AHB-Lite Platform Level Interrupt Controller

Roa Logic

Contents

1 AHB-Lite PLIC Datasheet **3**

1.1 Features . . . 3

2 Specifications **4**

2.1 Functional Description . . . 4

2.2 Interrupt Handling Handshake . . . 4

2.2.1 PLIC Configuration . . . 5

2.2.2 Interrupt Request . . . 6

2.2.3 Interrupt Notification . . . 6

2.2.4 Claim Response . . . 6

2.2.5 Interrupt Handler . . . 6

2.2.6 Interrupt Completion . . . 6

3 Configurations **7**

3.1 Core Parameters . . . 7

3.1.1 HADDR_SIZE . . . 7

3.1.2 HDATA_SIZE . . . 7

3.1.3 SRCES . . . 8

3.1.4 TARGETS . . . 8

3.1.5 PRIORITIES . . . 8

3.1.6 MAX_PENDING_COUNT . . . 8

3.1.7 HAS_THRESHOLD . . . 8

3.1.8 HAS_CONFIG_REG . . . 8

4 Interfaces **9**

4.1 AHB-Lite Interface . . . 9

4.1.1 HRESETn . . . 9

4.1.2 HCLK . . . 9

4.1.3 HSEL . . . 9

4.1.4 HTRANS . . . 10

4.1.5 HADDR . . . 10

4.1.6 HWDATA . . . 10

4.1.7 HRDATA 10
4.1.8 HWRITE 10
4.1.9 HSIZE 10
4.1.10 HBURST 11
4.1.11 HPROT 11
4.1.12 HREADYOUT 12
4.1.13 HREADY 12
4.1.14 HRESP 12
4.2 PLIC Interface 12
4.2.1 SRC 12
4.2.2 IRQ 12
4.3 Register Interface 13
4.3.1 CONFIG 13
4.3.2 EL 14
4.3.3 IE[] 14
4.3.4 ID[] 14
4.3.5 PRIORITY[] 14
4.3.6 THRESHOLD[] 15
4.4 Register Address Mapping 15
4.4.1 Itemising Register Requirements 15
4.4.2 Register Address Map 17

5 Resources **19**

6 References **19**

7 Revision History **20**

1 AHB-Lite PLIC Datasheet

The Roa Logic AHB-Lite PLIC (Platform Level Interrupt Controller) IP is a fully parameterized soft IP implementing a Interrupt Controller as specified by the *RISC-V Privileged 1.9.1 specification*.

The IP features an AHB-Lite Slave interface, with all signals defined in the *AMBA 3 AHB-Lite v1.0* specifications fully supported. Bus address & data widths as well as the number of Interrupt Sources and Targets supported are specified via parameters.

The controller further supports user defined priority levels and pending events, in addition to interrupt masking via programmable priority thresholds

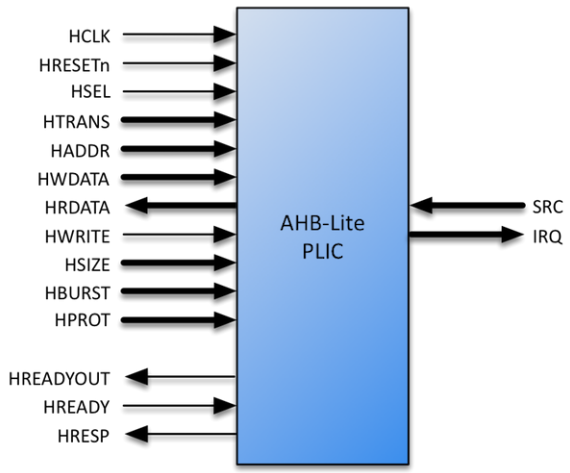


Figure 1: PLIC Port Diagram

1.1 Features

- AHB-Lite Interface with programmable address and data width
- User defined number of Interrupt Sources & Targets
- User defined priority level per Interrupt Source
- Interrupt masking per target via Priority Threshold support
- User defined Interrupt Pending queue depth per source

2 Specifications

2.1 Functional Description

The AHB-Lite PLIC IP is a fully parameterised Platform-Level Interrupt Controller, featuring a single AHB-Lite Slave interface and interfaces to a user-defined number of both Interrupt Sources and Targets.

The purpose of the PLIC core is to connect multiple interrupt sources to one or more interrupt targets. The core supports a programmable number of simultaneous pending interrupt requests per source and routing of those interrupt requests to individual targets.

Per the RISC-V Privileged Architecture Instruction Set specification (v1.9.1), the core performs full interrupt prioritisation of each interrupt source; each may be assigned a separate priority and enabled per target via a matrix of interrupt enable bits. Further, an optional threshold per target may be defined to mask lower priority interrupts.

To reduce latency, the PLIC core presents all asserted interrupts to the target in priority order, queuing them so that a software interrupt handler can service all interrupts without the need to restore the interrupted context.

An example use of the PLIC core is shown below:

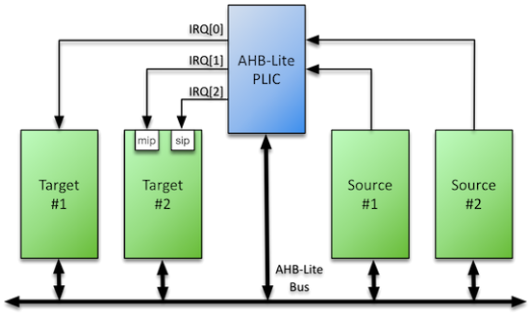


Figure 2: PLIC System Diagram

2.2 Interrupt Handling Handshake

The Roa Logic implementation of the handshake between Interrupt source, target and PLIC is illustrated below, and described in further detail in the following sections:

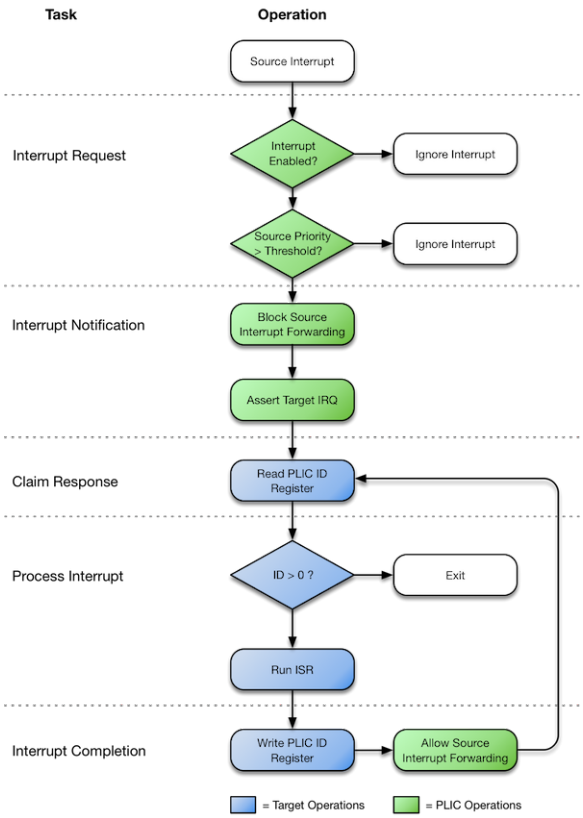


Figure 3:

2.2.1 PLIC Configuration

A matrix of Interrupt Enable vectors – one IE register per target – determines which target processes the interrupts of which source.

Each Interrupt Source attached to the PLIC is then assigned a Priority Level – an unsigned integer that determines the relative priority of the interrupt source. The greater the integer, the greater the priority level. A Priority Threshold per target may also be defined to mask lower priority interrupts such that interrupts will only be presented to a target if the assigned Priority Level Priority Threshold

In addition each source is assigned an Interrupt Identifier (`ID`) – an unique unsigned integer. This identifier determines interrupt priority when 2 or more interrupts with the same priority level are asserted. The lower the `ID` assigned to the source, the greater the interrupt priority.

2.2.2 Interrupt Request

A source asserts an interrupt request to the PLIC. The PLIC validates the request by first checking if an interrupt enable bit is set for each target and if the priority of the interrupt source exceeds any defined Interrupt Priority Threshold. If these conditions do not hold, the Interrupt Request is deemed invalid and ignored.

The PLIC also determines if a previous interrupt request has been made by the same source. If an interrupt is defined as level triggered and has already been asserted but not yet serviced, the request is ignored. If an interrupt is defined as edge triggered and has already been asserted but not yet serviced, the request is queued by incrementing a Interrupt Pending counter by one. The depth of this counter is programmable.

If the request is deemed valid the request is forwarded to the appropriate target. In the case of queued edge-triggered requests, the interrupt pending counter is decremented by one.

2.2.3 Interrupt Notification

A target is notified of an interrupt request by the assertion of the IRQ output for the relevant target. The PLIC also blocks the forwarding of any further requests from the interrupt source until the asserted request is serviced.

On each clock cycle the ID register is loaded with the unique identifier of the highest priority interrupt to be processed.

2.2.4 Claim Response

A target makes an interrupt claim response by reading the ID register, which also notifies the target of the interrupt source to service. The PLIC de-asserts the IRQ output for the target in response to the claim.

2.2.5 Interrupt Handler

If the ID read is greater than zero, the target services the identified interrupt source. If the ID read is zero, this indicates no outstanding pending interrupts remain and the handler may terminate.

2.2.6 Interrupt Completion

Once an interrupt has been serviced, completion is signalled to the PLIC by writing to the ID register. The act of writing to the register is the completion notification; the value written is irrelevant.

On receiving the completion notification the PLIC will again allow interrupts to be forwarded from the corresponding source.

The Interrupt Handler may then exit, however it is possible a new interrupt request may have been asserted while the handler was running. To reduce latency the handler may instead determine is a new interrupt has been received and if so again claim the interrupt as described in section *2.2.4 Claim Response*. In this way the interrupt handler can service all interrupts without the need to restore the interrupted context.

3 Configurations

3.1 Core Parameters

The size and implementation style of the PLIC module is defined via HDL parameters as specified below:

Parameter	Type	Default	Description
AHB Interface:			
`HADDR_SIZE`	Integer	32	Width of AHB Address Bus
`HDATA_SIZE`	Integer	32	Width of AHB Data Buses
PLIC Configuration:			
`SOURCES`	Integer	16	Number of Interrupt Sources
`TARGETS`	Integer	4	Number of Interrupt Targets
`PRIORITIES`	Integer	8	Number of Priority Levels
`MAX_PENDING_COUNT`	Integer	8	Max number of pending events
`HAS_THRESHOLD`	Integer	1	Is Threshold Implemented
`HAS_CONFIG_REG`	Integer	1	Is Config Reg. Implemented

Table 1: Core Parameters

3.1.1 HADDR_SIZE

The `HADDR_SIZE` parameter specifies the address bus size to connect to the AHB-Lite based host. Valid values are 32 and 64. The default value is 32.

3.1.2 HDATA_SIZE

The `HDATA_SIZE` parameter specifies the data bus size to connect to the AHB-Lite based host. Valid values are 32 and 64. The default value is 32

3.1.3 SRCES

The `SOURCES` parameter defines the number of individual interrupt sources supported by the PLIC IP. The default value is 16. The minimum value is 1.

3.1.4 TARGETS

The `TARGETS` parameter defines the number of targets supported by the PLIC IP. The default value is 4. The minimum value is 1.

3.1.5 PRIORITIES

The PLIC IP supports prioritisation of individual interrupt sources. The `PRIORITIES` parameter defines the number of priority levels supported by the PLIC IP. The default value is 8. The minimum value is 1.

3.1.6 MAX_PENDING_COUNT

The PLIC module supports multiple concurrently arriving interrupt requests. The maximum number of requests that are supported is defined by the `MAX_PENDING_COUNT` parameter.

The default value is 8. The minimum value is 0.

3.1.7 HAS_THRESHOLD

The PLIC module supports interrupt thresholds – the masking of individual interrupt sources based on their priority level. The `HAS_THRESHOLD` parameter defines if this capability is enabled.

The default value is enabled ('1'). To disable, this parameter should be set to '0'.

3.1.8 HAS_CONFIG_REG

The PLIC module supports a programmable Configuration Register, which is documented in section 0. The `HAS_CONFIG_REG` parameter defines if this capability is enabled.

The default value is enabled ('1'). To disable this parameter should be set to '0'.

4 Interfaces

4.1 AHB-Lite Interface

The AHB-Lite interface is a regular AHB-Lite slave port. All signals are supported. See the *AMBA 3 AHB-Lite Specification* for a complete description of the signals.

Port	Size	Direction	Description
HRESETn	1	Input	Asynchronous active low reset
HCLK	1	Input	Clock Input
HSEL	1	Input	Bus Select
HTRANS	2	Input	Transfer Type
HADDR	HADDR_SIZE	Input	Address Bus
HWDATA	HDATA_SIZE	Input	Write Data Bus
HRDATA	HDATA_SIZE	Output	Read Data Bus
HWRITE	1	Input	Write Select
HSIZE	3	Input	Transfer Size
HBURST	3	Input	Transfer Burst Size
HPROT	4	Input	Transfer Protection Level
HREADYOUT	1	Output	Transfer Ready Output
HREADY	1	Input	Transfer Ready Input
HRESP	1	Output	Transfer Response

Table 2: AHB Interface Signals

4.1.1 HRESETn

When the active low asynchronous HRESETn input is asserted ('0'), the interface is put into its initial reset state.

4.1.2 HCLK

HCLK is the interface system clock. All internal logic for the AMB3-Lite interface operates at the rising edge of this system clock and AHB bus timings are related to the rising edge of HCLK.

4.1.3 HSEL

The AHB-Lite interface only responds to other signals on its bus – with the exception of the global asynchronous reset signal HRESETn – when HSEL is asserted ('1'). When HSEL is negated ('0') the interface considers the bus IDLE.

4.1.4 HTRANS

HTRANS indicates the type of the current transfer.

HTRANS	Type	Description
00	`IDLE`	No transfer required
01	`BUSY`	Connected master is not ready to accept data, but intents to continue the current burst.
10	`NONSEQ`	First transfer of a burst or a single transfer
11	`SEQ`	Remaining transfers of a burst

Table 3: HTRANS Transfer Types

4.1.5 HADDR

`HADDR` is the address bus. Its size is determined by the `HADDR_SIZE` parameter and is driven to the connected peripheral.

4.1.6 HWDATA

`HWDATA` is the write data bus. Its size is determined by the `HDATA_SIZE` parameter and is driven to the connected peripheral.

4.1.7 HRDATA

`HRDATA` is the read data bus. Its size is determined by `HDATA_SIZE` parameter and is sourced by the APB4 peripheral.

4.1.8 HWRITE

`HWRITE` is the read/write signal. `HWRITE` asserted ('1') indicates a write transfer.

4.1.9 HSIZE

`HSIZE` indicates the size of the current transfer.

HSIZE	Size	Description
000	8 bit	Byte
001	16 bit	Half Word
010	32 bit	Word
011	64 bits	Double Word
100	128 bit	

HSIZE	Size	Description
101	256 bit	
110	512 bit	
111	1024 bit	

Table 4: HSIZE Values

4.1.10 HBURST

HBURST indicates the transaction burst type – a single transfer or part of a burst.

HBURST	Type	Description
000	`SINGLE`	Single access**
001	`INCR`	Continuous incremental burst
010	`WRAP4`	4-beat wrapping burst
011	`INCR4`	4-beat incrementing burst
100	`WRAP8`	8-beat wrapping burst
101	`INCR8`	8-beat incrementing burst
110	`WRAP16`	16-beat wrapping burst
111	`INCR16`	16-beat incrementing burst

Table 5: HBURST Types

4.1.11 HPROT

The `HPROT` signals provide additional information about the bus transfer and are intended to implement a level of protection.

Bit#	Value	Description
3	1	Cacheable region addressed
	0	Non-cacheable region addressed
2	1	Bufferable
	0	Non-bufferable
1	1	Privileged Access
	0	User Access
0	1	Data Access
	0	Opcode fetch

Table 6: HPROT Indicators

4.1.12 HREADYOUT

`HREADYOUT` indicates that the current transfer has finished. Note, for the AHB-Lite PLIC this signal is constantly asserted as the core is always ready for data access.

4.1.13 HREADY

`HREADY` indicates whether or not the addressed peripheral is ready to transfer data. When `HREADY` is negated ('0') the peripheral is not ready, forcing wait states. When `HREADY` is asserted ('1') the peripheral is ready and the transfer completed.

4.1.14 HRESP

`HRESP` is the instruction transfer response and indicates OKAY ('0') or ERROR ('1').

4.2 PLIC Interface

Blah

Port	Size	Direction	Description
`SRC`	`SOURCES`	Input	Interrupt Sources
`IRQ`	`TARGETS`	Output	Interrupt Requests

Table 7: PLIC Interface Signals

Note: Width of PLIC interface buses defined by Core Parameters.

4.2.1 SRC

Interrupt sources connect to the `SRC[SOURCES-1..0]` input of the PLIC module. The width of this interface is defined by the `SOURCES` parameter.

4.2.2 IRQ

Interrupt targets are sourced by the `IRQ[TARGETS-1..0]` output of the PLIC module. The width of this interface is defined by the `TARGETS` parameter.

4.3 Register Interface

The following registers are user accessible in the PLIC module:

Register	Registers	Width (bits)	Mode	Function
`CONFIG`	1	64	RO	Configuration
`EL`	1	`SOURCES`	RW	Edge/Level Trigger
`IE`	`TARGETS`	`SOURCES`	RW	Interrupt Enable
`ID`	`TARGETS`	clog_2(`SOURCES`)	RW	ID of Highest priority IRQ, Int. Claim (R), Int. Complete (W)
`PRIORITY`	`SOURCES`	clog_2(`PRIORITIES`)	RW	Priority Level
`THRESHOLD`	`TARGETS`	clog_2(`PRIORITIES`)	RW	Priority Threshold

Table 8: PLIC Register Interface

Note: $\text{clog}_2()$ refers to the System Verilog function by the same name, defined as:

> *The system function $clog2 shall return the ceiling of the log base 2 of the argument (the log rounded up to an integer value). The argument can be an integer or an arbitrary sized vector value. The argument shall be treated as an unsigned value, and an argument value of 0 shall produce a result of 0.*

4.3.1 CONFIG

The `CONFIG` register is a Read-Only register that enables a software routine to determine the hardware configuration of the PLIC module.

When enabled via the `HAS_CONFIG_REG` hardware parameter, the `CONFIG` register returns a 64 bit value constructed as follows:

Bit Position	63		49	48	47	32	31	16	15
Value	0	HAS_THRESHOLD	PRIORITIES	TARGETS	SOURCES				

The values, `HAS_THRESHOLD`, `PRIORITIES`, `TARGETS` and `SOURCES` correspond to the hardware parameters documented in section 3.1.

4.3.2 EL

The `EL` Read/Write register defines if an interrupt source is Edge or Level Triggered.

The number of interrupt sources, as defined by the `SOURCES` parameter, determines the width of the `EL` register. One bit within the register corresponds to an interrupt source, where a logic high ('1') defines a rising-edge triggered interrupt and a logic low ('0') defines a level triggered interrupt.

4.3.3 IE[]

The matrix of `IE[]` Read/Write registers define if an interrupt source is enabled or disabled for a specific target. When disabled, any interrupts generated by the source will be ignored by the PLIC.

The number of targets determines the number of `IE[]` registers. The number of interrupt sources, as defined by the `SOURCES` parameter, determines the width of each `IE[]` register. One bit within the register corresponds to an individual interrupt source, where a logic high ('1') defines an interrupt source as enabled and a logic low ('0') as disabled.

4.3.4 ID[]

The `ID[]` Read/Write register identifies to each target the ID of the highest priority pending interrupt request.

This register indicates to the target which of potentially multiple pending interrupts should be serviced rather than relying on this being resolved by the software Interrupt Service Routine.

When a target reads this register, this also indicates the target has claimed the interrupt for the defined source and will service then service the interrupt source.

A target then writes to this register to indicate completion of servicing the interrupt source. It is the action of writing to this register which generates the interrupt completion notification – the value written will not update the register which continues to identify the highest priority interrupt source to be serviced.

4.3.5 PRIORITY[]

The `PRIORITY[]` Read/Write registers define the priority level of each interrupt source.

There is one `PRIORITY[]` register per interrupt source as defined by the `SOURCES` parameter (see `SOURCES`), identified as `PRIORITY[SOURCES-1:0]`. The

width of each register is derived from the number of priority levels as defined by the PRIORITIES parameter (see TARGETS).

Interrupt priority increases with larger values of PRIORITY.

4.3.6 THRESHOLD[]

Each target may be assigned a priority threshold via the THRESHOLD[] registers. Only active interrupts that have a priority strictly greater than the threshold will cause an interrupt notification to be sent to the target. A THRESHOLD[] value of 0 means that no interrupts will be masked.

4.4 Register Address Mapping

The PLIC supports a wide variety of options and unlimited user-definable number of both interrupt sources and targets. To configure and control the PLIC requires a memory-mapped register interface that must be defined according to the specific implementation.

To ease the development of PLIC based systems, the Roa Logic PLIC implements a dynamic register interface based on the hardware parameters set during generation of the implementation, packing multiple bit-fields into registers where feasible to minimise the required address space.

The following sections describe the calculations performed during generation of the dynamic register interface so that the user may determine the registers available and the memory mapping of those registers for a given implementation.

A spreadsheet in Microsoft Excel format is available to perform these calculations based on user-defined parameters to show the registers and memory mapping. Further, simulation of the PLIC will also shows the registers and memory mapping.

4.4.1 Itemising Register Requirements

The section "Register Interface" provides a summary of the registers required to control and configure the PLIC. The following is a more detailed summary of those requirements.

CONFIG Register The CONFIG register is always 64 bits. For 32 bit implementations this means 2 physical registers are required, 1 each for the upper and lower word. For 64 bit implementations a single register will be implemented.

EL Registers Each interrupt source requires a single bit in the EL register to define if the source is level or edge triggered. These bits will be packed into the

minimum number of registers.

The physical number of registers implemented can be calculated as follows:

```
No. of Registers = ROUNDUP(SOURCES/HDATA_SIZE)
```

Example: For a 32 bit system supporting 48 interrupt sources

```
No. of Registers = ROUNDUP(SOURCES/HDATA_SIZE)
                 = ROUNDUP(48/32)
                 = ROUNDUP(1.5)
                 = 2
```

IE Registers Interrupt sources may be enabled or disabled per target requiring single bit per target. These bits will be packed into the fewest registers possible and the resulting number of registers calculated as follows:

```
No. of Registers = ROUNDUP(SOURCES/HDATA_SIZE)*TARGETS
```

Example: For a 32 bit system supporting 48 interrupt sources and 4 targets

```
No. of Registers = ROUNDUP(SOURCES/HDATA_SIZE)*TARGETS
                 = ROUNDUP(48/32)*4
                 = ROUNDUP(1.5)*4
                 = 2*4
                 = 8
```

ID Registers The `ID[]` Read/Write register identifies the ID of the highest priority pending interrupt request, with one ID register required per target.

```
No. of Registers = TARGETS
```

Priority Registers Each interrupt source can be assigned a priority, which is defined as positive integer. The PLIC parameter `PRIORITIES` defines the number of priority levels for a specific implementation, which then allows a source to be assigned a priority between 1 and `PRIORITIES`.

These priority levels are packed into `HDATA_SIZE` bit registers, as fields aligned to 4-bit nibble boundaries

```
No. of Registers = ROUNDUP(SOURCES/FPR)
```

where:

```
FPR = FIELDS_PER_REGISTER
    = HDATA_SIZE/(4*NPP)

NPP = NIBBLES_PER_PRIORITY
    = ROUNDUP($clog2(PRIORITIES+1)/4)
```

Example: For a 32 bit system supporting 48 interrupt sources and 8 priority levels

```
NPP = NIBBLES_PER_PRIORITY
    = ROUNDUP($clog2(PRIORITIES+1)/4)
    = ROUNDUP($clog2(8+1)/4)
    = ROUNDUP(4/4)
    = 1

FPR = FIELDS_PER_REGISTER
    = HDATA_SIZE/(4*NPP)
    = 32/(4*1)
    = 8

No. of Registers = ROUNDUP(SOURCES/FPR)
                 = ROUNDUP(48/8)
                 = 6
```

Note: $\text{clog}_2()$ refers to the System Verilog function by the same name and calculates the number of binary bits required to represent a given integer.

Threshold Registers Each target may be assigned a priority threshold and therefore the PLIC implements 1 register per threshold.

```
No. of Registers = TARGETS
```

4.4.2 Register Address Map

The order of the registers in the memory map is defined as follows:

Order	Registers
1	CONFIG Register(s)
2	EL Registers
3	PRIORITY Registers
4	IE Registers
5	THRESHOLD Registers
6	ID Registers

Table 10: Register Address Order

Registers a mapped to consecutive addresses based on this order and the number of registers required.

Using the previous example of a 32 bit system supporting 48 interrupt sources, 4 targets and 8 priority levels:

Parameter	Value
HDATA_WIDTH	32
SOURCES	48
TARGETS	4
PRIORITIES	8

The resulting number of registers is:

Registers	Number
CONFIG	2
EL	2
PRIORITY	6
IE	8
THRESHOLD	4
ID	4
Total	**26**

These registers will be then mapped as follows according to the order defined below:

Reg	Parameter	Value
0	0x0	CONFIG
1	0x4	CONFIG
2	0x8	EL
3	0xC	EL
4	0x10	PRIORITY
5	0x14	PRIORITY
6	0x18	PRIORITY
7	0x1C	PRIORITY
8	0x20	PRIORITY
9	0x24	PRIORITY
10	0x28	IE
11	0x2C	IE
12	0x30	IE
13	0x34	IE
14	0x38	IE
15	0x3C	IE

Reg	Parameter	Value
16	0x40	IE
17	0x44	IE
18	0x48	THRESHOLD
19	0x4C	THRESHOLD
20	0x50	THRESHOLD
21	0x54	THRESHOLD
22	0x58	ID
23	0x5C	ID
24	0x60	ID
25	0x64	ID

Note: A spreadsheet exists that can calculate the above Register Address Mapping and is downloadable from the Roa Logic web site.

5 Resources

Below are some example implementations for various platforms.

All implementations are push button, no effort has been undertaken to reduce area or improve performance.

Platform	DFF	Logic Cells	Memory	Performance (MHz)

Table 14: Resource Utilisation Examples

6 References

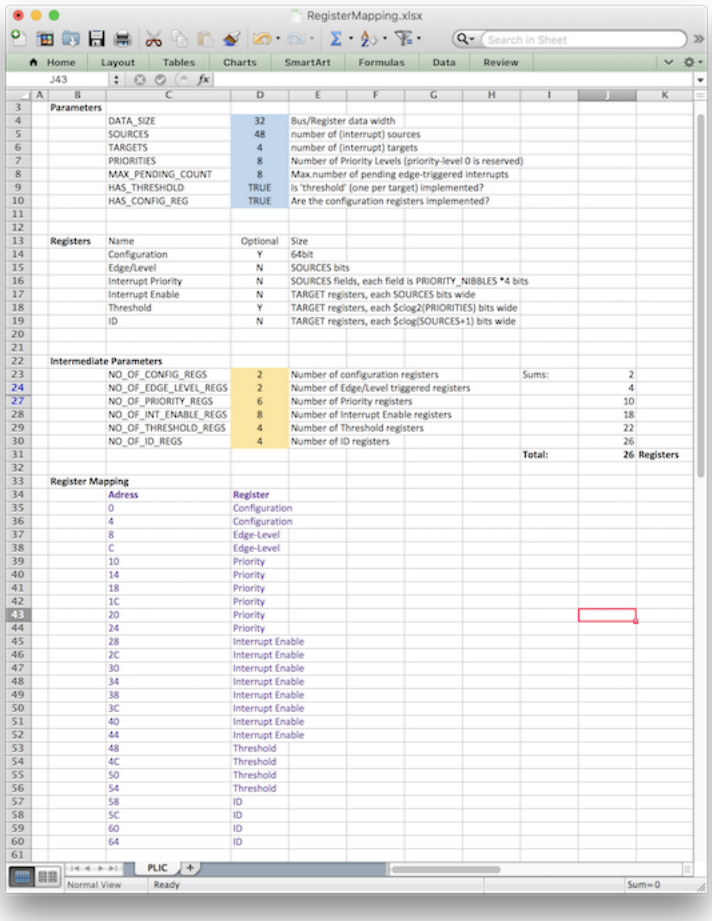
The PLIC is designed to be compliant with the RISC-V Privileged Architecture v1.9.1 and RISC-V User Level ISA v2.2 specifications, as licensed under the Creative Commons Attribution 4.0 International License:

> "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2", Editors Andrew Waterman and Krste Asanović,RISC-V Foundation, May 2017.
>
> "The RISC-VInstruction Set Manual, Volume II: Privileged Architecture, Version 1.9.1",Editors Andrew Waterman and Krste

7 Revision History

Date	Rev.	Comments
	1.0	
**		
**		
**		

Table 15: Revision History



RegisterMapping.xlsx

3	Parameters										
4		DATA_SIZE	32	Bus/Register data width							
5		SOURCES	48	number of (interrupt) sources							
6		TARGETS	4	number of (interrupt) targets							
7		PRIORITIES	8	Number of Priority Levels (priority-level 0 is reserved)							
8		MAX_PENDING_COUNT	8	Max.number of pending edge-triggered interrupts							
9		HAS_THRESHOLD	TRUE	Is 'threshold' (one per target) implemented?							
10		HAS_CONFIG_REG	TRUE	Are the configuration registers implemented?							
11											
12											
13	Registers	Name	Optional	Size							
14		Configuration	Y	64bit							
15		Edge/Level	N	SOURCES bits							
16		Interrupt Priority	N	SOURCES fields, each field is PRIORITY_NIBBLES *4 bits							
17		Interrupt Enable	N	TARGET registers, each SOURCES bits wide							
18		Threshold	Y	TARGET registers, each $clog2(PRIORITIES) bits wide							
19		ID	N	TARGET registers, each $clog($SOURCES+1) bits wide							
20											
21											
22	Intermediate Parameters										
23		NO_OF_CONFIG_REGS	2	Number of configuration registers					Sums:	2	
24		NO_OF_EDGE_LEVEL_REGS	2	Number of Edge/Level triggered registers						4	
27		NO_OF_PRIORITY_REGS	6	Number of Priority registers						10	
28		NO_OF_INT_ENABLE_REGS	8	Number of Interrupt Enable registers						18	
29		NO_OF_THRESHOLD_REGS	4	Number of Threshold registers						22	
30		NO_OF_ID_REGS	4	Number of ID registers						26	
31									Total:	26	Registers
32											
33	Register Mapping										
34		Adress	Register								
35		0	Configuration								
36		4	Configuration								
37		8	Edge-Level								
38		C	Edge-Level								
39		10	Priority								
40		14	Priority								
41		18	Priority								
42		1C	Priority								
43		20	Priority								
44		24	Priority								
45		28	Interrupt Enable								
46		2C	Interrupt Enable								
47		30	Interrupt Enable								
48		34	Interrupt Enable								
49		38	Interrupt Enable								
50		3C	Interrupt Enable								
51		40	Interrupt Enable								
52		44	Interrupt Enable								
53		48	Threshold								
54		4C	Threshold								
55		50	Threshold								
56		54	Threshold								
57		58	ID								
58		5C	ID								
59		60	ID								
60		64	ID								
61											

Figure 4: Register Mapping Worksheet