Design of Many Core Interrupt Controller Based on ARMv8 Architecture

BING LI
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
Southeast University Chengxian College
Dongda Road No.6, Nanjing, Jiangsu Province, China
CHINA
JUN LU
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
DEZHI WU
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
GUANYU LIU
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA
lj2012_happy89@163.com

Abstract: - This paper presents a design of many core interrupt controller based on ARMv8 architecture, which is known as Generic Interrupt Controller (GIC). Interrupt controller is one of the most important peripheral modules, which affects the performance of microprocessor directly. GIC deals with all the interrupt requests and sends interrupt requests to the each processor, which is connected to the GIC. This design of GIC adopts the principle of hierarchical coding to choice the interrupt which has the highest priority. supports 64 cores, 16 priority levels, level 7 interrupt preemption. After analyze of GIC signature, this paper presents a realization of GIC. Proposed GIC has been verified under Modelsim software and FPGA development board, and the simulation result is consistent with the expectations. Use DC to synthesis GIC with the condition of 40nm process library, 1 GHz, and the timing meet the requirement and the total area is 111490 μm^2.

Key-Words: - ARMv8 architecture; GIC; interrupt controller; many core; AXI; interrupt

1 Introduction

With the development of the integrated circuits, the application of the microprocessor has been permeated all the fields of human life. The microprocessor is the most critical part of the computer systems, the design and manufacturing are the heart of the computer technology. The processor is the core of the computer operation. The performance of processor affects the computer system directly. ARM release new processor architecture (ARMv8), which is the ARM first support 64-bit instruction. ARMv8 is based on 32-bit ARM architecture. Therefore, ARMv8 architecture is regarded as the core technology of next generation processor architecture.

With the improvement of the microprocessor, more and more external devices need to send interrupt requests to the processor, how to manage these external interrupts effectively has become one of the problems in the design of processor. Chip Multi-Processor(CMP) has more than two

processors that integrated on a silicon wafer, processors share L2 Cache or L3 Cache, and use high-speed bus interconnect between the processors [1-3]. This paper presents a design of many core interrupt controller based on ARMv8 architecture, which is known as GIC. GIC deals with all the interrupt requests, and sends interrupt request to the each processor core that connect on the GIC. GIC provides the programmable interface for processor visiting the GIC. We use Modelsim to simulate, and further validate on the Xilinx Virtex-7. Results show that GIC can judge interrupt correctly, and sends interrupt request to the processor.

2 Many Core Processor

Chip Multi-Processor(CMP) has more than two processors that integrated on a silicon wafer, processors share L2 Cache or L3 Cache, and use high-speed bus interconnect between the processors. The many core processor system can realize parallel computing, many core processor architecture depicted in Fig.1.

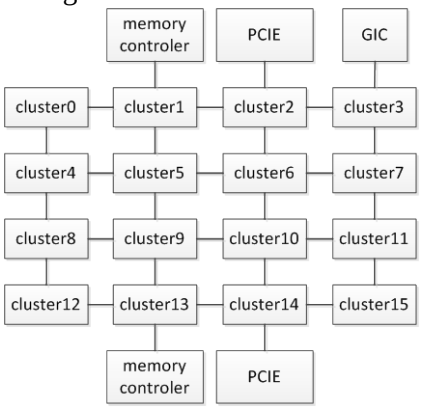

Fig.1 many core processor architecture

Fig.1 shows that many core processor is consisted of Cluster, Memory Controller, PCIE, GIC. The cluster is consisted of four cores, L2 cache and router. Every core uses Simultaneous multi-threading(SMT), so every core supports four threads [6]. The architecture of the Cluster depicted in Fig.2.

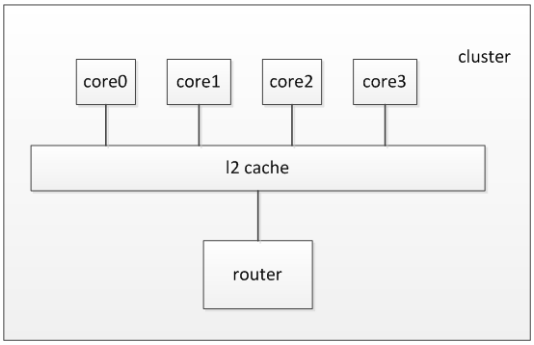

Fig.2 the architecture of the Cluster

This paper is design a GIC that supports 16 clusters, 64 cores, 256 threads, 8244 interrupt requests and Level 7 interrupt preemption.

3 Function of the GIC

Interrupt controller is one of the most important peripheral modules, it affects the whole performance of microprocessors directly. GIC supports 256 threads, GIC deals with all the interrupt requests, and sends interrupt requests to the each processor, which is connected to the GIC.

3.1 Interrupt Type

GIC has three interrupt types, Software Generated Interrupts(SGI), Private Peripheral Interrupts(PPI), Shared Peripheral Interrupts(SPI). Every thread of GIC supports 64 interrupts. SGI is generated only by write GICD_SGIR register, indicate the target processor and interrupt number. PPI and SPI are generated by the external interrupt signal or software to write GICD_ISPENDRn register, indicate the interrupt number.

3.2 Interrupt Register

GIC provides the programmable interface for processor visiting the GIC. GIC has some register. we can see Table 1.

Table 1 GIC register

Name	Purpose
GICD_CTLR	All interrupts are enabled to CPU interface
GICD_TYPER	Provide the interrupt controller configuration information, such as number of interrupts, number of CPU interfaces, whether to support the Security Extension and so on
GICD_IIDR	Provide the version of interrupt controller
GICD_ISENABLERn	Enable each interrupt to CPU interface
GICD_ICENABLERn	Disenable each interrupt to CPU interface
GICD_ISPENDRn	Each interrupt is pending state
GICD_ICPENDRn	Remove each interrupt pending state
GICD_ISACTIVERn	Each interrupt is active state
GICD_IPRIORITYRn	Priority of interrupt
GICD_ITARGETSRn	Target of interrupt
GICD_ICFGRn	Trigger type of interrupt
GICD_SGIR	Control SGI interrupts
GICC_CTLR	Enable interrupt to the processor

GICC_PMR	Mask interrupt priority
GICC_IAR	Interrupt acknowledge
GICC_RPR	Running priority
GICC_EOIR	End of interrupt
GICC_HPPIR	Interrupt with highest priority which in pending state

3.3 Interrupt State

GIC has four states, which is inactive, pending, active, active and pending. Inactive state is that interrupt is not in active or pending state. Pending state is that interrupt is generated by external signal or software, and wait for the target processor acknowledge. Active state is that the interrupt has been acknowledged by the target processor and the interrupt is being handled. Active and pending state is that processor has been responded to the interrupt request, the interrupt is being handled and also have the same interrupt gives interrupt request at the same time. State machine of interrupt processing depicted in Fig.3 and the condition of State machine depicted in Table 2.

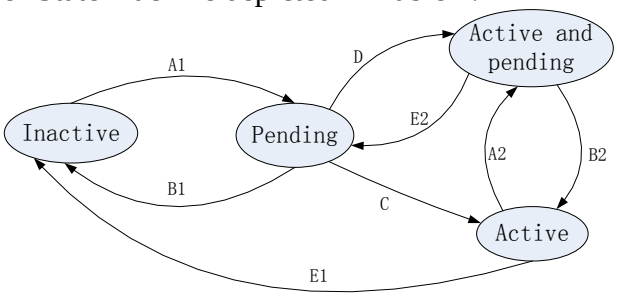

Fig.3 state machine of interrupt processing

Table 2 the condition of State machine

Condition of State	Depicted
A1, A2	SGI is generated by software, PPI or SPI is generated by external request signal or software.
B1, B2	SGI can't remove pending state. PPI or SPI, for level trigger can remove the pending state by remove external request signal or by software, for edge trigger remove the pending state only by software.
C	The target processor responses the interrupt request
D	The target processor responses the interrupt request and add the external request signl at the same time.
E1, E2	End of interrupt.

3.4 GIC Procedure

GIC working procedure is mainly divided into two steps. The first is software initial GIC, then GIC handle interrupt, until handle all the interrupts. Software initialization first is that write "0" to the GICD_CTLR register to mask all interrupts to the CPU interface. Then initialize the register, such as GICD_ISENABLERn, GICD_ITARGETSR, GICD_ICFGRn, GICD_IPRIORITYRn, GICC_CTLR, GICC_PMR, GICC_BPR. After the software configure, write "1" to GICD_CTLR register, enable interrupt to the CPU interface. GIC interrupt handle is to judge all interrupts, and pick the highest priority interrupt which is in the pending state, and then determine whether to send the interrupt request to the target processor, then the target processor responses interrupt request, when finish the interrupt subroutine, software write GICC_EOI register, it shows that processor responses the interrupt request and finish handling the interrupt. GIC working procedure depicted in Fig.4.

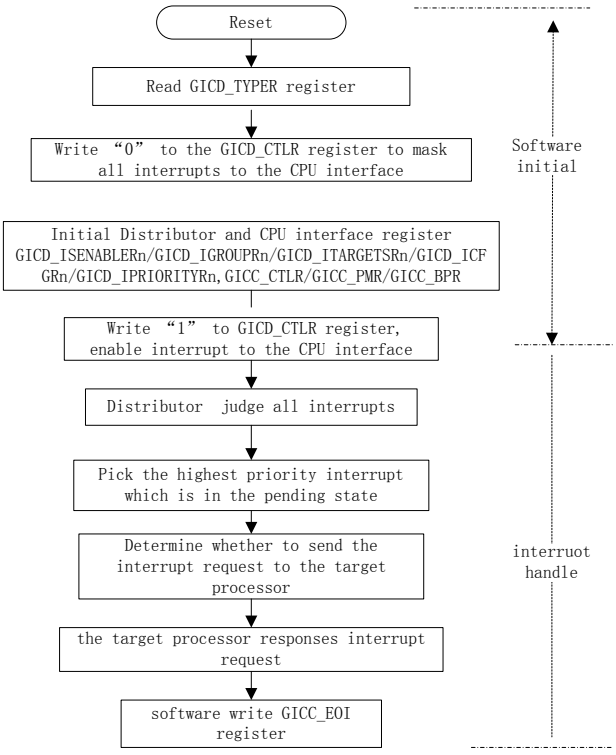

Fig.4 GIC working procedure

4 Design of GIC system

The introduction of the interrupt is to improve the utilization of computer resources, solve the problem of speed mismatch between computer system and various modules. Interrupt is generated by the signal from hardware or software, we call this kind of request signal as interrupt request. system structure of GIC depicted in Fig.5.

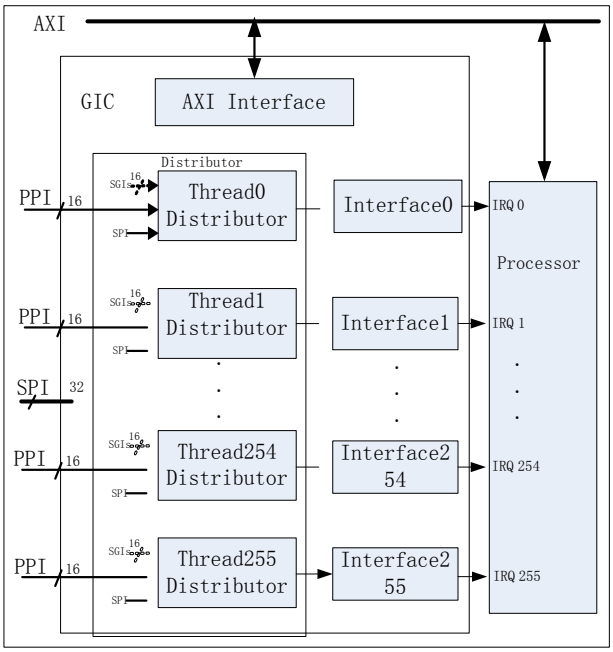

Fig.5 structure of GIC

From Fig.5 we can see that GIC has 3 modules. They are AXI interface, Distributor and Interface. Because of GIC has 256 threads, so width of CPUID is eight.

4.1 AXI Interface

GIC provides the programmable interface that for processor to configure and visit. Through the AXI(Advanced eXtensible Interface) bus protocol, the processor can configure and visit the GIC. Address/Control and data of AXI bus protocol is separated [4-5]. AXI supports the asymmetric data transmission, only need to give the first address in the burst transmission. AXI supports a significant transmission and out-of-order access. AXI protocol is based on burst transmission. Each channel has address and control information. AXI has five channel, they are raddr, rdata, waddr, wdata, wresp. Each channel consists of a set of signals.

4.2 Distributor

Distributor module judges all interrupts. Distributor sends the interrupt which is pending state to the target processor. GIC gets priority of the interrupt according to the GICD_IPRIORITYRn register, and sends the interrupt id and corresponding priority to the target Thread Distributor. Thread Distributor adopts the principle of hierarchy coding to choice the interrupt which has the highest priority and sends the interrupt to the target CPU interface. The principle of hierarchy coding depicted in Fig.6.

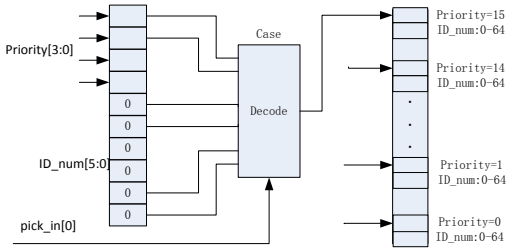

Fig.6 principle of classification coding

Priority and interrupt ID is regarded as the address, which is 16(priority) * 64(interrupt ID) = 1024 bits. For example, a interrupt ID is 2, and it's priority is 3. This interrupt is in pending state. So it's address is 10'b0011_000010, so the value of the 10'b0011_000010 address is "1", other is "0". Then find out the address of the lowest non-zero from 1024 bits register. This 10bits address is the highest pending interrupt's priority and interrupt ID.

It uses the principle of hierarchy coding to 1024 mux 1. It uses three level 8 mux 1 and one 2 mux 1 pipeline. 1024 mux 1 is depicted in Fig.7.

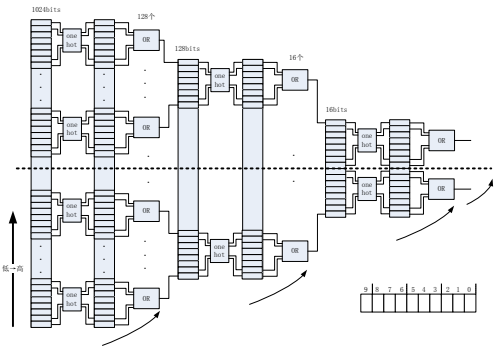

Fig.7 1024 mux 1

4.3 Interface

Interface module masks some interrupts, GIC can only send the interrupt request that priority is higher than mask priority, GIC can only send the interrupt request which interrupt priority higher than the current interrupt priority. GIC sends IRQ to the target processor, the processor responses the interrupt and then end of the interrupt [7-8]. GIC supports Level 7 interrupt preemption, so GIC has interrupt preemptive handling. The design adds some register to protect the field of preemptive,

when finish preempting, GIC restores the current state.

5 Example Result

This design uses verilog language to design the GIC. Use the Modelsim to simulate the each module, and then integrate all modules to debug. In Xilinx ISE14.5 programming environment, synthesis and optimizes the circuit, generate bit files to download to the Virtex-7 development board [9]. Use ChipScope to observe the waveform, compare the Modelsim simlution wave with ChipScope wave, whether meet the requirements of design. With the condition of 40nm process library, 1GHz, we use DC to synthesis GIC. Verify the timing whether meet the requirements and estimate the area of GIC.

5.1 Simulation Result

We use 4 threads for example, simulation result depicted in Fig.8. We can see from Fig.7 that GIC can handle the interrupt and end of the interrupt normally. And we also can see that it has preemption, interrupt ID 17 takes over the interrupt ID 18.

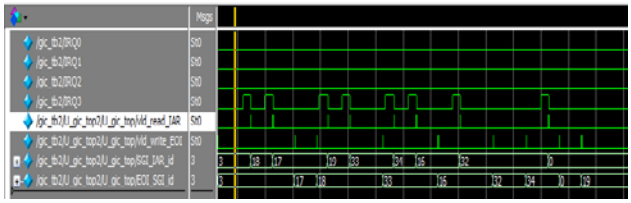

Fig.8 simulation result

5.2 FPGA Verification

In Xilinx ISE14.5 programming environment, the design uses 50 MHz to synthesis and optimize the circuit, and then generates bit files to download to the Virtex-7 development board, using ChipScope to observe the waveform, compare the Modelsim simlution wave with ChipScope wave, and the simulation result is consistent with the expectations. ChipScope result depicted in Fig.9. Rats of resource utilization listed in Table 3.

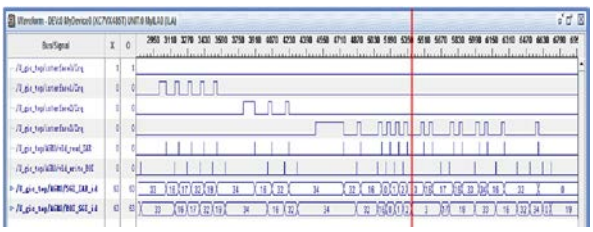

Fig.9 ChipScope result

Table 3 resource utilization

Name	Used number	Total nubmer	Ratio
Register	45960	607200	7%
Look-up-table	54428	303600	18%
IOBs	201	700	28%
BUFGs	2	32	6%

5.3 DC synthesis

Use DC to synthesis GIC with the condition of 40nm process library, 1GHz. Synthesis report is depicted in Fig.10. Area report is depicted in Fig.11. From Fig.10, we can see that GIC meets the timing requirement. From Fig.11, we can see that the area of GIC is 111490μm^2.

Fig.10 Synthesis report

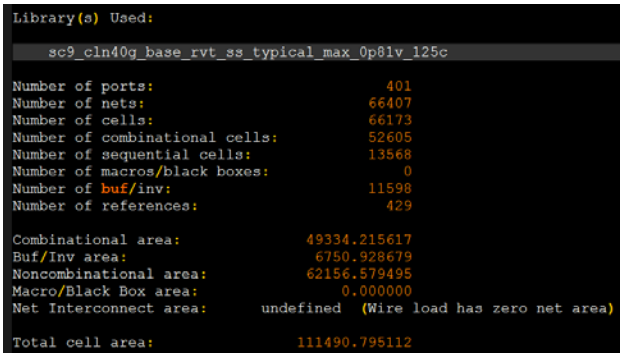

Fig.11 Area report

5.4 Performance Comparison

Interrupt of reference 2 supports 8 processors and 2 Clusters. Compared with reference 2, GIC supports 256 processors and 16 Clusters.

Interrupt of reference 6 supports 60 inputs and 12 outputs. Compared with reference 6, GIC supports 8224 inputs and 8224 outputs.

6 Conclusion

In this paper, we show that how to design of many core interrupt controller based on ARMv8 architecture. The result shows that, GIC supports 16 clusters, 64 cores, 256 threads, 8224 interrupt requests, Level 7 interrupt preemption. GIC deals with all interrupt requests, and sends interrupt request to each processor, which is connected to each GIC. The whole design is implemented in Verilog HDL. Functional simulation and FPGA

validation shown from Fig.8 to Fig.9 demonstrate that the system can work as expected. The design uses Design Compiler to synthesize, the report of timing and area meets the design requirements.

The design can be improved in the future. For example, some modules can be reused in the design to reduce area cost. The Interrupt Controller can be extended to 128 cores to improve the performance. And according to the AXI bus interface, building a SOC prototype verification system to fully verify the design should be considered.

References:

[1] Sugako Otani, Hiroyuki Kondo, Itaru Nonomura, An 80 Gbps dependable multicore communication SoC with PCI express I/F and intelligent interrupt controller. IEEE COOL Chips XIV, 2011.

[2] Huong Thien Hoang, Phong The Vo, Y Thien Vo, Design and Performance Evaluation of an 8-processor 8,640 MIPS SoC with Overhead Reduction of Interrupt Handling in a Multi-core System. IEEE Asian Solid-State Circuits Conference, 2008, pp. 193-196.

[3] K.S.Vijula Grace, Dr.K.Baskaran, Optimized speed multicore based open loop PMDC motor controller embedded system. International Conference on Computing, Electronics and Electrical Technologies, 2012, pp. 26-31.

[4] AXI Reference Guide[Z]. UG761 (v13.1) March 7, 2011.

[5] AMBA AXI and ACE Protocol Specification[Z]. ARM IHI0022D(ID102711),2011.

[6] Wei Chipin, Li zhaolin, Zheng Qingwei, Ye Jianfei, Li Shenglong, Design of a configurable multichannel interrupt controller. 2nd Pacific-Asia Conference on Circuits, Communications and System, v1, 2010, pp. 327-330.

[7] GaiNing Han, YongFeng Li, VxWorks system of touch screen interrupt handling mechanism design based on the ARM9. Communications in Computer and Information Science, v236 CCIS, nPART 6, 2011, pp. 39-44.

[8] HaiBing Guan, YaoZu Dong, Kun Tian, Jian Li, SR-IOV Based Network Interrupt-Free Virtualization with Event Based Polling. IEEE JOURNALON SELECTED AREAS IN COMMUNICATIONS, VOL.31, NO.12, 2013, pp. 2596-2609.

[9] Wojciech Kabacinski, Marek Michalski, The control algorithm and the FPGA controller for non-interruptive rearrangeable Log2(N, 0, p) switching networks. IEEE International Conference on Communications, 2013, pp. 3840-3845.