WAVE420L HEVC Codec IP

Datasheet

Version 1.8.14

WAVE420L HEVC Codec IP: Datasheet

Version 1.8.14

Revision History

Date	Revision	Change
2021/7/30	1.8.14	Release version

Proprietary Notice

Address and Phone Number

Chips&Media
NC Tower1, 7~8th FL, 509, Teheran-ro, Gangnam-gu, 125-880
Seoul, Korea
Tel: +82-2-568-3767
Fax: +82-2-568-3768

Homepage: *http://www.chipsnmedia.com*

Table of Contents

List of Figures

List of Tables

Preface

This preface introduces the WAVE420L Datasheet and its reference documentation. It contains the following sections:

1. About This Document

This document is the datasheet for WAVE420L Video Codec IP.

1.1. Intended Audience

This document has been written for experienced hardware engineers who want to integrate the video processing unit(VPU) into their SoC and to learn about how it works at system point of view.

1.2. Document Scope

This document describes mainly the supported features, system interface and required memory resources of VPU. It covers architecture description of IP, I/O ports and bus interface for hardware integration, video data flow from/to the IP, and optional display interface.

1.3. Typographical Conventions

The following typographical conventions are used in this document:

bold	Highlights signal names within text, and interface elements such as menu names. May also be used for emphasis in descriptive lists where appropriate.
italic	Highlights cross-references in blue, file names, and citations.
`typewriter`	Denotes example source codes and dumped character or text.

2. Further Reading

This section lists documents which are related to this product.

2.1. Other Documents

- *WAVE420L Verification Guide*
- *WAVE420L Programmer's Guide*
- *WAVE420L API Reference Manual*

Glossary

1080p	Progressive high-definition resolution of 1920 x 1080 pixels
2160p	Progressive resolution of 3849 x 2160 pixels
4k	Progressive resolution of 4096 x 2304 pixels
4:2:0	YCbCr sampling format, where Cb and Cr components are subsampled by two both horizontally and vertically
ACLK	Clock source for AXI bus
AHB	AMBA advanced high-performance bus protocol specification
AIR	Adaptive Intra Refresh
AMBA	Advanced Microcontroller Bus Architecture.
APB	Advanced Peripheral Bus - ARM open standard for peripheral buses. APB is used for host processor to communication with VPU as their interface protocol.
API	Application Programming Interface
AXI	AMBA Advanced eXtensible interface. ARM open standard for high-performance
BCLK	Clock source for BPU in V-CORE
BL	Burst Length
BPU	Bit Processor Unit composing of a highly optimized 16-bit fixed-point DSP for handling bitstream and controlling V-CORE to accelerate video codec processing.
BW	Bandwidth
BWB	Burst Write Back. BWB is a hardware module that collects write data and sends 8 bursts of data to the external memory. It aims to increase bus efficiency by reducing a lot of short burst accesses that happen from VPU due to CTU-based processing nature.
WTL	Write To Linear. This is an optional hardware module that allows a reconstructed output to be written in linear frame as well as in compressed frame. (if WTL is disabled, VPU writes only compressed frame for less bandwidth). WTL writes linear frame data by using BWB module.
CABAC	Context-Adaptive Binary Arithmetic Coding
CCLK	Clock source for VCE in V-CORE
CD	Compressed Data
CIF	Common Intermediate Format, a 352x288 pixel of image size
C&M	Chips&Media
CODEC	Encoder and decoder of digital signal or stream
ColMV	Collocated Motion Vector

CPB	Coded Picture Buffer
CTU	Coding Tree Unit
CU	Coding Unit
DCT	Discrete Cosine Transform
DFT	Design For Testability
DHT	Define-Huffman-Tables marker
DMA	Direct Memory Access unit
DMAC	Direct Memory Access Controller
DPB	Decoded Picture Buffer
DPCM	Differential Pulse Code Modulation
DQT	Define Quantization Table
DSC	Digital Still Camera
DSP	Digital Signal Processor
FBC	Frame Buffer Compressor for hardware block
FBD	Frame Buffer Decompressor for hardware block
FPS	Frame Per Second
FW	Firmware
GALS	Globally Asynchronous Locally Synchronous
GDI	General DMA Interface
HD	High Definition
HEVC	High Efficiency Video Coding
Host processor	This is host CPU which gives video commands to VPU such as picture decoding or picture encoding through API or host interface registers.
HW	Hardware
I/F	Interface
IP	Intellectual property indicating HEVC encoder/decoder IP
IP	Intra Prediction
IPU	Image processing unit which behaves like image signal processor (ISP)
ISO/IEC	International organization for standardization (ISO)/international electrotechnical commission (IEC)
KB	Kilo Byte
LF	Loop Filter

LSB	Least Significant Bit
MB	Macro Block
MSB	Most Significant Bit
MC	Motion Compensator
MIB	Memory Interface Block
MG	Memory Group
MV	Motion Vector
MVcol	Collocated motion vector information
NB	Neighbor Block
OT	Offset Table
PCLK	Clock sources for APB bus
PMU	Power Management Unit
PPS	Picture Parameter Set
PRI	Primary AXI
PRP	Name of hardware unit for pre-processing
PPU	Name of hardware unit for post-processing
PU	Prediction unit
QCIF	Quarter CIF, an 176x144 pixel of image size
QP	Quantization Parameter
RTL	Register Transfer Level
R/W	Read/Write
SAO	Sample Adaptive Offset
SEC	Secondary AXI
SEI	Supplemental Enhancement Information
SEQ	Sequence of video
SH	Slice Header
SPP	Stream Pumping Processor
SPS	Sequence Parameter Set
SoC	System on Chip
TQ	Transform/Quantization
TU	Transform Unit

UART	Universal Asynchronous Receiver/Transmitter
UHD	Ultra High Definition, 3840x2160
V-CORE	A group of video codec hardware blocks that perform a series of slice data processing -
V-CPU	V-CPU is a 32-bit processor that is the top layer of VPU hierarchical architecture. It is responsible for parsing bitstream syntax from sequence to slice header unit, controlling the underlying video hardware blocks called V-CORE.
VCE	Video Coding Engine, a group of video codec hardware blocks
VLC	Variable Length Code
VLD	Variable Length Decoder
VPU	Video Processing Unit. It stands for the Chips&Media's video IP core in this document. It is mainly composed of V-CORE and V-CPU.
VUI	Video Usability Information
WPP	wavefront parallel processing
3DNR	3D Noise Reduction

Chapter 1
OVERVIEW

1.1. Introduction

Chips&Media WAVE420L(hereinafter referred to as "VPU") is low-cost H.265/HEVC hardware encoder IP that is capable of encoding FHD H.265/HEVC main profile L4.1. This IP product is targeted for middle to low-end mobile application processor (AP), IP cameras/sports camcorders and other SoCs with small-sized area and low-cost system requirements.

A single VPU is able to encode any resolution up to 8192x4096. It guarantees real-time performance for encoding 2K 60fps based on its sophisticated, latency tolerant hardware architecture. To meet our SoC customers' needs, VPU can be highly optimized for memory bandwidth loading and excellent power management.

VPU contains a 32-bit processor called V-CPU, which is responsible for parsing bitstream syntax in decoder or encoding bitstream syntax in encoder from sequence to slice header unit, prescanning slice data, controlling the underlying video hardware blocks called V-CORE, and communicating with host CPU through host register interface. The V-CORE performs actual processing of coded slice data including mode decision, motion estimation/compensation, transform/quantization, entropy encoding, and loop filtering. This architecture combined with software and hardware can provide flexibility and high throughput at the same time.

It is easy for VPU to be integrated into a SoC. Because it can be connected through the industry standard interfaces: 32-bit AMBA3 APB bus for host CPU system control and 128-bit AMBA3 AXI for data transfer. There are two 128-bit AXI buses available: one for accessing external memory and the other for on-chip SRAM memory.

1.2. Features

1.2.1. Performance

H.265/HEVC Encoder

- Capable of encoding HEVC Main Profile @ L4.1 High-tier
  - Max resolution [1]: 8192x4096
  - Min resolution: 256x128
- Constraints
  - A picture width shall be multiple of 8.
  - A picture height shall be multiple of 8.

Table 1.1. HEVC Encoder Performance

PicWidth	PicHeight	fps	MHz
3840	2160	15	350
1920	1080	60	350
1920	1080	30	200
1280	720	30	100

[1]The max resolution means the largest image size that the hardware is able to process with. Performance is not considered in this term.

1.2.2. Codec related features

VPU is fully compatible with ISO/IEC 23008-2 High Efficiency Video Coding Main Profile . All coding tools in the profile are supported.

H.265/HEVC Encoder

- I/P slices
- CTU64
  - Supported Prediction Unit(PU) size : 16x16, 8x8
  - Supported Transform Unit(TU) size : 16x16 to 8x8
- Parallel tools
  - Multi slice : Independent slice segment and dependent slice segment
- High performance CABAC encoding
- In-loop deblocking filtering
- Loop filtering across slice
- Lossless coding
- Noise reduction
- Rate Control
  - VBR, CBR and ABR
  - ROI support

1.2.3. Non codec related features

Frame Buffer Compression

To overcome bandwidth suffering and at the same time to ensure real-time encode/decode performance, we have carefully designed lossless, great access patterned frame buffer compression technology and employed it into the WAVE IP series. WAVE420L achieves significant reduction in bandwidth while sustaining fast processing capability.

3DNR

VPU employs temporal noise reduction technology also called 3DNR (3D Noise Reduction) for better quality of image under low light. Video noise appears easily in a low light level. It is an important issue in surveillance camera, since images are captured often under dark condition or indoor and such can make interest of object or person hard to be identified.

VPU is able to detect and filter noise for each color component Y, Cb, and Cr of a frame. Not only good video quality but also enhancement of coding efficiency can be achieved. Also noise reduction is done while encoding so additional read/write bandwidth is not required.

Latency Tolerance

VPU can afford to reach real time performance under memory access latency as long as the number of cycle per CTU retains less than 500 cycles. VPU is designed to be less sensitive to pipeline delay especially at a peak bitrate which might eventually incur performance drop. With use of decoupling technique and inter-pipe queues, it can hide this sort of delay and deliver high performance at any situation.

Programmability

VPU embeds the 32-bit CPU and 16-bit DSP dedicated to bitstream processing and their video hardware control. Host interface registers and interrupt are provided for communication between a host processor and VPU.

Low Power Consumption

VPU can make sure ultra-low power operation by gating the clock sources for some internal hardware blocks in an idle state.

Frame-based processing

V-CPU processor operates video operation on a frame by frame basis. While frame operation is running, there is no need for communication between host processor and VPU. This is the key feature for promising the lowest burden to host processor for video operations.

By just issuing a picture processing, host application can perform its own operations until it receives an interrupt from VPU informing completion of picture processing.

Handling multi-instances

VPU supports multiple instances which can be very helpful for multi-channel encoder and/or decoder applications. In order to support this multi-instance operation, VPU uses an internal context parameter set for each instance. While creating a new instance and starting a picture processing, a set of these context parameters is created and updated automatically within VPU. Because of this internal context management scheme, different encoder and/or decoder tasks running on host processor can control VPU operations independently with their own instances.

When creating a new instance, application task will get a new handle specifying an instance if a new handle is available on VPU. And all the following operations for the given application task could be separately handled on VPU by using this task-specific handle. Since VPU can only perform one picture processing task at a time, each application task should check whether VPU is ready or not before starting a new picture operation. By calling a function for closing a certain instance, application can easily terminate a single task of video operation on VPU.

1.3. Block Diagram

Figure 1.1, "Block Diagram of VPU" shows the hierarchical architecture of VPU and external bus interfaces.

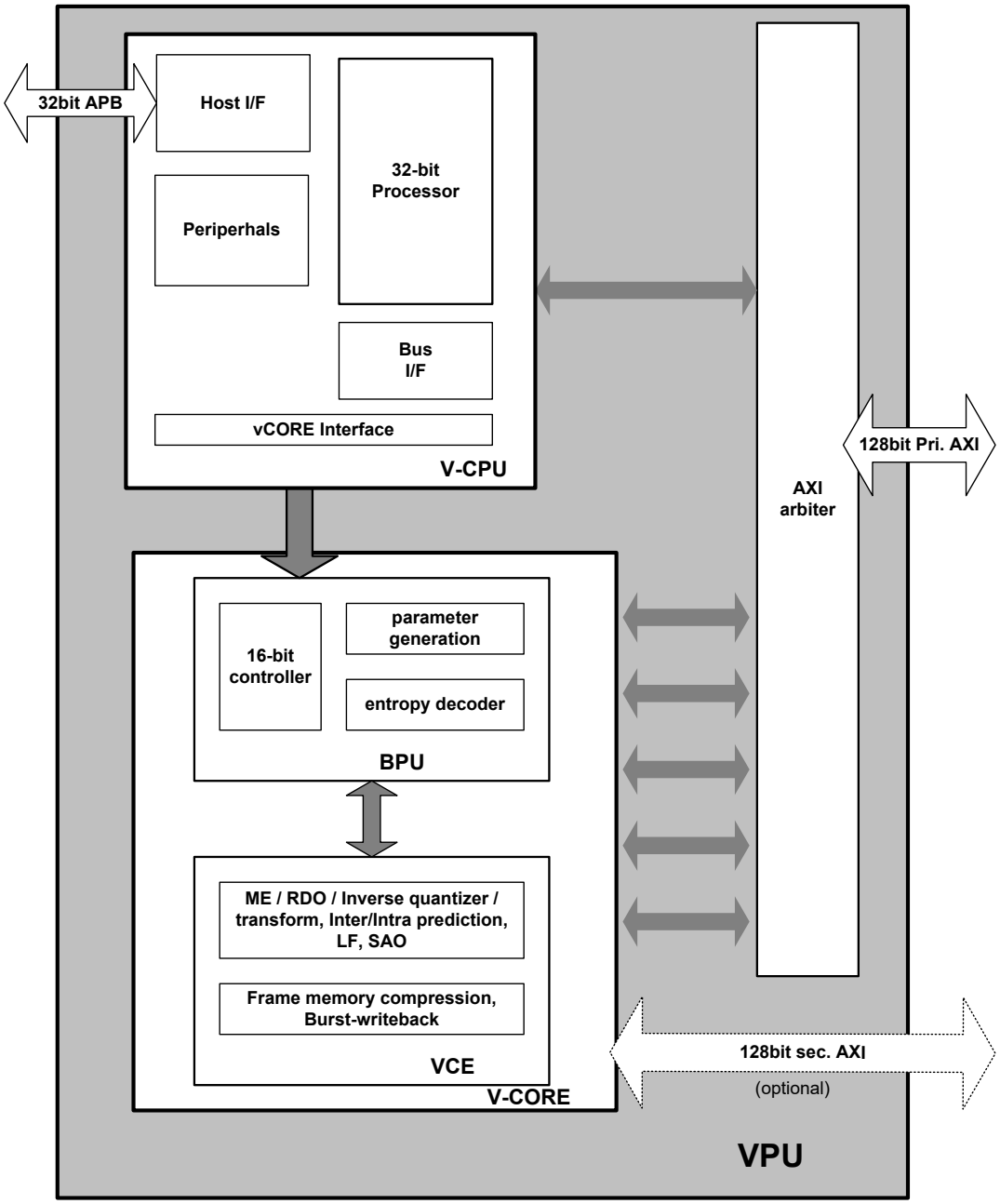


Figure 1.1. Block Diagram of VPU

VPU is built on layer structure for efficient execution of video processing. VPU can be partitioned into two main parts, V-CPU and V-CORE. V-CPU is a 32-bit processor where the VPU L1 Driver Software running on. It mainly does encode and decode a high level syntax of bitstream - SPS/PPS/SH, and communicate with host processor such as receiving a picture level command or sending the result of video task through the 32-bit AMBA3 APB bus . V-CPU can also parse necessary parameters and give necessary slice/tile-level information to the underlying V-CORE hardware.

V-CORE can encode and decode a slice unit of data. It consists of a 16-bit DSP called BPU (BIT Processor Unit) and a group of video codec hardware blocks called VCE (Video Codec Engine) .

- BPU is responsible for packing/ parsing of coded slice data, CTU/CU/PU/TU-level parameter derivation, and finally passing slice data over to the VCE. In order to speed up entropy encoding and decoding , some hardware accelerators are included in the BPU.

- On the other hand, VCE is an actual hardware core executing a series of encoding process motion estimation, intra-prediction, inter-prediction, transformation/quantization, and loop filter with their hardwired logics.

They are doing these tasks with CTU-based pipelines and FIFO queues to guarantee real-time speed and performance.

When it comes to system connection as shown in the *Figure 1.1, "Block Diagram of VPU"*, there is one 32-bit AMBA3 APB interface connecting to host processor, and there are 128-bit AMBA3 AXI bus interfaces connecting to external memory controller for reading and writing picture data and temporal data.

- Primary AXI: to access code, work data, coded picture(bitstream), decoded picture data and so forth
- Secondary AXI(optional): to connect on-chip-SRAM to alleviate required bandwidth by storing temporal data in it

Note For details about temporal data through secondary AXI, please see the *Section 3.3, "Optional SRAM on Secondary AXI"*.

Chapter 2
SYSTEM INTERFACE

This chapter presents a list of interface signals with some note for careful use and describes the VPU clock gating and reset scheme and bus interfaces. Those are useful information when integrating VPU into user's SoC environment.

2.1. I/O Port Signal

2.1.1. Signal Summary

I/O port signals of VPU are categorized into four following types :

- APB bus signals
- AXI bus signals
  - Primary AXI (mandatory) : program, data, bitstream, compressed frame data or neighbor pixel data
  - Secondary AXI (optional) : neighbor pixel data
- General signals
  - Interrupt
  - UART for debugging message
- A test signal

All signals are defined as either Input or Output. There is no tri-state or bi-directional signal in the port list of VPU. *Figure 2.1, "Port List of VPU"* shows the conceptual port list of each signal category.

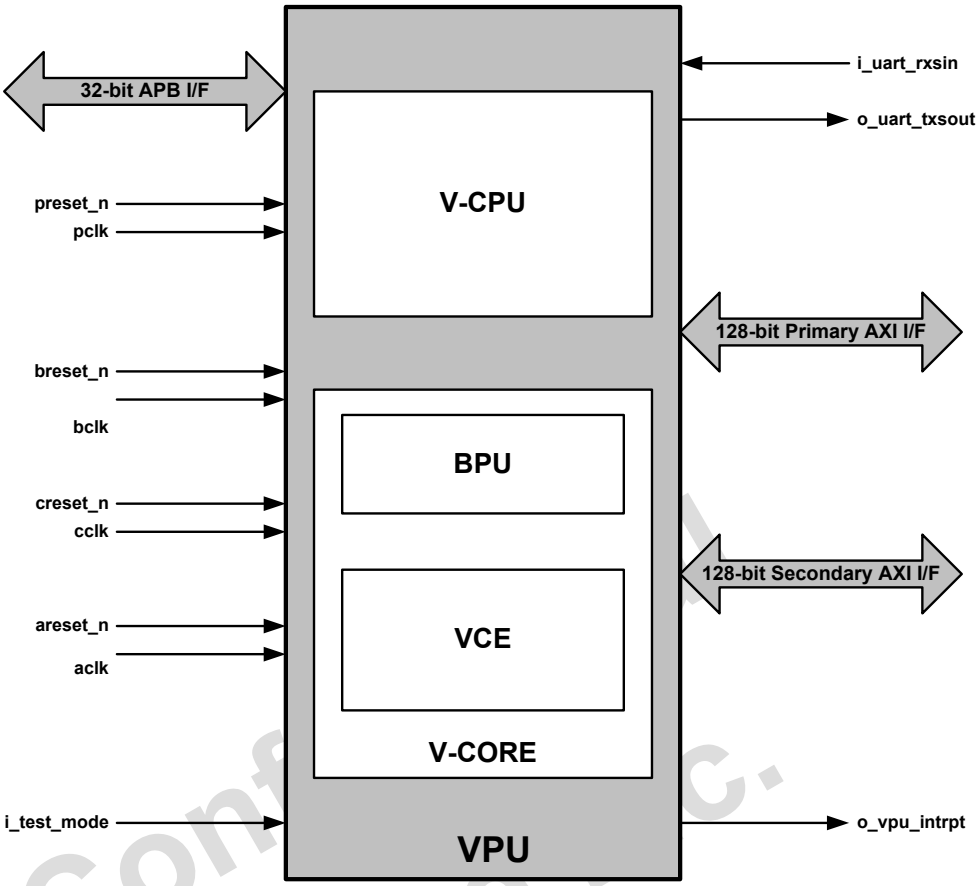


Figure 2.1. Port List of VPU

The APB and AXI are AMBA3 compliant bus interfaces. The APB is consisting of input/output port signals : 16-bit address signals and 32-bit data signals. They are used for interfacing with host processor on the hierarchical architecture of VPU.

VPU presents 128-bit AXI interfaces that allow SoC to have more flexible connectivity. Depending on SoC inter-connection with many other AXI buses on the system, all of the AXI output ports can be connected separately, or only two output ports (Primary AXI and Secondary AXI) can be connected with an AXI arbiter inside VPU.

The Secondary AXI is an optional AXI for bandwidth reduction purpose. It can connect VPU to On-chip SRAM directly which can be shared by other modules in the SoC and carry temporal data used for internal processing of VPU, so that these data are not transmitted to the external memory and bandwidth can be much more reduced.

WAVE420L codec uses 4 clocks which are pclk, cclk, aclk and bclk with the corresponding reset signals, preset_n, creset_n, areset_n and breset_n. Meantime, WAVE420L encoder uses 3 clocks which are pclk, cclk and aclk with the corresponding reset signals, preset_n, creset_n, and areset_n. For details, refer to *Section 2.1.2.3, "Clock and Reset Signals"*.

There is an interrupt signal sent to host processor from VPU, alarming the status of VPU such as completion of picture processing. UART and test signals are provided for debugging and test purpose respectively.

2.1.2. List of Signals

The following tables show a list of I/O port signals of VPU. The conventions are as follows.

Prefix Conventions

- i : Input

- o : Output
- N/A : Not Applicable
- [x:y] : Bus width
- xx'byy : Bit width is xx bits in decimal and the value is yy in binary

2.1.2.1. APB Signals

Table 2.1. AMBA3 APB signals

Name	Type	Destination	Description
i_host_apb_psel	Input	APB master	APB bus peripheral select
i_host_apb_penable	Input	APB master	APB bus enable
i_host_apb_pwrite	Input	APB master	APB bus write signal
i_host_apb_paddr[15:0]	Input	APB master	APB bus address
i_host_apb_pwdata[31:0]	Input	APB master	APB bus write data
o_host_apb_prdata[31:0]	Output	APB master	APB bus read data
o_host_apb_pready	Output	APB master	APB bus peripheral ready signal
i_host_apb_emudbg	Input	APB master	Debug enable signal 0: Normal mode access 1: Debug mode access

2.1.2.2. AXI Signals

There are types of AXI bus interfaces available for VPU. They have an identifier "one_pri" and "one_sec" in the middle of AXI signal name. i.e. i_one_pri_axi_wready, which is represented with *** in the *Table 2.2, "AMBA3 AXI signals - write channel"*.

The AXI bus signals are classified into write and read channel. The AXI write channels are AXI write address channel, AXI write data channel and AXI write response channel. The AXI read channels are AXI read address channel and AXI read data channel.

Table 2.2. AMBA3 AXI signals - write channel

Name	Type	Source/ Destination	Description
AMBA3 AXI signals - write address channel			
o_***_axi_awid[9:0]	Output	AXI slave	AXI write address ID This signal is the identification tag for the write address group of signals.
o_***_axi_awaddr[31:0]	Output	AXI slave	AXI write address The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
o_***_axi_awlen[3:0]	Output	AXI slave	AXI write burst length The burst length gives the exact number of transfers in a burst. This information

Name	Type	Source/ Destination	Description
			determines the number of data transfers associated with the address.
o_***_axi_awsize[2:0]	Output	AXI slave	AXI write burst size This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. **The value of this signal is always 3'b100 .**
o_***_axi_awburst[1:0]	Output	AXI slave	AXI write burst type The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **The value of this signal is always 2'b01, the incrementing-address burst.**
o_***_axi_awlock[1:0]	Output	AXI slave	AXI write lock type This signal provides additional information about the atomic characteristics of the transfer. For end request of overlapped burst or single request, this signal is "00" (normal). For non-end request of overlapped burst, this signal is "10" (lock). **The value of this signal is always 2'b00.**
o_***_axi_awcache[3:0]	Output	AXI slave	AXI write cache type This signal indicates the bufferable, cacheable, write-through, write-back, and allocates attributes of the transaction. **The value of this signal is always 4'b0000.**
o_***_axi_awprot[2:0]	Output	AXI slave	AXI write protection type This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **The value of this signal is always 3'b010, the normal non-secure data access.**
o_***_axi_awvalid	Output	AXI slave	AXI write address valid This signal indicates that valid write address and control information are available:

Name	Type	Source/ Destination	Description
			1 = address and control information available 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.
i_***_axi_awready	Input	AXI slave	AXI write address ready Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.
AMBA3 AXI signals - write data channel			
o_***_axi_wid[9:0]	Output	AXI slave	AXI write ID tag This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction.
o_***_axi_wdata[127:0]	Output	AXI slave	AXI write data
o_***_axi_wstrb[15:0]	Output	AXI slave	AXI write strobes This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. VPU uses this signal for memory compression and cache. External DRAM controller must support this WSTRB[15:0]
o_***_axi_wlast	Output	AXI slave	AXI write last This signal indicates the last transfer in a write burst.
o_***_axi_wvalid	Output	AXI slave	AXI write valid This signal indicates that valid write data and strobes are available: 1 = write data and strobes available 0 = write data and strobes not available.
i_***_axi_wready	Input	AXI slave	AXI write ready This signal indicates that the slave can accept the write data: 1 = slave ready 0 = slave not ready.
AMBA3 AXI signals - write response channel			

Name	Type	Source/ Destination	Description
i_***_axi_bid[3:0]	Input	AXI slave	AXI response ID The identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding.
i_***_axi_bresp[1:0]	Input	AXI slave	AXI write response This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR in the AMBA AXI Protocol Specification. **However, VPU regards any of the four responses as OKAY.**
i_***_axi_bvalid	Input	AXI slave	AXI write response valid This signal indicates that a valid write response is available: 1 = write response available 0 = write response not available.
o_***_axi_bready	Output	AXI slave	AXI response ready This signal indicates that the master can accept the response information. 1 = master ready 0 = master not ready.

Table 2.3. AMBA3 AXI signals - read channel

Name	Type	Source/ Destination	Description
AMBA3 AXI signals - read address channel			
o_***_axi_arid[9:0]	Output	AXI slave	AXI read address ID This signal is the identification tag for the read address group of signals.
o_***_axi_araddr[31:0]	Output	AXI slave	AXI read address The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
o_***_axi_arlen[3:0]	Output	AXI slave	AXI read burst length The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.

Name	Type	Source/ Destination	Description
o_***_axi_arsize[2:0]	Output	AXI slave	AXI read burst size This signal indicates the size of each transfer in the burst. **The value of this signal is always 3'b100.**
o_***_axi_arburst[1:0]	Output	AXI slave	AXI read burst type The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. **The value of this signal is always 2'b01, the incrementing-address burst.**
o_***_axi_arlock[1:0]	Output	AXI slave	AXI read lock type This signal provides additional information about the atomic characteristics of the transfer. For end request of overlapped burst or single request, this signal is "00" (normal). For non-end request of overlapped burst, this signal is "10" (lock). **The value of this signal is always 2'b00.**
o_***_axi_arcache[3:0]	Output	AXI slave	AXI read cache type **The value of this signal is always 4'b0000.**
o_***_axi_arprot[2:0]	Output	AXI slave	AXI read protection type **The value of this signal is always 3'b010, the normal non-secure data access.**
o_***_axi_arvalid	Output	AXI slave	AXI read address valid This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1: address and control information valid 0: address and control information not valid
i_***_axi_arready	Input	AXI slave	AXI read address ready This signal indicates that the slave is ready to accept an address and associated control signals:

Name	Type	Source/ Destination	Description
			1: slave ready 0: slave not ready
AMBA3 AXI signals - read data channel			
i_***_axi_rid[9:0]	Input	AXI slave	AXI read ID tag This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding.
i_***_axi_rdata[127:0]	Input	AXI slave	AXI read data
i_***_axi_rresp[1:0]	Input	AXI slave	AXI read response This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR in the AMBA AXI Protocol Specification. **However, VPU regards any of the four responses as OKAY.**
i_***_axi_rlast	Input	AXI slave	AXI read last This signal indicates the last transfer in a read burst.
i_***_axi_rvalid	Input	AXI slave	AXI read valid This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available.
o_***_axi_rready	Output	AXI slave	AXI read ready This signal indicates that the master can accept the read data and response information: 1= master ready 0 = master not ready.

2.1.2.3. Clock and Reset Signals

VPU uses clock sources and *Figure 2.2, "Clock and Reset Signals"* illustrates each set of clock and reset signals that are generated in different clock domains.

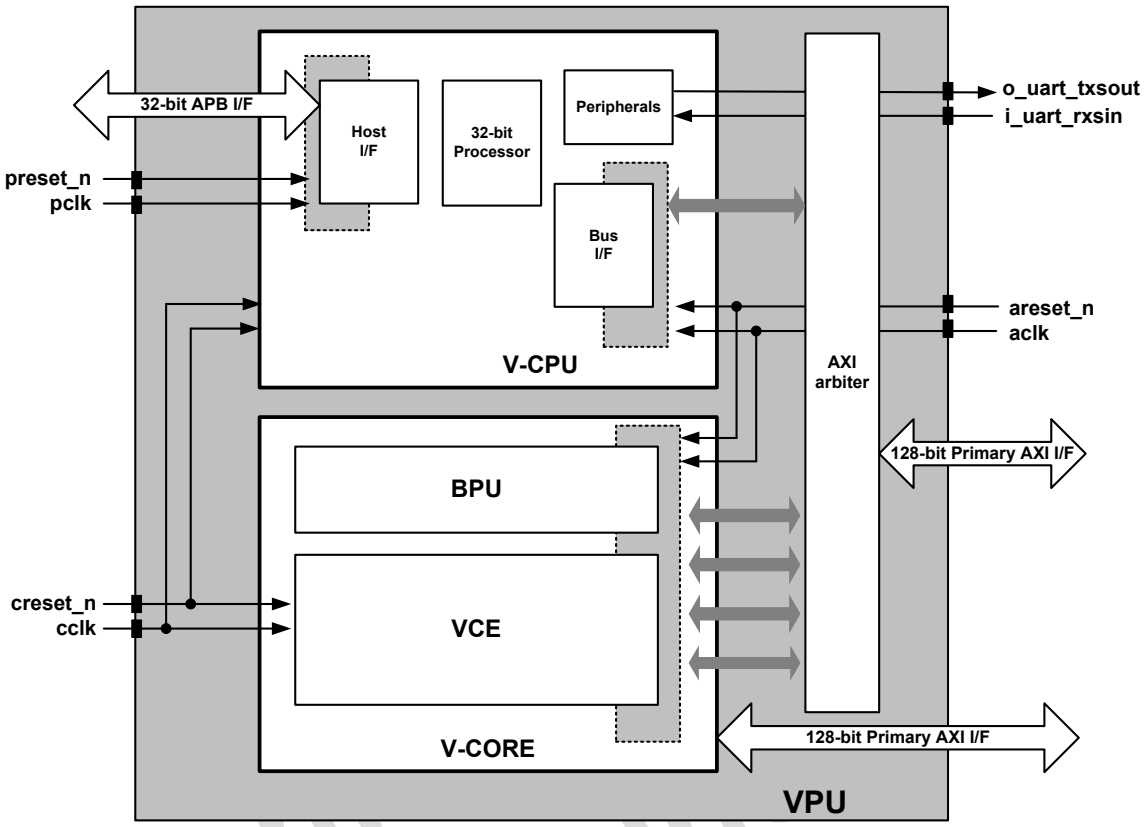


Figure 2.2. Clock and Reset Signals

Pairs of clock and reset signal can drive each subblock of VPU.

Table 2.4. Clock and Reset Signals

Name	Type	Source/ Destination	Description
pclk	Input	External clock controller	APB bus clock signal
preset_n	Input	External reset controller	APB bus reset signal
aclk	Input	External clock controller	AXI bus clock signal
areset_n	Input	External reset controller	AXI bus reset signal
bclk	Input	External clock controller	Clock signal for BPU in V-CORE and V-CPU
breset_n	Input	External reset controller	Reset signal for BPU in V-CORE and V-CPU
cclk	Input	External clock controller	Clock signal for VCE in V-CORE
creset_n	Input	External reset controller	Reset signal for VCE in V-CORE

Based on the GALS (Globally Asynchronous Locally Synchronous) architecture of VPU, each clock source can be asynchronous to the others. The duty cycles of all clocks are "don't care", because VPU uses only rising edge of them. Reset assertion is also asynchronous to its clock source which does not need to be active at the time of reset.

For better performance, all of the clocks may be set as the maximum synthesizable frequency for each block. However, you might want to reduce the number of clock/reset ports. Even in that case, it is recommended to separate CCLK and BCLK for decoder, because frequency of bclk is somewhat slower than that of cclk. Other clocks can use one source if you want.

2.1.2.4. Monitoring Signal

Table 2.5. VPU Monitoring Signal

Name	Type	Source/Destination	Description
o_vpu_idle	Output	-	VPU monitoring signal. This signal indicates that VCPU core stopped and the bus status is IDLE.

2.1.2.5. Interrupt Signal

Table 2.6. Interrupt Signal

Name	Type	Source/Destination	Description
o_vpu_intrpt	Output	Interrupt controller	Interrupt request signal (active high) This signal is retained until host CPU clears it. This signal is synchronized with positive edge of PCLK.

2.1.2.6. UART Signals

Table 2.7. UART Signals

Name	Type	Source/Destination	Description
i_uart_rxsin	Input	PAD	UART received serial data input. If you don't want to use this pin, please tie up the pin.
o_uart_txsout	Output	PAD	UART transmitted serial data output

2.1.2.7. DFT Signal

Table 2.8. DFT Related signal

Name	Type	Source/Destination	Description
i_test_mode	Input	Test logic	0: Normal operation 1: Scan test

2.2. Clock & Reset Control

VPU uses a multi-level clock gating technique for SoC customers who want to optimize power consumption.

Figure 2.3, "Internal Clock Gating Scheme" depicts how each of clock sources is gated inside VPU.

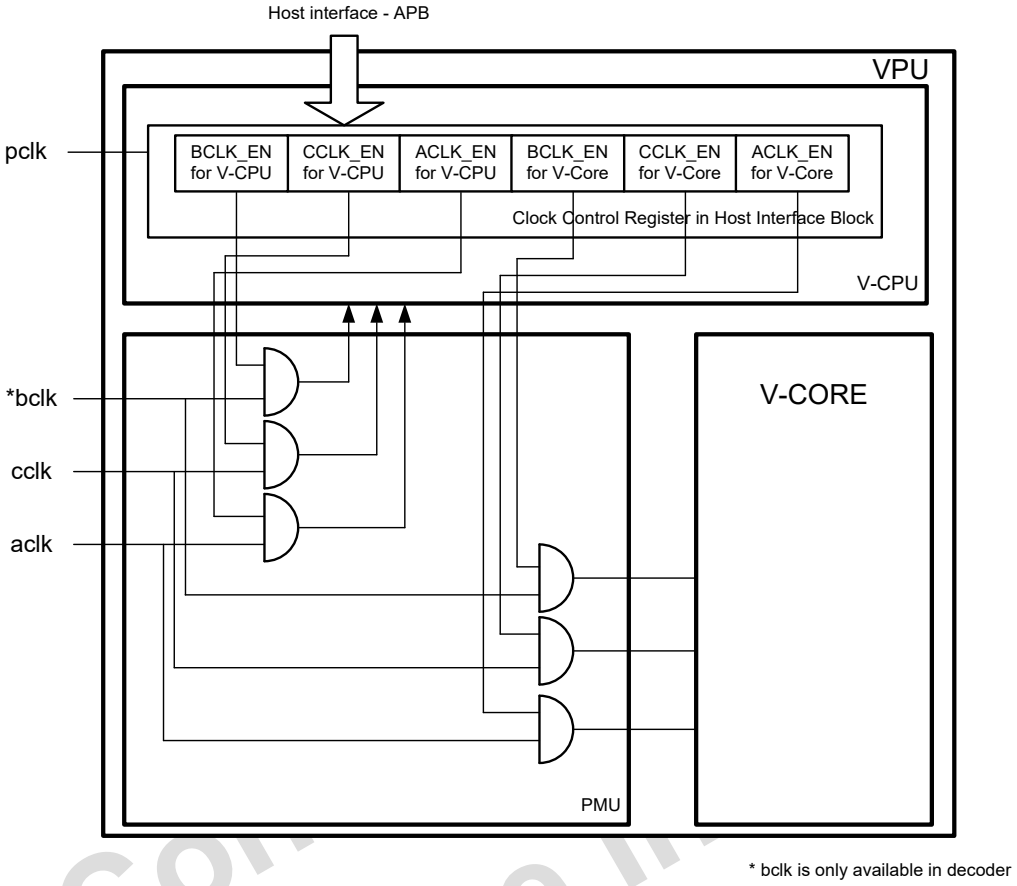


Figure 2.3. Internal Clock Gating Scheme

There are clock enable host I/F registers as many as clock domains except PCLK itself in Host Interface Block of V-CPU and PMU (Power Management Unit) for clock and reset control logic inside VPU. If host processor wants to shut off a certain unnecessary clock activity, they can set the associated register gating the clock in the software manner. With the register setting from host processor, PMU is able to shut off the any of the clock sources - ACLK_EN, BCLK_EN, and CCLK_EN. PMU is using simple two-input AND gates for clock gating in which one input to AND gate is a clock while the other input is a signal asserted from clock enable register.

Note For more details, please refer to the PMU relevant registers which are from VPU_RESET_REQ(0x50) to VPU_CLK_MASK(0x5C) in *programmer's guide*.

Inserting your own clock gating cell into *_vegas_cglp.v

The wave420l_vegas_cglp.v below the /design/rtl_v/prim/ directory includes a certain code describing a clock gating cell model, which is instantiated for gating clock of sub-block for RTL simulation, not for actual synthesis purpose. The enable signals for each gate are controlled by firmware and internal controller.

Otherwise, if you want to use your own clock gating cell and you don't define CNM_VEGAS_CGLP_RTL in rtl_v/top/*_cnm_defines.v, you should instantiate it in the *_vegas_cglp.v file. This is an example of instantiating user specific clock gating cell, the else part in the following *_vegas_cglp.v code.

Note * indicates the name of VPU product.

```
`ifdef CNM_VEGAS_CGLP_RTL
     // clock gating model for simulation
   reg   gate_en;
   always  @(CK or EN)
     begin
```

```
        if (~CK)
          gate_en=EN;
      end

    assign GCK = (TE | gate_en) & CK;

`else
    // user specific clock gating cell
  my_icg_cell icg_cell ( .CP(CK), .E(EN), .Q(GCK), .TE(TE) );

`endif
```

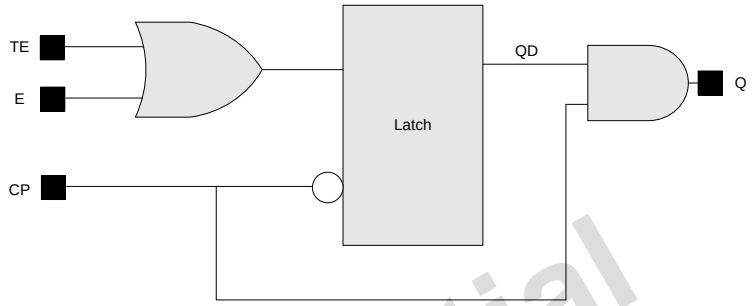


Figure 2.4. Schematic Diagram of icg_cell

Inserting your own clock mux cell into *_vegas_clk_mux.v

The wave420l_vegas_clk_mux.v below the /design/rtl_v/prim/ directory includes a certain code describing a clock mux cell model. It instantiates mux of reset signal in *_pmu_reset_gen.v module for RTL simulation, not for actual synthesis purpose. The enable signals for each gate are controlled by firmware and internal controller.

If you want to use your own clock mux cell, close the CNM_VEGAS_CGLP_RTL define in rtl_v/top/*_cnm_defines.v and write your own clock mux cell into the *_pmu_reset_gen.v file for better clock glitch and skew.

Note * indicates the name of VPU product.

```
`if CNM_VEGAS_CGLP_RTL
        // clock mux model for simulation
      assign O = (S) ? I1 : I0 ;
`else
    // user specific clock mux cell
  assign O = I0 ;
`endif
```

Figure 2.5, "Reset Scheme" depicts the internal reset scheme of VPU that is almost similar with clock gating.

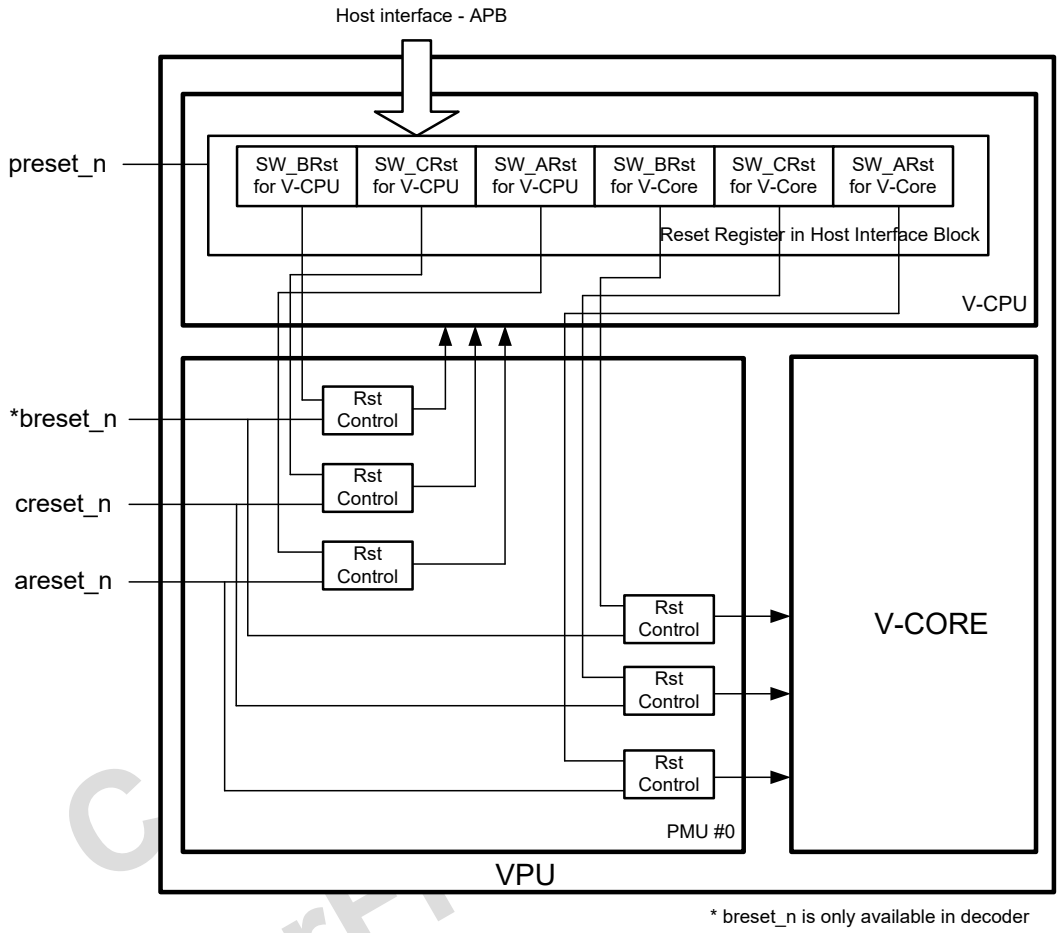


Figure 2.5. Reset Scheme

Figure 2.6, "Reset Connection" is an inner view of reset control logic in PMU.

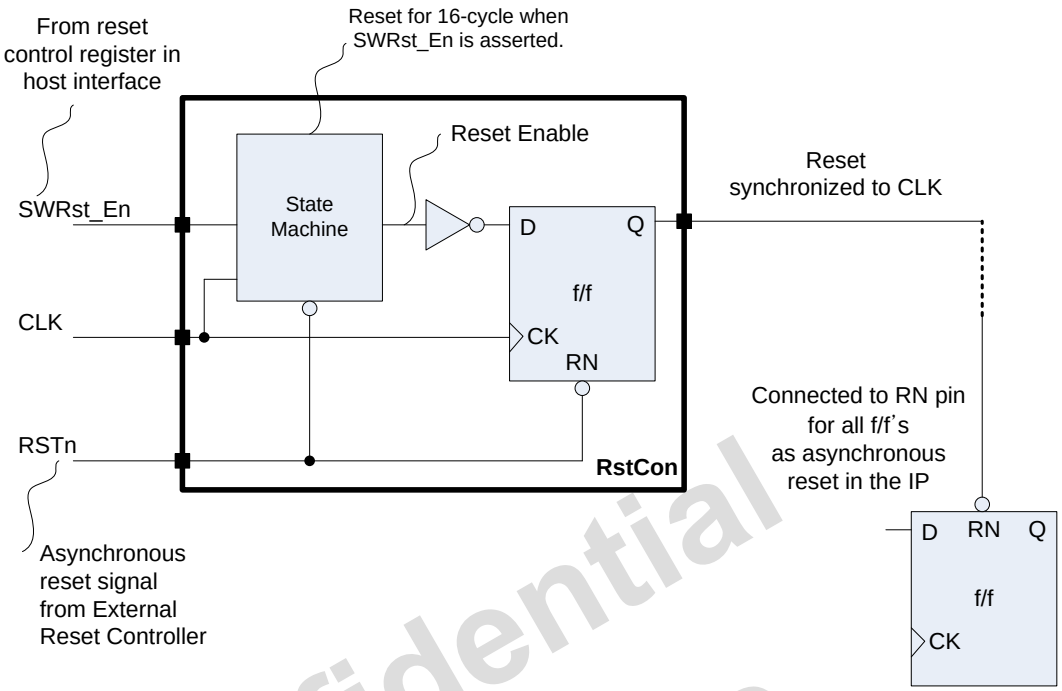


Figure 2.6. Reset Connection

Multi-level reset is driven by either hardwired reset signal `RSTn` or software reset signal `SWRst_En` by the host interface register setting. Both ways can be used to stop working of a certain clock domain.

SWRst_En works with input signal occurring when the reset control host I/F register is enabled. It requires the clock to be present and reset state to be retained for 16 cycles. On the other hand, the hardwired reset signal RSTn can generate reset with or without clock, and reset state stays as long as RSTn is asserted.

2.3. Bus Interface

2.3.1. APB

Host processor can communicate with VPU through AMBA3 APB bus. Host processor can set parameters for VPU configuration, give command, and get the status of VPU through APB interface. *Figure 2.7, "APB timing diagram"* shows the general APB timing diagram.

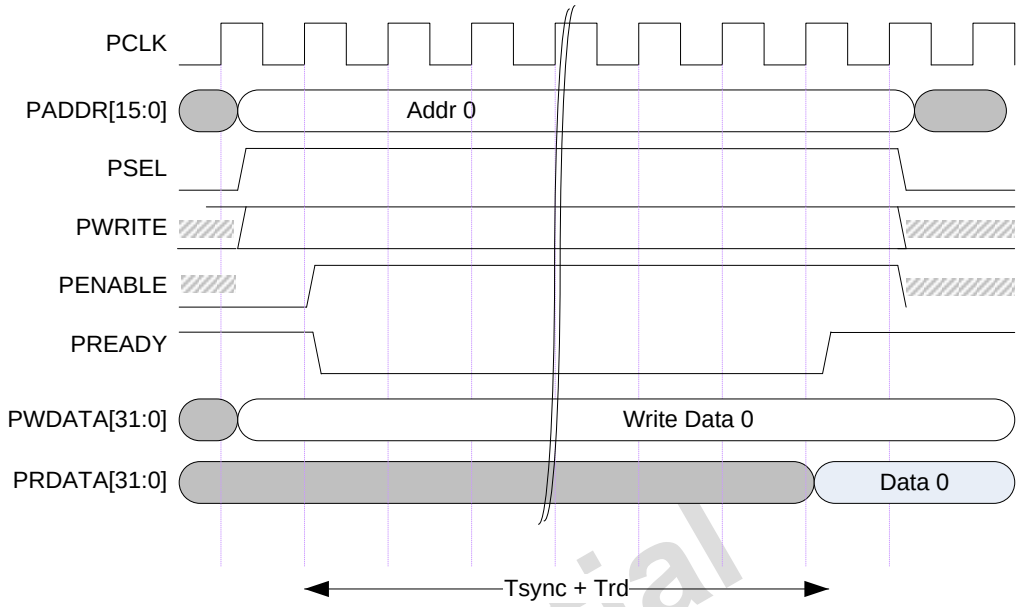


Figure 2.7. APB timing diagram

When VPU and Host processor are requesting access to a certain host interface register at the same time, VPU has a higher priority than host processor. Therefore, host processor has to wait until VPU's access to the register has completed. The wait control is performed by PREADY signal. In addition to those wait cycles, a few cycles are necessary to synchronize with CCLK, the VPU core clock.

The bit-width of the PADDR is 16. Among the sixteen bits, VPU actually uses most significant 14 bits (PADDR[15:2]). The least significant 2 bits, PADDR[1:0] is meaningless because VPU supports only word access.

2.3.2. Secondary AXI

Secondary AXI is an optional interface that is used for bandwidth reduction of DDR memory. The on-chip memory connected to VPU might be dedicated to VPU or other external modules.

In VPU, AXI ports of the internal blocks are connected to the primary AXI interface and the secondary AXI interface simultaneously but directed either to primary or secondary AXI by internal register setting.

If you can connect an additional internal memory at system level and want to reduce the bandwidth, connect the secondary AXI interface to the memory and set the relevant registers through API. Then two (primary and secondary) AXI interfaces could be utilized. On the other hand, if you cannot connect an additional internal memory or if the system can support sufficient bandwidth, use the primary AXI interface only.

The secondary AXI bus is used for transferring internal data only for VPU. Memory space for those data does not have to be disclosed to external AXI bus masters, for example, an external host processor. The memory for storing data transferring on the secondary AXI bus could be implemented with external DDR or SRAM, depending on the users' system environment. If they do not want to use this bus interface, it is also possible to connect all internal bus masters of VPU to the primary bus by setting the configuration register. In default, all internal AXI masters are connected to the primary AXI bus interface.

Chapter 3
SYSTEM REQUIREMENTS

This chapter describes the requirements of using VPU in the system point of view. Data flow, bus usage, buffer access pattern, format, and size are covered in this chapter.

3.1. Bandwidth

This chapter explains which type of data is read or written from/to the external memory through AXI bus channel for the entire decoding process and how much bandwidth would be used for each channel, and finally estimation on total amount of bandwidth for a few typical test cases.

3.1.1. Encoder Data Transactions on AXI

Figure 3.1, "Data Flow of WAVE420L Encoder" illustrates all kinds of AXI data transaction to the external that occur from VPU during encoding process.

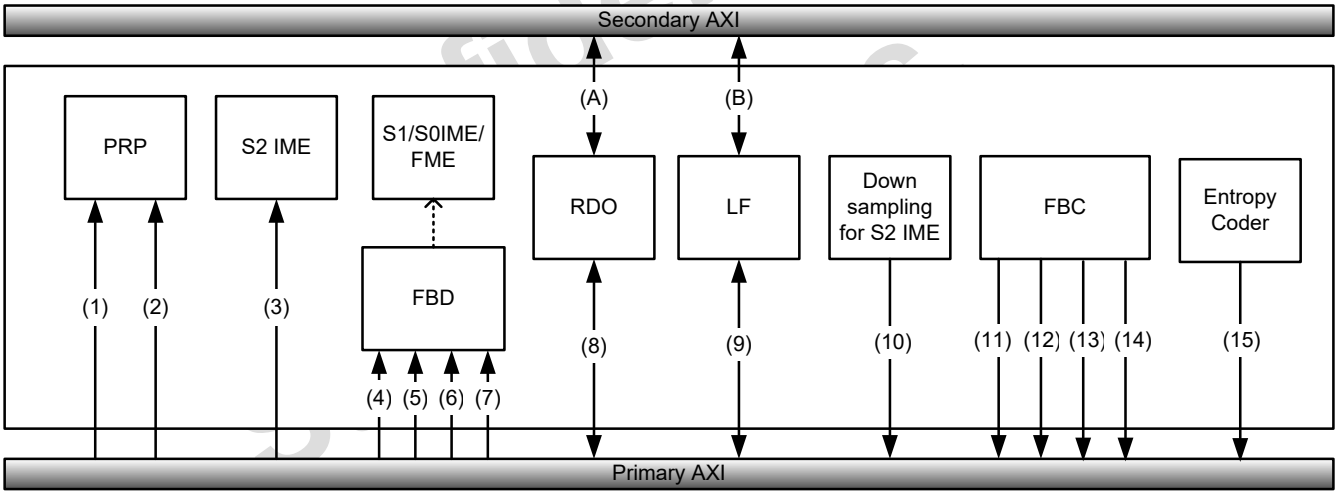


Figure 3.1. Data Flow of WAVE420L Encoder

Note (A) and (B) are read/write requests for access to on-chip SRAM through secondary AXI to save bandwidth. Others are the ones through general AXI channels for read or write data from/to the external memory.

The following is the brief explanation about each AXI data request.

First, PRP reads current luma and chroma samples from the external memory. (1) is luminance samples and (2) is chrominance samples. (1) and (2) take the below amount of bandwidth.

- 8bit : width * height * 1.5

Through the (3) request, S2 IME reads down-sampled (1/4 * width, 1/4 * height) reference luma samples for hierarchical motion estimation within search range [+/-64H, +/-32V] centered on PMV position. It works on a 32x32 block basis. Bandwidth of (3) is the following.

- 1/16 * width * height (This is approximated bandwidth. It depends on the cache hit/miss ratio for S2 cache.)

On (4),(5),(6), and (7), S1/S0 IME & FME also needs luma and chroma samples for further in depth motion estimations. Reference samples are the ones stored in compressed format, so FBD (Frame Buffer Decompressor) loads compressed reference luma samples and compressed reference chroma samples and performs decompression before being used for ME.

- (4) is taken as much as the BW of luma offset table data read.
- (5) as the BW of luma compressed data read.
- (6) as the BW of chroma offset table data read.
- (7) as the BW of chroma compressed data read.

(8) is read/write requests of CTU context, temporal neighbor pixels, and collocated motion vectors which are used in RDO module. If secondary AXI is used this is removed and replaced with (A).

(9) is read/write requests of temporal pixel data used in LF module. If secondary AXI is used this is removed and replaced with (B).

(10) is a write request of down-sampled reference samples which is used for S2 IME in next frame encoding.

- 1/16 * width * height height

(11),(12),(13), and (14) are write requests of compressed reference luma and chroma samples for reconstruction.

- (11) is taken as much as the BW of luma offset table data write.
- (12) as the BW of luma compressed data write.
- (13) as the BW of chroma offset table data write.
- (14) as the BW of chroma compressed data write.

(15) is a write request of coded bitstream as a final result of encoding process.

3.2. External Memory Access

This chapter describes buffers on the external memory that VPU has access to, buffer format and endianness, how to set the base address for each buffer, and required buffer size.

3.2.1. Types of Buffer

There are three types of buffers that are involved in VPU.

- Code buffer: buffer for storing program and read only data for VPU operation.
- Bitstream buffer: In decoder, V-CPU reads bitstream from this buffer. In encoder, V-CPU and BPU write bistream to this buffer. It is also called a CPB (coded picture buffer). Bitstream buffer exists for each instance.
- Frame buffer : V-CORE writes or reads frame data for reference to/from this buffer through Primary AXI. It is also called a DPB (decoded picture buffer). Frame buffer exists for each instance.
- Temporal buffer : V-CORE writes or reads neighbor pixel data that are required for RDO and loop filter in encoder and for intra-prediction and loop filter in decoder to/from this buffer. The data for temporal buffer can be transferred through primary AXI or secondary AXI if secondary AXI is used. (this document represents the total size of temporal buffer for one instance).
- Work buffer: VPU writes or reads the information for the codec instance to/from this buffer through Primary AXI. This buffer should be maintained while an instance is in active. One per instance.
- User data and Report buffer: This is an optional buffer for VPU to handle user data in decoding process or to report some information to host CPU.

3.2.2. Buffer Format

Figure 3.2, "Input and Output Format of WAVE420L" shows which types of data are used for VPU as its input source and output.

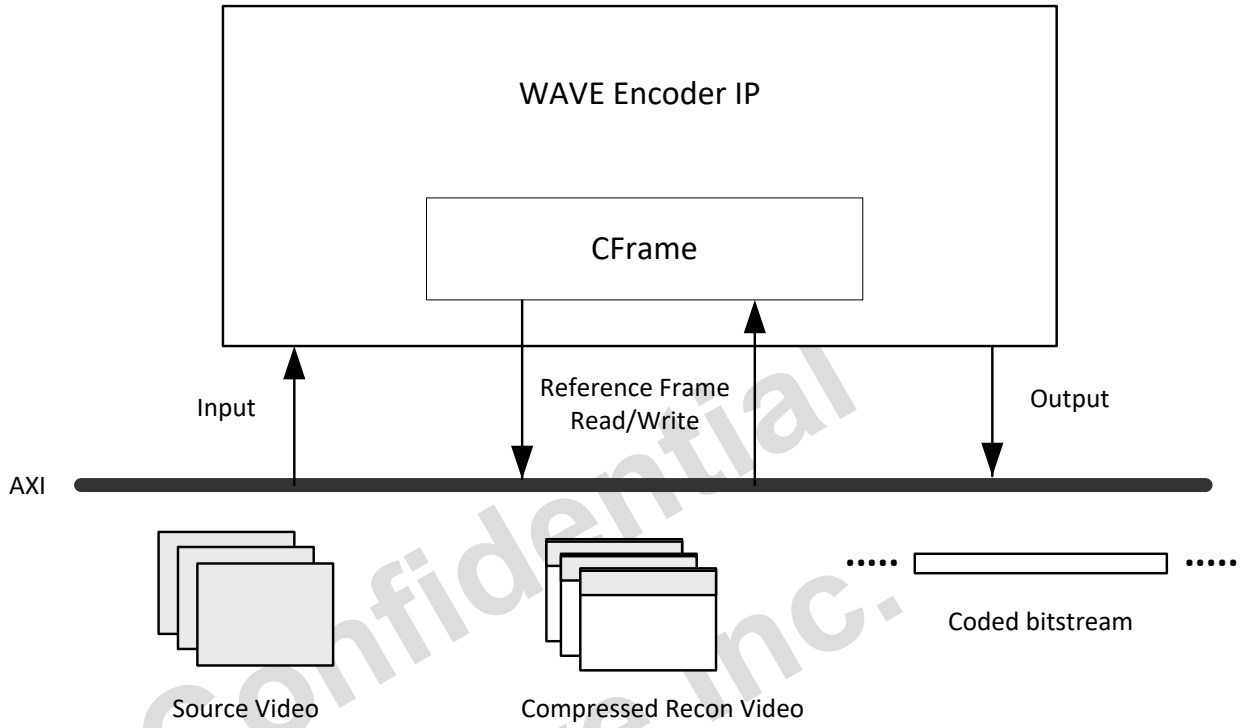


Figure 3.2. Input and Output Format of WAVE420L

WAVE420L involves not only a complementary pair of encoder and decoder hardware, but also frame buffer compression block called CFrame. Basically, in encoding case VPU reads source video, converts them into a compressed form occupying a reduced number of bits, write them back to the external memory. During this encoding, VPU also reads reference data for prediction which were compressed. Meanwhile, VPU decoder converts the bitstream back into a representation of the original video data and applies filtering to reduce block distortion. These reconstructed, filtered frames are compressed by CFrame and then written for future predictions.

3.2.2.1. Frame Buffer Format

3.2.2.1.1. Compressed Formats

The CFrame should allocate 4 memory regions for one compressed frame - 2 regions for compressed texture Y and C and 2 regions for offset table Y and C. The base address for each region should be assigned with the relevant host interface register.

3.2.2.1.2. Uncompressed Formats

WAVE420L encoder supports five kinds of frame formats - Planar, NV12, NV21, packed mode and 2D mode, while the decoder supports three different frame formats - Planar, NV12 and NV21.

- **Planar**

There are three planes in this format - Y, Cb and Cr. The address increment…..

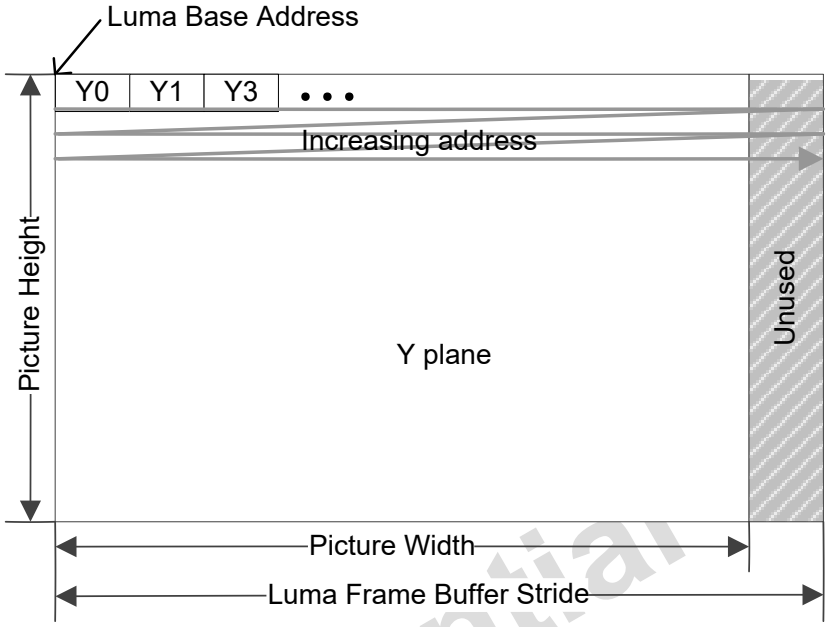


Figure 3.3. Luma Plane in Planar Mode

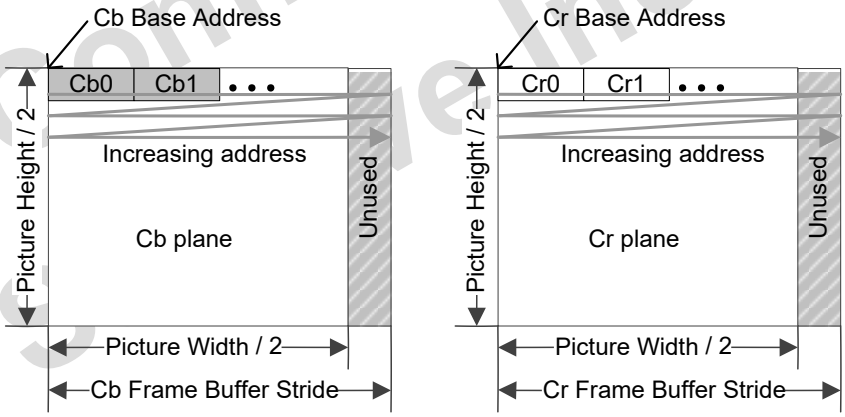
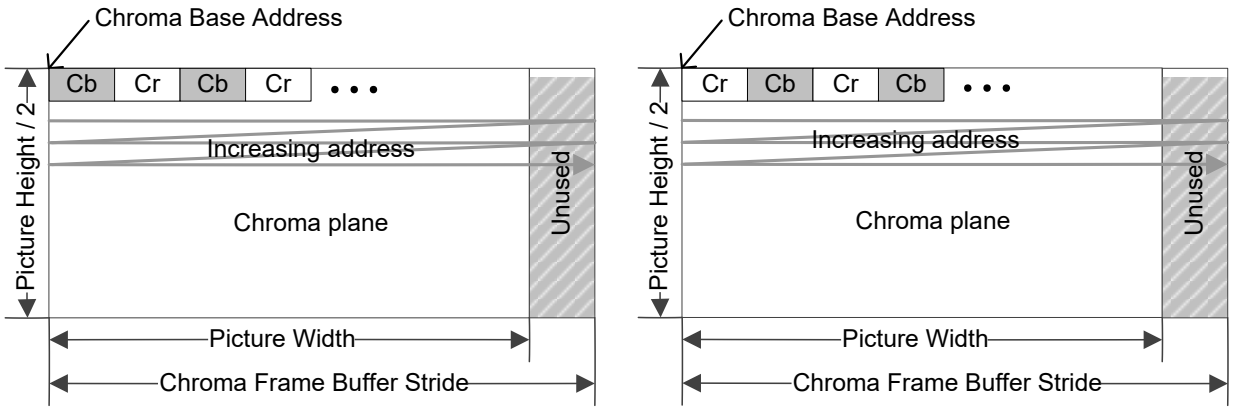


Figure 3.4. Cb and Cr Plane in Planar Mode

- **NV12 and NV21**

These have 2 Planes which are luma and chroma plane. The luma plane of NV12 and NV21 are the same. The chroma plane has interleaved Cb and Cr. The order of chroma sample in NV12 is Cb, Cr, Cb, Cr, … and NV21 has the opposite sample order such as Cr, Cb, Cr, Cb, …



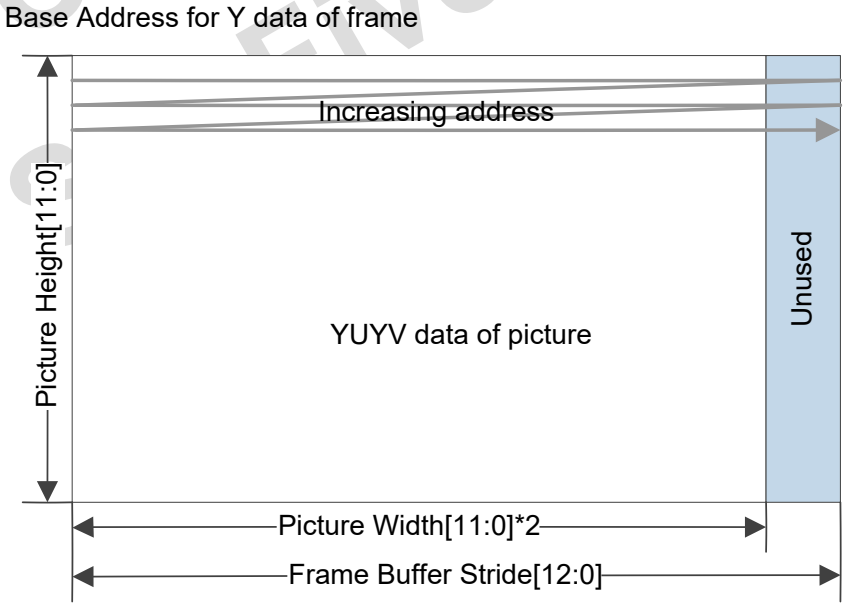
(a) NV12 chroma plane (b) NV21 chroma plane

Note: The luma plane of NV12 and NV21 is the same as the one of planar.

Figure 3.5. Chroma Plane of NV12 and NV21

- **Packed mode**

This is YUV4:2:2 format and has only 1 plane for Y, Cb and Cr. The order of samples in packed mode is Y,Cb,Y,Cr, … The width of plane in packed mode is double of picture width, because Y and C exist in the same plane. Even though packed mode has YUV4:2:2 image, WAVE420L encodes YUV4:2:0 image by ignoring the even line chroma format.



Frame buffer for packed mode

Figure 3.6. The Plane of Packed Mode

- **2D mode**

2D mode has 2 planes, Y and C. In the Y plane, the address increases in vertical direction until 32 lines as shown in *Figure 3.7, "2D Mode"*. Long burst (128-byte) access of 2D mode can improve bandwidth efficiency.

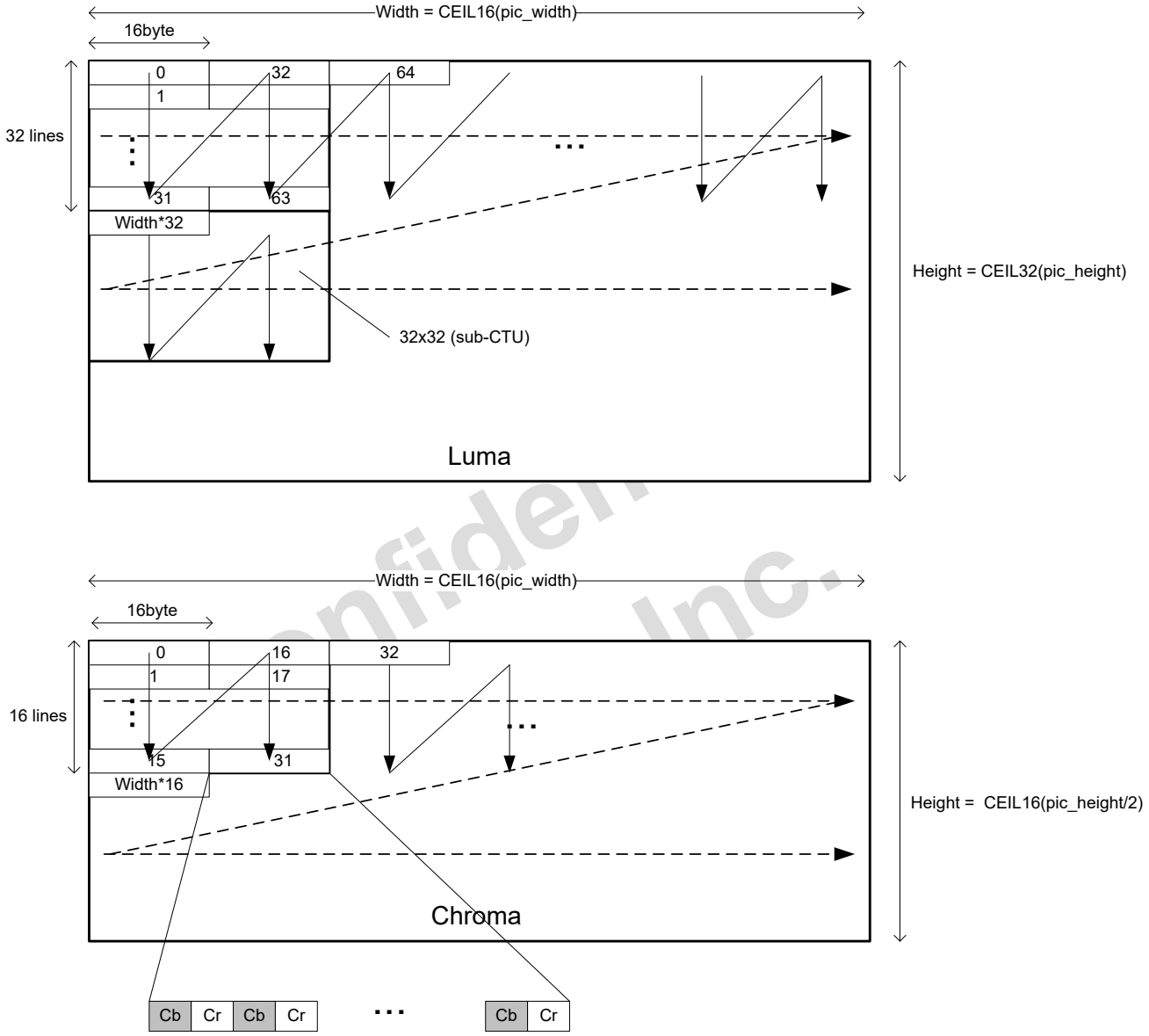


Figure 3.7. 2D Mode

A frame buffer is configured with the following parameters:

- Alignment
  - Luminance frame buffer base address in 16-byte alignment
  - Cb frame buffer base address in 16-byte alignment
  - Cr fame buffer base address in 16-byte alignment: In case of NV12 or NV21, a base address for Cr is meaningless because a base address for Cb is used to store or load the interleaved Cb/Cr samples.
  - In 2D mode, the height of frame buffer is the multiple of 32.
- Stride
  - A stride should be multiple of 16 for each luma and chroma.

3.2.3. Buffer Size

3.2.3.1. Encoder

3.2.3.1.1. Bitstream Buffer

A maximum bitstream buffer size is the same size of uncompressed frame size.

```
BitstreamBufferSize = UncompressedFrameSize
```

These are the examples of bitstream buffer size according to frame sizes.

- HD
  – 1080p @ L4.1 Main Profile : 2.97 Mbytes
- 4K
  – 3840x2160 @ L5.0 Main Profile : 11.87 Mbytes

3.2.3.1.2. Frame Buffer

Required frame buffer size is derived by this formula.

```
FrameBufferSize = source frame buffer size + reference frame buffer size
```

- source frame buffer size = # of needed source frame x (1 source frame size)
- reference frame buffer size = MaxDpbSize x (1 compressed frame size + 1 sub-sampled frame size + 1 Collocated MV buffer size)

The number of source frame is determined by GOP structure such as IPPP or IBP. A reference frame buffer for encoding a frame includes a compressed frame, a sub-sampled frame, and collocated MV information. MaxDpbSize is defined in the specification - Annex A.4 Tiers and levels of ISO/IEC 23008-2.

During derivation of FrameSize, we will use ceilN(K) function. This function means *ceiling to N*. For instance, we can write the body of ceilN(K) in C code like below because the result of integer division always floor(fraction part will be remove to 0) in C .

```
ceilN(K) = (K + (N-1) / (N)) * N
```

We will use variables for the resolution of picture. The variables are multiples of 2.

- W = pic_width_in_luma_samples
- H = pic_height_in_luma_samples

In a frame buffer, addresses are continuous on the physical memory. However, frame buffers for each color component or frame do not have to necessarily be in the same place with continuous physical addresses.

3.2.3.1.2.1. Frame Buffer Size Derivation Formula

Table 3.1. Frame Buffer size for 1080p @ L4.1

	MAIN profile
1 source frame size	W x H x 1.5
1 compressed frame size	(W x H x 1.5) + (ceil256(W)/256) x (ceil16(H)/16) x 128 x 1.5
1 sub-sampled frame size	ceil16(W/4) x ceil4(H/4)
1 collocated MV buffer size	(ceil16(W) x ceil16(H))/256 x 8

3.2.3.1.2.2. Example Frame Buffer Sizes

MAIN profile

- Example 1
  – 1080p @ L4.1

Table 3.2. Frame Buffer size for 1080p @ L4.1

GOP structure	Number of required Source Frame	Min. DPB Number	DPB size (MBytes)
IPPP	1	2	9.41

- Example 2
  – 4K @ 15fps (3840 x 2160)

Table 3.3. Frame Buffer size for 4K

GOP structure	Number of required Source Frame	Min. DPB Number	DPB size (MBytes)
IPPP	1	2	37.57

3.2.3.1.3. Work Buffer

A work buffer is a memory region that stores the information required during the encoding process. Every instance owns its work buffer, and once a work buffer is allocated, it cannot be allocated again within the same instance. VPU uses a fixed-size of work buffer as follows.

- Encoder : 128Kbyte
- Decoder : 3Mbyte

3.2.3.1.4. Code Buffer

The size of code buffer is the same as the sum of binary file for V-CPU and supervisor stack. The binary for V-CPU includes codes and read only data for processing and binary code of BIT processor. Code buffer should be aligned in 4-KB boundary. Code buffer should be set during VPU initialization process.

3.2.3.1.5. Temporal Buffer

A temporal buffer is a memory region where to store the information which is temporarily used for encoding a single frame. All of the instance opened can share this buffer in a V-CORE.

The following table shows formulas retrieving the required temporal buffer size in the worst case.

Once a temporal buffer is allocated, it can be used for all instances. Thus, please make sure to allocate enough memory to support the maximum size.

VPU accesses temporal buffer very frequently to read and write data. A temporal buffer in on-chip SRAM that is connected through 2nd AXI can be an alternative for faster access and less external memory bandwidth.

Table 3.4. External Memory size for BPU (MAIN profile)

Data	Description	Size (Byte)	Possible to access through 2nd AXI
Loopfilter	Row buffer	Luma = ceil64(PicWidth) x 5 byte Chroma = ceil64(PicWidth) x 3 byte	Yes

Data	Description	Size (Byte)	Possible to access through 2nd AXI
RDO	Neighbor information and collocated MV	(ceil64(PicWidth)/64) x 256 byte	Yes
Firmware	Miscellaneous information used by FW	1KB (entry point offset) + 2KB (ROI map) + 1KB (lambda)	No

3.2.3.1.6. Report and User Data

TBD

3.2.3.1.7. Summary with Examples

In this section, we present some examples of memory requirement for encoding HEVC Main profile L4.1 stream. The exact size might be a little different from byte alignment for each buffer. In the examples below, we assume that 1080p is 1920 x 1080 and 4K is 3840 x 2160.

Table 3.5. Required Buffer Size (Mbytes) for 1080p MAIN

GOP structure	IPPP
Num of needed source frame	1
MaxDpbNum	2
1 source frame size	2.97
1 compressed frame size	3.07
1 sub-sampled frame size	0.12
1 MVCol buffer size	0.03
Max Bitstream buffer size	2.97
Code Buffer	1.0
Working Buffer	0.13
Temporal Buffer	0.02
Total Size	13.52

Table 3.6. Required Buffer Size (MBytes) for 4K MAIN

GOP structure	IPPP
Num of needed source frame	1
MaxDpbNum	2
1 source frame size	11.87
1 compressed frame size	12.24
1 sub-sampled frame size	0.49
1 MVCol buffer size	0.12
Max Bitstream buffer size	11.87
Code Buffer	1.0
Working Buffer	0.13
Temporal Buffer	0.04
Total Size	50.61

3.2.4. Buffer Endianness

The VPU uses only 16-byte aligned access through the AXI bus interface and supports 16 endian modes when transferring data through the interface. The host processor can configure an endian mode using an API function provided by C&M.

3.2.4.1. Read and Write Operation

Example MEM[ADDR] = 0xFFEEDDCCBBAA99887766554433221100 where ADDR[3:0] is 4'b0000 always.

When the VPU writes 128-bit data 0xFFEEDDCCBBAA99887766554433221100 to address A, data ordering in wdata[127:0] are as followings:

ENDIAN_MODE = 4'b0000 (128-bit Little Endian)

127:120	**119:112**	**111:104**	**103:96**	**95:88**	**87:80**	**79:72**	**71:64**
FF	EE	DD	CC	BB	AA	99	88
63:56	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
77	66	55	44	33	22	11	00

ENDIAN_MODE = 4'b0001

127:120	**119:112**	**111:104**	**103:96**	**95:88**	**87:80**	**79:72**	**71:64**
EE	FF	CC	DD	AA	BB	88	99
63:56	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
66	77	44	55	22	33	00	11

ENDIAN_MODE = 4'b0010

127:120	**119:112**	**111:104**	**103:96**	**95:88**	**87:80**	**79:72**	**71:64**
DD	CC	FF	EE	99	88	BB	AA
63:56	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
55	44	77	66	11	00	33	22

ENDIAN_MODE = 4'b0011

127:120	**119:112**	**111:104**	**103:96**	**95:88**	**87:80**	**79:72**	**71:64**
CC	DD	EE	FF	88	99	AA	BB
63:56	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
44	55	66	77	00	11	22	33

ENDIAN_MODE = 4'b0100

127:120	**119:112**	**111:104**	**103:96**	**95:88**	**87:80**	**79:72**	**71:64**
BB	AA	99	88	FF	EE	DD	CC
63:56	**55:48**	**47:40**	**39:32**	**31:24**	**23:16**	**15:8**	**7:0**
33	22	11	00	77	66	55	44

ENDIAN_MODE = 4'b0101

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
AA	BB	88	99	EE	FF	CC	DD

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
22	33	00	11	66	77	44	55

ENDIAN_MODE = 4'b0110

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
99	88	BB	AA	DD	CC	FF	EE

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
11	00	33	22	55	44	77	66

ENDIAN_MODE = 4'b0111

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
88	99	AA	BB	CC	DD	EE	FF

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
00	11	22	33	44	55	66	77

ENDIAN_MODE = 4'b1000

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
77	66	55	44	33	22	11	00

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
FF	EE	DD	CC	BB	AA	99	88

ENDIAN_MODE = 4'b1001

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
66	77	44	55	22	33	00	11

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
EE	FF	CC	DD	AA	BB	88	99

ENDIAN_MODE = 4'b1010

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
44	55	77	66	11	00	33	22

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
DD	CC	FF	EE	99	88	BB	AA

ENDIAN_MODE = 4'b1011

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
55	44	66	77	00	11	22	33

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
CC	DD	EE	FF	88	99	AA	BB

ENDIAN_MODE = 4'b1100

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
33	22	11	00	77	66	55	44

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
BB	AA	99	88	FF	EE	DD	CC

ENDIAN_MODE = 4'b1101

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
22	33	00	11	66	77	44	55

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
AA	BB	88	99	EE	FF	CC	DD

ENDIAN_MODE = 4'b1110

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
11	00	33	22	55	44	77	66

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
99	88	BB	AA	DD	CC	FF	EE

ENDIAN_MODE = 4'b1111 (128bit Big Endian)

127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64
00	11	22	33	44	55	66	77

63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
88	99	AA	BB	CC	DD	EE	FF

3.2.4.2. Data Exchange with Host Processor

The VPU accesses external memory to do the followings. Endianness of buffers can be configured separately.

- Frame buffer access

The endianness of frame buffer can be configured separately. For detailed information, refer to the description about relationship between the endianness and the pixel layout (arrangement) for various frame buffer formats.

- Bitstream buffer access

The endianness of bitstream buffer can be configured separately.

- Code buffer access

The endianness of code buffer can be configured separately. The address of the code buffer should be aligned to 4KB boundary and should be set before sending INIT_VPU command. For more details, please see the VPU initialization sequence in programmer's guide.

- Task buffer, work buffer, and temporal buffer access

The task buffer, work buffer, and temporal buffer indicate regions of external memory which is dedicated to VPU. The region cannot be accessed by Host processor or other external peripherals. While access to the buffers, VPU uses its own endianness regardless of configuration for other buffers.

- External data for host processor buffer access

The endianness of external data for host processor such as user data buffer and report data can be configured separately.

3.3. Optional SRAM on Secondary AXI

Temporal memory can be used as internal memory which is connected with VPU through secondary AXI so that VPU can alleviate memory traffic and required bandwidth while decoding. VPU can replace as much of the temporal buffer as described in and *Section 3.2.3.1.5, "Temporal Buffer"* with optional SRAM.

The table below shows the amount of bandwidth reduced by using optional SRAM.

Table 3.7. Reduced BW when use of optional SRAM for Encoding

HW Block	Size for MAIN(8-bit)	BW saving
Loopfilter	ceil64(PicWidth) x 5 byte ceil64(PicWidth) x 3 byte	((ceil64(PicHeight)/64)-1) x LF size x 2 byte
RDO	(ceil64(PicWidth)/64) x 256 byte	(ceil64(PicHeight)/64) x RDO size x 2 byte

How much bandwidth is saved from using option memory might be different on CTU size of picture. In this table CTUsize means the width of a CTU for the video sequence.

For instance, the following table represents the required SRAM size for decoding 4096x2160 and the amount of bandwidth saved from use of these memories. In this table CTUwidth of 32 is used, since level should be 5.0 or higher to support the frame size.

The following table represents the required SRAM size for 1920x1080 and 3840x2160 encoding. Supported CTU size in encoder is 64.

Table 3.8. Optional SRAM and Reduced BW for Encoding 1920x1080 Video

Temporal Buffer Type	SRAM size (Byte)	Reduced BW (MB/s)
Loopfilter	15,360	480
RDO	7,680	255
Total	23,040	735

Table 3.9. Optional SRAM and Reduced BW for Encoding 3840x2160 Video

Temporal Buffer Type	SRAM size (Byte)	Reduced BW (MB/s)
Loopfilter	30,720	1,980

Temporal Buffer Type	SRAM size (Byte)	Reduced BW (MB/s)
RDO	15,360	1,020
Total	46,080	3,000

Appendix A
EMBEDDED MEMORIES

A.1. Overview

This chapter describes embedded memories used in the VPU. The following is the naming conventions of the memories.

- Single port register file
  - spregDxWeM
  - D = depth (number of word)
  - W = bits per word
  - M = bits per write mask bit

example

```
module spreg80x64e32 (clk, cen, wen, a, d, q);
    input   clk;
    input   cen;                  // chip select - "active low"
    input   [1:0] wen;            // write enable - "active low"
    input   [MemAddrWidth-1:0] a;
    input   [MemDataWidth-1:0] d;
    output  [MemDataWidht-1:0] q;

endmodule
```

Note If there is not any write mask bit in the memory, the bit-width of the wen signal is 1, and the eM is omitted in the name. But, if there is the write-mask bit per M-bit, the name has eM, and the bit-width of the wen is D/M-bit. The n-th bit, from LSB, of the write-mask bits is 0, [M*n -1 : M*(n-1)] of the data-bus, d, is written.

- Two port register file
  - A two port register file has 1 read-only port and 1 write-only port, and has a name of following:
  - dpregDxWeM
  - D = depth (number of words)
  - W = bits per word
  - M = bits per write mask bit

example

```
module dpreg192x64e32 (wa, wen, d, wclk, ra, ren, q, rclk);
    input  wclk;                  // clk for write port
    input  [MemAddrWidth-1:0] wa;  // write address
    input  [1:0] wen;             // write enable - "active low"
    input  [MemDataWidht-1:0] d;   // write data
    input  rclk;                  // clk for read port
    input  [MemAddrWidth-1:0] ra;  // read address
    input  ren;                   // read enable - "active low"
    output [MemDataWidth-1:0] q;   // read data

endmodule
```

- Single port SRAM
  - spsramDxWeM
  - D = depth (number of words)
  - W = bits per word
  - M = bits per write mask bit

example

```
module spsram384x16(clk, a, cen, wen, oen, d, q);
    input  clk;
    input  [MemAddrWidth-1:0] a;
    input  cen;                 // chip enable - "active low"
    input  wen;                 // write enable - "active low"
    input  oen;                 // output enable - "active low"
    input  [MemDataWidth-1:0] d;
    output [MemDataWidth-1:0] q;

endmodule
```

Note The functionality of single port SRAM is same as the one of single port register file. The only difference is area depending on the bit width and depth (number of words). User can use either a single port register file or a single port SRAM based on the area.

Memory instances of the VPU are grouped in the design/rtl_v/top/wave420l_mg_vcore_top.v.

A.2. List of Memories

Table A.1. List of WAVE420L Internal Memories

Module	Memory	Depth	Width	EA.	Byte size
PRP	dpreg128x128	128	128	1	2048
IMD	spreg30x64	30	64	1	240
ME	dpreg256x128	256	128	1	4096
	dpreg128x128	128	128	1	2048
RDO_TOP	spreg128x64	128	64	3	3072
	spreg64x128e64	64	128	1	1024
	spreg64x64e32	64	64	1	512
	spreg48x46	48	46	1	276
	spreg256x26	256	26	1	832
	dpreg16x100	16	100	1	200
	dpreg64x128	64	128	1	1024
	dpreg32x128	32	128	1	512
	dpreg128x64	128	64	1	1024
	dpreg128x32	128	32	1	512
RDO_CU08	spreg48x128	48	128	1	768
	spreg32x64	32	64	1	256
	spreg48x64	48	64	1	384
	spreg32x32	32	32	1	128
RDO_CU16	spreg64x128e64	64	128	1	1024
	spreg128x128e64	128	64	1	1024

Module	Memory	Depth	Width	EA.	Byte size
	spreg128x64e32	128	64	2	2048
	spreg64x64e32	64	64	1	512
	spreg128x32e16	128	32	1	512
	spreg64x16	64	16	4	512
Loopfilter/SAO	dpreg48x128e32	48	128	1	768
	dpreg64x128	64	128	1	1024
	spreg11x128e8	11	128	1	176
	spreg32x128e8	32	128	1	512
	dpreg140x128e8	140	128	1	2240
	dpreg188x128e8	188	128	1	3008
DCI	dpreg32x16	32	16	1	64
	dpreg32x32	32	32	2	256
	spsram408x32	408	32	1	1632
	spsram1024x32	1024	32	1	4096
	dpreg32x64	32	64	1	256
	spsram1920x64	1920	64	1	15360
output buffer	spreg128x128	128	128	1	2048
Sub-pixel WB	spreg64x128e32	64	128	1	1024
current buffer	spreg192x64e16	192	64	7	10752
Reference Buffer FBD for L0	spreg116x128e8	116	128	8	14848
	spreg112x64e8	112	64	4	3584
	spreg32x42	32	42	1	168
	spreg256x128	256	128	1	4096
	spsram512x28	512	28	1	1792
	spsram384x128	384	128	2	12288
	dpreg64x128	64	128	1	1024
tBPU for encoder	spsram3072x16	3072	16	1	6144
	spsram1536x16	1536	16	1	3072
	dpreg64x128e8	64	128	1	1024
	dpreg40x32	40	32	1	160
	spreg24x32	24	32	1	96
	spreg128x64e8	128	64	1	1024
	dpreg32x128e32	32	128	2	1024
vCPU for encoder	dpreg64x128e8	64	128	1	1024
	spreg64x32	64	32	1	256
	spreg32x3e1	32	3	1	12
	spreg32x44e1	32	44	2	352
	spreg512x32e8	512	32	4	8192
	spreg32x5e1	32	5	1	20
	spreg128x64e8	128	64	1	1024
	dpreg16x128	16	128	2	512
	dpreg32x128	32	128	1	512

Module	Memory	Depth	Width	EA.	Byte size
FBC	spreg40x21	40	21	1	105
	spreg40x22	40	22	1	110
	spreg32x29	32	29	1	116
	spreg136x128e16	136	128	1	2176
	dpreg352x128e8	352	128	1	5632
Total				97	138191 (134.95KB)

Note *Table A.1, "List of WAVE420L Internal Memories"* is a preliminary version, so there might be some change.

Appendix B
cnm_defines.v

There are RTL define statements in *wave420l_cnm_defines.v* . The RTL defines are necessary for simulating RTL and synthesis, and most of them are fixed ones. *Table B.1, "Defines for Target Library"* lists the RTL defines that are configurable by user.

Table B.1. Defines for Target Library

Parameter	Description	Configu-ra- bility
ASIC_MEM	It defines use of compiled memory for ASIC synthesis or verification.	YES
TSMC28HPM	It defines TSMC28HPM for ASIC Lib. It is valid only when ASIC_MEM is defined.	YES
FPGA_MEM	It defines use of compiled Xilinx FPGA memory for FPGA synthesis or verification.	YES

About Chips&Media

Chips&Media, Inc. is a leading video IP provider, headquartered in Seoul, South Korea. The company was established in 2003 by leading members with years of profound experiences in video standard technologies and the semiconductor industry. Our extensive catalogue of IP solutions includes video postprocessing, video frame buffer compression as well as video codecs covering vast range of video standards from MPEG-2, MPEG-4, H.263, Sorenson, H.264, VC-1, AVS/AVS+, VP8/9, HEVC(H.265), and AV1 for HD to UHD (4K/8K) resolution.

Chips&Media has been developing a line of reliable, high-quality IP solutions that allow our customers and partners to satisfy the growing consumer demand for high-performance multi-media digital devices. Especially as a leading multi-standard video codec solution provider, we have been providing our advanced ultra-low power multi-codec video IPs to top-tier semiconductor companies.

With a mission to become a global top SoC IP provider, Chips&Media has introduced Image Signal Processing (ISP) IP and Deep learning-based super resolution IP to the market and continues to widen our solution portfolio to cope with growing demand on image processing and related solutions. To find further information, please visit the company's web site at *http://www.chipsnmedia.com*