**ISP\_Bypass spec**

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Description | Author |
| 0.1 | 2016-12-16 | Initial | Hubio.Zhong |
| 0.2 |  |  |  |
|  |  |  |  |

# IP overview

## Introduce the IP’s features, functions



The audio path is as below shows:

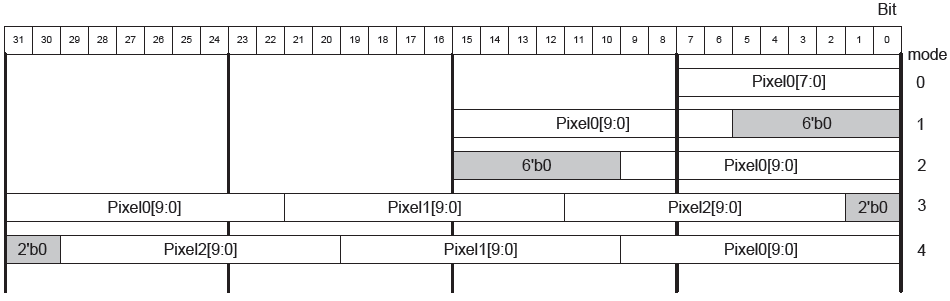


**Sirius Audio data path**

**Function:**

* + 1. Store video for ISP/CODEC
* Mapping for ISP
* Mapping for JPEG
* Mapping for H264
* Mapping for HEVC
* Mapping for Display controller

For CODEC(data is YUV422/420):



Currently we support two mode

DDR’s burst length is 8, data width is 64bit.

ISP\_bypass’ data width is 128 bit, burst is **~~2/~~ 4/8/16**

1. Planar Mode Mapping

Planar mode, For HEVC encoder, Each Y read is 64Byte, and Cr/Cb is 32B, But DDR’s request is 64x8, so Planar mode is not suitable for this design, so Planar mode may be removed.



Note:

1. Y/Cb/Cr Base Addr is configurable by register.
2. Luma/Chroma Stride is configurable by register
3. Cb\_Height = (Sampling mode == 4:2:2 ? Height : Height/2 );

//Cr\_Height = Cb\_Height

Pixel Format:

8bit mode : 1pixel/per byte,

10bit mode: 1pixel/2 bytes // (LSB/MSB 待定)

3pixel/4 bytes // (LSB/MSB 待定)

12bit mode: 1pixel/2 bytes (JPEG only)

1. Cb/Cr Interleave Mode Mapping

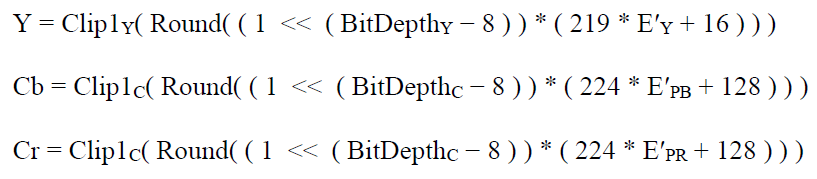


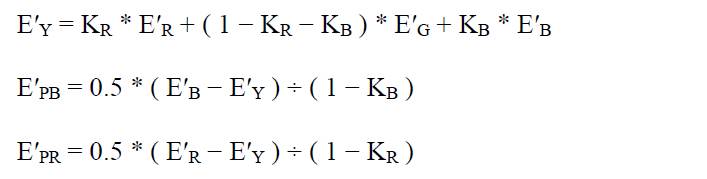
Note:

1. Y/Cb/Cr Base Addr is configurable by register
2. Luma/Chroma Stride is configurable by register
3. Cb\_Height = (Sampling mode == 4:2:2 ? Height : Height/2 ); //Cr\_Height = Cb\_Height

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | For VIDEO CODEC | | | |
|  | Data | 128bit | Line step | Frame step |
| Planar mode | 8bit Y or 444 U/V | 16pixel | ((Pic\_width+63)  >>6) <<6) | Line step \* pic\_height |
| 10bit Y | 12pixel+8bitdummy | ((Pic\_width\*10/8+63)  >>6) <<6) | Line step \* pic\_height |
| 12bit Y | 10pixel+8bitdummy | ((Pic\_width\*12/8+63)  >>6) <<6) | Line step \* pic\_height |
| 8bit 422 UV | 16pixel | ((picture\_width/2+63)>>6)<<6 | Line step \* pic\_height |
| 10bit 422 UV | 12pixel+8bitdummy | ((picture\_width\*12/16/2 +63)>>6)<<6 | Line step \* pic\_height |
| 12bit 422 UV | 10pixel+8bitdummy | ((picture\_width\*10/16/2 +63)>>6)<<6 | Line step \* pic\_height |
| 8bit 420 UV | 16pixel | ((picture\_width/2+63)>>6)<<6 | Line step \* pic\_height |
| 10bit 420 UV | 12pixel+8bitdummy | ((picture\_width\*12/16/2 +63)>>6)<<6 | Line step \* pic\_height/2 |
| 12bit 420 UV | 10pixel+8bitdummy | ((picture\_width\*10/16/2 +63)>>6)<<6 | Line step \* pic\_height/2 |
| Interleave  mode | 8bit Y | 16pixel | ((Pic\_width+63)  >>6) <<6) | Line step \* pic\_height |
| 444 UV | 16pixel | ((Pic\_width\*2+63)  >>6) <<6) | Line step \* pic\_height |
| 10bit Y | 12pixel+8bitdummy | ((Pic\_width\*10/8+63)  >>6) <<6) | Line step \* pic\_height |
| 12bit Y | 10pixel+8bitdummy | ((Pic\_width\*12/8+63)  >>6) <<6) | Line step \* pic\_height |
| 8bit 422 UV | 16pixel | ((2\*picture\_width/2+63)>>6)<<6 | Line step \* pic\_height |
| 10bit 422 UV | 12pixel+8bitdummy | ((2\*picture\_width\*12/16/2 +63)>>6)<<6 | Line step \* pic\_height |
| 12bit 422 UV | 10pixel+8bitdummy | ((2\*picture\_width\*10/16/2 +63)>>6)<<6 | Line step \* pic\_height |
| 8bit 420 UV | 16pixel | ((2\*picture\_width/2+63)>>6)<<6 | Line step \* pic\_height |
| 10bit 420 UV | 12pixel+8bitdummy | ((2\*picture\_width\*12/16/2 +63)>>6)<<6 | Line step \* pic\_height/2 |
| 12bit 420 UV | 10pixel+8bitdummy | ((2\*picture\_width\*10/16/2 +63)>>6)<<6 | Line step \* pic\_height/2 |
|  |  |  |  |
| **FOR ISP** | | | | |
|  | Data | 128bit | Line step | Frame step |
|  | Raw6 | 21pixel+2bitdummy | ((Pic\_width\*6/8+63)  >>6) <<6) | Line step \* pic\_height |
|  | Raw7 | 18pixel+2bitdummy | ((Pic\_width\*7/8+63)  >>6) <<6) | Line step \* pic\_height |
|  | Raw8 | 16pixel | ((Pic\_width+63)  >>6) <<6) | Line step \* pic\_height |
|  | Raw10 | 12pixel+8bitdummy | ((Pic\_width\*10/8+63)  >>6) <<6) | Line step \* pic\_height |
|  | Raw12 | 10pixel+8bitdummy | ((Pic\_width\*12/8+63)  >>6) <<6) | Line step \* pic\_height |
|  | Raw14 | 9pixel+2bitdummy | ((Pic\_width\*14/8+63)  >>6) <<6) | Line step \* pic\_height |
|  | RGB444 | Same as YUV444 8bit |  |  |
|  | RGB555 | Same as YUV444 8bit |  |  |
|  | RGB565 | Same as YUV444 8bit |  |  |
|  | RGB666 | Same as YUV444 8bit |  |  |
|  | RGB8 | Same as YUV444 8bit | | |
|  | RGB10 | Same as YUV444 10bit | | |
|  | RGB12 | Same as YUV444 12bit | | |
|  |  |  |  |  |
|  |  |  |  |  |

* + 1. Store Audio data for audio\_subsystem
* Audio
* Packet
  + 1. Simple color space convert according requirement:
* YUV444🡪YUV420
* YUV444🡪YUV422
* RGB2YUV





Where E’Y,E’R, E’G, E’B are real numbers with values in the range of 0 to 1 inclusive ([0,1]), and E’PB and E’PR are in the range of -0.5 to 0.5 inclusive([-0.5,0.5]).

The variable KR, KB is configurable by register . The value of KR, KB isdifferent by selecting BT.601/BT.709/BT.2020 or other standards .

* + 1. **~~Bypass the video source to Display Port.~~**
* **~~No need to store in DDR, just some BT656 interface~~** (TBD)
  + 1. ~~Bypass the video to PCIE (AXI interface)~~
* ~~Mapping~~

Clock reset.

Debug bus.

System interrupt:

* For sync between video and audio stream
* For video low delay encoding.

Ipu\_end\_of\_low (end of line, pulse)

Ipu\_new\_frame (start of frame, level, 0 or 1)

Ipu\_current\_buffer[2:0] ( buffer state, each bit denotes current buffer, set by bypass, clear or overwrite by CODEC )

* Error state detect.

## Block diagram



其中MEM 采用单块sram结构实现，容量为768x128。

其中地址A~C存储Y分量，地址C~E存储U分量，E~G存储V分量。

以Y分量来说。

A~B存储view0

B~C 存储view1.

A~G的值均可以通过寄存器配置值，可以根据视频分辩率的不同来分配存储空间。



# Hardware interface description

## Input/output signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port name | Input/output | Width  (bit) | Clock (domain) | Description |
| **System signal** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Interface with Line buffer** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| HDMI interface | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| NOC (AXI interface) | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## More description on the input/output signals

### Interface protocol

### Requirements on the signals (clock, reset, …)

RESETN: it is asynchronous reset signal for this module.

CLK: it can be gated by ACTIVE signal to save power.

# Software interface description

## Register definition

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register name | width | offset | R/W | description |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Memory requirement

(With below description for each UMAC agent or PMAN node)

## Memory space requirement

## Memory bandwidth requirement in each use case scenario

### Memory access pattern (burst characteristics)

### Memory latency requirement (consequence if the latency is not guaranteed)

### Peak bandwidth and average bandwidth requirement

# Power saving modes

## Clock gating/throttling, automatic or register controllable.

## Clock frequency per use case

.

## Power down control for ROM/RAM/analog macros.

No.

## Power Island (if applicable)

No.

# Synthesis tips

## Suggestions for Placement in terms of performance, routing, timing, etc.

## Suggestions for Routing

## Suggestions for clock tree

## Information on LVT/SVT ratio to achieve timing closure

## Information on gate count, timing constraints

# DFT/test

## Test scheme for each of ROM/RAM/OTP/analog macros

### Input/output signal requirement

### Register setting

## Test scheme for special logic (asynchronous logic, clock generation logic, reset logic, etc)

# Performance tuning, IP profiling and debugging

# ROM/RAM/analog macro spec (requirement)

## Macro list with detail requirement

## Interface description (waveform)

# I/O pad requirement

## Clock I/O

## Fast speed I/O

## I/O with different power (other than 3.3V digital and DDR I/O)