A Dynamic Reconfiguration Scheme for Embedded System Based on Multi-core DSP

Xiaojian Liu*, Shuo Liu, Hao Lv
Computing Technique Research Institute, Aviation Industry Corporation Xi’an, PR, China

*Corresponding author: lxjiansky@163.com

Abstract. High-performance embedded system is needed in high-speed and accurate avionics control systems, but its hardware resources are limited. It is of great significance to reuse hardware resource through static or dynamic reconfiguration. A dynamic local reconfiguration scheme is designed based on c6678 isomorphic multi-core DSP to realize the dynamic reconfiguration of software on a single core under the premise of ensuring the normal operation of the whole system. This reconfiguration scheme can realize the complete switching of software on the whole core and realize the time division multiplexing of multiple algorithms in the same memory space. Experimental results show that the scheme can fully meet the requirements of real-time.

1. Introduction
With the increasing precision requirements of avionics control system, the performance requirements of embedded processor are also higher and higher. The development of embedded processor from single core to multi-core has become the major trend. The improvement of the overall performance of the system not only needs the improvement of hardware performance, but also needs the cooperation of software to make full use of hardware resources. The most important thing in the development of multi-core platform is how to maximize the performance of each core under the premise of multitasking [1].

Processor hardware resources are limited, but in some application scenarios a processor is required to complete the processing of multiple algorithms. The idea of reconfiguration first appeared in the field of FPGA. Its principle is to divide the system into static area and dynamic area. The static area remains unchanged during the run time of the program. As the circuit to maintain the basic work of the system, the dynamic area reconfigures the circuit to update the function of the system when the external reconfiguration is needed [2]. Transplanting the idea of reconfiguration to multi-core processor can not only provide multiple functional features for the system with the lowest consumption to improve the resource reuse rate of the hardware system, but also realize time-division multiplexing of various algorithms on one core of the multi-core processor. The difference between multi-core processor reconfiguration and FPGA reconfiguration is that the former object is the software running on each core, while the latter is aimed at hardware logic [3].

For the application of reconfigurable technology in multi-core processor, there are related researchs. The basic idea is to design a reconfigurable software framework, as a static part of the software on each core to manage the basic operation of the software on the core and realize the loading of other applications. The specific functional modules or algorithms are made into dynamic link library, which
is moved from outside to memory by the core when it needs to be run by the processor [4]. This method has great limitations. The static part of the software (as the management part of the software) cannot be changed, so only the local part of the software can be reconfigured. If it is necessary to change the operation logic of the software on the whole core, because a set of software cannot reconfigure itself, it cannot realize the overall software reconfiguration. In this paper, a new reconfiguration scheme is designed based on homogeneous multi-core C6678 DSP. With the combination of hardware and software, not only can the algorithm on the processor be changed dynamically, but also the working logic of the whole core can be modified. One core of C6678 is used as the management core of reconfigurable system, which does not participate in the reconfiguration, but controls the overall operation of the system and the reconfigurable logic of other cores. When the softwares of the remaining 7 cores need to be reconfigured, the management core can control the software replacement on the corresponding core. The software takes up the same address space before and after replacement, which will not waste extra memory space, and the whole process will not affect the operation of other cores. For a single core, this scheme is not constrained by the existing software framework, nor is it a minor repair to the original software, but a complete replacement. For example, UCOS system can be replaced by sys / BIOS system.

2. System design based on reconfigurable technology

A new high-performance multi-core DSP C6678 based on keystone architecture of Texas Instruments (TI) is used as the reconfiguration processor. The DSP has eight identical cores, each with 32KB L1D data memory, 32KB L1P program memory and 512KB L2 memory [5]. Each core can run independently and has communication mechanisms such as hardware communication interruption IPC and shared memory. In practical application, the reconfiguration system often needs the cooperation of multiple processors. In this scheme, the sparten-6 series FPGA of Xilinx is used as the host computer of DSP, which is responsible for the system management. The block diagram of the system is shown in Fig.1.
As the host computer, FPGA maintains the operation of the whole system, controls the DSP’s power-on sequence, clock and reset, and sends reconfiguring instructions to DSP. DSP is mainly responsible for the operation of the algorithm and the reconfigurable function of the system. FPGA and DSP transmit data and send or receive instructions through EMIF16 bus. FPGA interacts with PC through CAN bus to facilitate developers to debug. FPGA can control the power-on reset of DSP through POR pin, and reset DSP globally through RST pin. The former is used for DSP power-on self-starting, and the latter is used for global reset or restarting of DSP. There are four CORESEL pins and one LRESEL pin on the chip 6678, which are connected with FPGA to control the local reset of any DSP core. FPGA provides external interrupt and control information to DSP through 16 GPIO pins. In addition, FPGA can also detect the running state of DSP through corresponding pins.

3. Dynamic reconfiguration scheme of DSP

3.1. Multi-core Startup Mechanism of DSP

6678 DSP supports a variety of startup modes, which can be divided into two modes according to the source of program loading: local memory startup mode and external host online loading mode. The boot mode of local memory needs to solidify the program to the local memory before startup. After DSP is powered on and reset, the program is moved from memory chip to internal RAM by bootloader. When using the external host loading mode, the program does not need to be solidified. After the slave is reset and powered on, the bootloader will initialize the corresponding external interface according to the startup mode, and the external host will load the program to the running address of the slave through the corresponding interface [6]. At present, the most commonly used method is to make the program of multiple cores into a mirror file through the tool chain, and load it into the memory space of each core at one time. In order to realize the local reconfiguration function of multi-core DSP, the multi-core program is made into image file and loaded separately at startup. Core 0 is started by bootloader, while other cores are started by core0 respectively.

No matter what loading method is adopted, after the program loading is completed, the program running entry address _C_int00 should be written in 0x1x87fffc (x is the core number) which is the address of the last word of the L2 memory space of each core. This address on L2 is called the Boot_Magic address. After the CPU starts, it always runs from the address in Boot_Magic by default. When the program loading is completed, core 0 runs first, while other cores are in sleep state. Core 0 needs to wake up other cores through inter core interrupt (IPC) [7]. The multi-core startup process is shown in Fig.2.
3.2. Implementation Mechanism of Dynamic Reconfiguration

6678 DSP adopts the homogeneous multi-core architecture. Eight identical cores on the chip can perform one task in parallel or perform eight different tasks respectively. With the idea of centralized management and decentralized work, the core 0 of DSP is taken as the key of system’s reconfigurable task management, which is responsible for managing the operation logic and core reconfiguration of the remaining seven cores’ programs. The other seven cores that do not participate in the management perform their own algorithms. When the software of a certain core needs to be reconfigured, core 0 is responsible for receiving a new program from the external host or reading from the local memory to replace the current program in the memory. After the replacement, the core executes a new program to realize the dynamic switching from one function to another. The whole process only works on the reconfiguration core and does not affect the operation of other cores. The reconfiguration idea of this scheme is similar to the local dynamic reconfiguration of FPGA.

After the multi-core start-up is completed, the program cannot be rewritten and replaced during the running period. A mechanism is needed to make the core temporarily stop running, replace the original program by other core without affecting other functional modules, and the CPU needs to resume running after software replacement. The local reset function of C6678 can fully meet this requirement. Through local reset, the CPU of one core can be suspended without affecting other modules. After the program reloading, the CPU can be resumed by releasing the reset. In addition, the memory access permission of each core in C6678 can be set. After the core is reset, one core can access the memory of another, which makes it possible to replace one core’s program with another.

When reconfiguring, the program loading is similar to bootloader program, and the program needs to be made into a specific image format through the tool chain. After reading the image file, it needs to parse according to the specified format, and read the first address, size and content of each segment. Then each segment is loaded into the corresponding memory address according to the parsed information. In practical application, it is necessary to protect the scene before reconfiguration in some scenarios and restore the scene after reconfiguration. The flow of dynamic reconfiguration is shown in Fig.3.
3.3. Local Reset of DSP
The configuration scheme in this paper mainly uses the local reset function of DSP. C6678 DSP has four reset modes: hardware reset, software reset, power-on reset and local reset. The first three kinds of reset will cause a global reset and trigger the restart of BootLoader. However, local reset only affects the operation of single core and does not trigger BootLoader [8].

There are two trigger modes for local reset: hardware trigger and software trigger. The way of C6678 software reset is to shut down the core power domain by power sleep controller (PSC). The method of hardware local reset is adopted in this paper. As shown in the system block diagram of Fig. 1, the pins related to local reconfiguration are 4 CORESEL pins and 1 LRESET pin, which are controlled by FPGA. FPGA can select the core to be reset through four CORESEL pins, set LRESET to low level to reset the core locally, and pull up the pin to realize releasing the reset. After release reset, CPU will run from the address indicated by Boot_Magic by default.

4. Software design of dynamic reconfiguration
Before DSP start-up, it is necessary to solidify the program into the ROM on chip, and the program loading address should all adopt the global address to avoid the address conflict between multiple cores. Before the DSP starts, the FPGA configures the input clock, power-on time sequence and start mode of
DSP. DSP begins to power-on reset, and then multiple cores start in turn and run their own programs. When the system needs to be reconfigured, an external reconfiguration instruction is given to the FPGA through CAN bus. After the reconfiguration instruction is parsed by FPGA, the reconfiguration information is transmitted to DSP through EMIF16 bus, and an external interrupt signal is sent to DSP through GPIO pin to inform DSP to process the reconfiguration task. After DSP receives the reconfiguration request, as the management key of reconfiguration, core 0 parses the reconfiguration instructions, determines the core to be reconfigured, and saves the standby program of the core to memory. Save the field data as needed and then core 0 sends the local reset instruction to FPGA. FPGA resets the corresponding core of DSP according to the instruction, and the CPU of the core stops running. Then core 0 replaces the original program with the standby program, writes the first address _c_init00 of the program to the Boot_Magic address of corresponding core and restores the scene. Then DSP sends the releasing reset instruction to FPGA, and the FPGA will release reset of the core. At this time, the reconfigured core goes into sleep state, and core 0 wakes up the core through IPC interrupt. Then the reconfiguring core starts to run the new program, and the local reconfiguring task is completed. In this process, the tasks of other cores are executed normally, and only the software of the reconfigured core is updated. The flow chart of software reconfiguring is shown in Fig.4.

![Software flow chart of reconfiguration system](image)
5. Verification of reconfiguration scheme

In order to verify the function of system reconfiguration, two sets of running programs are designed for each core of DSP, one is running at startup, the other is used as standby program to replace the original program when reconfiguration is needed. Program A writes 0xAAAA and 0xBBBB alternately in 1K space starting with 0x1X800400 (X represents the core number) in its own L2 storage space. Standby program B writes 0xCCCC and 0xDDDD alternately in 1K space starting from 0x1X800400 (X represents core number) in L2 storage space of each core.

In order to observe the experimental phenomenon, the DSP is directly attached to the emulator, and the CCS platform Memory Browser interface on the PC is used for intuitive observation. The timer is used in the program to measure the time taken by the core to complete the reconfiguration, so as to evaluate it. The figure below shows the phenomena before and after the reconfiguration of the core software. Fig.5 shows the phenomenon before core 1 reconfiguration, in which 0xAAAA and 0xBBBB are alternately written in a space starting from the address of 0x11800400. Fig.6 shows the phenomenon after core 1 is reconfigured, and the alternate writing of 0xCCCC and 0xDDDD in the space starting from the address of 0x11800400. The experiment shows that it takes 4 ms to complete a 128KB software reconfiguration, which achieves the goal of real-time software reconfiguration.

![Fig 5. Phenomenon before core 1 reconfiguration.](image1)

![Fig 6. Phenomenon after core 1 is reconfigured.](image2)
6. Conclusion and prospect
In this paper, the method of hardware reset and software is used to replace the software on a core completely. It makes up for the defect that pure software reconfiguring can't reconfigure the whole software on the core. Experiments show that it takes only 4 ms to complete the reconfiguration of a 128 K program when the image is stored in on-chip RAM. Using this reconfiguration scheme to realize the time division multiplexing of multi-task can meet the requirement of real-time. If the reconfigured image file is stored in ROM memory, the file reading will take certain amount of time, and the time cost of the reconfiguration process will depend on the read-write speed of ROM memory. However, the restriction of memory read-write speed to reconfiguration speed will be overcome with the application of high-speed communication interface. In practical applications, local reconfiguring programs are often loaded through external interfaces. The external host can directly load the programs needed locally into the memory space of the reconfiguration core online through SRIO, PCIe and other interfaces, which will greatly save time and achieve a qualitative leap in reconfiguration efficiency. With the development of the Internet and some high-speed interfaces, remote online reconfiguration through external hosts is bound to become a trend.

References
[1] G.X.Wang. Research and Implementation of Dynamic Reconfiguration Framework for Embedded Multi-core DSP[D]. Xi'an: Xi'an Electronic and Engineering University, 2015:pp.2-7.
[2] S.W.Bai. Design of Multi-core DSP Local Reconfiguration System [D]. Tianjin: Tianjin Polytechnic University, 2017:pp.35-36.
[3] Z.L.Lu,Y.F.Zhao,L.D.Lan. Research on DSP Task Dynamic Loading Method Based on Reconfigurable Technology[J]. Application of Electronic Technique, 2015,41(10):pp.18-21.
[4] Y.M.Sun,R.Yao,Q.X.Guo. Design of Self-reconfigurable System Based on Dynamic Local Reconfiguration[J]. Electronic Measurement Technology, 2015,38(7):pp.28-33.
[5] Texas Instruments. TMS320C6678 Multicore Fixed and Floating-point Digital Signal Processor User Guide[Z]. USA: Texas ,2012.
[6] T.H.Chen,J.Xiao,W.Feng. Application Development and Practice of Embedded Multi-core DSP [M]. Beijing: Beijing University of Aeronautics and Astronautics Press,2017:pp.253-258.
[7] J.H.Niu. Introduction and Example Refinement of TMS320C66x Keystone Architecture Multi-core DSP[M]. Shanghai: Shanghai Jiaotong University Press,2014:pp.275-280.
[8] Texas Instruments. TMS320C6678 Boot Loader for the C66x DSP User Guide[Z]. USA: Texas, 2012.