Design and Verification of Platform Level Interrupt Controller

Submitted by:

2019-FYP-33

Maja 2019-EE-xxx

Saja 2019-EE-xxx

Basantay 2019-EE-xxx

Billa 2019-EE-xxx

Supervised by: Dr. Ubaid Abdullah Fayyaz

Mr. Umer Shahid

Department of Electrical Engineering

University of Engineering and Technology Lahore

Contents

List of Figures

List of Tables

Abbreviations

PLIC	**P**latform **L**evel **I**nterrupt **C**ontroller
ie	**i**nterrupt **e**nable
Reg Map	**Reg**ister **Map**ping
FPGA	**F**ield **P**rogrammable **G**ate **A**rrays

Abstract

Abstract of the project is written here (and usually kept to just this page). (Maximum 350 words)

Chapter 1

Introduction

Discuss the opening perspective of the problem area, the challenge in that area, and refine the challenge into a concise statement. $(1-2$ pages)

Chapter 2

Problem Statement

Unmet need or problem, what is the unmet need or problem the FYDP is aiming to solve? How significant is the problem? Quantify as much as possible. In case of a research problem, show the significance of the unsolved problem. Who needs it? List the type of customers who will be interested in the solution of the problem. For each type of customer, indicate the potential market size. In case of a research problem, identify its scope. (1 page)

Chapter 3

Literature Review

Interrupts are Asynchronous events generated by a external source via hardware. In RISC-V interrupts are classified into timer, software and external interrupts. The external interrupts are also called as global interrupts. Timer and software interrupts are handled by a Core Local Interrupt. External interrupts are handled by the PLIC.

3.1 Working

The PLIC connects the global interrupt sources to the interrupt target i.e., core. The PLIC consists of the "PLIC core" and the "Interrupt gateways". There are multiple interrupt gateways, one per interrupt source. Global interrupts are sent from their source to one of the interrupt gateway. The interrupt gateway processes the arriving interrupt signal from each source and sends a single interrupt request to the PLIC core. The PLIC core contains a set of interrupt enable bits to enable individual interrupt sources in the PLIC. The PLIC core contains pending interrupt bits to signal that an interrupt is waiting to be processed. Also, PLIC core performs interrupt prioritization. Each interrupt source is assigned a separate priority. The PLIC core latches the interrupt request into the Interrupt Pending bits. Whenever, the priority of the pending interrupt exceeds a per-target threshold, the PLIC core forwards an interrupt notification to the interrupt target. The PLIC Claim register holds the highest priority interrupt waiting to be processed. On interrupt completion the interrupt Gateways can now send another interrupt request to the PLIC.

3.1.1 Operation Parameters

Register blocks that perform PLIC operation parameters are Interrupt Priorities registers for the selection of interrupt priority for each interrupt source. The second one is Interrupt Pending Bits register for the interrupt pending status of each interrupt source. The third one is Interrupt Enables register to perform the enablement of interrupt source of each context. The fourth one is Priority Thresholds register to select the interrupt priority threshold of each context. The fifth one is Interrupt Claim register to acquire

interrupt source ID of each context. And finally the Interrupt Completion register to send interrupt completion message to the associated gateway.

3.1.2 Memory Register Map

The base address of PLIC Memory Map is platform implemented and its width is 32-bit.

Register Adress.	Data Width	Description
0x0C000000	4 bytes	Source zero priority(base Addr)
0x0C000004	4 bytes	Source 1 priority
. .		
0x0C002003	8 bytes	Interrupt Enabled Source 24 to 27
0x0C000000	4 bytes	Priority Threshold register
0x0C010010	4 bytes	Interrupt claim

3.2 Interrupt Prioritues

Interrupt priorities are small unsigned integers, with a platform-specific maximum number of supported levels. The priority value 0 is reserved to mean "never interrupt", and interrupt priority increases with increasing integer values. Each global interrupt source has an associated interrupt priority held in a memory-mapped register. Different interrupt sources need not support the same set of priority values. A valid implementation can hardwire all input priority levels. Interrupt source priority registers should be WARL fields to allow software to determine the number and position of read-write bits in each priority specification, if any. To simplify discovery of supported priority values, each priority register must support any combination of values in the bits that are variable within the register, i.e., if there are two variable bits in the register, all four combinations of values in those bits must operate as valid priority levels. The base address of Interrupt Source Priority block within PLIC Memory Map region is fixed at 0x000000.

PLIC register block name	Register Block size in byte	Function
Interrupt Source Priority	1024*4 = 4096(0x1000) bytes	Interrupt Source Priority 0 to 1023

3.2.1 Interrupt Pending Register

The current status of the interrupts pending in the PLIC core can be read from the interrupt pending register. The interrupt pending register is a set of 2, 32 bit words. It can be seen as a array of 8 bytes. The pending bit of interrupt id 0 is stored in LSB of first pending register. The pending bit for interrupt ID N is stored in the N mod

8th bit of N/8th byte. The PLIC has 2 interrupt pending registers. Bit 0 of byte 0 represents the non-existent interrupt source 0 and is hardwired to zero. A pending bit in the PLIC core can be cleared by setting the associated enable bit then performing a claim as described in section. The content of the Interrupt pending register is read-only.

3.3 Interrupt Enablers and Priority Thresholds

Each global interrupt can be enabled by setting the corresponding bit in the enables register. The enables registers are accessed as a contiguous array of 32-bit registers, packed the same way as the pending bits. Bit 0 of enable register 0 represents the non-existent interrupt ID 0 and is hardwired to 0. PLIC has 15872 Interrupt Enable blocks for the contexts.In PLIC a large number of potential IE bits might be hardwired to zero in cases where some interrupt sources can only be routed to a subset of targets. A larger number of bits might be wired to 1 for an embedded device with fixed interrupt routing. Interrupt priorities, thresholds, and hart-internal interrupt masking provide considerable flexibility in ignoring external interrupts even if a global interrupt source is always enabled.

PLIC provides context based threshold register for the settings of a interrupt priority threshold of each context. The threshold register is a WARL field. The PLIC will mask all PLIC interrupts of a priority less than or equal to threshold. For example, a‘threshold‘ value of zero permits all interrupts with non-zero priority.

3.3.1 Interrupt Completion

When PLIC signals completes executing an interrupt handler by writing the interrupt ID it received from the claim to the claim register. The PLIC does not check whether the completion ID is the same as the last claim ID for that target. If the completion ID does not match an interrupt source that is currently enabled for the target, the completion is silently ignored. After a handler has completed service of an interrupt, the associated gateway must be sent an interrupt completion message, usually as a write to a non-idempotent memory mapped I/O control register. The gateway will only forward additional interrupts to the PLIC core after receiving the completion message.

Chapter 4

Project Overview and Objectives

The Program Level interrupt controller deals with interrupts generated by the peripherals and the processors, handles the interrupt priorities, and delegates the execution to a processor. An interrupt request is an asynchronous signal that is typically triggered by an I/O device which needs to be serviced.

The PLIC in our case has finite interrupt sources. Some of these will be exposed at the top level via the GPIO pins. Other Interrupt sources are muxed with GPIO pins. They can be used by configuring pinmux registers. If pinmux value is zero, all the pins are GPIO configured. And if one pin goes to the I2C. The source of interrupts for PLIC are the devices connected to the SoC (GPIO, UART, I2C, etc...). As per the RISC-V specification these are termed as global interrupt sources. Global interrupt sources can take many forms, including level-triggered and edge-triggered. In our case, all the interrupts will be positive edge triggered. Our concept will be more clear by looking onto the above fig.

4.1 Design Management Challanges

4.1.1 Design Flexibility and Complexity

- Potentially hundreds and even thousands of registers in Memory Mapped Management Interface.
- Interface Complexity discourages design changes.
- Documentation is time-consuming and error-prone.

4.1.2 Interface Design Goals

- Keep memory map as small as possible.
- Maintain an intuitive, logical arrangement of registers.
- Avoid need for manual maintenance of interface code.

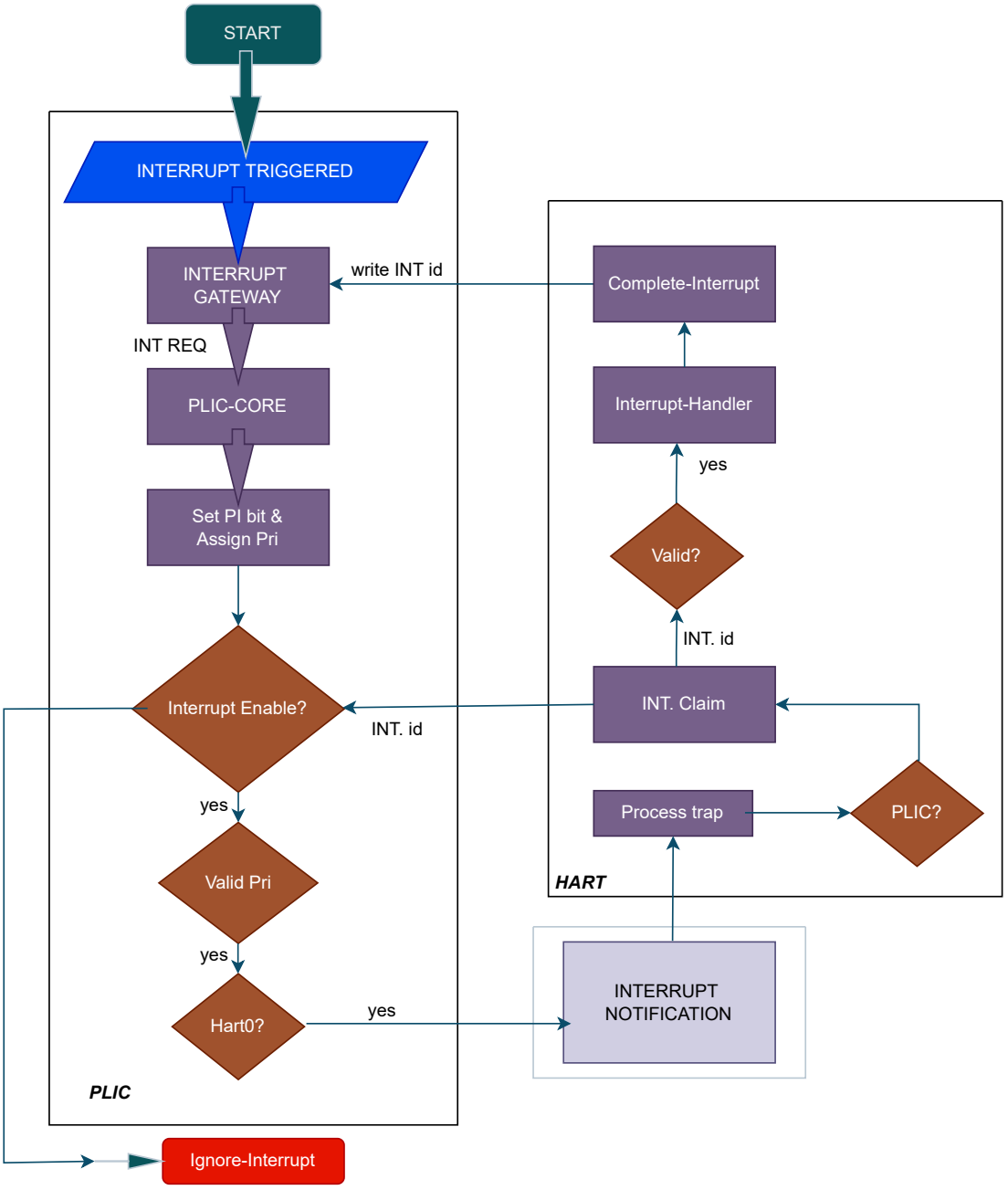

FIGURE 4.1: Interrupt Controller Overview

4.2 Design Management Solutions

4.2.1 Dynamic Memory Map

- Easily adapt to wide range of parameters.
- Automate practical register arrangement.
- Automate documentation of memory map.

4.2.2 No. of Registers

- Re-Use Read ID register as “Interrupt Claim”.
- Re-Use Write ID register as “Interrupt Complete”.

4.3 PLIC Design Goals

- Easy integration with external bus interfaces.
- Support user defined number of Interrupt Sources and Targets.
- Enabling and disabling of individual interrupt sources per target.
- Full Priority Level and Priority Threshold support.
- Low latency handling of queued interrupt requests.
- Programmable depth queue of pending interrupts.

Chapter 5

Project Development Methodology/Architecture

Distribute the project goals into smaller objectives/modules and outline deliverables for each objective. Explain the modules of the project through a system-level block diagram. Students may also mention tools, technologies, and suitability of the method(s) to be employed with justification. In case of a research problem, outline the approaches that will be investigated in the project. $(2-3$ pages)

Chapter 6

Project Milestones and Deliverables

Clear milestones should be defined at the start of the project in the form of a Gantt chart. It is recommended to use excel or some equivalent software to make a Gantt chart. $(1-2$ pages)

Chapter 7

Block Diagram

As interrupts come from sources (from the same or different sources), we also tell whether the interrupt is **Level or edge triggered**? Then gateway passes these intterupts to target in the form of ip (interrupt pending).

Target then segregates the interrupts with respect to their priority. Then finally, using valid-ready interface interrupt is sent to RISC-V processor to service it. High priority interrupt is serviced first.

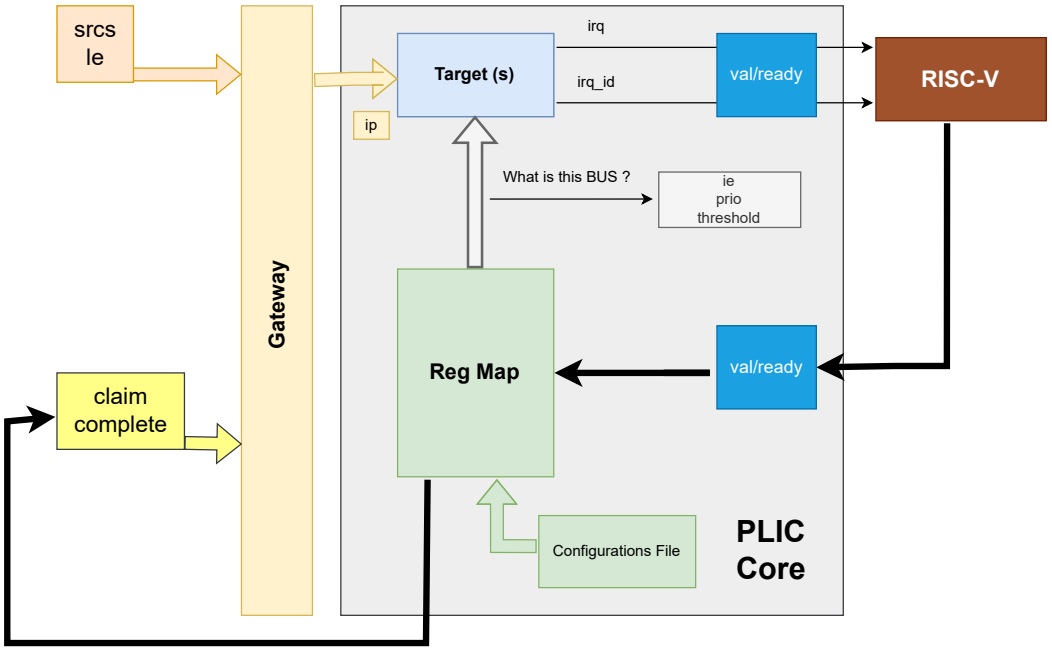

FIGURE 7.1: Block Diagram.

As we see in the figure 7.1, there is also a block Reg Map (Register Map). It is used to store a particular interrupt is enabled or not, what is its priority, what is the threshold of the target to take that interrupt? etc. We program all this stuff using configuration file. At last, Processor also tells PLIC core that particular interrupt is taken and PLIC takes action to mark it as complete.

Chapter 8

Flow Chart

Once interrupt source triggers the interrupt, it then goes to gateway and then gateway will request that interrupt to PLIC Core which is shown in the Figure 8.1.

After the interrupt is requested to PLIC core, it will then **alert/notify** target(s). Now the important step comes, after the target has received the interrupt, it will send a signal back to PLIC core that this interrupt is claimed (Also the claim ID). Receiving the claim from target, PLIC will send claim response that this interrupt should be serviced or not. Then the interrupt handler services the that particular interrupt.

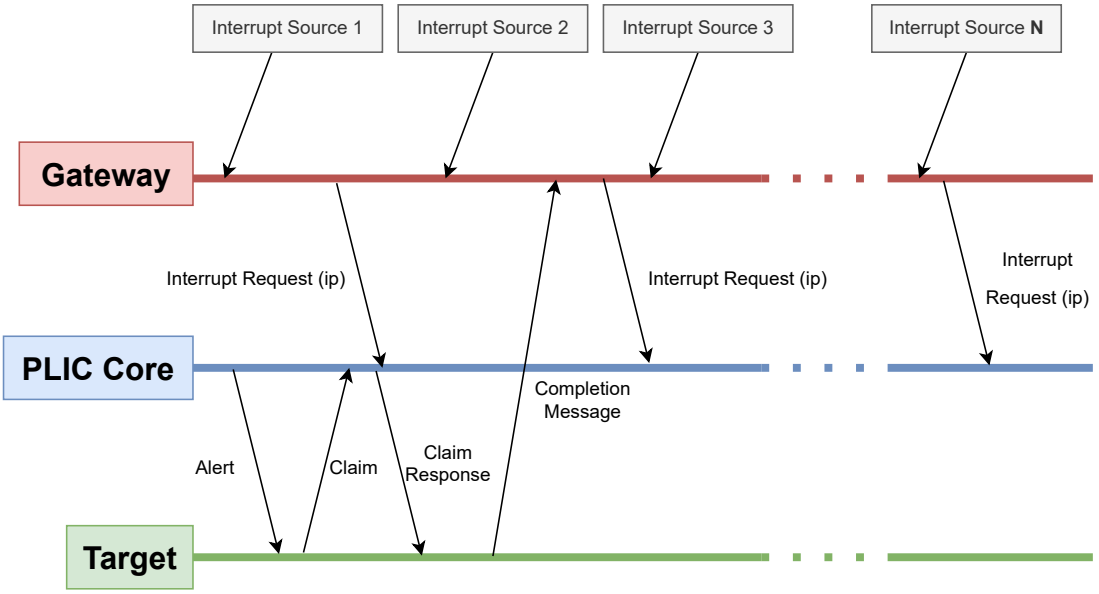

FIGURE 8.1: Flow Chart.

Finally, the interrupt completion message will be sent to gateway that this interrupt is completed and no need to send any other interrupt from that source. So is the case with all other pending interrupts from other sources.

Chapter 9

Work Division

The work division among the four members is given below:

9.0.1 Fatima's Work

She is responsible to Design and Verify the **Gateway(s)**. This is a block or a device which will convert the interrupts from different sources to interrupt requests based on their triggering type (Level/Edge). It will also take care of interrupt claim and complete message from RISC-V.

9.0.2 Shaheer's Work

He is responsible to Design and Verify the **Target**. This is a block or a device which will prioritize the interrupt requests to be sent to RISC-V. It will be based on mainly two algorithms. User will have to decide the algorithm for prioritization.

9.0.3 Ahsan's Work

He is responsible to Design and Verify the **Register Map (Reg Map)**. This module is a set of registers which will be programmed externally to give data to almost every device inside the PLIC. For example, If Gateway wants ie (interrupt enable) signal, then it will take from register map. If **Target** wants priorities of the enabled interrupts, then it will take from register map. And so on.

9.0.4 Ateeqa's Work

She is responsible to interface and merge all the modules to a new device which is eventually going to be the PLIC Core to communicate among all the devices like Gateway, Target, Register Map etc.

Chapter 10

Costing

This project requires quite low budget becaue it is 60 to 70 percent software based. The only hardware required is FPGA.

Sr No.	Device Name	Software Name
1	Xilinx's Nexys A7 FPGA	xc7a100tcsg324-1

Rest of the 30 to 40 percent project will be hardware based where Verification of the project will be done on test benches as well as FPGA.

References