data.

It is the hardware that decides whether a board is a master, slave or arbiter, and it cannot be determined by software.

III. VME BUS ADDRESS MAPPING

A. *Mapping Principle*

It's consist of I/O(A16),Standard(A24),Extend(A32). A16 dimensional size was 64 K, A24 was 16 M, A32 was 4G to maximum in theory, but it could use some address space of all actually.

Local CPU could map external VME BUS address space on local CPU address space so as to make local CPU visit VME BUS.

The other mapping method called Slave window mapping, which mapped part of the local RAM on VME BUS so as to visit local RAM by VME BUS.

Address Modifier is mainly used for designing VME BUS address and visiting method, which had six control lines to make the following set:

1) Authority Level: It made no sense to VxWorks between root users or normal users' mode, for all VxWorks were root users' mode, otherwise I/O cards used hardware jumper wire to set;

2) The mode of visiting: the data, the program or block transmission;

3) The size of address: Short I/O、Standard or Extended.

B. *Configure BSP*

VxWorks BSP had nine macro definitions to configure the three address space. Define the following nine macros to set the VME Master window mapping in the card VG5's BSP.

VME_A16_MSTR_SIZE the size of the window of A16 storage space

VME_A16_MSTR_BUS the first address of the bus of A16 window

VME_A16_MSTR_LOCAL the local address of A16 window

VME_A24_MSTR_SIZE the size of the window of A24 storage space

VME_A24_MSTR_BUS the first address of the bus of A24 window

VME_A24_MSTR_LOCAL the local address of A24 window

VME_A32_MSTR_SIZE the size of the window of A32 storage space

VME_A32_MSTR_BUS the first address of the bus of A32 window

VME_A32_MSTR_LOCAL the local address of A32 window

When these values set up, the Master window mapping completed, also was the VME bus address mapped on the local CPU address. If to disable a window wanted, as long as to set the size of the corresponding window zero. The default memory mapping of VG5 was show as Fig3.

Slave window mapping made local RAM map on the VME bus so as to serve other masters for visitation. The BSP macro definition of the Slave window mapping is familiar with the Master window mapping, VG5 mapped all the local memory on the A32_Slave window.

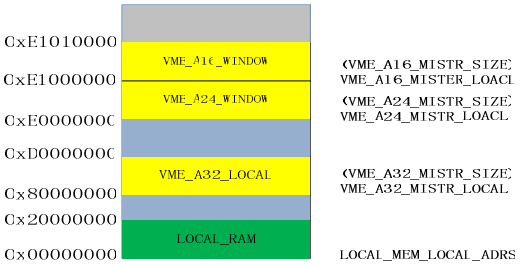

Fig3 .Memory Mapping of VG5

If several cards needed to map the local memories on VME, address space setting shouldn't have overlapping conflicts and to set the space size as 4K as far as possible.

The method of the CPU to visit some registers of VME: the base address of the local VME address space + the address of the card in VME space + registers offset value.

Assuming that a piece of VME card mapped inner register on VME BUS (A24), the value of the VME address is 0x200000, and a register offset value is 0x100. Take Fig8 as example, VxWorks could visit the value of register as long as visiting 0Xe0200100 in the VG5. The value of 0xE0200100 could also got by sysBusToLocalAdrs () provided by VxWorks system.

C. *Experiment*

Several VG5 boards work in VME master mode. Assuming we put 4 VG5 boards in one VME case, all of them will access to VME sub-board and working in VME_master mode. Meanwhile, local memory should mapped to VME bus, where other boards can access to contents in other memory.

Assuming we choose VME A24 mapping mechanism. We map 64K contents in the last 1M in each board to VME bus for accessing. Setting parameters are as follows:

Parameters in VME master: default settings.

Parameters in VME laver:

VME_A24_SLV_LOCAL……0x1ff00000 /* from the last 1M of memory */

VME_A24_SLV_SIZE ……0x10000 /* 64k size shared to the VME bus */

The following parameter settings are different in each board as follows:

CPU0: VME_A24_SLV_BUS……0x0

CPU1: VME_A24_SLV_BUS……0x100000

CPU2: VME_A24_SLV_BUS……0x200000

CPU3: VME_A24_SLV_BUS……0x300000

Meanwhile, we should set the default VME_A32_SLV_SIZE to 0 in BSP, or change VME_A32_SLV_BUS in each board to a different value to avoid conflict. After the parameter settings of each VG5 board, we should compile them to a VxWorks mirror image respectively.

IV. VME BUS INTERRUPT

A. *VME Bus Interrupt Principle*

VME bus supported 7 level priorities. Several VME bus cards could requested the same lever interrupts at the same time. In the VME bus system which contained several processors, an interrupt lever could only be used by one processor card, that was to say VME bus had 7 interrupt to use, a processor could use any lever for interrupting, certainly a processor card could use several VME bus to interrupt.

For several equipments could use one interrupt and different processor card could use different interrupt lever. The interrupt handling steps were shown as following:

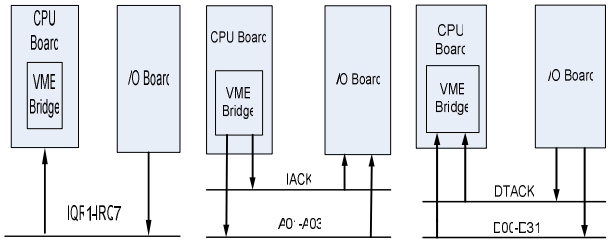

Figure 4.The Process of. Interrupt

First, A VME interrupt was generated at first; I/O card selected an interrupt line to set, which would generate a corresponding lever interrupt. If CPU approved the interrupt, it stepped into the interrupt response phase.

Second, CPU put the values of the interrupt lever on the A01-A03 address lines, and then set IACK signal lines. IACK signal lines was transmitted as the daisy chain mode on the VME backplane, every card should confirm whether the interrupt was generated by itself from the first tank, if do the card would step into the interrupt response phase continuously, otherwise it transferred to the next one.

At last, I/O card put the interrupt data on the data bus, and set DTACK signal line. VME bridges on the CPU card read the interrupt data, and then mapped them on the corresponding interrupt. Above all the interrupt response phase was completed.

B. *Interrupt in VxWrosks*

The interrupt vendor table in a VxWorks system has 256 entries. As some vendors has been used in VG5 system, we recommend to use even vendors ranged from 0x70 to 0xff.

VME interrupt priority ranges from 1 to 7, and 7 is the highest priority. VxWorks operating system enables VME interrupt by calling sysIntEnable () function.

1) Interrupt processing flow is as follows:

a) VxWorks operating system binds the interrupt service program to a appropriate interrupt vendor by calling intConnect() function.

b) VxWorks operating system enables the interrupt by calling sysIntEnable() function. Step 1 and 2 are usually called when system processing initialization or the board driver initialization.

c) VME sub-board requests VME interrupt and informs the host controller of the interrupt priority and value via interrupt response mechanism.

d) When interrupt response completed, system triggers the CPU interrupt of the host controller.

e) When CPU interrupt occurred, system suspends the processing programs. According to the interrupt vendor table, system locates and processes the interrupt service program via interrupt vendor.

f) After interrupt service program executed, system quit interrupt and reprocess suspended programs.

C. *Experiments*

1) Experiment 1: Trigger VME interrupt on VG5 board.

For VG5, composed of PowerPC, if we make VME as a slave, how do we trigger a VME interrupt?

In VxWorks operating system, we only need to call sysBusIntGen() function, provided by BSP. This function usually exists in the driver code in VME chip. The original function is as follows:

……STATUS.sysBusIntGen
……int.lever……./*interrupt.lever.generate…*/
……int.vector……/*interrupt.vector.for.interupt*/

2) Experiment 2: board reset function setting

We propose to insert several VG5 board into one case. When we reset one VG5 board, other board may be reset. This is because reset VG5 is implemented by reset VME bus, other VG5 boards in this bus may receive the reset signal.

It can be software configured to determine whether do rest action to VME bus in VG5 board. When we call sysVmeResetOn() function to reset VG5 board, rest signal triggered by VME bus will reset other boards in this case.

This function can be forbidden by calling sysVmeResetOff() function.

3) Experiment 3: how to schedule VME interrupt in different I/O board into different CPU board.

For some special applications, such as there are 2 CPU boards and several I/O sub-board in VME case. How can we implement that VME interrupts triggered by some I/O sub-boards impact on CPU0 and others triggered by other I/O sub-boards have an effect on CPU.

Assuming that, CPU0 locates in slot 1 and CPU resides in lot 2 in a VME case. TMB0 and TMB1 act as VME slave and they are inserted in slot 5 and 6 respectively. We can do the following settings:

Setting the TMB0 interrupt priority to TRQ3 and TMB1 to IR4; setting IRQ3 interrupt enable and IRQ4 disable in CPU0; setting IRQ3 disable and IRQ4 enable. VME interrupt priority enable and disable functions are sysIntEnable () and sysIntDisable ().

Then, interrupt triggered by TMB0 and TMB1 will respectively occurred in CPU0 and CPU1.

V. CONCLUSION

In this paper, we introduced the analysis and configuration methods of VME bus in VxWorks operating system. We mainly focus on the bus address space mapping principle and interrupt settings. The VME bus driver has been widely applied in practical project at present.

ACKNOWLEDGE

This project is supported by Shanghai Microelectronics Equipment Co., Ltd.

REFERENCES

[1]. Wind River Systems Inc, VxWorks Programmer's Guide. Wind River Systems,2001

[2]. Wind River System Inc, VxWorks BSP Reference: VG5

[3]. GE Fanuc Automation, VME_UNIVERSE: VME bus Driver and Tools for Linux. 2004

[4]. A. Aloisio, P. Branchini, F. Cevenini, "Timing analysis of asynchronous block transfer cycles on VME and VME 64x physical layers", IEEETrans. on Nucl. Sci., vol. 51, n. 3, pp. 401-406, Jun. 2004

[5]. B. G. Penaflor, J. R. Ferron, M. L. Walker, D. A. Piglowski, and R. D.Johnson, "Real-time control of DIII-D plasma discharges using a Linuxalpha computing cluster," Fus. Eng. Des. Vol. 56-57, pp. 739-742,2001

[6]. K. G. Ricks, D. J. Jackson, and W. A. Stapleton, "An evaluation of the VME architecture for use in embedded systems education," in Proc.Workshop on Embedded Systems Education (WESE 2005) Held in Conjunction with the EMSOFT 2005 Embedded Systems Software Conference, Jersey City, NJ, Sep. 22, 2005, pp 59-65.

[7]. K. Hwang, Advanced Computer Architecture: Parallelism, Scalability,Programmability. New York: McGraw-Hill, 1993.

[8]. W. D. Peterson, "VME Technology Frequently Asked Questions," [Online].Available: http://www.vita.com/vmefaq.html.

[11]A. Aloisio, P. Branchini, F. Cevenini, V. Izzo, S. Loffredo, R. Lomoro,"Signal integrity and timing issues of VME64x double edge cycles",IEEE Trans. on Nucl. Sci., vol. 53, n. 2, pp. 520-525, Apr. 2006.