Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 3906 – 3910

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineeringand Information Science

Dynamic Management of Hardware Multi-threading for Network Processors

Yue Qi, XinQiang Luo, Qin Wang a*

University of Science and Technology Beijing, Beijing, 100083, China

Abstract

In this paper, we presented a dynamic management mechanism of hardware multi-threading for pipelined hardware multi-threading architecture. And a set of special instructions are provided. In view of the workload and traffic of network are uncertainty, the dynamic multi-threading architecture in this paper can adaptively adjust processor performance according to the workload, and achieve the effective power savings.

Keywords: Network processor; dynamic multi-threading; hardware multi-threading

1. Introduction

In research of micro processor, in order to reduce the cost of data exchange in thread switching to further improve the utilization of processor, hardware multi-threading processor architecture is a good choice[2]. Compared to the thread scheduling by software programming, hardware multi-threading have better performance such as faster activation time[3][4], switching time can be predicted[5] and so on.

Now researching on hardware multi-threading are more focus on performance, such as thread switching frequency, pipeline depth or pipeline utilization. Ref. [1] proposed a hardware multi-threading architecture of network processor. Which realize multi-thread parallelism through the optimization of

* Corresponding author. Tel.: 86-10-62334781-13.
E-mail address: qiyuee@ustb.edu.cn.

doi:10.1016/j.proeng.2011.08.731

processor pipeline. The architecture of the pipeline multi-threading prevents the power cost from thread switching, can achieve low power without missing performance.

But for network processor, the workload and traffic are random, which result the processor running without program. How to achieve high throughput and low power while satisfy the real-time is a challenge in the design of network processor. In this paper, we focus on how to adaptive adjust the hardware thread number dynamic according to the external network traffic and load in order to reduce power consumption.

Based on the Ref. [1], we presented a dynamic multi-thread management mechanism for network protocol processor. And the interface between hardware and software is realized by providing a special instruction set. The paper is organized as follows: Section 2 describes the hardware multi-threading architecture, Section 3 discusses the dynamic management mechanism of hardware multi-threading, Section 4 describes the special instruction design and timing constrains, Section 5 gives the summary.

2. Hardware multi-threading architecture

The architecture in Ref. [1] is based on the typical single-threading RISC architecture. As shown in Fig. 1, the modifications can be found in sub-stages inserted into main stages on the original pipeline to shorten the delay of critical path, multiple duplications for thread context storage, and control logic for thread context switching and pipeline stall.

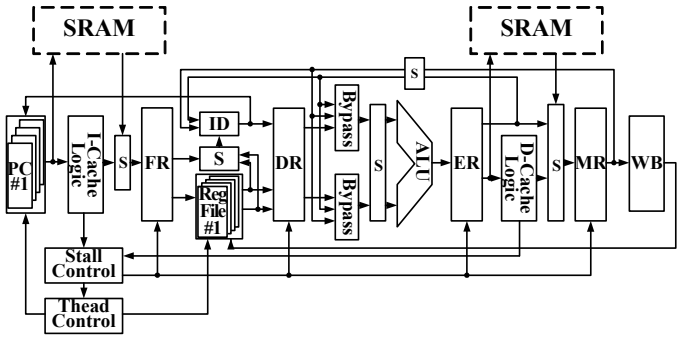

Figure 1. Hardware multi-threaded protocol processor architecture block diagram.
S represents registers between two sub-stages

When the network traffic changes, software thread of network data processing will immediately change, if we can adaptive adjust the number of hardware threads according to network traffic, that is guaranteed satisfy the needs of data processing with active hardware threads as little as possible, we can reduce power consumption of the network processor and achieve workload adaptive. Due to the architecture in Figure 1, that adaptive change the number of hardware threads means to dynamic change the frequency of hardware multi-threading.

For the convenience of discussion, assume processor supports up to 4 threads. By adjusting the clock frequency, to achieve 4 threads, 2 threads and one thread processor. Assume the maximum clock frequency supported by the processor is F Hz, when the network loads are full, we need to run the four hardware threads. at this time set the clock of each sub-stage clk1, clk2, clk3, clk0 frequency of F Hz, each clock with the same phase.

When the network load decreases, the software thread may reduced to two threads, then change the frequency of pipeline to F/2 Hz, by adjusting the clock phase, making the thread interval with the same phase. The timing is shown in Figure 2, two active hardware threads mode will receive.

Similar, by setting the frequency of the four clock to F/4 Hz, adjusting the clock phase difference of 1/4 cycles, one hardware thread mode achieved, the timing shown in Figure 3.

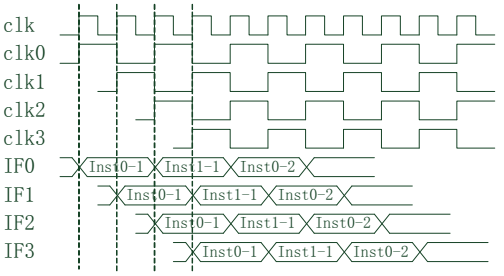

Figure 2. Timing of two hardware thread

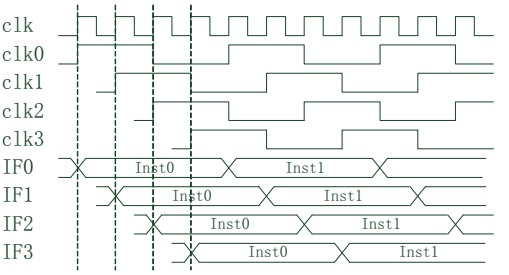

Figure 3. Timing of single hardware thread

3. Soft control of hardware multi-threading management mechanism

This paper presents a thread management mechanism of dynamic frequency change. In other words, according to the current network traffic, the processor dynamically change the number of hardware threads, including thread dynamically open and close. The control of the thread is realized by software and hardware co-design.

3.1. Close thread control

Since this hardware multi-threading architecture is multiple threads sharing the pipeline, open or close the thread means to change the frequency of the pipeline dynamically. Clock frequency is usually generated multiple, so the thread would be doubling the number of changes, for example, the original number of hardware threads are four, the dynamic close can only to the two threads and not be three threads, that is separated by 2 threads are reserved. This will result in thread migration in the closure process, if it is adjacent two threads need to close. Proposal is as follows.

Number the four hardware thread as {0,1,2,3}, different hardware thread will correspond to a different register file.

Define: Tno as the thread number being closed, Tstatus as the thread status table to record the current status of all hardware threads, MapTable is used to record the mapping between the hardware thread and the register file. MapTable [i] == j means the j^{th} register file is corresponded to the i^{th} thread.

The close mode of thread were divided from the different conditions when the thread being closed, including: only close one thread, no thread migration. And close the interval between the two threads, no migration. And close the two adjacent threads, thread migration needs.

When a software thread is finished, the thread will initial the closing thread process. First, read the current value of Tstatus, then revise the Tstatus table according current thread number being closed.

For example, assume the current Tstatus is 1011, and the hardware thread number to be closed is 0. There will have 2 hardware thread idle when turn off the 2 soft thread, so it is time to close hardware thread from four to two. After mode switching, the soft thread which run on the 2^{nd} and 3th hardware thread will be reserved. That result in one of the two soft thread need to migrate to another hardware thread in order to ensure the program running correct.

3.2. Open thread control

When network traffic becomes large, the current hardware thread can not support the real-time processing, the software will open more hardware threads and load the new soft threads onto the opened hard threads to improve the processor performance. The detail of open thread is as follows.

(1) If the current Tstatus is one of [0001], [0010], [0100] or [1000], that three hard threads are idle, set the interval thread bit to 1and send open instruction to hardware when open.

(2) If the current Tstatus is one of [1010] or [0101], set the corresponding bit which thread will be active to 1, and send open instruction to hardware when open.

(3) If the current Tstatus is one of [0111], [1011], [1101] or [1110], that means one hard thread is idle, only need to set the idle bit to 1.

(4) In other cases, the work mode needn't change.

4. Design and implementation of special instructions

In order to control the processor's dynamic frequency, on the one hand to control the software process, on the other hand a corresponding circuits increase in the hardware control logic, support dynamic control of software algorithms. This paper using special instructions achieve software control interface of hardware. It is easy to software programming. The new extended instruction comply with the hardware multi-threading processor [1] instruction format (RISC instruction set).

4.1. Special instruction designed

According to the needs of the software, the extended instruction set include: close thread instruction, TEST & SET instruction, CLEAR instruction, GETVALUE instructions and open thread instructions.

Close thread instruction used to close the hardware thread. Parameters include: Mode, Row and Value. Mode indicates thread closed mode. Row indicates the number of MapTable registers will be changed. Value means the value of MapTable register will be changed.

TEST&SET instruction used to return the value of the LOCK register to the general register, and set the value of the LOCK register to 1. When LOCK reg is 1, indicating a thread is accessing to shared areas, prohibit other threads access the shared area.

CLEAR instruction is used to set the value of LOCK register to 0.

GETVALUE instruction used to assigned the value of specified MapTable register to general register.

Open thread instruction used to open more hardware threads.

4.2. Special instruction timing

In the architecture of hardware multi-threading, thread contexts are stored in different duplicated register. So dynamic frequency may override the context of the thread is not closed or cause corresponding error of the context. It is important to define a reasonable clock switching timing constrains.

The timing of closing thread includes four cases: (1) Close current thread and the interval thread in four threads to two. (2) Close current thread and the one before this thread in four threads to two. (3) Close current thread and the one after this thread in four threads to two. (4) Close current thread in two threads to one. This paper will take the 3rd case as example to illustrate.

Assume the 3rd hard thread send the close thread instruction to close the 3rd and 4th hard thread. The timing switch may take a few clock cycle to correctly close the thread. The close timing constrains as shown in figure 4.

Clk0, clk1, clk2 and clk3 are connected to four sub-stage registers of the pipeline. IF0, IF1, IF2, IF3 represent the four sub-stages register of Instruction Fetch Module, and ID0, ID1, ID2, ID3 represent the four sub-stages register of Instruction Decode Module. In fig.4 different number indicate different soft thread instruction. Sm_n represents n^{th} instruction of m^{th} soft thread. The figure shown that the frequency switching start from T clock, after 3 clk cycles, two hardware thread, 1^{st} and 2^{nd} , are reserved in the pipeline. The frequency switches from F to F/2. And the registers are correct in the pipeline. In other cases of open or close, the timing constrains can be defined in the same way.

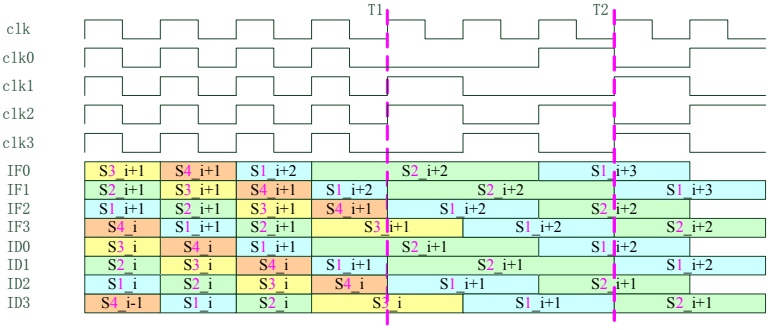

Figure 4. Timing constrains of close thread

5. Conclusion

This paper uses the Xilinx ISE development tools for RTL-level implementation and simulation. Simulation results show that the dynamics multi-thread management mechanisms and specific instructions in this paper make the hardware multi-threaded dynamic frequency can be achieve. According to the dynamic power consumption formula $p = cv^2f$, power and frequency is directly proportional. When the network load is a little or less, multiple threads closed can reduce dynamic power exponentially.

For hardware multi-threading network processor architecture this paper proposed a hardware load-adaptive multi-threaded scheme, the hardware multi-threaded dynamic frequency management and related special instructions. It is applicable to the environment that network loads are uncertainty. The specific saving of power consumption is related with the load distribution.

References

[1] Wang Qin, Wang Lei, Qi Yue,ect. A Hardware Multi-threading Architecture for Protocol Processors. IEEE ICCT 2010, Nan jing

[2] Sisalem D, Wolisz A. Towards TCP-friendly Adaptive Multimedia Applications Based on RTP[C]//Proceedings of the 4th IEEE Symposium on Computers and Communications. 1999-07: 166-172.

[3] Lehoczky J, Lui S, Ye D. The rate monotonic scheduling algorithm: exact characterization and average case behavior [A]. IEEE 9th Real2Time Systems Symposium[C]. Santa Monica: IEEE Communication Society , 1989 :166 - 171.

[4] Audsley N, Burns A, Richardson M. Applying new scheduling theory to static priority preemptive scheduling [J]. Software Engineering Journal, 1993, 8 (5): 284 - 292.

[5] Sisalem D, Wolisz A. Towards TCP-friendly Adaptive Multimedia Applications Based on RTP[C]//Proceedings of the 4th IEEE Symposium on Computers and Communications. 1999-07: 166-172.