AiOnIc Specification

AiOnIc TOP

Revision 1.0

13 April 2025

English edition

Table of Contents

1. Overview

1.1. Introduction

- 「**AiOnIc**」 is an optimal solution for embedded applications requiring low cost and high power performance on edge AI chips. **AiOnIc** uses ArchiTek's proprietary **virtual engine architecture**, fusing innovative software flexibility and custom hardware optimized for low power consumption and low latency into a programmable engine for image processing and AI processing. **AiOnIc** provides AI solutions for everything from computer vision to deep learning.

  「**AiOnIc**」 can **dynamically reconfigure hardware components** to flexibly accommodate various algorithms. It combines the advantages of existing CPUs, GPUs, and dedicated LSIs, enabling **real-time concurrent execution of multiple processes**.

  Below is a conceptual diagram:

  You prepare an operation scenario, which the hardware scheduler (pss) then fragments into tasks and automatically executes various transfers and AI processing.

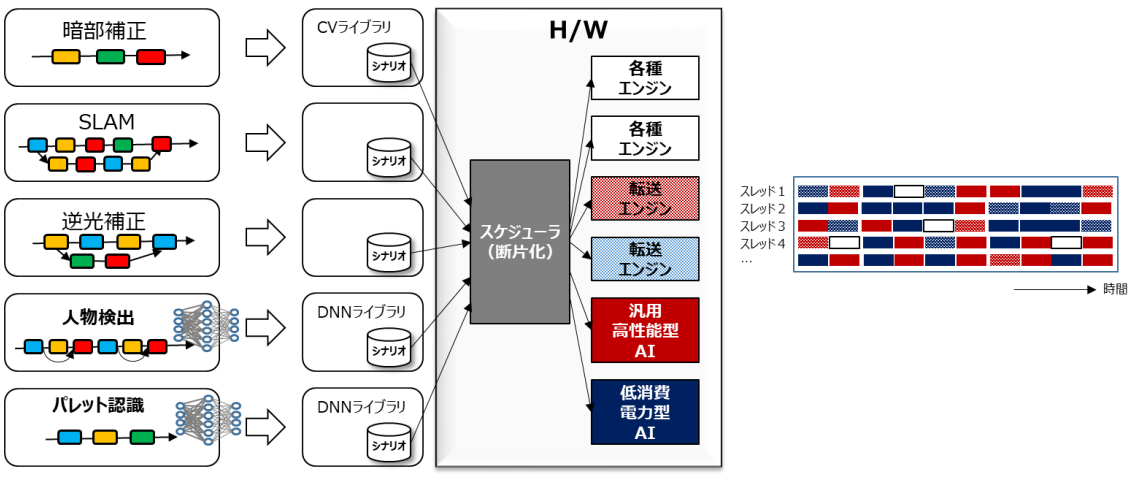

Figure 1 Conceptual diagram

Each hardware component (IP or engine) of the current version is shown below. Hardware components can be easily added, deleted, and replaced by the user.

Note: For details of each hardware component, please refer to the dedicated specification sheet.

IP Name	Description
blt	Compact data transfer engine, mapping
label	Labeling core, connectivity, numbering
frcomp	Image processing engine, image manipulation and transfer
fft	FFT computation
imp	Inverse matrix computation
nnlp	Neural network block
ppUnit	RISC-V processor block: includes **pp**: A SIMD-type high-performance processor **mmap**: Matrix multiply-add processor **dmap**: Transfer block
sigcomp	Digital signal filtering, rate conversion, modulation, mixing
sort	Sorting block
vin	Video input
vout	Video output
xpeg	DCT computation

In addition to the hardware components, AiOnIc contains several control IPs that control it. The control IPs are outlined below.

Note:*Please refer to the specifications for* ***mcxc*** *and* ***pss****, which are especially important for the system architecture.*

IP Name	Description
axiMaster	AXI master block
axiSlave	AXI slave block

IP Name	Description
Mcxc	Memory control block
pss	High-performance scheduler
userCntl	User control (register)
busArb	Bus arbiter
busBridge	Bus bridge (AXI)
busCntl	Bus control

In addition, 8MB of 16-Bank Cache and 1MB of SRAM for GPU processing are implemented.

AiOnIc internal block diagram is shown below.

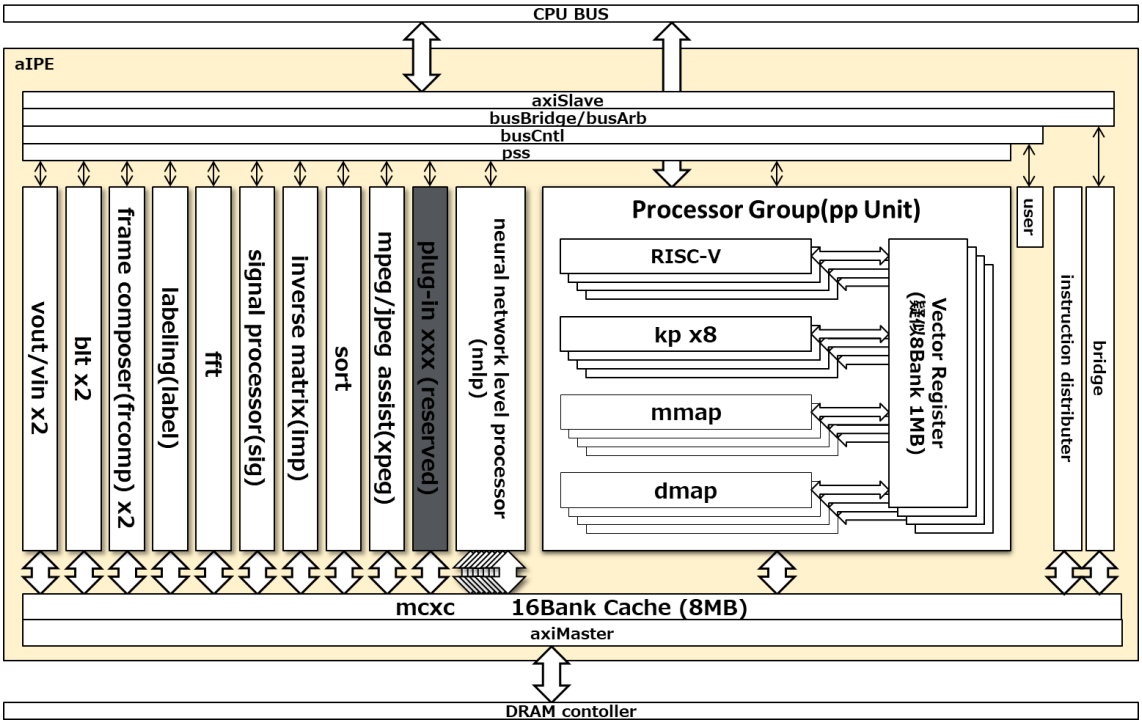

Figure 2 AiOnIc Block Diagram

1.2. Implementation Parameters

Parameter Name	Description	Default Value
UID	user ID	0x71F9
UREV	user revision	0x44DC
CLR	pss logical number radix	6
SLR	scalar length radix	11
LNR	sram line number radix	12
BKR	sram bank number radix	3
SNR	sram word number radix	3
PNR	pp number radix	3
MNR	mmap number radix (must be > 1)	3
DNR	dmap number radix	3
PLR	mcxc bank number radix	4
WYR	mcxc way number radix	2
LLR	mcxc line number radix	10
BLR	mcxc burst number radix	3
MLR	mcxc word number radix	4
AID	AXI ID Length	8
ABD	AXI Burst Definition (AXI3=4, AXI4=8)	8
ENUM	engine number	20

---

2. Signal Lines

2.1. Control Bus Interface

Signal Name	I/O	Pol.	Source	Description
clk	I	+	-	System clock (not gated)
clk_cg	I	+	-	System clock
clk_axi	I	+	-	AXI clock

Signal Name	I/O	Pol.	Source	Description
reset_n	I	–	–	Asynchronous reset0: reset asserted1: reset deasserted
sysIrq[ENUM-1:0]	O	+	clk_cg	Interrupt

2.2. AXI Slave Interface (from CPU BUS)

Signal Name	I/O	Pol.	Source	Description
axiAWburst[1:0]	I	+	clk_axi	AXI Write Burst Type
axiAWlen[ABD-1:0]	I	+	clk_axi	AXI Write Burst Length
axiAWsize[2:0]	I	+	clk_axi	AXI Write Burst Size
axiAWvalid	I	+	clk_axi	AXI Write Address Valid
axiAWready	O	+	clk_axi	AXI Write Address Ready
axiAWaddr[35:0]	I	+	clk_axi	AXI Write Address
axiAWid[AID-1:0]	I	+	clk_axi	AXI Write Address ID
axiWvalid	I	+	clk_axi	AXI Write Valid
axiWready	O	+	clk_axi	AXI Write Ready
axiWid[AID-1:0]	I	+	clk_axi	AXI Write ID
axiWdata[63:0]	I	+	clk_axi	AXI Write Data
axiWstrb[7:0]	I	+	clk_axi	AXI Write Strobe
axiWlast	I	+	clk_axi	AXI Write Data Last
axiARburst[1:0]	I	+	clk_axi	AXI Read Burst Type
axiARlen[ABD-1:0]	I	+	clk_axi	AXI Read Burst Length
axiARsize[2:0]	I	+	clk_axi	AXI Read Burst Size
axiARvalid	I	+	clk_axi	AXI Read Address Valid
axiARready	O	+	clk_axi	AXI Read Address Ready
axiARaddr[35:0]	I	+	clk_axi	AXI Read Address
axiARid[AID-1:0]	I	+	clk_axi	AXI Read Address ID
axiRvalid	O	+	clk_axi	AXI Read Valid
axiRready	I	+	clk_axi	AXI Read Ready

Signal Name	I/O	Pol.	Source	Description
axiRid[AID-1:0]	O	+	clk_axi	AXI Read ID
axiRdata[63:0]	O	+	clk_axi	AXI Read Data
axiRlast	O	+	clk_axi	AXI Read Data Last
axiBid[AID-1:0]	O	+	clk_axi	AXI Write Response ID
axiBresp[1:0]	O	+	clk_axi	AXI Write Response
axiBvalid	O	+	clk_axi	AXI Write Response Valid
axiBready	I	+	clk_axi	AXI Write Response Ready

2.3. AXI Master Interface (to DRAM)

Signal Name	I/O	Pol.	Source	Description
axoAWburst[1:0]	O	+	clk_axi	AXI Write Burst Type
axoAWlen[ABD-1:0]	O	+	clk_axi	AXI Write Burst Length
axoAWsize[2:0]	O	+	clk_axi	AXI Write Burst Size
axoAWvalid	O	+	clk_axi	AXI Write Address Valid
axoAWready	I	+	clk_axi	AXI Write Address Ready
axoAWaddr[31:0]	O	+	clk_axi	AXI Write Address
axoAWid[AID-1:0]	O	+	clk_axi	AXI Write Address ID
axoWvalid	O	+	clk_axi	AXI Write Valid
axoWready	I	+	clk_axi	AXI Write Ready
axoWid[AID-1:0]	O	+	clk_axi	AXI Write ID
axoWdata[MBL-1:0]	O	+	clk_axi	AXI Write Data
axoWstrb[ML-1:0]	O	+	clk_axi	AXI Write Strobe
axoWlast	O	+	clk_axi	AXI Write Data Last
axoARburst[1:0]	O	+	clk_axi	AXI Read Burst Type
axoARlen[ABD-1:0]	O	+	clk_axi	AXI Read Burst Length
axoARsize[2:0]	O	+	clk_axi	AXI Read Burst Size
axoARvalid	O	+	clk_axi	AXI Read Address Valid
axoARready	I	+	clk_axi	AXI Read Address Ready

Signal Name	I/O	Pol.	Source	Description
axoARaddr[31:0]	O	+	clk_axi	AXI Read Address
axoARid[AID-1:0]	O	+	clk_axi	AXI Read Address ID
axoRvalid	I	+	clk_axi	AXI Read Valid
axoRready	O	+	clk_axi	AXI Read Ready
axoRid[AID-1:0]	I	+	clk_axi	AXI Read ID
axoRdata[MBL-1:0]	I	+	clk_axi	AXI Read Data
axoRlast	I	+	clk_axi	AXI Read Data Last
axoBid[AID-1:0]	I	+	clk_axi	AXI Write Response ID
axoBresp[1:0]	I	+	clk_axi	AXI Write Response
axoBvalid	I	+	clk_axi	AXI Write Response Valid
axoBready	O	+	clk_axi	AXI Write Response Ready

2.4. AXI Master Interface (to CPU BUS)

Signal Name	I/O	Pol.	Source	Description
axrAWburst[1:0]	O	+	clk_axi	AXI Write Burst Type
axrAWlen[ABD-1:0]	O	+	clk_axi	AXI Write Burst Length
axrAWsize[2:0]	O	+	clk_axi	AXI Write Burst Size
axrAWvalid	O	+	clk_axi	AXI Write Address Valid
axrAWready	I	+	clk_axi	AXI Write Address Ready
axrAWaddr[31:0]	O	+	clk_axi	AXI Write Address
axrAWid[AID-1:0]	O	+	clk_axi	AXI Write Address ID
axrWvalid	O	+	clk_axi	AXI Write Valid
axrWready	I	+	clk_axi	AXI Write Ready
axrWid[AID-1:0]	O	+	clk_axi	AXI Write ID
axrWdata[31:0]	O	+	clk_axi	AXI Write Data
axrWstrb[3:0]	O	+	clk_axi	AXI Write Strobe
axrWlast	O	+	clk_axi	AXI Write Data Last
axrARburst[1:0]	O	+	clk_axi	AXI Read Burst Type

Signal Name	I/O	Pol.	Source	Description
axrARlen[ABD-1:0]	O	+	clk_axi	AXI Read Burst Length
axrARsize[2:0]	O	+	clk_axi	AXI Read Burst Size
axrARvalid	O	+	clk_axi	AXI Read Address Valid
axrARready	I	+	clk_axi	AXI Read Address Ready
axrARaddr[31:0]	O	+	clk_axi	AXI Read Address
axrARid[AID-1:0]	O	+	clk_axi	AXI Read Address ID
axrRvalid	I	+	clk_axi	AXI Read Valid
axrRready	O	+	clk_axi	AXI Read Ready
axrRid[AID-1:0]	I	+	clk_axi	AXI Read ID
axrRdata[31:0]	I	+	clk_axi	AXI Read Data
axrRlast	I	+	clk_axi	AXI Read Data Last
axrBid[AID-1:0]	I	+	clk_axi	AXI Write Response ID
axrBresp[1:0]	I	+	clk_axi	AXI Write Response
axrBvalid	I	+	clk_axi	AXI Write Response Valid
axrBready	O	+	clk_axi	AXI Write Response Ready

3. Configuration and Operation Description

3.1. Configuration Overview

AiOnIc can be accessed through the CPU BUS. AXI slave accesses the address space inside AiOnIc using a 36-bit address as described below. If the CPU bus is 32-bit, it must be expanded to 36 bits using TLB, etc. If the CPU bus is 32-bit, it must be expanded to 36 bits using TLB or other means. To start each engine, it accesses the Pipeline Slice Scheduler (pss). pss retrieves the necessary context from memory, generates information such as coordinates, and starts each engine.

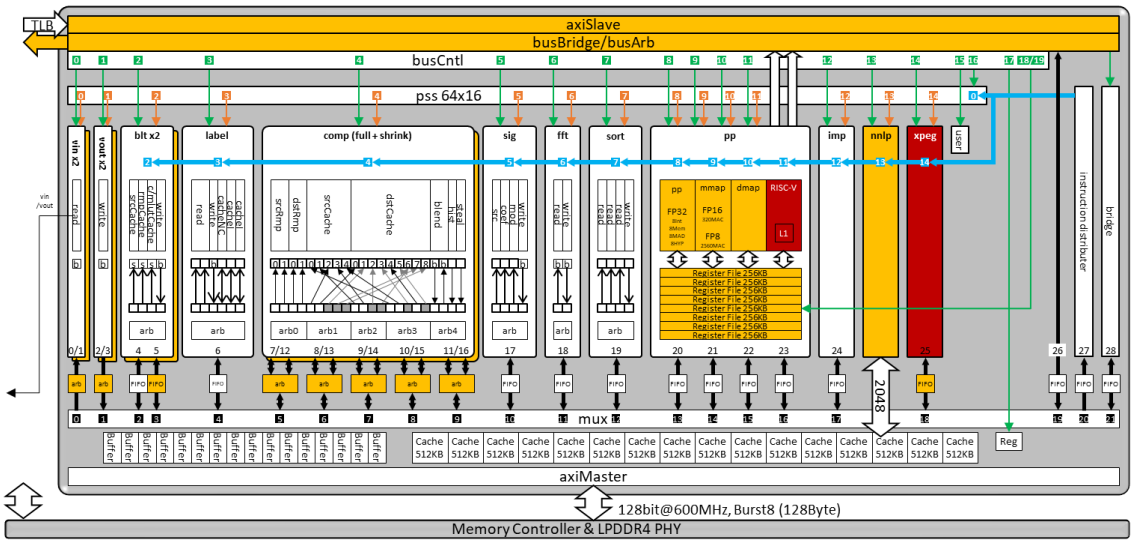

Figure 3 AiOnIc Block Detail

Inside the AiOnIc, each engine's register address map is shown below.

Note: For register details, please see the register specifications of each engine.

bus/irq ch.	addr[31:16]	engine
0	000n	vin 0-3c
1	001n	vout 0-3c
2	002n	blt 0-4
3	0030	label 0-c
4	004n	comp 0-30c
5	0050	sig 0-4
6	0060	fft 0-4
7	0070	sort 0-4
8	0080	pp 0-100
9	0090	mmap 0-7c
10	00a0	dmap 0-4
11	00b0	riscv
12	00c0	imp 0-4
13	00d0	nnlp 0-c
14	00e0	xpeg
15	01d0	user
16	01e0	pss 0-8000
17	01f0	mcxc 0-8000
18	0200	scalar register
19	0400	vector register

※n = 0,1

The "pss" that controls each engine is assigned a physical ch.

Note: For details, please refer to the pss specification.

Below is an example of the current specification assigning channel numbers to pss.

pss physical ch.	engine
0	vin
1	vout
2	blt
3	label
4	comp
5	sig
6	fft
7	sort
8	pp
9	mmap
10	dmap
11	riscv
12	imp
13	nnlp
14	xpeg
15	-

The instruction distributer is assigned the following ch.

instruction ch.	engine
0	pss
1	-
2	blt
3	label
4	comp
5	sig
6	fft
7	sort
8	pp
9	mmap
10	dmap
11	riscv
12	imp
13	nnlp
14	xpeg
15	-

The engines assigned to each mcxc channel are listed below.

mcxc ch.	engine
0	vin
1	vout
2	blt0
3	blt1
4	label
5	comp0
6	comp1
7	comp2
8	comp3
9	comp4
10	sig
11	fft
12	sort
13	pp0
14	pp1
15	pp2
16	pp3
17	imp
18	xpeg
19	cpu
20	arb
21	bridge

In AXI slave, the address bus expects 36-bit input, so if the CPU bus is 32-bit, it must be expanded to 36 bits using TLB, etc. (see CPU Space below). The upper [35:32] bits are the extended address, Mem/Buf selector, and CCF bits..

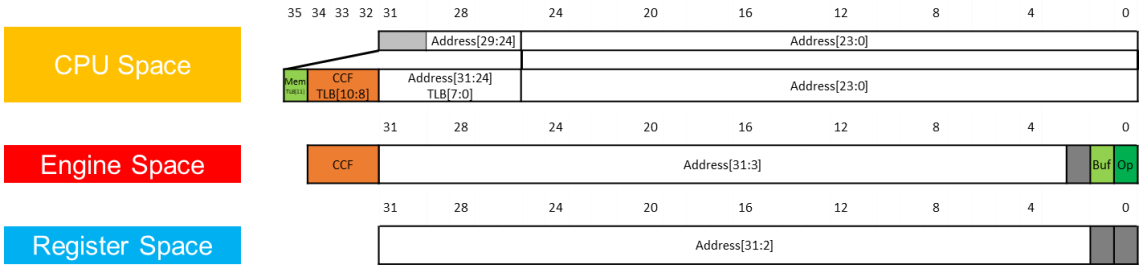

Figure 4 Address Space

See the figure below for Mem/Buf selector and address space selection by CCF bits.

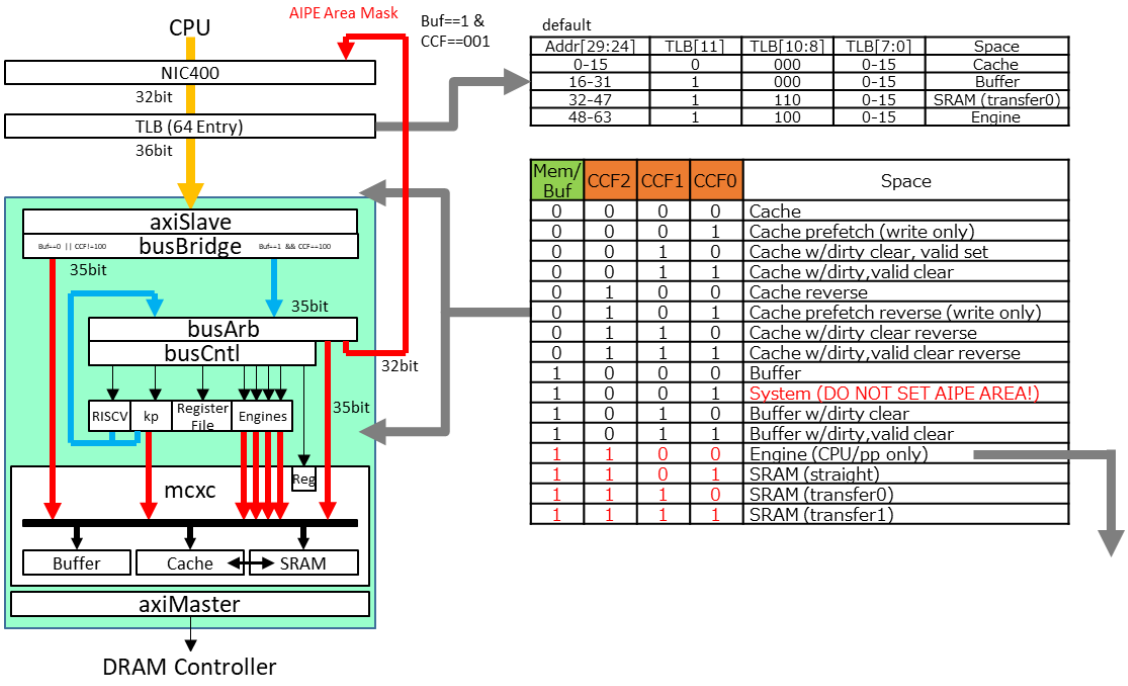

Figure 5 Address Space Control

3.2. Interrupt Overview

The interrupt signal **sysIrq** integrates the interrupts of each IP inside AiOnIc and outputs 20 bits.

The following IPs are assigned to each interrupt number..

irq ch.	engine
0	vin
1	vout
2	blt
3	label
4	comp
5	sig
6	fft
7	sort
8	pp
9	mmap
10	dmap
11	riscv
12	imp
13	nnlp
14	xpeg
15	xxx
16	user (Fix 0)
17	pss
18	mcxc
19	scalar register
20	vector register

---

4. Application Note

4.1. Other Information

No additional content provided in this section.
