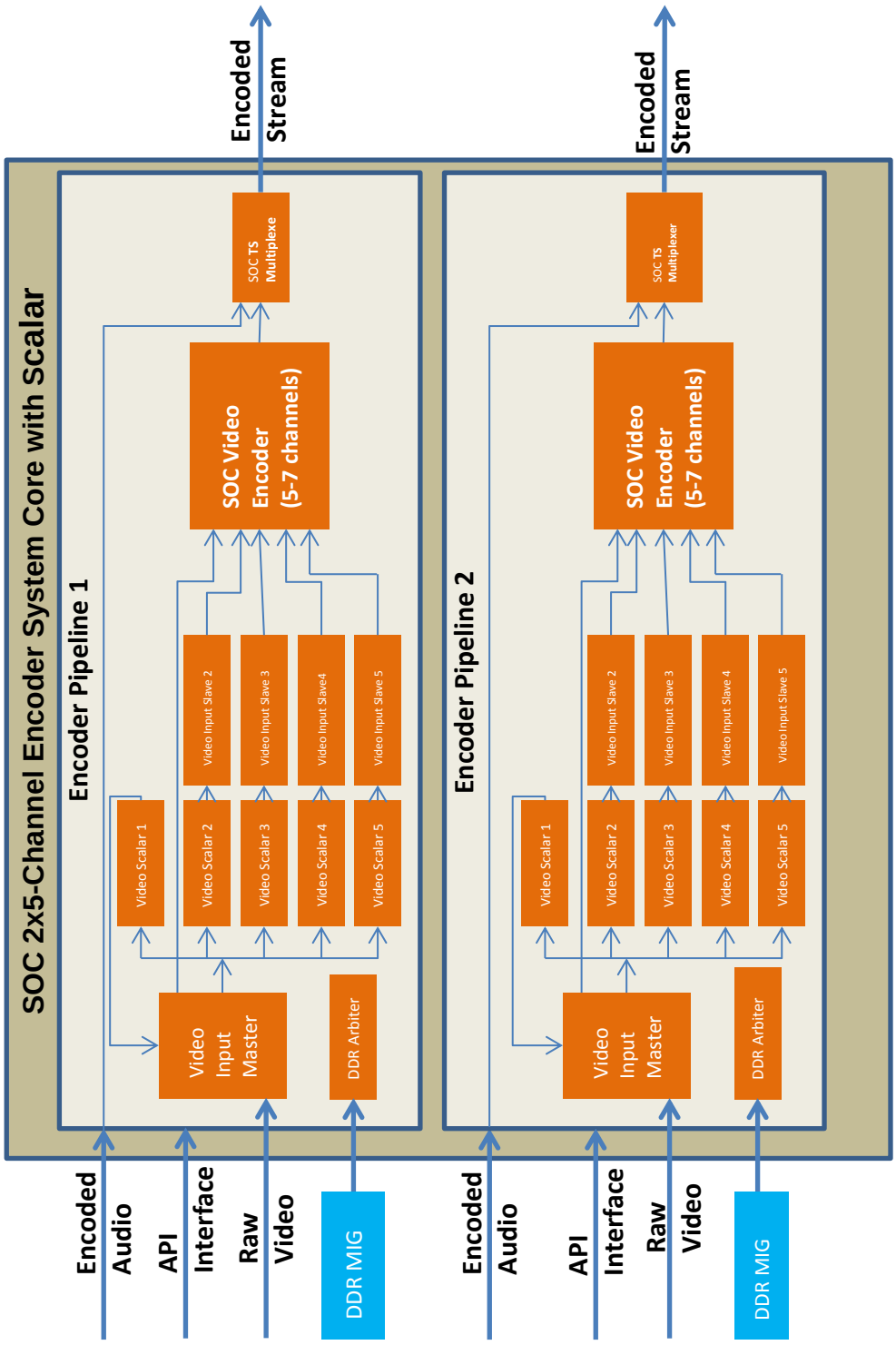
2x5-Channel Encoder System Core

With Scalar

System Level API Specification

1. Hardware API interfaces

The diagram of the encoder core is as following:



As shown in above diagram, the SOC 2x5-Channel Encoder System Core includes two exactly same hardware encoder pipeline copies sitting in the core, each of which is responsible to scale and encode 5 video channels. The two pipelines work in parallel. Therefore the system core is able to scale and encode 10 (2x5) channels totally.

The two pipelines of the core are exactly identical. Within each of the two pipelines, it takes one raw video input, after which there are 5 scalar modules work in parallel and one H264 encoder core works in time-shared mode to encode the 5 scaled video channels.

For each pipeline, there is one API interface to control and retrieve information from the pipeline. So there are two API interfaces for the core ports. For the API signals detail, please reference the integration sheet of the core.

The two encoder pipelines work independently and the two API interfaces work independently as well.

In the remaining part of this document, all description target only one of the two API interfaces while the other are exactly same.

2. Important rule for genera API access

Any bit in any register that is not listed in any published document are reserved by SOC internal use for purpose such as SOC debug or future extension. The client MUST not use them in any way. If the user failed to follow this rule, the behavior of the core is not defined and the results are neither protected nor supported by SOC.

3. API channel selection

Within each pipeline, there are 256 physical API addresses to access directly. In this document, the term "register" is refer to an API register. In order to access 5 channels with the only one API interface, there is a special channel select register at address 0x92, which is used to select the channel to access with other API addresses. In this document, the channel selection register is referred as **channel selector**.

All other addresses are for the selected channel by channel selector register 0x92. The channel selector register 0x92 is listed in Table-1 below.

Table-1: Encoder System Configuration

Addr (HEX)	Information	Access
92	Bit[2:0]: Multi-Channel/Time Sharing API Core Selection - 0 – All-channel mode (for write only). - 1-7 = One-channel mode(for read and write) - Note only 1 core can be read at a time. A value of 0 should not be used for reading registers. - This register is also used for indexing sub-API blocks such as Video Input, H264 Encoder	R/W

There are two access modes for channel selection:
- One-channel mode
- All-channel mode.

- One-channel mode

With one channel mode, only the registers in the selected one channel is accessed.
The steps to access registers in one channel are:

Step 1: Write the channel index into register 0x92 to select channel. The channel index is from 1 to for this version.
Step 2: Access (read or write) the expected registers in the selected channel.

For example:
Write 1 to register 0x92
Write 100 to register 0x97
With above two steps, the target bit-rate of channel 1 of the pipeline will be modified to 10.0Mbps.

- All-channel mode

With all-channel mode, the user is able to modify the specified register in all channels within the pipeline. The limitation of this mode is only write access is possible.

Step 1: Write the value 0 into register 0x92 to select all channels.
Step 2: Write the expected value to expected register address to modify all registers with same address in all channels.

For example:

Write 0 to register 0x92
Write 100 to register 0x97

With above two steps, the target bit-rate for every channel within the pipeline will be modified to 10.0Mbps.

4. API Address Segments Map

Except the channel selection register (0x92), all other registers are mapped into several segments. The segments are listed in below Table-2.

Table-2: API Address Segments Map

Address Range (HEX)	Sub Core	Separate Document	Note
0x00 to 0x7F	H264 Encoder Core	Yes	1,2
0x80 to 0x8F	Video Input and scalar module	Yes	1,3
0xB0 to 0xB5	DDR Arbiter	No	4
Others (except 0x92)	System Level	No	5

Note:

1. The regarding API manuals describe only one channel and the channel is selected by the channel selector register 0x92.
2. The H264 Encoder API is described in another document:
   H264 Encoder_API_vx_y.pdf
   (vx_y is the version x.y, same as other documents)
3. The video input and scalar module API is described in another document:
   Video Input Core with Scalar API Specification_vx_y.pdf
4. The DDR arbiter is not affected by the channel selector because there is only one arbiter within one pipeline. All other modules and channels share the one arbiter to access the DDR.
5. The system level uses all the registers that are not used by any sub core except channel selector 0x92.

5. Encoder System Level API registers description

The system level registers are listed in Table-3.

Table-3: Encoder System Level API registers

Addr (HEX)	Information	Access
The following registers are unique to each time-shared core (when used)		
94	Bit[15:8]: Real-Time Encoding Frame Rate Bit[7:0]: Target Encoding Frame Rate	R
95	Real-Time bit rate. Bit[15:4]: Decimal Bit Rate (Mbps) Bit[3:0]: Fractional Bit Rate (0.x Mbps)	R
96	Bit[15:0]: Presentation Latency / 2.84ms - Example: 0x1E = 85.2ms (=30x2.84ms)	R/W
97	Bit[9:0]: Target Bit Rate Value / 10 (Mbps) - Example: 150(decimal) = 15Mbps	R/W
9E	Bit[9:0]: Constant Target Bit Rate Value / 10 (Mbps) - Example: 150(decimal) = 15Mbps Bit[15]: Enabled NULL packet insertion. - When the target bitrate drops below the threshold, NULL transport packets will be inserted if enabled.	R/W
The following registers are shared between all channels even when muli/time-sharing cores are used		
9D	Bit[31]: Time limited Core Expired Bit[16]: Authentication Passed Bit[15]: EEPROM passed Bit[14]: DNA Passed Bit[13]: User TX Ready (reflect the port tx_rdy of the core) Bit[11]: Output Buffer Ready Bit[9]: DDR Fifo Buffer Present Bit[8]: EEPROM Authentication Present Bit[7]: DNA Authentication Present Bit[0]: Bypass Transport Stream Encoder* (W1T, Note 3)	R/W*
9F	Bit[15:0]: Signed Audio Offset from Video Frames	R/W
E0	Bit[5]: Final Encoder Reset Bit[2]: WatchDog Reset Bit[1]: Enable WatchDog * (Note 1) Bit[0]: API Controlled Reset* (Note 2)	R/W*
E1	Bit[3]: Disable Video (W1T, Note 3)	R/W

	Bit[2]: Disable Audio (W1T, Note 3)	
E5	Bit[7:0]: DSP Audio Stream Type	R
E6	Bit[31:0]: Encoder Core Build Data	R
E7	Bit[31:0]: Encoder Core Time Stamp	R

Note 1:

There is a watchdog to monitor the encoder pipeline running. The watchdog monitors the output data. The watchdog resets the pipeline if there is not any data produced by the encoder pipeline within 100 milliseconds.

To enable watchdog, write 0x4320 to register 0xE0.

To disable watchdog, write 0x4321 to register 0xE0.

Note 2:

The user is able to reset the encoder pipeline through API. The user API reset command will produce a reset signal for 0.5 seconds.

To issue an API reset command, write 0x55AA to register 0xE0.

Note 3:

W1T is short for "write 1 to toggle". For all the bits with "W1T", to change the regarding bit, write a logic '1' to that bit. The logic '0' on that bit of the write value does not change the regarding bit.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/06/2015	Initial Version	1.0