Arinc429 Bus-based Multi-Dsp, Fpga Component Program Serial Load

Yongchao Zeng
Department of Instrument Science and
Optical-Electronics Engineering
Beihang University
Beijing, China
zengyongchao@aspe.buaa.edu.cn

Jianhui Zhao
Department of Instrument Science and
Optical-Electronics Engineering
Beihang University
Beijing, China

Fan Li
Department of Instrument Science and
Optical-Electronics Engineering
Beihang University
Beijing, China

***Abstract*—The load of DSP and FPGA program are usually accomplished by the way of JTAG. But in some occasions, the circuit of DSP and FPGA are sealed up in the equipment; it is not convenient to update the program by dismounting device. For example, certain aviation equipment left only one interface of ARINC429 to achieve the program load and verification of internal chips. Based on this application, this paper proposed a way of loading and verifying the serial program of multiple DSP and FPGA. Under this way, the load and verification of multiple internal DSP chip programs, external connection FLASH program and FPGA configuration program can be accomplished in the complete machine state. Currently, this way has been applied in the program load and verification of an aviation equipment, and the result is effective and reliable.**

Keywords- serial load ;DSP; FPGA; ARINC429

I. INTRODUCTION

In a certain aviation equipment, two DSP (TMS320F2812 and TMS320C6414) and one FPGA were used as the core processing part following the request of processing system. The equipment used airtight structure, and the updating of program could only be accomplished by a reserved ARINC429 bus interface. In the process of whole updating of equipment program, only TMS320F2812 could have communications with upper computer by ARINC429bus, the updating of other chips and FLASH program needed to be accomplished by transfer of TMS320F2812. The structure of system hardware is shown in Figure 1, the program load and read-back check of whole system are divided into two parts, the TMS320F2812 (TMS320F2812 internal chip FLASH program, external connection FLASH program) and TMS320C6414 (TMS320C6414 internal chip FLASH program, external connection FLASH program and FPGA configuration chip program).There are two important communication interfaces: ARINC429 communication bus, MCbsp (Multichannel Buffered Serial Port) interface. ARINC429 bus communication is used between upper computer and TMS320F2812; TMS320F2812 used MCbsp interface to communicate with the MCbsp1 interface of TMS320C6414.

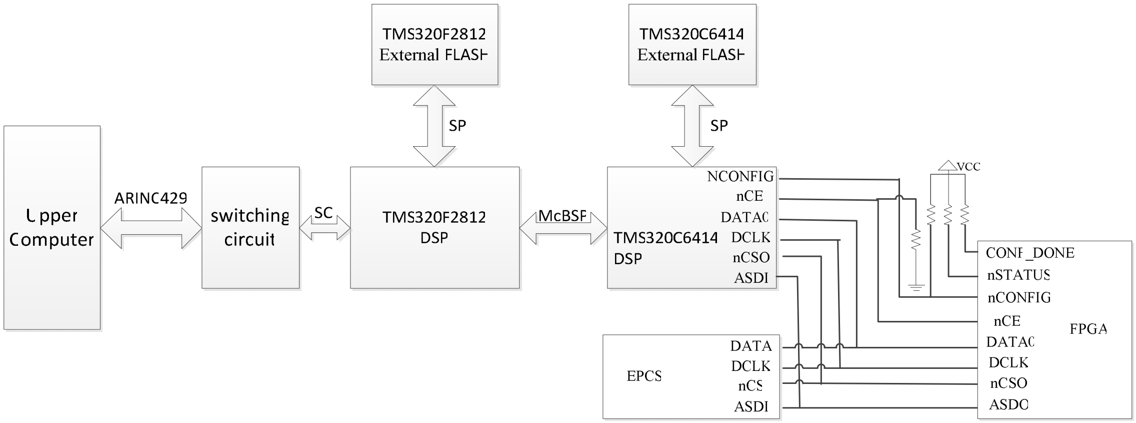


Figure 1 the structure of system hardware

II. THE ACCOMPLISHMENT OF BUS COMMUNICATION PROTOCOL

To accomplish the serial load of system program by ARINC429 bus, the reliability of bus transmission must be guaranteed. No mistake can be allowed in the process of data sending, transmission and receiving. Otherwise, wrong program will be loaded in the system, leading unpredictable mistakes in the running of system. Also, the loaded programs are usually large, so the transmission of whole programs cannot be accomplish once. It needs to divide the programs into numerous data packets to transmit. The form of data packets includes 36-byte packet-head and unloaded program data (the length is the number of 22~25th bytes of the packet-head) and 2-byte end sign. The detailed form of packet-head is shown in the Table 1, the sign of packet-head is 0xAA08, and the end sign is 0x0000.

TABLE 1 HEAD FORMAT OF DATA PACKET

byte	0~1	2~9	10~17	18~21	22~25	26~29	30~31	32~35
meaning	Head sign	Command type	The radix-minus-one Complement of Command type	the entrance address of TMS320F2812	Loaded data length	The destination address of load	Loaded data length	The data buffer address transmitted to the TMS320F2812

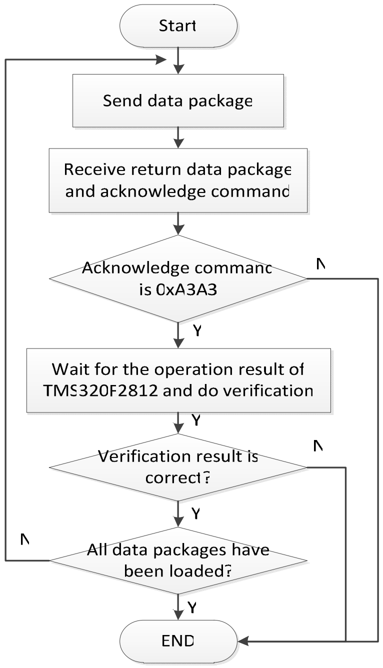
The system programs include TMS320F2812 on-chip FLASH programs, TMS320F2812 external FLASH programs, TMS320C6414 on-chip FLASH programs, TMS320C6414 external FLASH programs, FPGA data code, etc. And each part is separated into program load and program read-back. Load or read-back targets are distinguished by different command types in the data packet.

A. ARINC429 communication

Only when the system is powered on and at loading work status (enabled status), the communication between upper computer and TMS320F2812 could be carried out. Using the ARINC429 bus interface, the upper computer could send a 0x41 to TMS320F2812, then the TMS320F2012 should send back a 0x41 as the correct handshaking signal.

When loading the programs, according to the size of 16KB and the loading format, we should firstly separate the data file into several data packets. The relevant process of loading data packets is finger 2:

Firstly, the upper computer would send the loading data packets to TMS320F2812.After receiving the data packets, the TMS320F2812 would send back signals to the upper computer. At the same time, TMS320F2812 should check out the data packets and send back the result to the upper computer. If the result is right, the loading process would be carried out. If the loading process is relevant to TMS320F2012, the relevant process would be carried out. If the loading process is relevant to the TMS320C6414, the data packets would be sent to the TMS320C6414 through the MCbsp interface. The TMS320C6414 would carry out the relevant process. After the process is finished, the TMS320F2012/TMS320C6414 will send the results of the process. If the result of checking out the data packet is not right, the TMS320F2012/TMS320C6414 will wait for the next data packet from the upper computer. After receiving the data packet and response command sent back from TMS320F2012, the upper computer would check out the response command. If the result is right (the right result is 0xA3A3),it will wait for the result of the process carried out by the TMS320F2812 and check out the result. Otherwise, it will stop waiting. If the result is right (0xA7A7 for TMS320F2812 and 0xABAB for TMS320C6414), it means the loading process is successful. Next data packet will be loaded. Otherwise, the loading of this item will be stopped.


Start
Send data package
Receive return data package and acknowledge command
Acknowledge command is 0xA3A3
N
Y
Wait for the operation result of TMS320F2812 and do verification
Y
Verification result is correct?
N
Y
N
All data packages have been loaded?
Y
END

Finger 2 Program loading flow chart

B.MCbsp communication

The MCbsp interface of TMS320F2812 and the MCbsp1 interface of TMS320C6414 transmit data to each other reciprocally. The format of the data is single phase, each frame is a data unit, a data unit includes 32 bytes. When a jump from high level to low level on the GES0/GES1 pin of the hardware GPIO interface on TMS320F2012 is detected by the supervisory computer, the fever write begins. Then the TMS320F2812 begins to transmit data packet with relevant commands to MCbsp interface. After receiving a data packet, TMS320C6414 will adopt different load/read-back operations to different chips according to the command types in the data packet.

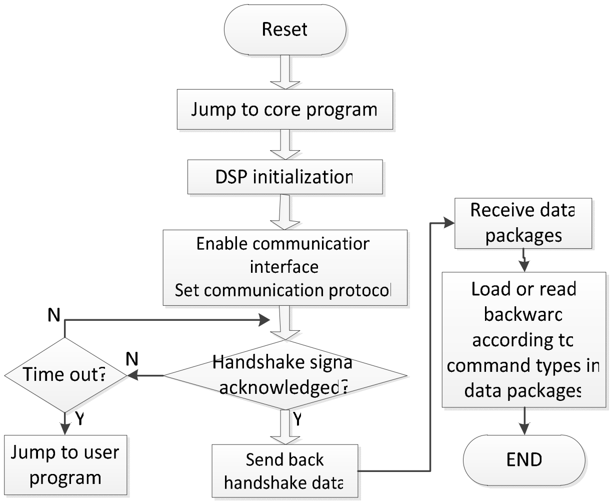
III. DSP PROGRAM DESIGN

TMS320F2812 is the core part of the system when loading programs. It is in charge of the communication with the upper computer, the communication with the TMS320C6414 and the load/read-back of some FLASH programs. Before the serial loading of the system programs, curing some core code on the TMS320F2012 chip through JTAG is needed. This part of code could carry out the process of the program data which is sent by the supervisory computer through the bus, and it will carry out the relevant load/read-back process according to the type of the command. At the same time, when the serial loading is not processed, this part of code is in charge of jumping to the user programs.

Due to the kernel code will not be changed in the process, the following three requirements are needed to be achieved: firstly, the kernel code cannot be rewritten in the process of serial load; secondly, the program pointer can jump to the code segment and then return to the user code segment when DSP is resetting; finally, the curing kernel code will not be broken in case of power cut by accident in the process of loading.

Therefore, according to the property that the flash in TMS320F2812 slice is allowed to write by segments based on user's requirements, the serial-loading code can be loaded to the last two blocks(block A and block B, addressing space distributes from 0x3F4000 to 0x3F7FFF)which never to be rewritten.

The program pointer will jump to the kernel code segment as long as DSP resets, its function is to enable the ports of SCI and Mcbsp of DSP, set communication protocol, judge whether serial received handshaking signal, jump to user's code segment automatically when the handshaking signal is out of emerging for a certain time. DSP will receive the data packet sent by supervisory computer if handshaking signal has been received successfully, and then perform an appropriate action on the basis of the instructions of the received data packet. Core program flow chart of TMS320F2812 shown in Figure 3:


Reset
Jump to core program
DSP initialization
Enable communication interface
Set communication protocol
N
Time out?
N
Handshake signal acknowledged?
Y
Y
Jump to user program
Send back handshake data
Receive data packages
Load or read backward according to command types in data packages
END

Finger 3 TMS320F2812 Core program flow chart

IV. PC PROGRAM DESIGN

To ensure the code is loaded correctly in the loading processing, it is necessary for verification. The data packet is verified by the embedded program of DSP by the way of send-back (After the packet has been received, DSP will send the packet back to the upper computer which then compares received packet with sent packet and stops loading if not match). Because of COFF file (.out file) is generated after DSP project file was compiled and assembled, it is necessary to split it into the hex file to meet the requirements of serial loading. The usual practice is to convent .out file into hex file by using TI own tools. Because the program file is sent by section by the way of packet, so we use our own program to conversion, divide COFF file and write the packet-head and end into divided file for convenience. When transmitting to TMS320F2812, just read every file orderly and sent. Compare the DSP send-back file with the corresponding file when verifying. CCS has written the information of every section into the .out file when it was generating.

The whole structure of COFF file is:

File Header ；*Header*

Optional Header ；*Optional header*

Section Header 1 ；*Section header*

......

Section Header n

Section Data ；*Section data*

Relocation Directives ； *Relocation table*

Line Numbers ；*Line number table*

Symbol Table ；*Symbol table*

String Table ；*String table*

The first three structs are defined as：

```
struct head_section{
        UInt16 id;
        UInt16 amount;
        int data_time;
        int file_ptr;
        int symbol_amount;
        UInt16 option_bytes;
        UInt16 flag;
        UInt16 dest_id;
    }
struct Option{
        short flag;
        short verision;
        int code_size;
        int data_size;
        int uninit_size;
        int entry_point;
        int code_addr;
        int data_addr;
    }
struct Section{
        char[] symbol;
        int phsical_addr;
        int virtual_addr;
        int word_size;
        int orignal_ptr;
        int relocation_ptr;
        int line_entry_ptr;
        uint relocation_amount;
        uint line_amount;
        uint flag;
        UInt16 reserved;
        UInt16 page_number;
    }
```

We mainly focused on the file header, option header and section header. Firstly, read the number of the section (“amount” in struct *head_section*) and the size of the Option section(“option_bytes” in stuct *head_section*) from the file header, and then obtain the entry address of the procedure (“entry_point” in struct *Option*) from the option header. Finally, get the important data of each section header such as the physical address (“physical_addr” in struct *Section*), virtual address (“virtual_addr” in struct *Section*) and the section size (“word_size” in struct *Section*). Based on these data to read all the data files and split it into multiple files to save. At the same time the beginning and ending of each file was written into the corresponding packet-head and packet-end according to the format of Table 1.

V. CONLUSIONS

This paper introduced a method of serial load and verification program for multiple DSP and FPGA through the ARINC429. The method had been applied to certain aviation equipment, and the result shows that the load and verification method was effective and reliable. It has a certain reference value to the program load of others multi-DSP system.

REFERENCES

[1] TMS320C6414 Hardware Designer’s Resource Guide[J]. Texas Instruments.2005

[2] Cui Weicheng.Serial loading and verification technology of TMS320F240DSP on-chip Flash based on SCI[J]. ELECTRONIC MEASUREMENT TECHNOLOGY.2007(3):71-73.

[3] Ren Linxiang, Ma Shufen, Li fanghui, Principle and Application of the TMS320C60 DSP series, Beijing: Electronic Industry Press,2000.