Design of Embedded Multiprocessor Platform with Cooperating Synchrony

Haiyong Wang Bing Zhang Shu Xu Zhiyi Wang
School of Astronautics, Beihang University
37, Xueyuan Rd., 100191, Beijing, China
Email: *why@buaa.edu.cn*

Abstract *–To meet the requirement of high-speed data processing of complicated system with multi-sensor inputs and multi-controller outputs, an embedded ARM+FPGA +DSP multiprocessor platform is designed, which is characterized by synchronous cooperation and mutual exchange of shared data. ARM completes interface expansion and task scheduling configuring peripheral interfaces including RS422、CAN and RJ45, and it loads program into the 3 DSPs via HPI bus. Shared data stored in multi FIFO core sets in FPGA can be transmitted between any two processors controlled synchronously by interrupt signal. The interrupt signal from ARM starts a new turn of operation, in which period all processors exchange necessary shared data mutually, after ARM collecting 3 DSP interrupt signals, it means the end of this turn and the beginning of next operation cycle. With characters of high-speed, synchronization and real-time, this system platform is designed for the semi-physical simulation system suitable for the on-orbit spacecraft ground test. The principle and frame have reference value for navigation computer upgrading improvement.*

Keywords *–ARM, FPGA, FIFO, HPI, Multiprocessor*

I. INTRODUCTION

Nowadays, data acquisition systems and control systems for spacecraft are characterized by complicated task, real-time computing and periodic task. The traditional single DSP system whose computing ability is limited doesn't meet a higher level of real-time requirement. Therefore, a cooperative parallel signal processing system is designed, composed of high-speed floating-point multiDSPs completing their tasks under the unified scheduling. Obviously, it greatly improves processing speed. This system can be achieved by using ARM +FPGA +DSP digital hardware system. ARM is responsible for task scheduling and interface expansion, multiDSPs complete high-speed floating-point computing, and shared memory and logical design are realized in FPGA [1].

The on-orbit spacecraft constantly processes input data from various sensors, such as IMU, star sensor, earth sensor and GPS, and outputs signals to control various engines, such as flywheel engine, pulse-jet engine, and propulsion engine so as to realize spacecraft attitude and orbit control, which is a cycle process. Therefore, the complicated ground simulation system needs a high-performance embedded platform as a link of semi-physical simulation loop. The paper presents a general parallel multi-DSP embedded frame cooperating with ARM and FPGA, which accomplishes cycle synchronization by external interrupt.

978-1-4244-3864-8/09/$25.00 ©2009 IEEE

II. SYSTEM ARCHITECTURE DESIGN

Fig. 1 shows the architecture of the system. It is composed of 1 ARM, 1 FPGA and 3 DSPs. This system chose NXP company's ARM LPC2468 as the host processor, whose clock frequency is 72MHz, which has abundant on-chip and external resources[2]. The system employed TI company's TMS320C6713 floating-point DSP as the signal processing unit, whose working frequency is up to 300 MHz and processing speed is 1800 MFLOPS [3]. The design adopted Xilinx company's FPGA Spartan-3AN XC3S400AN [4].

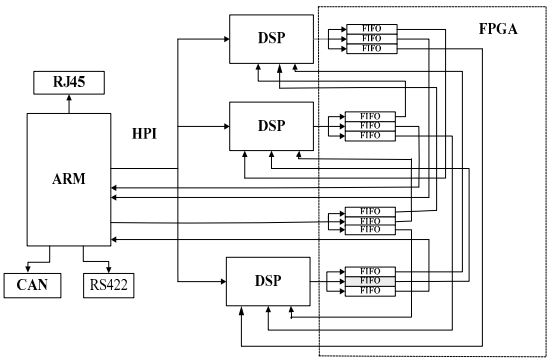

Fig.1. Whole structure of multi-processor system

ARM achieves functions of controlling DSP system, which is suitable for task scheduling and data exchange. ARM is connected to DSP through the HPI of DSP; it downloads program into DSP, and then boots DSP. The multi-processor
connection is realized through connecting every processor to FPGA when the FIFO is designed with FPGA. All processors use external interrupt to synchronize access to shared data in FIFO.

III. THE IMPLEMENTATION OF HPI BETWEEN ARM AND DSP

A.Connecting the C6713B to ARM

Host Port Interface (HPI) of TMS320C6713 is a 16-bit parallel port through which a host processor can

directly access memory space of DSP. The host also has direct access to memory-mapped peripherals. Here ARM is host processor. The HPI16 of C6713 connects ARM with 16-bit data lines, HD0~HD15, and 10 control lines. HPI has three registers: HPI control register (HPIC), HPI data register (HPID) and HPI address register (HPIA). By selecting access to HPIC, ARM can read and write the entire internal RAM [5], [6]. The hardware connection between ARM and DSP through the HPI is shown in Fig. 2.

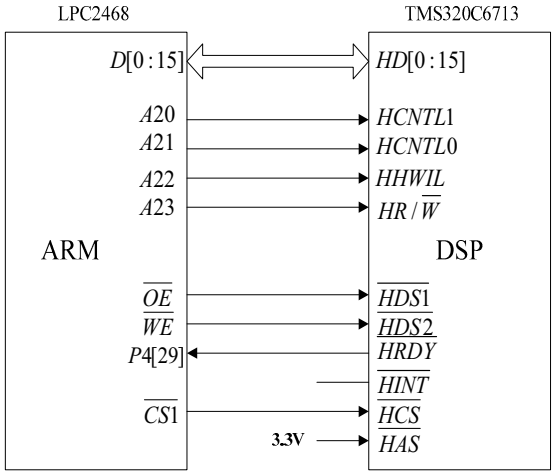

Fig. 2. Hardware connection between LPC2468 and TMS320C6713

Two address lines A20 and A21 of LPC2468 are connected to HCNTL1 and HCNTL0 on C6713, which are used to select a HPI register. While the data interchanged with ARM is 32-bit wide, HPI interface is a 16-bit-wide parallel port. So, each data access needs two fetches, HPI will automatically combine two successive 16-bit data into 32-bit data. HHWIL signal identifies the data on HD [15:0] is first or second half word, which is connected to ARM address pin A22; LPC 2468 does not have $HR/\overline{W}$ signal, its A23 is used instead to connect with DSP. $\overline{HDS1}$ and $\overline{HDS2}$ are data strobe pins, all address and control lines will be latched at its falling edge; reading and writing pins, $\overline{OE}$ and $\overline{WR}$, are connected to $\overline{HDS1}$ and $\overline{HDS2}$ separately to accomplish the function. $\overline{HRDY}$ is host ready pin, whose low level indicates that HPI is ready to achieve a data transmission; high level indicates that HPI is busy completing with a previous HPID writing or reading.

Reading and writing should be done during $\overline{HRDY}$'s validate period. $\overline{HRDY}$ is connected with LPC2468's P4[29], so that ARM can detect if $\overline{HRDY}$ is low. Chip selection signal $\overline{HCS}$ is connected to $\overline{CS1}$, the chip selection pin of memory range 0x81000000-0x81FFFFFF in LPC2468. Address strobe pin $\overline{HAS}$ is not used in the design, tied high as shown in Fig.2. In this design, only two of the five BOOTMODE bits are required because C6713 only has one memory map, which places internal memory at address 0.The HD [4:3] pins which maps to the BOOTMODE [4:3] pins of the C6713 should be connected to external pulldown resistors.

B.Program design of HPI boot

The system has three DSPs. As their connection and function are similar, the paper takes DSP1 for example. Other DSPs can be explained in a similar way [7]. Program of HPI boot is composed of three parts.

1) Building a C Array of values from a COFF file

The first step is to use Hex6x.exe to convert the COFF file *.out which has been compiled in Code Composer Studio 3.3 into two hex files: *.a00 for the code sections and *.a10 for the initialized data sections.

hex6x *.cmd

Then hex2aray.exe should be invoked to build code.h from *.a00 and init.h from *.a01.

hex2aray -i *a00 -o code.h

hex2aray -i *.a10 -o init.h

Finally, two header files should be included in ARM's main program.

2) Initialization of ARM

ARM processor first needs to complete a series of initialization such as setting working mode, GPIO, EMC, only after that it can read or write the HPI correctly. Parts of C code are as follows:

```
Void TargetInit (void)
{
/*initialize target including EMC, GPIO, etc.*/
 EMC_CTRL = 0x00000001;
 // EMC enable; disable address mirror
 PCONP |= 0x00000800;
 //turn on EMC PCLK
 PINSEL4 = 0x5000000A;
 // [14:15] function: /CS2, /CS3
 PINSEL6 = 0x55555555;
 //P3[0:15] function: D[0:15]
 PINSEL8 = 0x55555555;
 //P4[0:15] function: A [0:15]
 PINSEL9 = 0x50055555;
 // P4[24:25], P4[30:31] function: /OE, /WE, /CS0, /CS1
 EMC_STA_CFG0 = 0x081;          //disable page mode
 EMC_STA_WAITWEN0 = 0x0; //delay 1cclk
 EMC_STA_WAITWR0 = 0x06;  //delay 33cclks
 EMC_STA_WAITOEN0 = 0x0;  //delay 1cclk
 EMC_STA_WAITRD0 = 0x1f;  //delay 32cclks
 EMC_STA_WAITPAGE0 = 0x0; //delay 1cclk
 EMC_STA_WAITTURN0 = 0x0; //bus idle 1cclk
 ……
}
```

3) Program of the HPI

Firstly, define the C6713 HPI registers as macros. In this design, $\overline{HCS}$ of DSP1 is connected to $\overline{CS1}$ of ARM, which selects memory address range: 0x81000000-0x81FFFFFF. Codes are follows:

```
#define HPIC_1st (*(volatile WORD *) 0x81000000)
#define HPIC_2nd (*(volatile WORD *) 0x81400000)
#define HPIA_1st (*(volatile WORD *) 0x81200000)
#define HPIA_2nd (*(volatile WORD *) 0x81600000
#define HPID_1st (*(volatile WORD *) 0x81100000)
#define HPID_2nd (*(volatile WORD *) 0x81500000)
```

Key part of the HPI boot program is as follows:

```
……
TargetInit (); //initialize ARM
// initialize HPIC register
HPIC_1st = 0x0001;
HPIC_2nd = 0x0001;
//calculate the length of code array
length_code = sizeof(code) / sizeof(code[0]);
// initialize HPIA register
HPIA_1st = 0x0000;
HPIA_2nd = 0x0001;
Write_Data(length_code);
// date from init array is written to DSP
length_init = sizeof(init) / sizeof(init[0]);
HPIA_1st = 0x0000;
HPIA_2nd = 0x0001;
Write_Init(length_init);
// detect P4[29] pin, when low, DSPINT is written to 1.
While ((FIO4PIN3 & (1 << 5)));
HPIC_1st = 0x0002;
HPIC_2nd = 0x0002;
```

IV. THE IMPLEMENTATION OF DATA SHARING AND SYNCHRONIZATION

A.Synchronous data sharing based on the FIFO

Shared memory, which can be used for data sharing, between processors, in particular, for coordination and synchronization, provides convenient communication between processors in a multiprocessor system.

This design uses FIFO group as the shared memory module. FIFO can hold shared data, allowing other processor to read in order through the external interrupt. FIFO is used in one direction, that is, the processor writing to FIFO expects some other processors to read from the FIFO, or processor reading from FIFO expects some other processors to write to the corresponding FIFO in advance. As shown in Fig. 1, each processor is connected to three FIFOs, whose inputs are connected together and outputs are connected to other three processors separately, which can avoid timing conflict on reading and guarantee the reliability of the system. Any processor can read shared data from the other processor. FIFOs are divided into four groups, and each group has 3 isomorphic FIFOs[8].

Communication between processors is conducted through external interrupt. All interrupt signals go through the FPGA, which greatly simplifies the hardware connection. Fig.3 illustrates all interrupt connections.

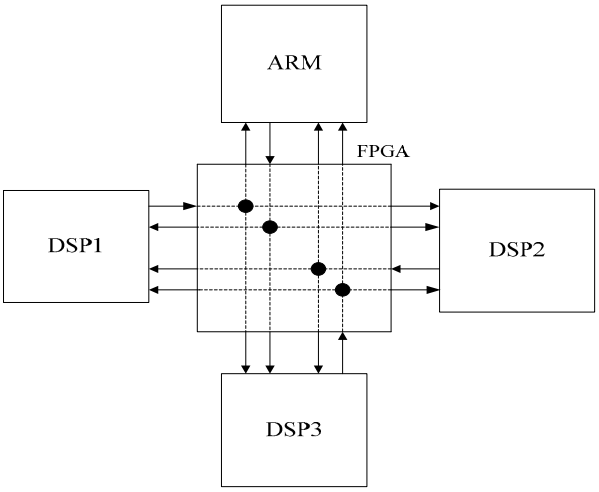

Fig.3. interrupt connections

Each processor can send interrupt to other 3 processors, and each processor also can receive interrupts from other three processors. For example, the ARM receiving data from various sensors dispatches the task, writes them into a FIFO group and sends an interrupt to three DSPs. Three DSPs response to the interrupt and read data from the FIFO and run the corresponding algorithm programs. If the data is expected by the second DSP but whose actual contents are determined by the first DSP, the second DSP also responses to the interrupt from the first DSP, reading the shared data which is written into FIFO by the first DSP. After processing, three DSPs send interrupts to ARM. ARM read results from FIFO and transmits them to various controllers via interfaces of ARM. After ARM collecting three DSP interrupts signals, it means the end of this turn and the beginning of next cycle. Hence, the system can achieve cycle synchronization by external interrupt.

B.FIFO implementation using FPGA

The design adopts Xilinx FIFO Generator from ISE 9.2 to create a FIFO buffer that makes use of the block RAM for memory in XC3S400AN[8]. Embedding many reusable modules is the advantage of FPGA device. If concrete FIFO chip is chosen, the complexity of the PCB layout could be increased greatly.

FPGA logic design using verilog HDL is composed of three parts: FIFO read-write logic design，FIFO generation by Xilinx FIFO core and interrupt hardwired logic design. The critical issue in the design of FPGA is to complete the interface logic design between processor and FIFO, which converts address

and read/ write signals of processors for FPGA into control signals for FIFO. In order to make efficient use of the RAM cells, parts of configuration are as follows: Input /output data width is 16 bits; depth is 128; read / write clock frequency is 100MHz[9]. Four groups of FIFOs are configured in asynchronous mode. Interrupt connections are realized by hardwired logic in FPGA.

Top-down design method is used. TOP_FIFO is the top module. In the top-level module, different functional submodules are instantiated. According to who performs writing, these FIFO submodules can be divided into the following 3 types, A_FIFO_D, D_FIFO_A and D_FIFO_D. A_FIFO_D module indicates that ARM writes the data into a FIFO and DSP read them from the FIFO. The system has 4 groups of FIFOs, and each group is assigned an unique address, which is used to distinguish which FIFO the processor should access . Some modules can reuse, because their input/output ports and function are the same. Fig. 4 shows the relationship between top-level module and submodules.

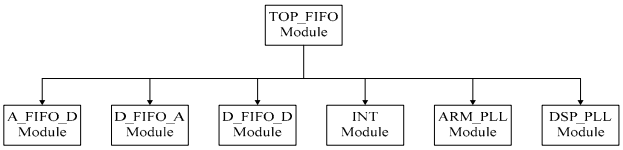

Fig.4. Relationship between top-level module and submodules

Read and write timing is another important issue .Take A_FIFO_D module as an example. D_FIFO_A module and D_FIFO_D module can be explained in a similar way. Parts of verilog code in A_FIFO_D module are as follows:

```
module A_FIFO_D (fifoclk, nWE1, nCS1, AB1, DBin,
ECLKOUT2, nOE2, nCE2, nRE2, AB2, DBout);
……..
assign
WrPulse=(AB1[2:0]=3b`000)&&(!nWE1)&&(!nCS1);
……..
always@(negedge fifoclk)
        if(!WrPulse1) state1<=3'b001;
        else
        case(state1)
                3'b001:  state1<=3'b010;
                3'b010:  state1<=3'b011;
                3'b011:  state1<=3'b100;
                3'b100:  state1<=3'b101;
                3'b101:  state1<=3'b110;
                3'b110:  state1<=3'b000;
                default: state1<=3'b000;
      endcase
 always@(negedge fifoclk)
        if(!WrPulse1)    WR_EN<=0;
        else
        case(state1)
                3'b001:  WR_EN<=0;
                3'b010:  WR_EN<=1;
                3'b011:  WR_EN<=0;
                3'b100:  WR_EN<=0;
                3'b101:  WR_EN<=0;
                3'b110:  WR_EN<=0;
                  default:  WR_EN<=0;
          endcase
        ……..
endmodule
```

Suppose if FIFO address of ARM is "3`b000". When nCS1 and nWE1 of ARM are low, as well as address is "3`b000", WrPulse becomes high. State machine is used to generate write timing of ARM, namely, generate a WR_EN pulse which directly inputs to WR_EN pin of FIFO; Data lines of ARM are connected to DBin of FIFO, and then ARM writes data into a FIFO.

When nCE2、nOE2and nRE2 of DSP are low, as well as address is "3`b000", State machine is used to generate read timing of DSP, namely, generate a RD_EN pulse which directly inputs to RD_EN pin of FIFO; Data lines of DSP are directly connected to DBout of FIFO, and then DSP read data from the FIFO.

V. CONCLUSION

The multiprocessor system of previous version, whose main functions has been tested successfully, is composed of 3 ARMs, 1 FPGA and 2 DSPs. Fig.5 shows its pictorial diagram. Test results including RS422、CAN、RJ45、HPI and external interrupt illuminate that this system can acquire various data via interfaces of ARM and realize the boot of DSP via HPI, data sharing between processors, and, in particular, coordination and synchronization with external interrupt.

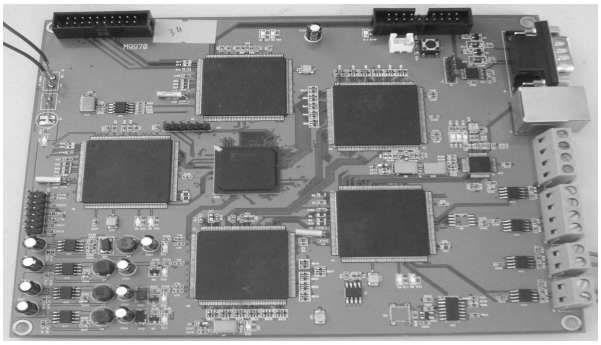

Fig. 5. previous multiprocessor system

The multiprocessor platform of revised version consists of 5 key chips, 1 ARM, 1 FPGA and 3 DSPs, more suitable for high-speed and real-time data processing. As a universal signal processing platform, it can process real-time data faster. Utilizing such a parallel hardware structure of high efficiency, it has created such a good condition for software algorithm that a higher performance can be acquired. In a word, the system can reach a level of engineering application.

ACKNOWLEDGMENT

The authors wish to thank all colleagues who previously provided technical supports.

REFERENCES

[1] PAN Fangsheng, ZHAO Feng, XI Jun, LUO Yi, "Implementation of Parallel Signal Processing System Based on FPGA and Multi-DSP", *Computer Engineering*, Vol. 32, No. 23, pp. 247-249, Dec. 2006.

[2] NXP Semiconductors, LPC2468 User manual, July. 2007.

[3] Texas Instruments, TMS320C6713 Floating-Point Digital Signal Processors, June. 2006.

[4] Xilinx Inc., Spartan-3AN FPGA Family Data Sheet, June. 2008.

[5] Texas Instruments, TMS320C6000 DSP Host Port Interface (HPI) Reference Guide, Jan. 2006.

[6] Jun Wu, Zhi-Tao Xiao, "Design of Communication Interface in a Video Vehicular Detection System Based on ARM and DSP", *ICSP 2008. 9th International Conference,* pp. 431- 434, Oct. 2008.

[7] Texas Instruments, Implementing the TMS320C6201/C6701/C6211 HPI Boot Process, Jan. 1999.

[8] YAN Lu-xin, ZHANG Tian-xu, ZOU sheng, ZHONG sheng, "Parallel system architecture of multi-DSP interconnected by FPGA", *Systems Engineering and Electronics,* Vol. 27, No. 10, pp. 1757-1759, Oct. 2005.

[9] Xilinx Inc., Spartan-3AN FPGA Family Data Sheet, June. 2008.

[10] Xilinx Inc., LogiCORE™ FIFO Generator v3.3 User Guide, April. 2007.