Programmable Interrupt Controller (PIC) v1.0

- **IP Documentation**

  October 2019

Contents

Specifications

Programmable Interrupt Controller (PIC) is a fully parameterised, configurable and portable soft IP core. It is a bare RTL design with AHB3-Lite interface to communicate with the host processor.

- ✓ AHB3-Lite interface.
- ✓ Statically configurable parameters:
  - No. of external interrupt sources; supports up to 63 interrupts.
  - No. of priority levels; supports up to 63 levels.
  - No. of nesting levels; supports up to 8 level of nesting.
  - Bus width; 32 or 64.
- ✓ Globally and locally maskable interrupts.
- ✓ Dynamically configurable priority level for each interrupt.
- ✓ Two modes of operation – *Fully Nested Mode* and *Equal Priority Mode*, see here.
- ✓ Supports active-high level sensitive interrupts.
- ✓ Supports only a single core of processor.

Overview

Programmable Interrupt Controller (PIC) receives multiple interrupts from external peripherals and merges them into a single interrupt output to a target processor core. All PIC registers are memory mapped, and accessed through AHB3-Lite bus interface.

Fig.1 shows the top-level block diagram of PIC.

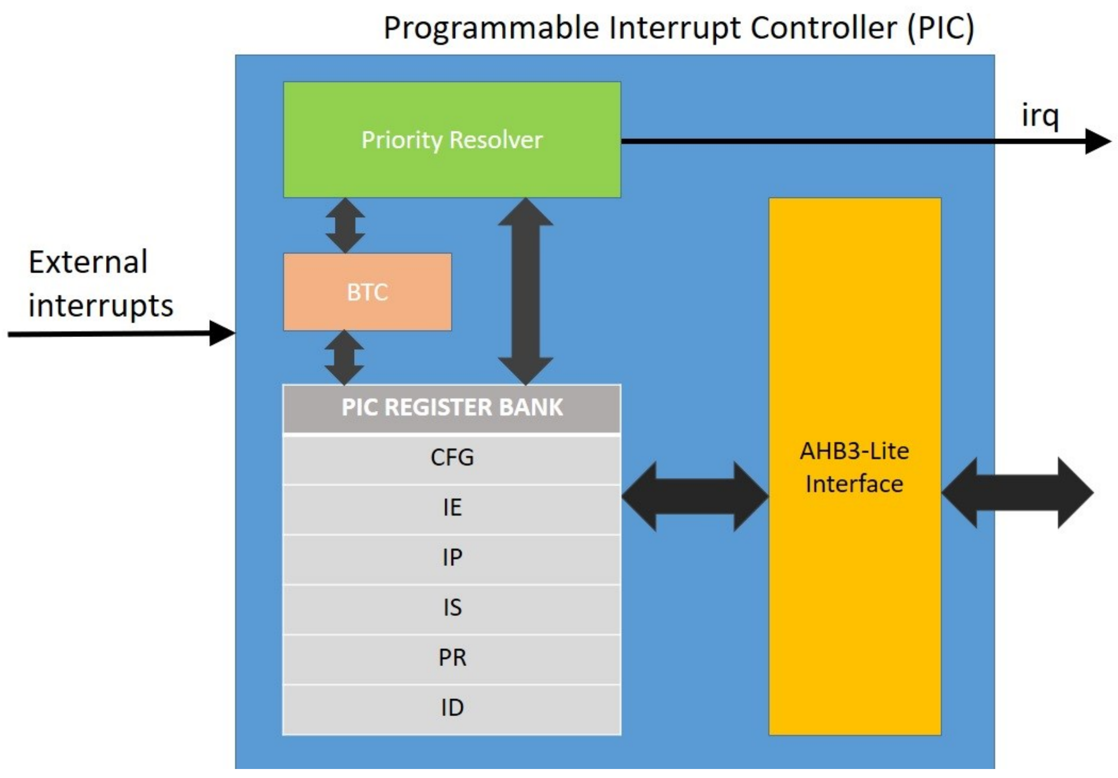

Fig 1: PIC Block Diagram

- **PIC Register Bank**: Contains all set of control and status registers. They are accessed through AHB3-Lite bus interface, see here.
- **BTC**: Binary-Tree-Comparators module is responsible for finding the ID of highest-priority-pending Interrupt (HPPINTR).
- **Priority Resolver**: Responsible to generate interrupt to the target core. It resolves the priorities of HPPINTR and the interrupt which is being currently serviced, and makes decision on whether to assert the `irq` line or not. It takes care of nesting of interrupts as well.

- **External Interrupts:** Various interrupt lines from external peripherals.
- **Irq**: Interrupt line to the target processor core.

Interrupt IDs and Priorities

Each interrupt in PIC has an ID associated with it. The ID is hardcoded for each pin in `ext_intr_i`.

For SOURCES = N, the external interrupt ports are grouped from `ext_intr_i(1)` to `ext_intr_i(N)`, where 1 to N are their respective IDs.

'0' is reserved ID, which means 'no interrupt'.

Each interrupt can be assigned a priority level via corresponding `prreg`. The priority value ranges from 0 to PRLEVELS.

Priority level increases with value. Thus, '1' has the least priority and PRLEVELS has the highest priority. If two interrupts have same priority, then the one with lower ID gets precedence over the other.

Priority level of '0' is special. It means 'never interrupt'. It is effectively another way of masking the interrupt.

RTL Diagram

Fig 2. shows the top-level ports of the IP Core with AHB3-Lite slave interface.

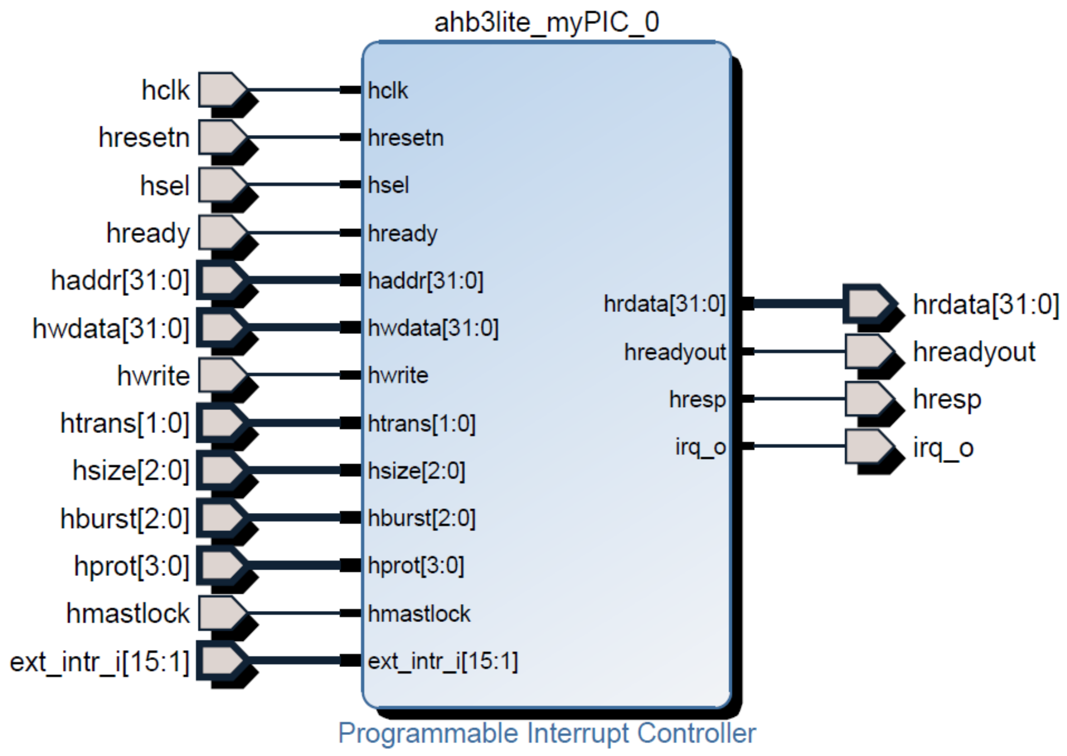

Fig 2: RTL Diagram of PIC

Port Descriptions

Parameters				
S.No	Name	Description		
1	SOURCES	No. of interrupt sources to be supported (1 to 63)		
2	PRLEVELS	No. of priority levels to be supported (1 to 63)		
3	NESTLEVELS	No. of nesting levels needed; 1 to 8 1 ⟶ No nesting …		
4	BWIDTH	Data/Address bus widths for AHB3-Lite interface; 32 or 64		
Signals				
S.No	Name	Direction	Width	Description
1	hclk	in	1	Global clock signal
2	hresetn	in	1	Active-low global sync. reset
3	hsel	in	1	Slave select signal
4	hready	in	1	Ready signal from Master
5	haddr	in	BWIDTH	Address to be accessed
6	hwdata	in	BWIDTH	Data to be written to PIC
7	hwrite	in	1	Write/Read signal
8	htrans	in	2	Transfer type
9	hsize	in	3	** Not used **
10	hburst	in	3	** Not used **
11	hprot	in	4	** Not used **
12	hmastlock	in	1	** Not used **
13	hrdata	out	BWIDTH	Data read from PIC
14	hreadyout	out	1	Transfer finish signal
15	hresp	out	1	Response status signal
16	ext_intr_i	in	SOURCES	External interrupt sources
17	irq_o	out	1	Interrupt to Processor Core

Table 1: Top-level Ports of the PIC

Interrupt Flow

Interrupt Flow in PIC can be categorised into four stages.

○ Interrupt Request

An external interrupt asserts interrupt. This is known as 'Interrupt Request'. The interrupt is then latched as pending. Further requests from the source are blocked by PIC hereafter, until the current request gets serviced. Thus, only outstanding request can be pending at a time from the same source.

○ Interrupt Notification

Each interrupt has ID ranging from 0 to SOURCES, and can be assigned a priority from 0 to PRLEVELS, see here. Each interrupt can be globally or locally enabled/masked as well. Only pending and enabled interrupts are candidates for getting notified to the Processor. The priority levels of all such interrupts are compared and ID register is updated with the highest-priority-pending interrupt (HPPINTR) every clock cycle. If HPPINTR has a priority level greater than the one being currently serviced by the Processor, then the `irq_o` line to the Processor is asserted. This is known as 'Interrupt Notification'.

○ Interrupt Claim-Response

After receiving Interrupt Notification, the Processor will send a read request for ID register. This is known as 'Interrupt Claim'. PIC responds with the value in ID register. This is known as 'Response'. On claiming, the corresponding pending bit is cleared and the servicing bit is set, signifying that the interrupt is being serviced right now. The `irq_o` line goes low as well until a higher priority interrupt is qualified to pre-empt.

○ Interrupt Completion

After the complete execution of ISR, the Processor signals 'Interrupt Completion' to PIC by writing to ID register. PIC then clears the corresponding servicing bit and unblocks the source to forward further requests. The value written to ID register is irrelevant.

Modes of Operation

PIC has two primary modes of operation which can be configured via `cfreg` – *Fully Nested Mode* and *Equal Priority Mode*. They are described below.

○ Fully Nested Mode

This is the default mode of operation of PIC after reset. In this mode, all interrupts can be assigned different priorities and hence can be nested up to NESTLEVELS , if the Processor core supports nesting and pre-emption. In this mode of operation, it is assumed that the last claimed/acknowledged interrupt is the first one to be completed.

○ Equal Priority Mode

In this mode, all interrupts have the same priority level regardless of what value is written on their `prreg`s. No nesting is hence possible with this mode. All interrupts are polled from ID 0 to SOURCES each time and the first pending interrupt is claimed and serviced.

PIC Registers

PIC has 6 set of memory-mapped control and status registers which can be accessed through AHB3-Lite interface.

○ Config Register (`cfreg`)

It is a R/W register of width BWIDTH. It is used to configure the mode of operation of PIC and control global masking. Reset value of the register is 0x0.

RESERVED	MODE	GIE
BWIDTH-1	1	0

GIE: ‘1’ => All interrupts are globally enabled, ‘0’ => All interrupts are globally masked.

MODE: ‘1’ => Equal Priority Mode, ‘0’ => Fully Nested Mode.

○ Interrupt Enable Registers (`iereg`)

They are R/W registers of width BWIDTH. It is used to enable/disable interrupts from individual sources. Reset value of the registers is 0x0.

ID BWIDTH-1		ID 1	ID 0
BWIDTH-1		1	0

Each interrupt can be enabled and disabled by setting or clearing respective bits in `iereg`. If all interrupts are globally masked, then the enable bit has no effect. If all interrupts are globally enabled, then the enable bit decides whether the interrupt is masked or not. The number of `iereg`s in PIC depends on SOURCES and BWIDTH.

N = [[(SOURCES+1)/ BWIDTH]] , where [[x]] is the least integer greater than or equal to x.

The `iereg` and the bit to be accessed to enable/disable an interrupt ID can be found as:

Reg index = [ID/BWIDTH], where [x] is the greatest integer less than or equal to x.

Bit index = (ID mod BWIDTH).

Reg index varies from 0 to **N-1** and Bit index varies from 0 to BWIDTH-1.

Interrupt Pending Registers (`ipreg`)

They are Read-only registers of width BWIDTH. Only PIC can write to this register. It keeps the pending status of each interrupt. Reset value of the registers is 0x0.

ID BWIDTH-1		ID 1	ID 0
BWIDTH-1		1	0

The number of `ipreg`s in PIC depends on SOURCES and BWIDTH.

N = [[SOURCES+1/ BWIDTH]] , where [[x]] is the least integer greater than or equal to x.

The `ipreg` and the bit that stores the pending status of an interrupt ID can be found as:

Reg index = [ID/BWIDTH], where [x] is the greatest integer less than or equal to x.

Bit index = (ID mod BWIDTH).

Reg index varies from 0 to **N-1** and Bit index varies from 0 to BWIDTH-1.

Interrupt Service Registers (`isreg`)

They are Read-only registers of width BWIDTH. Only PIC can write to this register. Each bit in the register indicates whether the corresponding interrupt is being serviced or not. Reset value of the registers is 0x0.

ID BWIDTH-1		ID 1	ID 0
BWIDTH-1		1	0

The number of `isreg`s in PIC depends on SOURCES and BWIDTH.

N = [[SOURCES+1/ BWIDTH]] , where [[x]] is the least integer greater than or equal to x.

The `isreg` and the bit that stores the pending status of an interrupt ID can be found as:

Reg index = [ID/BWIDTH], where [x] is the greatest integer less than or equal to x.

Bit index = (ID mod BWIDTH).

Reg index varies from 0 to **N-1** and Bit index varies from 0 to BWIDTH-1.

○ Priority Registers (`prreg`)

They are R/W registers of width BWIDTH. They store the priority levels of each interrupt. Reset value of the registers is 0x01.

The number of `prreg`s in PIC, **N** = SOURCES+1.

`prreg[x]` corresponds to interrupt ID = x.

○ ID Register (`idreg`)

It is a Read-only register of width BWIDTH. It stores the ID of HPPINTR. A read access to ID register is deemed as 'Interrupt Claim'. Writing access to this register is deemed as 'Interrupt Completion'. However, the written value will not be reflected here as only PIC can write to this register. Reset value of the register is 0x0 or 'no interrupt'.

The register map for all the PIC registers are given in Table 2.

S. No	Register	Address Range (12-bit Addresses)
1	`cfreg`	0x000
2	`iereg`	0x100 to 0x1[**N-1**] Least significant 2 bytes –> Reg Index
3	`ipreg`	0x200 to 0x2[**N-1**] Least significant 2 bytes –> Reg Index
4	`isreg`	0x300 to 0x3[**N-1**] Least significant 2 bytes –> Reg Index
5	`prreg`	0x400 to 0x4[SOURCES] Least significant 2 bytes –> Reg Index = ID
6	`idreg`	0x500

Table 2: Register Address Map

Timing Diagrams

All PIC register accesses (read and write) are zero-wait accesses via AHB3-Lite bus. Following figures show the timing diagram for read and write to a register in PIC, assuming a 32-bit bus.

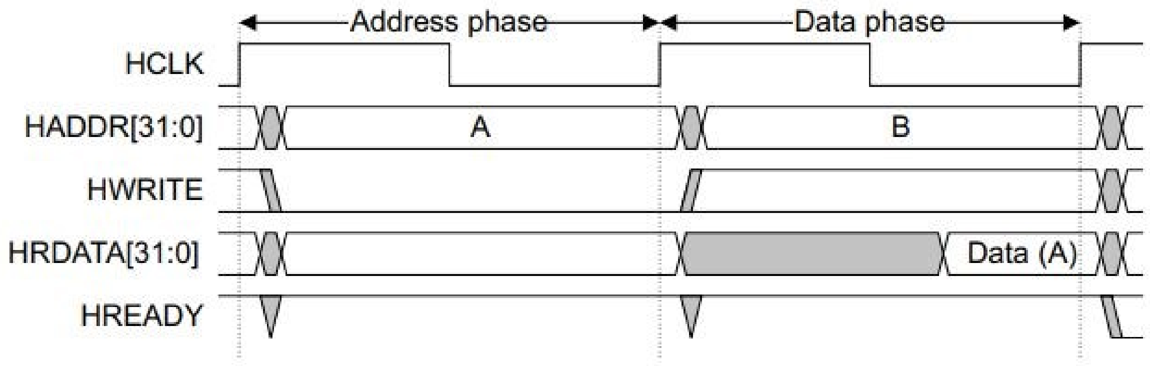

Fig 3: Reading a PIC Register

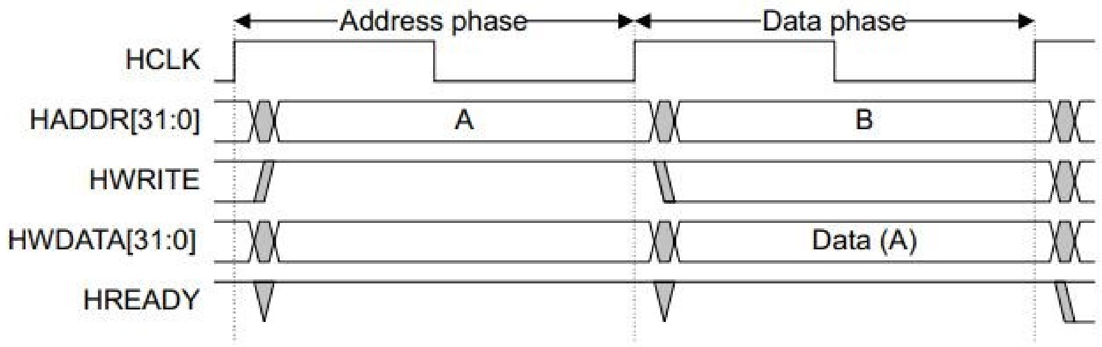

Fig 3: Writing to a PIC Register

Interrupt generated by a source takes t_d = two clock cycles before getting asserted at `irq_o`.

Hence, **Interrupt Latency** added by PIC is = t_d + one read cycle for claim.

Important Notes

- Logical complexity in the critical path in binary tree of comparators that update ID register every cycle can be approximated as:

  C = $\log_2[SOURCES]$

  where [x] is the largest integer which is a power of 2, and which is less than or equal to x.

  Hence, the performance of PIC heavily depends on the value of SOURCES.

- Interrupt level supported at `ext_intr_i` is active-high. If using positive edge-triggered interrupts, note that only one outstanding request can be pending at an given time.

- Interrupt gets registered as pending on the very first assertion from the peripheral. It cannot be retracted by de-asserting the line before claim. However, it can still be masked so that BTC and Priority Resolver ignores it.

- NESTLEVELS >1, gives support for nesting only if PIC is in *Fully Nested Mode* and only if the Processor allows it. In code, nesting can still be disabled by clearing `gie` when entering ISR.

- Re-entrancy is not supported in Fully Nested Mode.

Licensing Notice

Programmable Interrupt Controller v1.0 © 2019

Open-source licensed

Developer: *Mitu Raj*, ***iammituraj@gmail.com***