APV21B - Real-time Video 16X Bicubic Super-resolution IP, AXI4-Stream Video Interface Compatible, 4K 60FPS

Deng LiWei
Nijigasaki IC Design Club

August 11, 2022

INTRODUCTION

The APV21B Real-time Video 16X Bicubic Super-resolution core is a soft IP core. It provides fully real-time 16X Bicubic interpolation video super-resolution, and its high performance design allows it to support video output resolutions in excess of 4K 60FPS.

The APV21B is compatibled with the AXI4-Stream Video protocol as described in the **Video IP: AXI Feature Adoption** section of the *Vivado AXI Reference Guide* (Xilinx Inc. UG1037) and **AXI4-Stream Signaling Interface** section of the *AXI4-Stream Video IP and System Design Guide* (Xilinx Inc. UG934).

FEATURES

- AXI4-Stream Video Interface input/output
- Supported single 8-bit channel input
- Supported 4 pixel per clock (4 x 8-bit) output to reduce the interface frequency
- Real-time Bicubic computation core
- Parametric configurable input resolution
- Output supported to 4K 60FPS (@150 MHz)
- Optimized design for DSP48E2 unit of Xilinx UltraScale+ architecture
- Complete synthesizable HDL design
- Macro switches to using different implementations (DSP Macro/Verilog Inferring)

FACTS TABLE

Core Specifics

Supported Device[1]	Xilinx, Intel and Lattice FPGAs Digital ASICs
Supported User Interface	AXI4-Stream
Resources	*Resources Utilization Section*

Provided with Core

Design Files[2]	System Verilog
Example Design	Provided
Test Bench	Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Microarchitecture Design[3]	Open Source

Tested Design Flows

Design Entry	Vivado Design Suite
Simulation	Mentor Modelsim
Synthesis	Synopsys Synplify Premier and Vivado Synthesis

[1] To maximize the performance and minimize the resource usage, a Xilinx UltraScale+ architecture device is recommended.
[2] Fully synthesizable subset of System Verilog syntax.
[3] Detailed description of the microarchitecture design can be found in this document

Contents

1 Overview

Currently, the resolution of the display devices is getting higher and higher. Outputting high resolution (HR) graphics has become one of the general ability of various image processing systems. However, process HR graphics directly makes processing or rendering systems face greater challenges in terms of resources, bandwidth and storage.

Resources consumption of systems can be reduced if graphics can be processed or rendered at low resolution (LR). Then, a scale subsystem can scaled up the graphics to HR using super-resolution algorithms. Nvidia's *Deep Learning Super Sampling* (DLSS), AMD's *FideltiyFX Super Resolution* (FSR) and Intel's *Xe Super Sampling* (XeSS) are all well-established methods for obtaining HR graphics under LR rendering using super-resolution techniques.

The real-time Bicubic super-resolution IP core is designed to scale up the LR video streams. It is based on the effective and moderately complex Bicubic algorithm, and supports AXI4-Stream video stream input/output.

Figure 1 illustrates the real-time bicubic super-resolution IP block diagram.

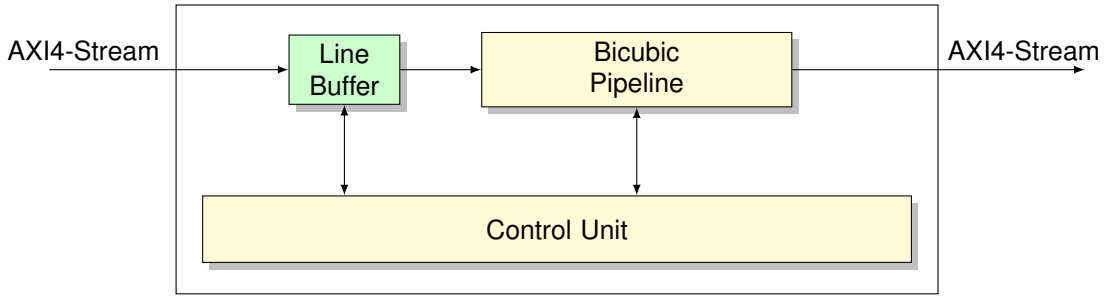

Figure 1: Real-time Bicubic Super-resolution IP Block Diagram

1.1 Feature Summary

AXI4-Stream Video Interface Compliant

The real-time Bicubic super-resolution IP is fully compliant with the AXI4-Stream Video interface. Both input and output inteface has *Start of Frame* and *End of Line* signal.

Detailed protocol of AXI4-Stream Video Interface is provided in **AXI4-Stream Signaling Interface** section of the *AXI4-Stream Video IP and System Design Guide* (Xilinx Inc. UG934).

Please note that the output interface has not TREADY signal which means backpressure blocking on output interface is unsupported. If the design need this function, please consider using an AXI4-Stream FIFO.

AXI4-Stream Data Width of Input Side

The AXI4-Stream interface of input side of the Bicubic IP is 8-bit data width. This 8-bit data can be connected to a channel of 8 bits per channel (BPC) input video stream.

To process multi-channel video, create multiplThe Real-time Video Bicubic Super-resolution IPe instances of the IP, and make them work in parallel. It is recommended that video streams in YUV format be processed for best results.

10 BPC or 12 BPC video streams are not supported in this IP.

AXI4-Stream Data Width of Output Side

The AXI4-Stream interface of output side of the Bicubic IP is 32-bit data width. This 32-bit data has 4x 8-bit pixels parallel output.

When transmitting high resolution video streams, using a higher pixel-clock ratio (or PPC) can reduce clock frequency pressure, and being more complatible with parallel processing units, to improving system performance.

Line Buffer

The Bicubic algorithm uses the pixels around the super-resolution pixels (hereinafter called the 'super-pixels') to perform the computation, which requires the IP core to buffer a certain number of pixels before the computing unit (CU) can start the computation. To maximize compatibility with real-time video streaming data, this IP uses full-line buffering, which using block RAM to buffer the entire line of input frame.

The super-resolution calculation also requires the image to be padded around. Controller of this IP can do the padding automatically with line buffers and pixel block buffers without user intervention.

Note that due to the special line buffering and padding mechanism, please ensure that the input video stream supproted backpressure and blocking (this can supported by using an AXI4-Stream FIFO). Also, please make sure that the resolution and start of frame (SOF) position of input video stream are correct. In this IP, the SOF signal of the input video stream is only used to pass-through to the output video stream and cannot reset the line buffer counter. If an error frame transmission occurs, please use the synchronous reset pin (SCLR) of this IP to reset before starting the correct frame transmission, otherwise, you may encounter an unexpected situation.

Bicubic Scaling Up Pipeline

The Real-time Video Bicubic Super-resolution IP uses a fully pipelined Bicubic interpolation unit. It can scale up the resolution of the input video stream by a fixed factor of 16X (4X each for width and height). The latency of the scaling up process of any video stream is fixed.

Multiple Resolution Support

The Real-time Video Bicubic Super-resolution IP has ability to process various resolutions of videos by modifying parameters. This modification supported in the hardware synthesis phase only, and it is not possible to configure the IP dynamically to support multiple resolutions. Depending on the required resolution, the resource usage of the Real-time Video Bicubic Super-resolution IP may vary.

1.2 Applications

The Real-time Video Bicubic Super-resolution IP provides the ability to scale up the LR video streams to HR in real time.

1.3 Unsupported Features

The following features are not supported by the Real-time Video Bicubic Super-resolution IP.

- Error frame detection and automatic recovery. Error frame includes resolution mismatched and SOF position misaligned.

- Output stream backpressure and blocking.
- Dynamic resolution.

2 Product Specification

2.1 Performance

The Real-time Video Bicubic Super-resolution IP is benchmarking by timing analysis tools in Vivado and Synopsys Synplify Premier. The parameters used in the benchmark test were configured as follows

- Input Video Width: 960
- Input Video Height: 540

Table 1 shows the results of the maximum frequencies benchmark.

FPGA Device Family	**Analysis Tool**	**Fmax (MHz)**
Xilinx Virtex UltraScale+	Synopsys Synplify Premier 2020.03[1]	628.6
Xilinx Kintex UltraScale+	Synopsys Synplify Premier 2020.03[1]	417.2
Xilinx Zynq UltraScale+	Vivado 2021.1[1][2]	428.4
	Synopsys Synplify Premier 2020.03[1]	394.1
Xilinx Kintex 7	Synopsys Synplify Premier 2020.03[3]	361.7
Xilinx Artix 7	Synopsys Synplify Premier 2020.03[3]	192.2
Intel Stratix 10	Synopsys Synplify Premier 2020.03[3]	260.7
Intel MAX 10	Synopsys Synplify Premier 2020.03[3]	220.5
Intel Arria V	Synopsys Synplify Premier 2020.03[3]	142.5

Table 1: Maximum Frequencies

Latency

Table 2 shows the Real-time Video Bicubic Super-resolution IP latency cycles measured on real-time video path. It does not include system dependent latency or throttling. Suppose the width of the input video frame is W.

Description	**Clocks**
Bicubic pipeline input to output	13
First pixel input to First pixel (group) output	$10 \cdot W + 28$
Last pixel input to last pixel (group) output	$13 \cdot W + 31$

Table 2: Latency

[1] Using DSP48E2 Macro.
[2] Using XPM Macro.
[3] Using pure Verilog inferring synthesis.

Throughput

Table 3 shows the Real-time Video Bicubic Super-resolution IP throughput measured for different input frame size.

Input Resolution	Output Resolution	Throughput (FPS/MHz)	FPS @150MHz
320 x 240	1280 x 960	3.23	484.7
480 x 270	1920 x 1080	1.92	287.6
640 x 360	2560 x 1440	1.08	162.1
960 x 540	3840 x 2160	0.48	72.1

Table 3: Throughput

2.2 Resource Utilization

Table 4 shows the Real-time Video Bicubic Super-resolution IP resource utilization on specified FPGA devices with specified settings for reference.

Resource utilization results of Xilinx Zynq UltraScale+ devices is evaluated under with Vivado synthesizer and using DSP48E2 and XPM Macros. The evaluation result of other devices are complete using Verilog automatically inferring, may different caused by the difference of data width of DSP module of each device. The Real-time Video Bicubic Super-resolution IP is specially optimized for DSP48E2 block of Xilinx UltraScale+ series devices. In order to maximum the resource utilization, it is recommended to use these devices for synthesis.

Device	Configuration Parameters		Resource Utilization			
	Input Resolution	fCLK (MHz)	LUTs	FFs	DSPs	BRAMs
XCZU15EG	960 x 540	300	431	694	49[1]	2.5[3]
XC7K325T	960 x 540	150	1979	2713	30[2]	2[3]

Table 4: Resource Utilization

[1] Xilinx UltraScale+ architecture DSP48E2 unit
[2] Xilinx 7 series FPGA DSP48E1 unit
[3] Xilinx FPGA Block RAM 36K unit

2.3 Port Descriptions

This section describes the details for each interface. The Real-time Video Bicubic Super-resolution IP signals are described in Table 5

Signal Name	Interface	Signal Type	Description
Clock, Reset and Clock Enable Signals			
clk	Clock	I	AXI4-Stream interface clock
aresetn	Reset	I	Active-Low asynchronous reset. When asserted Low, resets entire Real-time Video Bicubic Super-resolution core.
dsp_reset	Reset	I	Active-High synchronous reset. When asserted High, resets all DSP blocks (or computation pipelines). Must be synchronous to aclk and asserted for a minimum of sixteen clock cycles.
bram_reset	Reset	I	Active-High synchronous reset. When asserted High, resets all BRAM blocks (line buffers). Must be synchronous to aclk and asserted for a minimum of 16 clock cycles.
sclr	Reset	I	Active-High synchronous reset. When asserted High, resets all control units (line and column counters). Must be synchronous to aclk and asserted for a minimum of 16 clock cycles.
clken	Clock Enable	I	Active-High clock enable. When asserted High, the entire IP has clock to work, otherwise the clock is disabled.
AXI4-Stream Video Input Interface Signals			
s_axis_video_in*	S_AXIS_VIDEO_IN	-	AXI4-Stream Video Interface (Sink)
AXI4-Stream Video Output Interface Signals			
m_axis_video_out*	M_AXIS_VIDEO_OUT	-	AXI4-Stream Video Interface (Source)

Table 5: I/O Signal Description

2.4 Timing Diagrams

For the timing diagrams of the AXI4-Stream Video Interface, you can refer to the **AXI4-Stream Signaling Interface** section of the *AXI4-Stream Video IP and System Design Guide* (Xilinx Inc. UG934).

3 Algorithm Description

In mathematics, bicubic interpolation is an extension of cubic interpolation for interpolating data points on a two-dimensional regular grid. The interpolated surface is smoother than corresponding surfaces obtained by bilinear interpolation or nearest-neighbor interpolation.

In image processing, bicubic interpolation is often chosen over bilinear or nearest-neighbor interpolation in image resampling. In contrast to bilinear interpolation, which only takes 4 pixels (2x2) into account, bicubic interpolation considers 16 pixels (4x4). Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.

3.1 Terminology

Original-image The source (or input) image of the super-resolution algorithm, which has low resolution (W_i, H_i).

Original-pixels Pixels in the *Original-image*. The index of these pixels are from $(0, 0)$ to (W_i, H_i)

Super-image The result (or output) of the super-resolution algorithm. Its resolution (W_o, H_o) is a specific multiple of the *Original-image* ($k \cdot (W_i, H_i)$).

Super-pixels Pixels in the *Super-image*. The index of these pixels are from $(0, 0)$ to (W_o, H_o).

Pixel reference In the image interpolation algorithm, a new *Super-pixel* is generated from the *Original-pixels* (These pixels are generally around the target *Super-pixel*). The *Original-pixels* associated with a new *Super-pixel* are called the *Reference of this Super-pixel*.

Super-block The *Super-block* refers to the area between every four 2 x 2 *Original-pixels*. At a super-resolution of 4 (i.e. $(W_o, H_o) = 4 \cdot (W_i, H_i)$), this area is filled with 16 *Super-pixels*, which are referenced to the 16 *Original-pixels* surrounding the area. In other words, the *Original-pixels* referenced by the *Super-pixels* within a *Super-block* are identical, except that the distance between each *Super-pixel* and the *Original-pixel* is different.

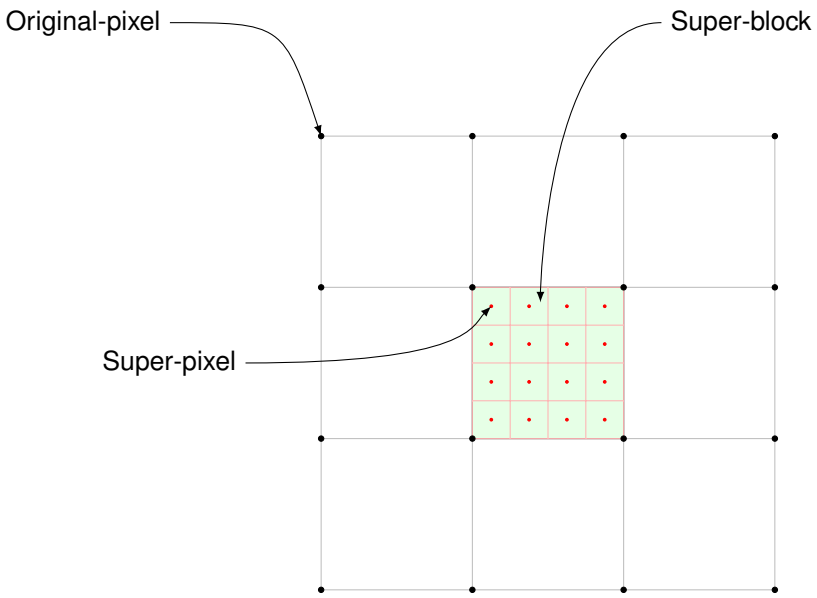

Figure 2: Schematic of reference pixels of the Super-block, Original-pixels, Super-pixels

3.2 Bicubic Algorithm Process

As a whole, the Bicubic interpolation algorithm computes the distance between the center of the *Super-pixel* and the center of the *Original-pixel*. Then, calculates the coefficient of this pair by a Bicubic interpolation function with the distance. Use this coefficient to multipiles with the value of the *Original-pixel*. Finally, the total of the product of 16 *Original-pixels* in the *Super-block* is the value of new *Super-pixel*.

Here is the formula of the Bicubic algorithm.

$$\mathrm{SP}_{\vec{s}} = \sum_{\vec{o} \in \mathrm{RSB}} f_B\left(|\vec{s} - \vec{o}|\right) \cdot \mathrm{OP}_{\vec{o}}$$

where $\mathrm{SP}_{\vec{s}}$ is the value of the *Super-pixel*, index is $\vec{s}$. RSB is the set of all *Original-pixels* index of the *Super-block* where the current *Super-pixel* is located. $OP_{\vec{o}}$ is the *Original-pixel* value on the index $\vec{o}$. The function f_B is the Bicubic interpolation, which returns a distance-dependent coefficient.

3.3 Bicubic Interpolation Function

The Bicubic interpolation function is a function dependent on distance between *Super-pixel* and reference *Original-pixel*. Denoting the distance by $|x|$, the Bicubic interpolation function takes the form

$$f_B(|x|) = \begin{cases} (\alpha+2)\cdot|x|^3 - (\alpha+3)\cdot|x|^2 + 1 & \text{, if } |x| \le 1 \\ \alpha\cdot|x|^3 - 5\cdot\alpha\cdot|x|^2 + 8\cdot\alpha\cdot|x| - 4\cdot\alpha & \text{, if } 1 < |x| < 2 \\ 0 & \text{, otherwise} \end{cases}$$

where $\alpha = -0.5$ for a general image quality.

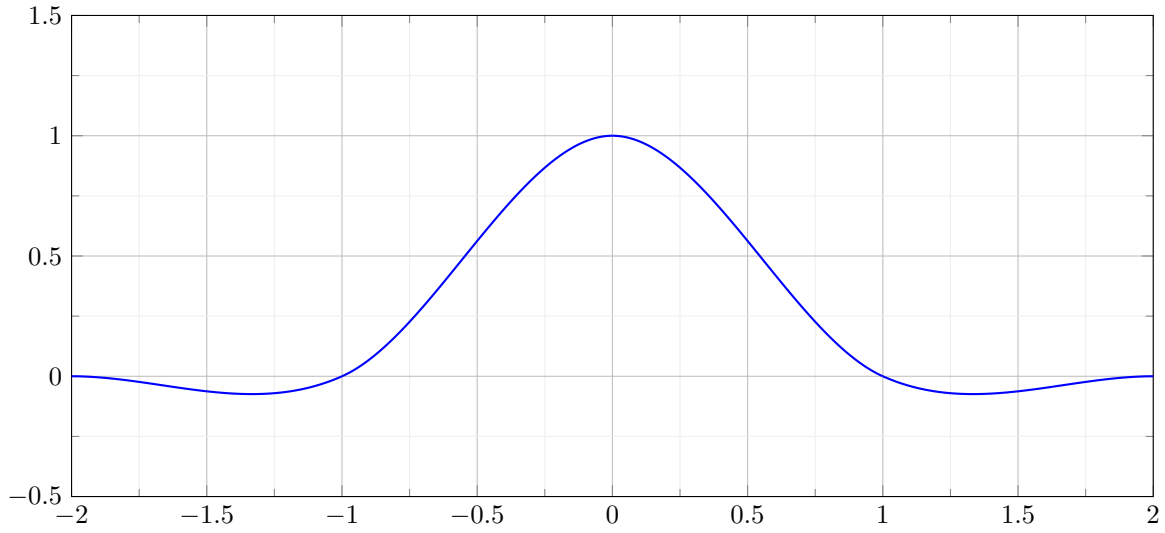

Figure 3: Plot of the Bicubic Interpolation Function

3.4 Super-pixel Symmetry in Super-block

The Bicubic interpolation function has one and only one return value at the same distance input, which means that any two *Super-pixels* and *Original-pixels* at same distance have the same coefficient. In a *Super-block*, there exist several groups of *Super-pixels* and *Original-pixels* pair, which have the same distance between each of them, because of their symmetric positions. For example, a symmetric group of the 4x super-resolution *Super-block* is shown in Figure 4. The four blue arrows shows the distance of corresponding *Super-pixel*-*Original-pixel* pair are same.

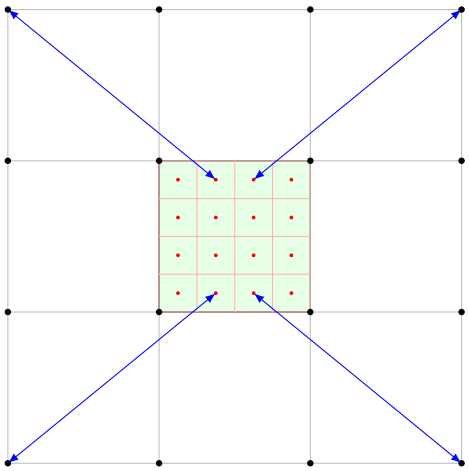

Figure 4: Example of a Symmetric Group of a 4x Super-resolution Super-block

Divide the 4x super-resolution *Super-block* into 4 parts, and then number the *Original-pixels* and *Super-pixels* separately as $O(0,0)$ to $O(3,3)$ and $S(0,0)$ to $S(3,3)$ like shown in the Firuge 5. Denote the distance from any *Super-pixel* $S(x,y)$ to any *Original-pixel* $O(m,n)$ as $d\left[(x,y) \rightarrow (m,n)\right]$.

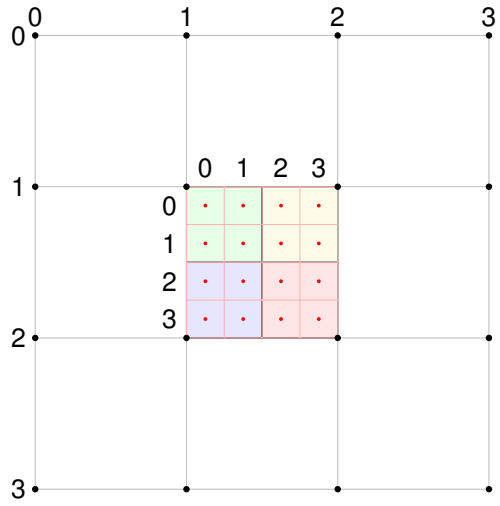

Figure 5: Symmetric Groups in the Super-block

We can find that for a single *Super-pixel*, there are 16 distance values (i.e. 16 coefficients) to 16 *Original-pixels*. For example, the *Super-pixel* $S(0,0)$ has $d\left[(0,0) \rightarrow (0,0)\right]$, $d\left[(0,0) \rightarrow (1,0)\right]$, ..., $d\left[(0,0) \rightarrow (3,3)\right]$. However, when we consider the symmetry of the *Super-pixel* in the *Super-block*, we can find that

$$d\left[(0,0) \rightarrow (0,0)\right] = d\left[(3,0) \rightarrow (3,0)\right] = d\left[(0,3) \rightarrow (0,3)\right] = d\left[(3,3) \rightarrow (3,3)\right]$$

In this way, we only need to consider the coefficients of the *Super-pixels* in the green part of Figure 5. We can find the coefficients of the remaining part of the *Super-pixels* by symmetry. The yellow and green parts are mirrored vertically along the center; the blue and green parts are mirrored horizontally along the center; and the red part is mirrored both vertically and horizontally. Using this property, the number of calculations of the coefficients is reduced to a quarter.

3.5 Pixel Alignment and Edge Padding of the Super-pixels

Super-pixels are not exist on the input LR image. Any *Super-pixels* indexed by $S(x,y)$ need mapped to the absolute coordinate on the input LR image using this way as follows:

$$\vec{S}_{map}(x_m, y_m) = \frac{\vec{S}(x,y) + (0.5, 0.5)}{k}$$

where k is the scale coefficient.

When $k = 4$, the top-left *Super-pixel* is mapped to the coordinate $(-0.375, -0.375)$. This means that the center of the top-left *Super-pixel* $S(0,0)$ is on the left and top of the *Original-pixel* $O(0,0)$.

In Figure 6, we can see that the center of $S(0,0)$ is located to the left and top of the center of O(0,0) and belongs to the *Super-block* in the blue box. 16 pixels from O(-2,-2) to O(1,1) are referenced to this *Super-block*.

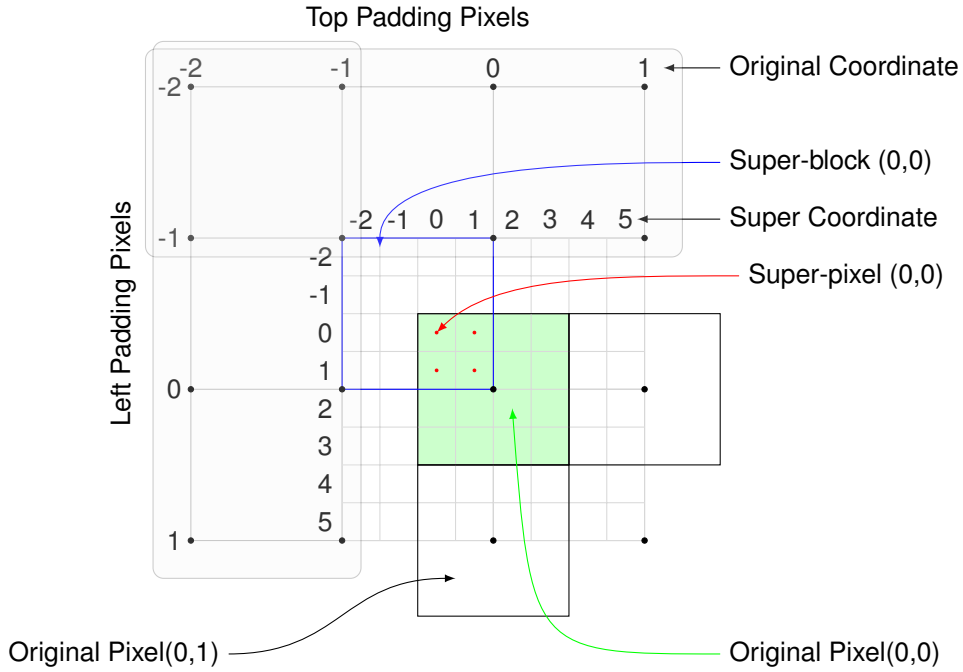

Figure 6: Super-pixel Alignment and Padding at the Top-Left of the Image

Note that $S(0,0)$ is not at postion $(0,0)$ but at position $(2,2)$ in the *Super-block*. If $S(0,0)$ is incorrectly computed using the coefficients of the $(0,0)$ position in *Super-block*, it will cause pixel alignment problems and affect the output result.

Since the *Original-image* does not have pixel values in negative coordinates and the coordinates out of the image size for reference, we must padding 2 rows and 2 columns of the edge of the image to provide them. One way is to padding the pixel using the nearest pixel, i.e., copy 2 rows(or 2 columns) of pixels on the edge of the *Original-image*. In this way, when we using the value of $O(-1,0)$ or $O(1,-2)$, we are actually using the value of $O(0,0)$ and $O(1,0)$.

After pixel alignment and padding, the entire super-resolution image has (W_i+1) x (H_i+1) *Super-blocks*, where only 2 rows (or columns) of *Super-pixels* are used in the boundary *Super-blocks*. And only a quarter *Super-pixels* of the *Super-pixels* located on the corners are used.

4 Microarchitecture Design

The Real-time Video Bicubic Super-resolution IP provides a complete Bicubic processing implementation. Including a real-time Bicubic interpolation pipeline, a line buffer module and a control unit for controlling the above 2 modules.

Because of the requirements of padding and pixel alignment processing, this IP also contains a pixel realignment module.

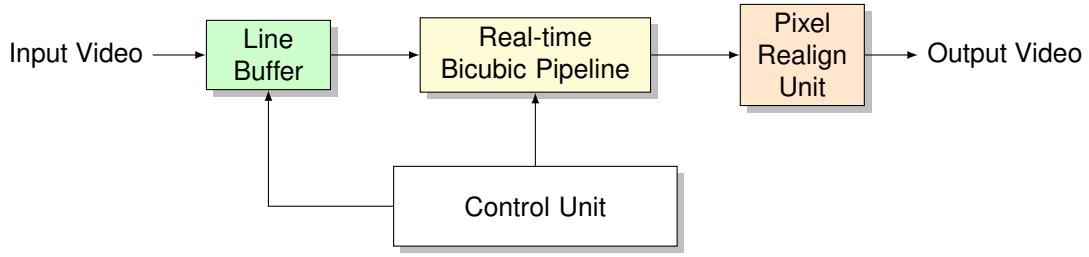

Figure 7: Real-time Video Bicubic Super-resolution IP Processing Units

4.1 Operational Bit-width and Quantization

In order to save resource consumption, maximize DSP resource utilization, and to optimize for the DSP48E2 unit, the Real-time Video Bicubic Super-resolution IP performs 9-bit signed quantization of the Bicubic coefficients during the operation, which ensures image quality while fully utilizing the multiplier of the DSP48E2 unit. This quantization method supports 8-bit image channel inputs.

4.2 Real-time Bicubic Pipeline

The real-time Bicubic pipeline contains a *Super-block* buffer, a Bicubic coefficient lookup table (LUT), a set of parallel multiplier-adder units, a set of parallel adders, a set of parallel rounding units and a limit unit. The structure of the pipeline is shown in Figure 8.

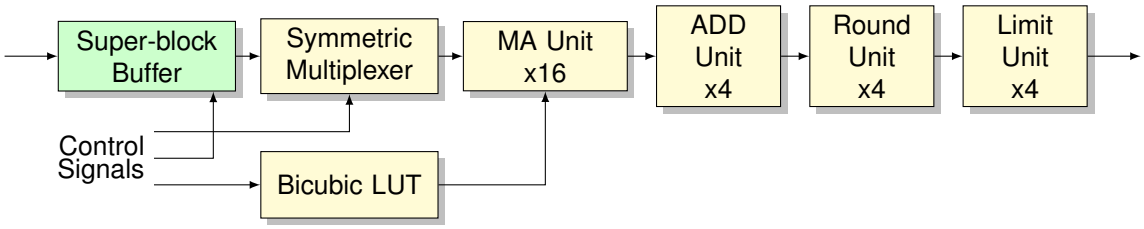

Figure 8: Real-time Bicubic Pipeline

The value of each *Super-pixel* is the sum of the product of 16 referenced *Original-pixel* and the corresponding Bicubic coefficient. A single *Super-pixel* requires at least 16 multiplication and addition operations for it.

The Real-time Video Bicubic Super-resolution IP performs 4 *Super-pixels* per clock cycle.

4.2.1 Super-block Buffer

The *Super-block* Buffer sotres the current *Reference-pixels* used by the *Super-block* for computation. It has 4 pixel inputs which shifts in 1-column x 4-row of *Original-pixels* each clock cycle. It also has an 4-column x 4-row pixel buffer matrix. Each row of which is a shift register that allows the *Super-block* to slide horizontally on the *Original-image*.

The *Super-block* Buffer can output 16 buffered *Original-pixels* at a clock cycle for computation. To meet the needs of pixel alignment and edge padding, the data of each output column is selected using a multiplexer. There are five modes of the multiplexer, as shown in Table 6, available for the 4 padding cases at the left and right edges of the *Original-image*, and the normal case at the middle of the *Original-image*.

Mode	**Column Outputs**				**Description**
	0	**1**	**2**	**3**	
0000	Col0	Col1	Col2	Col3	Normal
0001	Col1	Col1	Col2	Col3	Padding left 1 column
0010	Col2	Col2	Col2	Col3	Padding left 2 columns
0100	Col0	Col1	Col2	Col2	Padding right 1 column
1000	Col0	Col1	Col1	Col1	Padding right 2 columns

Table 6: Super-block Buffer Output Multiplexer Modes

Figure 9 shows the structure of the *Super-block* Buffer.

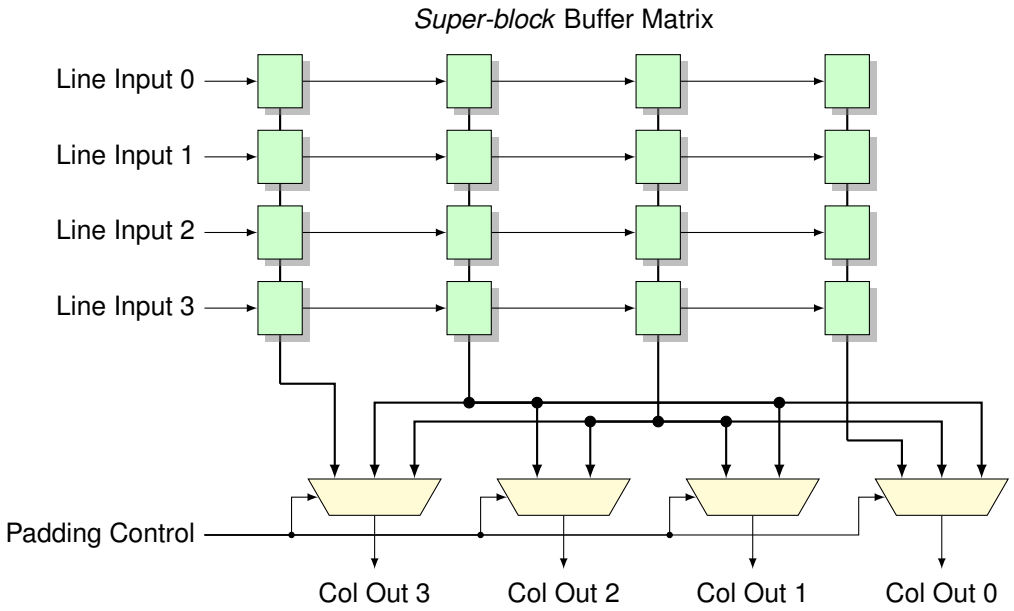

Figure 9: Diagram of the Super-block Buffer

4.2.2 Symmetric Multiplexer

The Symmetric Multiplexer is used to flip the pixel matrix of the Super-block Buffer output vertically. We are using symmetry to reduce the size of the Bicubic LUT. When the Super-pixel to be computed is located in row 2 or 3 of a Super-block, the coefficient of its corresponding Original-pixel is vertically mirrored with the Super-pixel located in row 1 or 0. For example, in Figure 10, $d[(1,2) \rightarrow (0,3)] = d[(1,1) \rightarrow (0,0)]$(Red arrows) and $d[(2,3) \rightarrow (2,1)] = d[(2,0) \rightarrow (2,2)]$(Blue arrows).

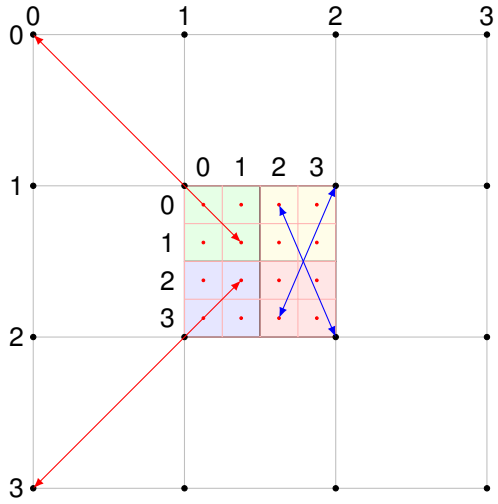

Figure 10: Example of Vertical Symmetric in a Super-block

Thus, by directly mirroring the Original-pixels in the entire *Super-block* horizontally along the center, it is possible to compute the values of the *Super-pixel* located in rows 1 or 2 without changing the order of the coefficients.

4.2.3 Bicubic Coefficient LUT

The Bicubic Coefficient LUT is used to lookup the Bicubic coefficients used in the product with the *Original-pixels*. This IP exploits the symmetry of *Super-blocks* in horizontal and vertical directions. In the horizontal direction, *Super-pixels* located in columns 2 or 3 (in each *Super-block*) can using the coefficients of columns 1 or 0. In the vertical direction, *Super-pixels* located in rows 2 or 3 can using the coefficients of rows 1 or 0. This allows the Bicubic Coefficient LUT to store a total of only 2-row x 2-*Super-pixel* x 16 coefficients. The Bicubic Coefficient LUT outputs 1-row x 2-*Super-pixel* x 16 coefficients each clock cycle. The outputs of rows located in 0 and 3 are same, and the outputs of rows located 1 and 2 is also the same. The symmetric conversion is done by the Symmetric Multiplexer.

Coefficient output count in single clock cycle in the Bicubic Coefficient LUT can meet the requirement of 4 Super-pixels output in a single clock cycle of this IP.

4.2.4 Multiplier Adder Unit (MA Unit)

The Real-time Bicubic Pipeline contains 16 MA Units, each of them multiplies 4 sets of *Original-pixels*-Coefficients and adds them two by two in one cycle.

The mathematical structure of the MA unit is shown in Figure 11.

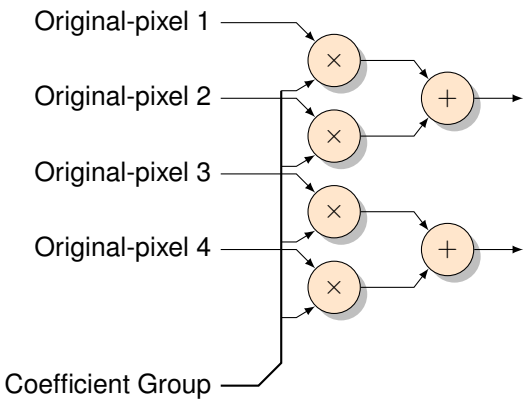

Figure 11: Mathematical Structure of MA Unit

Using the horizontal symmerty, when computing 4 *Suepr-pixels* located in the same row of a *Super-block* in parallel, a single Bicubic coefficient can be multiplied with 2 different *Original-pixels* for different *Super-pixels*. For example, since the coefficient $C_0 = f_B(d[(0,0) \rightarrow (0,0)]) = f_B(d[(3,0) \rightarrow (3,0)])$, multiplying the *Original-pixel* located at (0,0) and (3,0) in the *Super-block* reference pixel matrix with C_0 at same time, the partial values used for *Super-pixel* (0,0) and (3,0) can be obtained, respectively.

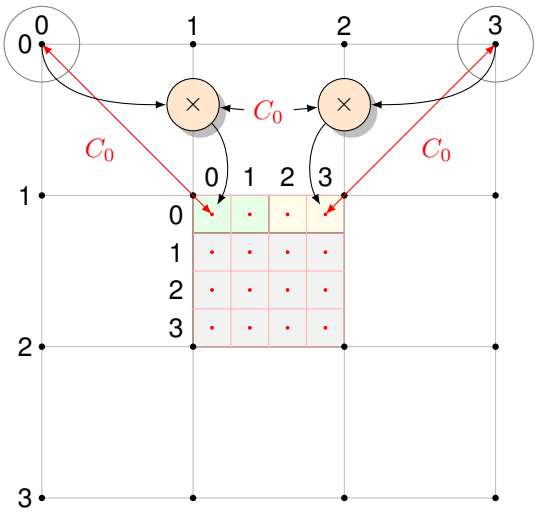

Figure 12: Example of Multiplication Operation Using Horizontal Symmetry

Split-multiplier Structure (SMS) To save multiplier resources and fully utilize the 18x27-bit multiplier of the DSP48E2 slice in the Xilinx UltraScale+ FPGA architecture, this IP innovatively proposes a single DSP48E2 split-multiplier structure (SMS), which splits a signal 18x27-bit multiplier into two 8x9-bit multipliers. This design not only halves the resource consumption of the multiplier, but also reduces the length of the wiring and improves the IP performance.

The SMS structure uses the mathematical relationship of binary multiplication to split the multiplier. The basic principle is derived from the following equation

$$M = C \cdot (B \cdot 2^n + A) = C \cdot A + C \cdot B \cdot 2^n$$

where C is the common coefficient to be multiplied and A and B are the two multipliers.

When we want to compute $M_1 = A \cdot C$ and $M_2 = B \cdot C$, we can first shift B left by n bits, i.e., $B \cdot 2^n$, then add it to A, and multiply it with C. Get the result $M = C \cdot (B \cdot 2^n + A)$, which is the result of adding M_1 and M_2 shifting left by n bits.

In binary multiplication, if the multipliers A and B are both x-bit binary numbers and C is a y-bit binary number, then neither $A \cdot C$ nor $B \cdot C$ will result in more than $x + y$ bits.

Therefore, when $n > x + y$, M_1 in $M = C \cdot (B \cdot 2^n + A)$ will not round to the high bit where M_2 is, so that M_1 and M_2 can be separated directly. Of course, when computing with signed numbers, if the result of M_1 is negative, it will cause it to round to the high bit by 1 bit, which can be compensated by the sign bit of M_1.

In this IP, the 9-bit signed Bicubic coefficients and 8-bit unsigned pixel data are used to multiply. Due to the symmetry of the Bicubic coefficients, the form of the split multiplication $M = C \cdot (A + B)$ is exactly satisfied, so that the splitting method can be used to operate two multiplication operations in a single 18x27-bit multiplier in a single clock cycle.

In this IP, the SMS structure multiply unit contains an 18x27-bit multiplier that inputs 2 *Original-pixels* A and B and a Bicubic coefficient input C. The 18-bit input of the multiplier is directly connected to the C input. And the 27-bit input of the multiplier is connected to the merged lines consists of A and B which is left-shifted by 18 bits, i.e., $A + B \cdot 2^{18}$.

Thus, in the output result of the multiplier, the bits [17:0] are the result M_1 and the bits [35:18] are the result M_2. Since the Bicubic coefficients are signed, the result of M_2 also needs to be compensated based on the most significant bit (sign bit) of M_1. The computation is shown in the Figure 13.

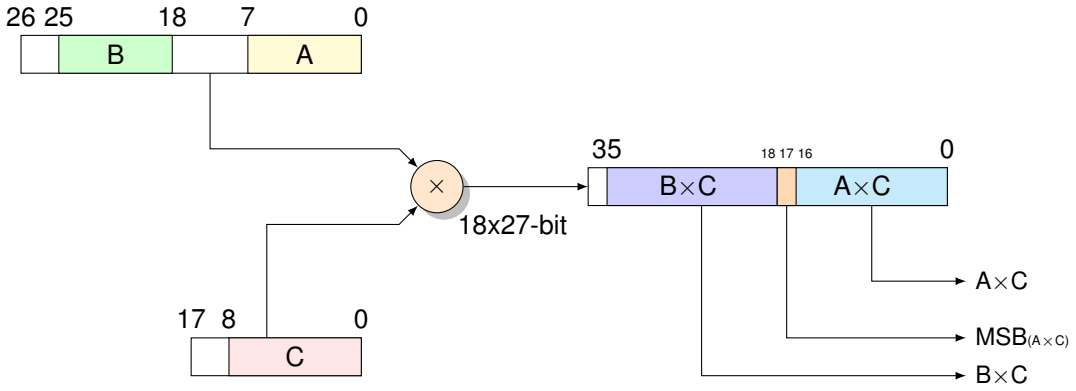

Figure 13: Bicubic Computation Process of SMS Structure

By connecting two such SMS units in series, an MA unit is obtained. The two SMS cells are used as two stages. The 1st stage can perform 2 multiplications (2 *Original-pixels* x 1 Coefficent), and output 2 results in parallel. The 2nd stage, while completing the 2 multiplications, can add the result of the 1st stage and 2nd stage together using the 48-bit adder in the DSP48E2 slice (or DSP slice). And finally when a MA unit completes the operation, it can get 2 summation result in parallel. The structure and data flow of the MA unit is shown in Figure 14.

The MA unit uses the equations are

$$M = C_1 \cdot \left(A_1 + B_1 \cdot 2^{18}\right) + C_2 \cdot \left(A_2 + B_2 \cdot 2^{18}\right)$$

$$M_1 = M_{[17:0]} = C_1 \cdot A_1 + C_2 \cdot A_2$$

$$M_2 = M_{[35:18]} - M_{[17]} = C_1 \cdot B_1 + C_2 \cdot B_2 - M_{[17]}$$

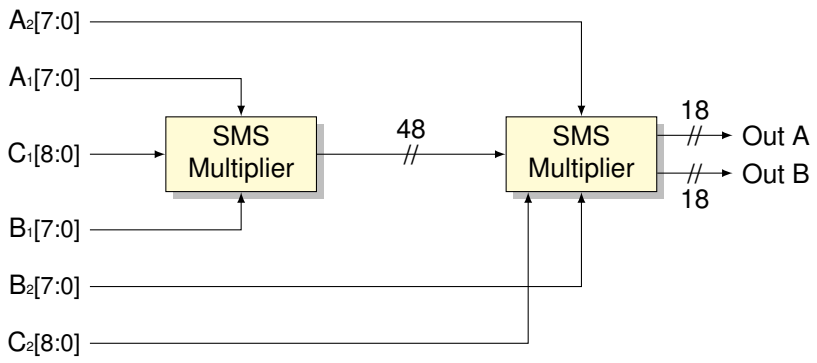

Figure 14: Structure and Data Flow of MA Unit

Because of the feature of the SMS structure for signed multiplication, all the *Original-pixels* input as B need to be compensated with the most significant bit of the result of $C \cdot A$ from same SMS unit.

Low Segment and High Segment We call the result of the A-way computation without compensation *Low Segment*, and the result of the B-way computation with compensation *High Segment*.

Bicubic Coefficient Assignment Taking advantage of the symmetry of the Bicubic coefficients, this IP assigns the coefficients and *Original-pixels* to the 16 MA units reasonably, and the assignment results are shown in Table 7. Each word string of the form $X \rightarrow (x, y)$ refers to a Bicubic coefficient. The X means X-direction index of the *Super-pixel* in the *Super-block* and (x, y) is the corresponding *Original-pixel* in the *Super-block*.

With this assignment, each *Super-pixel* has 4 data in *Low Segment* and 4 data in *High Segment* of the 4 *Super-pixels* to be computed in a single clock cycle. In this way, we can input the 8 results of each *Super-pixel* into an 8-input 4-carry adder and get the result value of the *Super-pixel*.

Multiplication result of *Super-pixel* contains 4 *High Segment* results that need to be compensated, which can be compensated by the most significant bit of the corresponding *Low Segment* feed to the 4 carry inputs.

Table 7: Multiplier Adder Unit Coefficient Assignments

DSP Unit	**DSP Stage 1**			
	Coefficient	**Symmetry**	**Coefficient**	**Symmetry**
0	$0 \rightarrow (0,0)$	$3 \rightarrow (3,0)$	$0 \rightarrow (0,1)$	$3 \rightarrow (3,1)$
1	$1 \rightarrow (0,0)$	$2 \rightarrow (3,0)$	$1 \rightarrow (0,1)$	$2 \rightarrow (3,1)$
2	$2 \rightarrow (0,0)$	$1 \rightarrow (3,0)$	$2 \rightarrow (0,1)$	$1 \rightarrow (3,1)$
3	$3 \rightarrow (0,0)$	$0 \rightarrow (3,0)$	$3 \rightarrow (0,1)$	$0 \rightarrow (3,1)$
4	$0 \rightarrow (1,0)$	$3 \rightarrow (2,0)$	$0 \rightarrow (1,1)$	$3 \rightarrow (2,1)$
5	$1 \rightarrow (1,0)$	$2 \rightarrow (2,0)$	$1 \rightarrow (1,1)$	$2 \rightarrow (2,1)$
6	$2 \rightarrow (1,0)$	$1 \rightarrow (2,0)$	$2 \rightarrow (1,1)$	$1 \rightarrow (2,1)$

Continued on next page

Table 7: Multiplier Adder Unit Coefficient Assignments (Continued)

7	$3 \rightarrow (1,0)$	$0 \rightarrow (2,0)$	$3 \rightarrow (1,1)$	$0 \rightarrow (2,1)$
8	$0 \rightarrow (0,2)$	$3 \rightarrow (3,2)$	$0 \rightarrow (0,3)$	$3 \rightarrow (3,3)$
9	$1 \rightarrow (0,2)$	$2 \rightarrow (3,2)$	$1 \rightarrow (0,3)$	$2 \rightarrow (3,3)$
10	$2 \rightarrow (0,2)$	$1 \rightarrow (3,2)$	$2 \rightarrow (0,3)$	$1 \rightarrow (3,3)$
11	$3 \rightarrow (0,2)$	$0 \rightarrow (3,2)$	$3 \rightarrow (0,3)$	$0 \rightarrow (3,3)$
12	$0 \rightarrow (1,2)$	$3 \rightarrow (2,2)$	$0 \rightarrow (1,3)$	$3 \rightarrow (2,3)$
13	$1 \rightarrow (1,2)$	$2 \rightarrow (2,2)$	$1 \rightarrow (1,3)$	$2 \rightarrow (2,3)$
14	$2 \rightarrow (1,2)$	$1 \rightarrow (2,2)$	$2 \rightarrow (1,3)$	$1 \rightarrow (2,3)$
15	$3 \rightarrow (1,2)$	$0 \rightarrow (2,2)$	$3 \rightarrow (1,3)$	$0 \rightarrow (2,3)$

4.2.5 Add Unit

The Real-time Bicubic Pipeline contains 4 8-input 4-carry adders used to sum up all multiplication results of *Super-pixel*.

The arithmetic equation of the 8-input 4-carry adder is

$$Y = A + B + C + D + E + F + G + H + C_0 + C_1 + C_2 + C_3$$

where A to H is 18-bit multiplication result input, and C_0 to C_3 is the 1-bit carry input, used to compensate for the multiplication results from *High Segment*.

In order to perform so many addition operations in fewer cycles using a minimum number of DSP units, this IP is optimized for the DSP48E2 slice. An 8-input 4-carry adder consists of 2 3-input adders with carry input and 2 DSP cascaded adders with carry input. Their respective structure are shown in the Figure 15 and Figure 16.

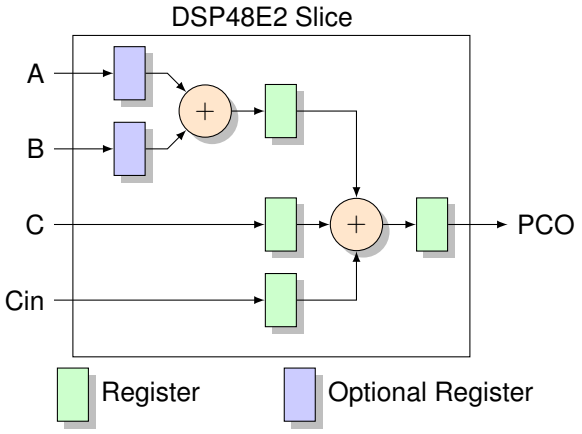

Figure 15: 3-input Adder Implemented by DSP48E2 Slice

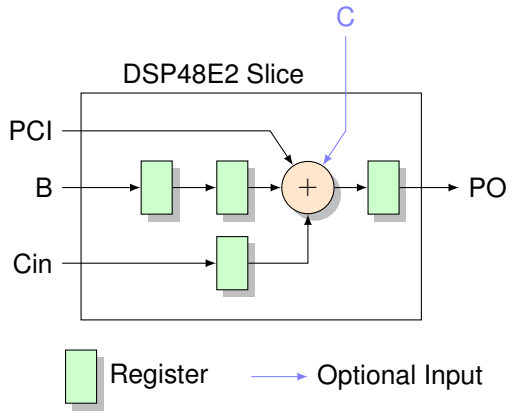

Figure 16: DSP Cascade Adder Implemented by DSP48E2 Slice

The DSP48E2 architecture has a DSP Slice Cascade Function that enables the use of cascaded outputs (PCO) and inputs (PCI) to minimize the data path length between two DSP48E2 Slices. The 8-input 4-carry adders in this IP use this design when optimizing for the DSP48E2 Slice, cascading multiple DSP48E2 slices through cascade pins to achieve an ultra-low latency adder unit. The specific structure of which is shown in the Figure 17.

The latency of the 8-input 4-carry adders in this IP is only 4 clock cycles.

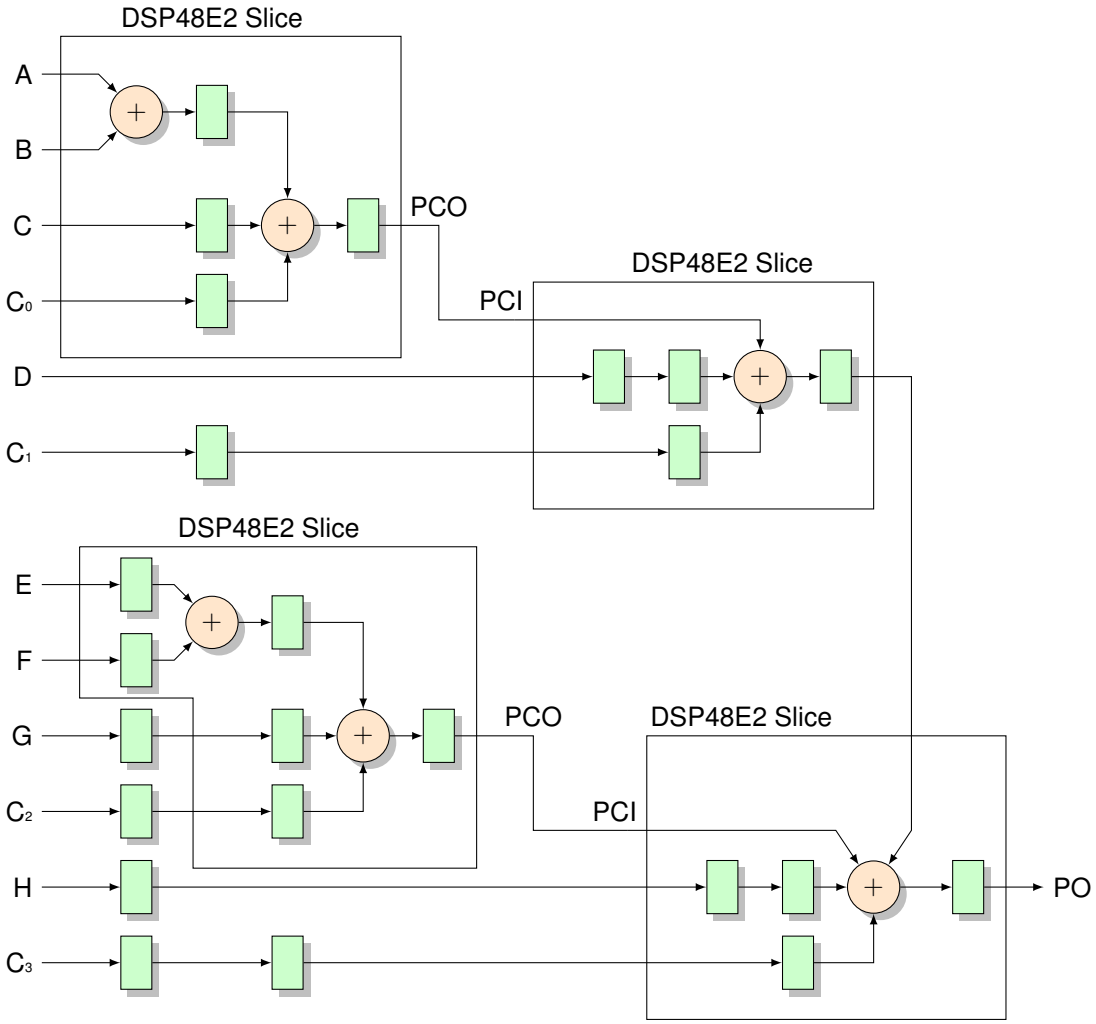

Figure 17: Structure of 8-Input 4-Carry Adder

4.2.6 Round Unit and Limit Unit

Since the fixed-point quantization is used to the Bicubic coefficients, the result of *Super-pixels* must be right-shifted and rounded before it output, and the output is limited to the standard 8-bit pixel values (0 to 255).

The Real-time Bicubic Pipeline uses a SIMD 4x12-bit adder for the rounding operation (optimized for the DSP48E2). All rounding and limiting operations are done in 2 clock cycles.

4.3 Line Buffers

There are 5 line buffers in Real-time Video Bicubic Super-resolution IP. All of them can be dynamically configured to either *working* or *buffering* state. At the same time, 4 of them will be working lines, and the remaining one will be buffering line. So that the input video stream data will not affect the super-pixel calculation. After the working lines have been calculated, the controller will automatically switch

to make the latest buffered line participate in the calculation, and configure the oldest line as buffering line to continuously receive new video pixels.

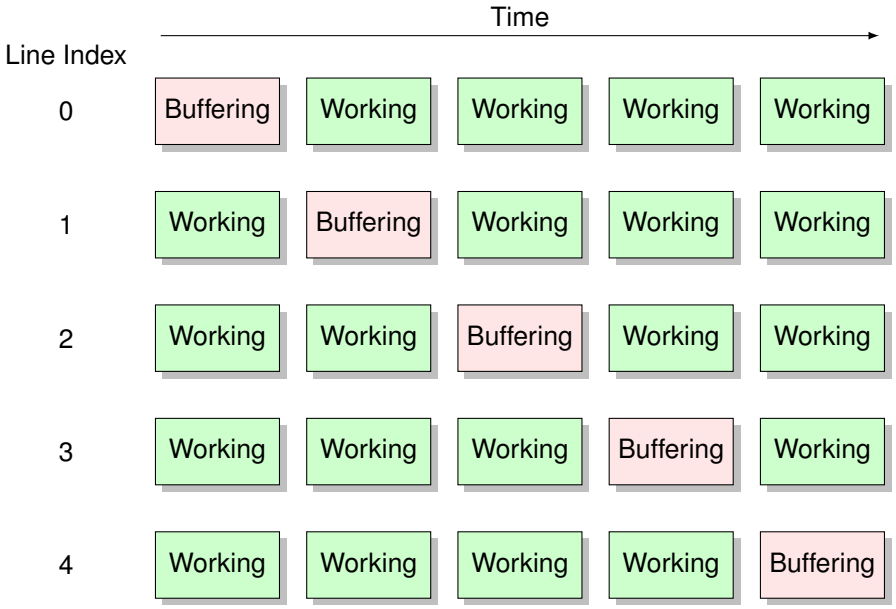

Figure 18: Working Lines and Buffering Line in Line Buffer

The line buffer contains an AXI4-Stream Video Sink interface that can be connected directly to the external video input stream. This interface has a blocking signal (TREADY) that blocks the input stream while edge padding is performed on the input video. The line buffer also has internal line counter for controlling the working lines and buffering line; and line pixel counter, for controlling the pixel out.

The line buffer outputs 1 column x 4 rows pixels from working lines in a clock cycle. Since a *Super-block* contains 4 rows of *Super-pixels*, and the *Original-pixels* corresponding to each of them are the same, the pixel counter in line buffer will perform a repeated 4 output.

The 1 column x 4 rows pixel data output from the line buffer will be directly connected to the *Super-block* buffer of pipeline. Due to the padding of the left and right edges of each line, a handshaking protocol is available between the line buffer and the pipeline controller in order to block the data input during padding.

The diagram of the line buffer is shown in the Figure 19;

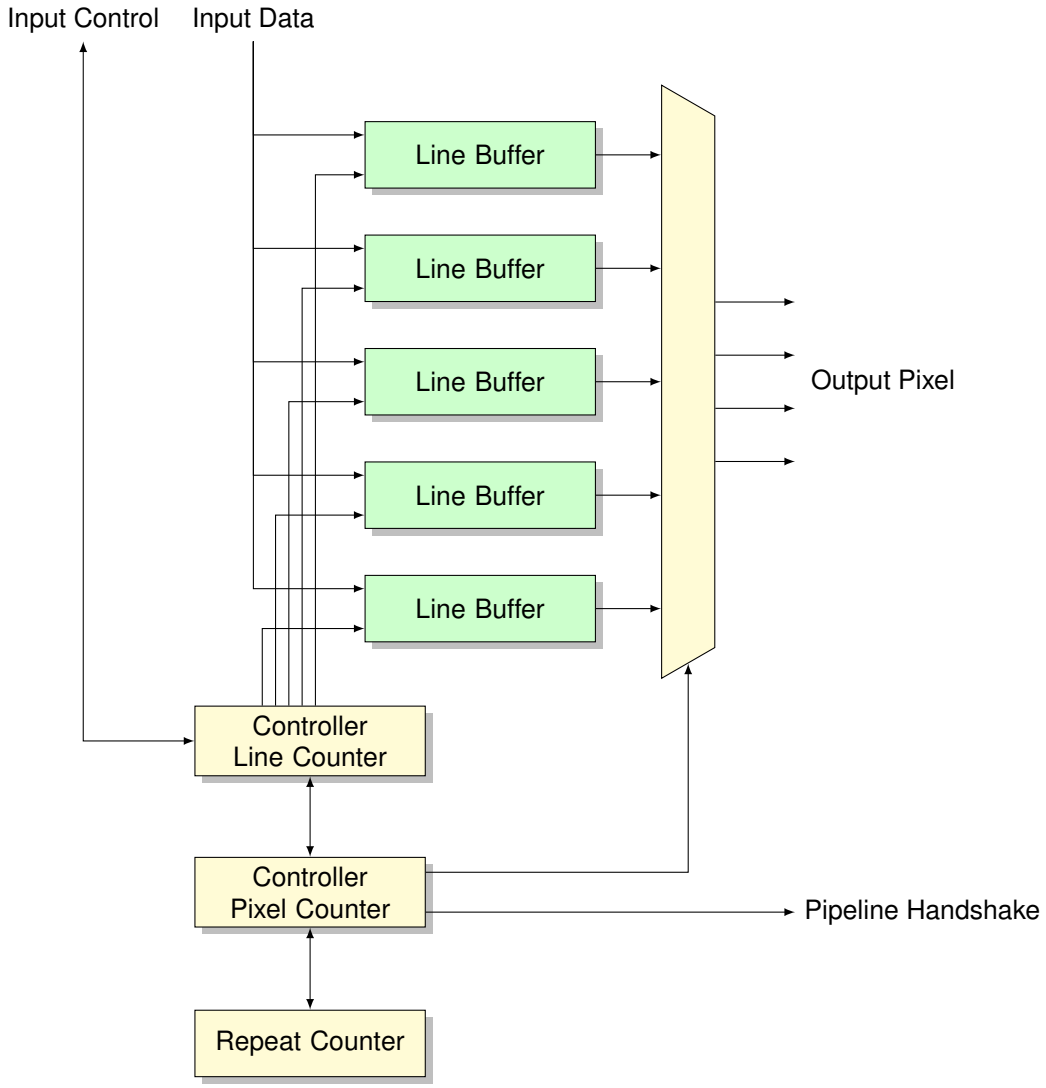

Figure 19: Diagram of Line Buffer

Padding Control in Line Buffer The line buffer blocks the input from the AXI4-Stream Video Sink interface while padding on the top and bottom eages of the frame. Each input frame has $H+1$ *Super-blocks* in the Y-direction, and we takes the time of 1 *Original-image* line input to process 1 line of *Super-blocks*, so each frame require at least 1 line of blocking time.

The line buffer will block while processing the $H-1$-line *Super-block* (when the last line of the previous frame is already stored in the line buffer), blocking the first line of the new frame until the $H-1$-line *Super-block* completes its operation. When the H-line *Super-block* starts its operation, new frames are allowed to be received again. If the input video stream is continuous, by the time the H-line and $H+1$-line *Super-block* operations are completed, the first 2 lines is already stored in the line buffer, available for the 0-line *Super-block* of new frame. In order to keep the data of the previous and the next frame from affecting each other during padding and to achieve copy-padding, the multiplexer in the line buffer can copy the data from valid working line.

The Figure 20 shows how line buffer blocking and padding. Note the **F-1** means the line or *Super-block* is from previous frame, and **F** means it is from the new frame.

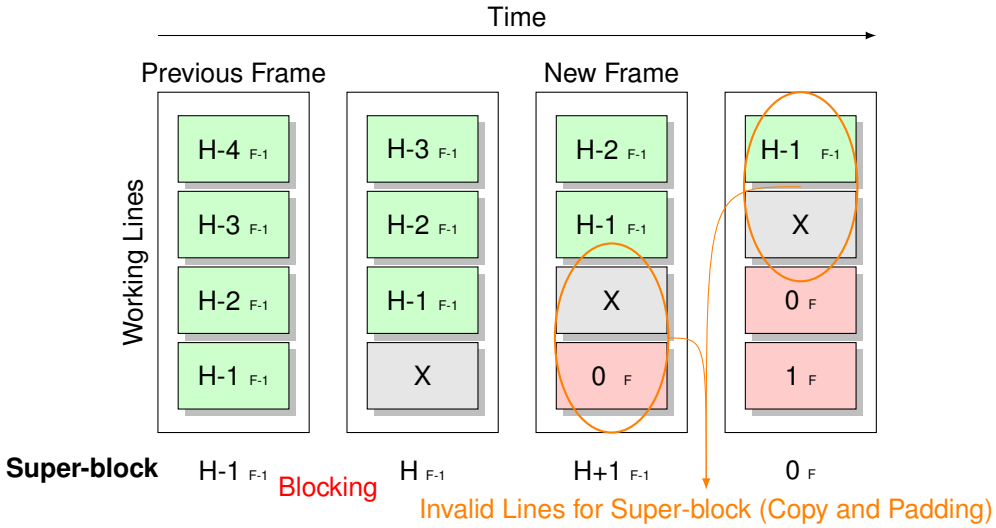

Figure 20: Line Buffer Line Padding (Top and Bottom) Between Frames

4.4 Control Unit

The control unit in the Real-time Video Bicubic Super-resolution IP is mainly used to control the following functions

1. Padding the left and right edges(Column Padding) of *Original-image*.
2. *Super-block* counting and generate line symmetry signal.
3. Generate the *Super-pixel* realign control signal.
4. Generate the control signals of AXI4-Stream Video Source interface.
5. Delay the control signals to match the latency of pipeline.

4.4.1 Column Padding

The control unit uses the same principle as the line buffer for column padding. Since the *Original-image* has $W-1$ *Super-blocks* in the X-direction, at least 1 *Super-block* blocking time is needed for padding when processing each *Super-block* line.

The control unit has an X-direction *Super-block* counter, which is able to generate a blocking signal to the line buffer during the cycle of the operation on the $W-1$-column *Super-block* of the previous line. *Reference-pixels* of new *Super-block* line will be allowed during the cycle of the W-column *Super-block* is operating.

The Figure 21 shows how control unit blocking and padding. Note the **L-1** means the *Super-block* column is from previous *Super-block* line, and **L** means it is from the new *Super-block* line.

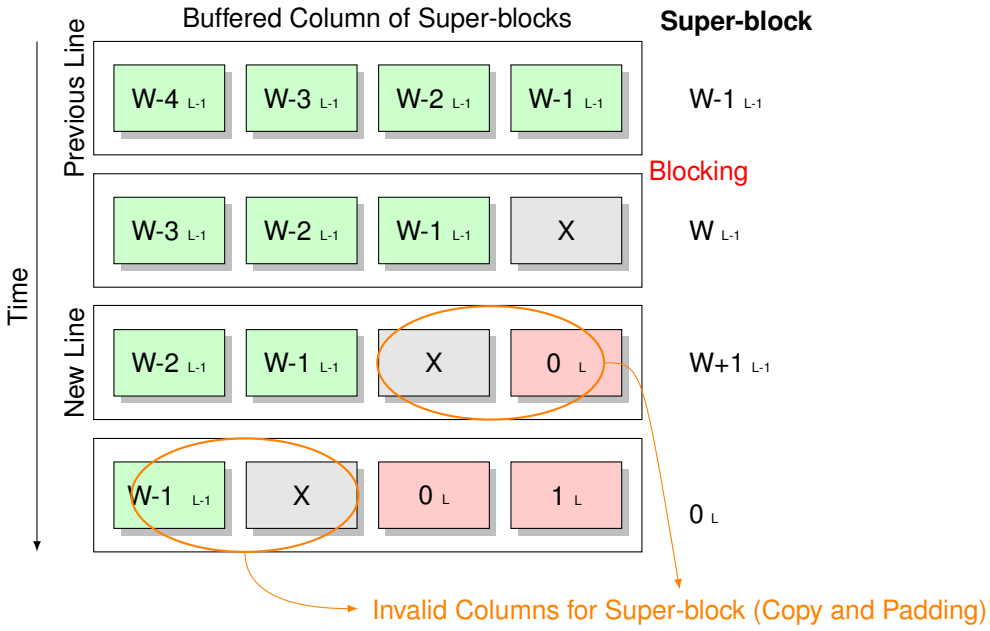

Figure 21: Control Unit Column Padding (Left and Right) Between Super-block Lines

Unlike the line buffer, the control unit completes the padding operation by controlling the multiplexer of the *Super-block* buffer in the pipeline.

4.4.2 Symmetry Control Signal Generating

There is a Y-direction *Super-pixel* counter in the control unit, which can get whether the current *Super-pixel* is in row 0, 1 or 2, 3 in the *Super-block* based on the current number of line, and generate the control signal to the symmetric multiplexer in the pipeline.

4.4.3 Super-pixel Realign Signal Generating

Due to padding and pixel alignment, only 2 *Super-pixels* are available for output when the first *Super-block* finishes its calculation, and it needs to wait until the second *Super-block* finishes its calculation to have enough data to output 4 pixels at one clock cycle. The control unit contains an X-direction *Super-pixel* counter, which can determine whether the current *Super-block* is the first column of a line, and if so, control the pixel realign module to buffer two valid *Super-pixels* in the first *Super-block*. And then starts the outputting when the next *Super-block* is processed. When processing the last *Super-block* of a line, since it also has only 2 valid *Super-pixels*, combining the last 2 *Super-pixels* with the 2 *Super-pixels* buffered in the realign module can yield the last 4 *Super-pixels* of a line of the output frame.

4.4.4 AXI4-Stream Video Source Control Signal Generating

The control unit uses the *Super-pixel* counters to determine the video frame signals (SOF and EOL), to meet the protocol of AXI4-Stream Video Interface.

4.5 Pixel Realign Unit

The pixel realign unit contains a *Super-pixel* buffer, which can buffer 2 *Super-pixels*. This unit can split the input 4 *Super-pixels* to 2 parts. Combining a part and the buffered *Super-pixels* for output, and buffering the other part into the *Super-pixel* buffer.

5 Design Verification

5.1 Verification Platform

The APV21B Real-time Video 16X Bicubic Super-resolution IP is a dedicated IP for image processing, and a bottom-up layered verification and UVM-like verification platform is used to verify the correct functionality of the IP.

To improve the algorithm verification coverage and accuracy, a mixed random + extreme verification pattern is used for the verification of computation units. The *AXI-Stream Video Image VIP* is used for bus transaction driving and monitoring for the overall verification of the IP, and a mixed verification pattern of preset bitmaps and random bitmaps is used.

The verification of the underlying arithmetic unit is performed by the SystemVerilog non-synthesizable subset for reference arithmetic and comparison. The overall verification of the IP is done by the external C model reference program and the pixel comparison program.

The block diagram of the verification platform is shown in Figure 22 and Figure 23.

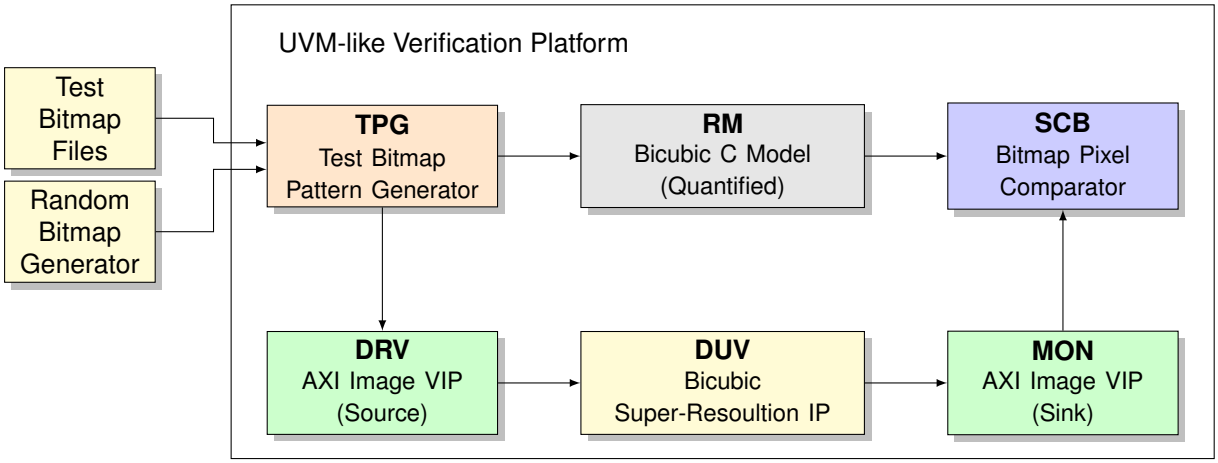

Figure 22: Diagram of UVM-like Verification Platform (Overall IP)

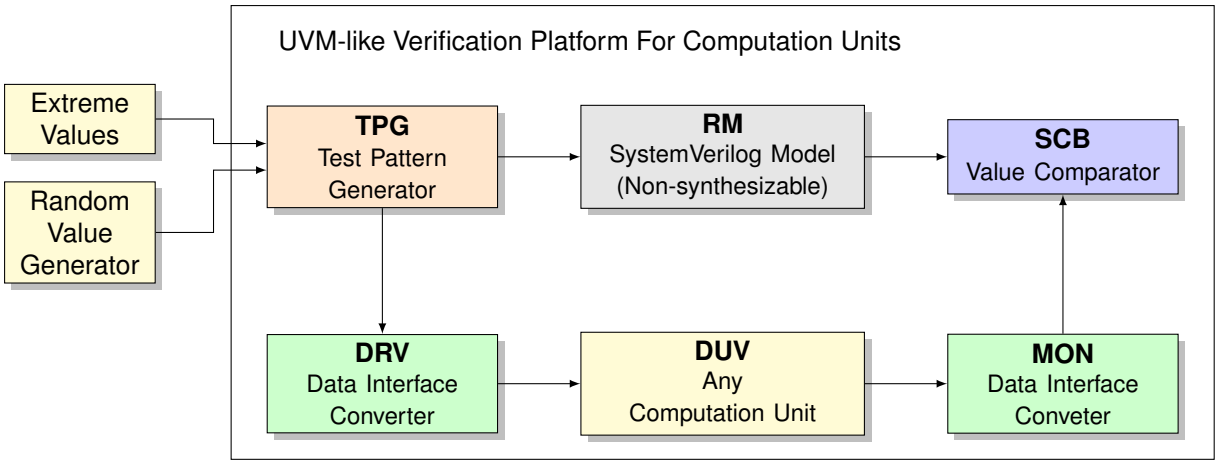

Figure 23: Diagram of UVM-like Verification Platform (Computation Units)

5.2 Software Platform

The detail information of the software platform used for IP verification is shown in Table 8.

Table 8: Software Verification Platform

Software Name	Mentor Modelsim
Version	2020.04
Referenced Libraries	Xilinx Secure IP, Xilinx XPM, Xilinx Unisims Verilog

5.3 Verification Methodology

The detail information of the verification methodology of this IP is shown in Table 9.

Table 9: Verification Methodology

Verification Language	System Verilog
Reference Model Language	System Verilog, C
Verification Platform	System Verilog Implemented, UVM-like
Test Patterns	Random Values, Extreme Values, Real-world Bitmaps

5.4 Verification Plan

The detail information of the verification plan of this IP is shown in Table 10.

Table 10: Verification Plan

Design Under Test	Test Pattern	Count of Patterns
Overall IP	Real-world Bitmap	60
	Random Bitmap	100
Multiplier Adder Unit	Extreme Values	1296
	Random Values	50000
8-input Add Unit	Extreme Values	104976
	Random Values	200000
Round Unit	Extreme Values	36
	Random Values	1000
Limit Unit	Extreme Values	3
	Random Values	1000
Overall Pipeline	Extreme Values	147456
	Random Values	200000

5.5 AXI-Stream Video Image VIP

The AXI-Stream Video Image VIP using the bitmap processing library which can read and write windows bitmap files (.BMP) into a bit array (virtural memory) by System Verilog for IP Verfication. The ”axi_stream_video_image_in_vip” IP can read a bitmap file into the memory, and send it by a AXI-Stream Video Interface (Defined in Xilinx User Guide UG934). And the ”axi_stream_video_image_out_vip” IP can monitor a AXI-Stream interface, obtain a frame which transmitting on the interface and save it to a bitmap file.

Please refer this GitHub Link (https://github.com/Aperture-Electronic/SystemVerilog-Bitmap-Library-AXI-Image-VIP) for detailed information of this important VIP in the verification platform.

The verifaction IP and the bitmap processing library were developed by the Nijigasaki IC Design Club.

5.6 Test Result

The detailed result report of the verification is shown in Table 11.

Table 11: Test Result

Design Under Test	**Test Pattern**	**Tested**	**Mismatched**	**Pass/Fail**
Overall IP	Real-world Bitmap	60	0	Pass
	Random Bitmap	100	0	Pass
Multiplier Adder Unit	Extreme Values	1296	0	Pass
	Random Values	50000	0	Pass
8-input Add Unit	Extreme Values	104976	0	Pass
	Random Values	200000	0	Pass
Round Unit	Extreme Values	36	0	Pass
	Random Values	1000	0	Pass
Limit Unit	Extreme Values	3	0	Pass
	Random Values	1000	0	Pass
Overall Pipeline	Extreme Values	147456	0	Pass
	Random Values	200000	0	Pass

6 Design with the Core

6.1 Operation

The Real-time Video Bicubic Super-resolution IP is a standalone co-processing pipeline and can work completely independently from any processor. If a processor is required to work with it for video frame synchronization, video frame access, etc., Video Direct Memory Access (VDMA) can be used to work with this IP.

6.2 Clock

The Real-time Video Bicubic Super-resolution IP operates on the clock input from the pin **clk**.

6.3 Clock Enable

The Real-time Video Bicubic Super-resolution IP has a clock enable pin **clken**, which is a active-High clock enable signal synchronous to **clk**. When the clock enable de-asserted, the IP will not continue to work, but all internal work in progress will be retained.

6.4 Reset

The Real-time Video Bicubic Super-resolution IP is fully reset when **aresetn** is asserted. This is an asynchronous active-Low reset.

There are also some synchronous reset used to reset some hardcore components (DSPs and BRAMs), they are **dsp_reset** for DSPs and **bram_reset** for BRAMs. These reset are active-High reset, and synchronous to **aclk**.

There is a synchronous reset siganl **sclr** in the Real-time Video Bicubic Super-resolution IP. This reset signal can reset entire IP to default state, an it is synchronous to **clk**. When the IP received a error frame (mismatched SOF signal, or mismatched resolution), this synchronous reset can be used to clear the error counters in the IP and resume the work.

7 Design Flow Steps

This section describes customizing and generating the IP core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core.

7.1 File Directory Structure

The directory structure of the source files for Real-time Video Bicubic Super-resolution IP is shown in Table 12.

Directory	Sub Directory	Description
doc		Documentations
ip		Packaged IP for Vivado Design Suite
modelsim		Modelsim Simulation Project
sim	pkg	Verification IPs and Packages
	ref	Reference Models
	scb	Scoreboard for UVM-like Testbenches
	tb	Testbenches
src		Design Sources
vivado		Vivado Synthesis Project

Table 12: Directory Structure

7.2 Source Files

The design source files in the **src** directory are shown in Table 13.

Table 13: Source Files

File	Type	Description
Top Level Design File		
bicubic.sv	System Verilog Source File	Top level design file of the IP
Design Unit Design File		
bicubic_line_buffer.sv	System Verilog Source File	Line buffer module
bicubic_pipeline_controller.sv	System Verilog Source File	Pipeline controller module
bicubic_pixel_out_realign.sv	System Verilog Source File	Pixel re-align module

Continued on next page

Table 13: Source Files (Continued)

bicubic_pipeline.sv	System Verilog Source File	Top level design file of the computation pipeline
Computation Pipeline Design File		
bicubic_coeff_rom.sv	System Verilog Source File	Bicubic coefficient LUT
bicubic_symmertial_mux.sv	System Verilog Source File	Symmertial multiplexer
bicubic_refpx_block_buffer.sv	System Verilog Source File	Super-block buffer
simd4x_round.sv	System Verilog Source File	4x Parallel rounding unit
pixel_limit_output.sv	System Verilog Source File	Pixel limit unit
DSP Computation Unit Design File		
dsp_simd2x_int9xuint8.sv	System Verilog Source File	2x parallel INT9xUINT8 multiplier uint (SMS architecture)
dsp_simd2x_int9xuint8_cascade_add.sv	System Verilog Source File	2x parallel SMS multiplier unit with cascade adder
dsp_simd4x_int12_add.sv	System Verilog Source File	4x parallel INT12 adder (for rounding)
dsp_add8_cin.sv	System Verilog Source File	8-input 4-carry INT18 adder
dsp_add_cascade_cin.sv	System Verilog Source File	Adder with cascade input and carry input
dsp_add3_cin.sv	System Verilog Source File	3-input adder with carry input
DSP Unit Parallel Instantiation File		
bicubic_nx_dsp_add.sv	System Verilog Source File	Nx parallel 8-input 4-carry INT18 adder instantiation
bicubic_nx_pixel_limit.sv	System Verilog Source File	Nx parallel pixel limit unit instantiation
bicubic_nx_simd_mul.sv	System Verilog Source File	Nx parallel SMS multiplier unit instantiation
bicubic_nx_simd_round.sv	System Verilog Source File	Nx parallel rounding unit instantiation
Global Marco Settings File		
bicubic_global_settings.sv	System Verilog Source File	Global macro settings
Hardware Resource Wrapper File		

Continued on next page

Table 13: Source Files (Continued)

sdpram_wrapper.sv	System Verilog Source File	Wrapper for simple dual-port RAM instantiation
Basic Unit Design File		
sfr.sv	System Verilog Source File	Customizable parameterized shift register
sfr_ce.sv	System Verilog Source File	Customizable parameterized shift register with clock enable
sfr_ce_sclr.sv	System Verilog Source File	Customizable parameterized shift register with clock enable and synchronous reset

7.3 Parameters

The Real-time Video Bicubic Super-resolution IP has customizable parameters that can be modified in the IP core top-level file. The parameters are shown in Table 14.

Parameter	**Format**	**Default Value**	**Range**	**Description**
INPUT_VIDEO_WIDTH	Integer	960	16 to 3840	The width of the input video frame
INPUT_VIDEO_HEIGHT	Integer	540	16 to 2160	The height of the input video frame

Table 14: Customizable Parameters

7.4 Macro Definitions

The Real-time Video Bicubic Super-resolution IP has customizable macro definitions in the global settings file (**bicubic_global_settings.sv**). These macros control the synthesis method used during synthesis to adapt to the different devices.

In this IP, all macro definitions are switch types. If you want to use a macro switch, please remove its comment in the file, otherwise, please comment the corresponding macro.

The macros are shown in Table 15.

Macro	Description
USE_DSP48E2_PRIMITIVE[1]	**On**: Using the DSP48E2 primitive in Xilinx UltraScale+ devices, which can force the synthesizer to use our optimized DSP48E2 wiring and architecture, maximizing performance and minimizing resource usage. **Off**: Disable the use of DSP48E2 primitive and use Verilog inference to complete the implementation of all computational statements, maximizing compatibility.
USE_LUT_FOR_ROUNDING	**On**: Use LUT instead of DSP unit for rounding operation to reduce DSP usage, minimizing resource usage. **Off**: Using DSP units for rounding operations, maximizing performance.
USE_XPM_MEMORY[2]	**On**: Use Xilinx Parameterized Macros (XPM) to implement memory elements (Line buffers), maximizing performance and minimizing resource usage. **Off**: Implement memory components using Verilog inference, maximizing compatibility.

Table 15: Macro Switches

Note For more information about Xilinx Parameterized Macros (XPM), please refer the following Xilinx documentations.

1. UltraScale Architecture Libraries Guide, UG974
2. Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide, UG953

7.5 Constraining the Core

Constraints are delivered when the IP is generated.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

[1] Only valid on Xilinx UltraScale+ devices.
[2] Only valid on Xilinx devices.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

8 Example Design

This chapter contains information about the example design of the Real-time Video Bicubic Super-resolution IP.

8.1 Overview

This example design for scaling live video from 540p to 4K resolution. The design is done based on the Xilinx Zynq UltraScale+ platform.

This design is capable of inputting 540p video in real time through the HDMI interface, scaling it up in real time, and finally outputting a 4K 30 fps video stream from the DP interface on the PS side.

The top-level module block diagram of this design is shown in Figure 24.

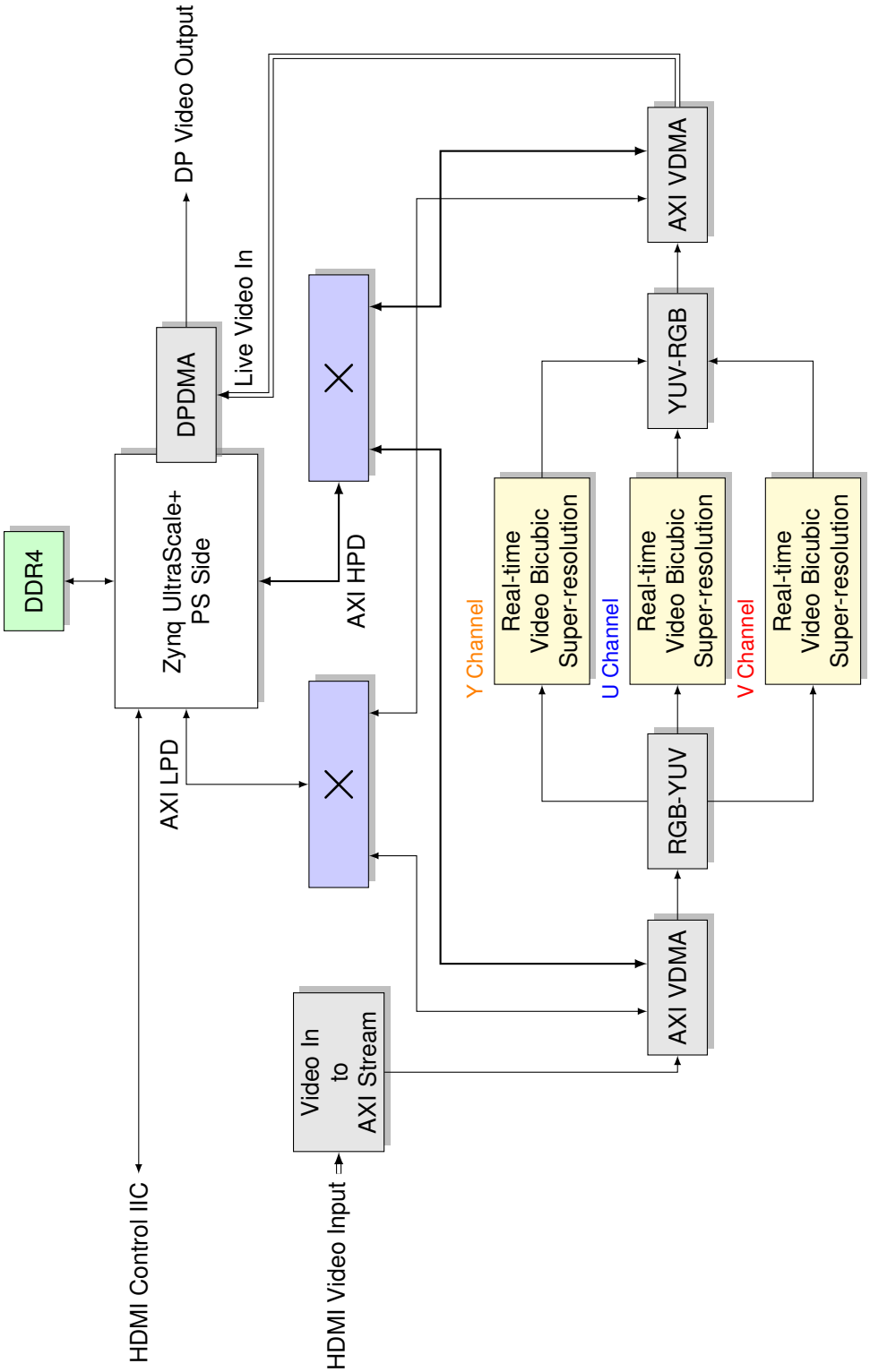

Figure 24: Top Level of the Example Design

8.2 IP Cores

Many third-party IP cores are used in the example design, and the description and functions of these IP cores are as follows.

AXI VDMA The Xilinx AXI Video Direct Memory Access provides high-bandwidth direct memory access between memory and AXI4-Stream video type target peripherals. The AXI VDMA supports a mechanism to synchronize writing and reading of frames in the frame buffer through Genlock synchronization.

RGB-YUV The Xilinx RGB to YCrCb Color-Space Converter is a simplified 3x3 matrix multiplier converting three input color samples to three output samples in a single clock cycle. The optimized structure uses only four XtremeDSP™ slices by taking advantage of the dependencies between coefficients in the conversion matrix of most RGB to YCrCb 4:4:4 or RGB to YUV 4:4:4 standards.

YUV-RGB The Xilinx YCrCb to RGB Color-Space Converter is a simplified 3x3 matrix multiplier converting three input color samples to three output samples in a single clock cycle. The optimized structure uses only four XtremeDSP™ slices by taking advantage of the dependencies between coefficients in the conversion matrix of most YCrCb 4:4:4 or YUV 4:4:4 to RGB standards.

Video In to AXI4-Stream The Xilinx Video In to AXI4-Stream is designed to interface from a video source (clocked parallel video data with synchronization signals - active video with either syncs, blanks or both) to the AXI4-Stream Video Protocol Interface.

8.3 Peripherals of the CPU

The example design uses several peripherals on the PS side of the Zynq UltraScale+, and the functions of these peripherals are described below.

DisplayPort Controller

The DisplayPort controller implements a flexible display and audio pipeline architecture.

The DisplayPort controller can source data from memory (non-live input) or the (live input) programmable logic (PL). The DisplayPort processes data, and sends it out through the DisplayPort source-only controller block to external display devices or to the PL (live output). The DisplayPort pipeline consists of the DisplayPort direct memory access (DMA) for fetching data from memory, a centralized buffer manager, a display rendering block, an audio mixer block, and the DisplayPort source controller, along with the PS-GTR block. The DisplayPort pipeline supports an ultra-high definition (UHD) aggregate video bandwidth of 30 Hz.

The DisplayPort DMA controller (DPDMA) supports up to six input channels as non-live input. Video/graphics, and audio streams can be sourced from the PL as live streams. The video processing stage involves mixing video and graphics streams, color space conversion, and chroma sub-sampling. The audio processing stage involves mixing two audio streams and volume control. The output of the audio/video processing pipeline can be output to the DisplayPort source controller or optionally be routed to the PL as live output.

I2C Controllers

The I2C controllers can function as a master or a slave in a multi-master design. They can operate over a clock frequency range up to 400 kb/s.

8.4 Buses of the CPU

The example design uses the bus connections (AXI LPD and AXI FPD) between the Zynq UltraScale+ PS and PL, and their functions are described below.

AXI LPD (AXI HPM0 LPD)

The low-latency interface port (AXI HPM0 LPD) from the high-performance interface port (LPD) to the PL includes the following features.

- Configurable to 32, 64, or 128-bit data widths on the PL side.
- AXI4 access in the PL, but is limited to a burst length of 16.
- Responds to lowest 512 MB memory in LPD's 32-bit address space.
- Enables direct access to the PL (for example for block RAM, DDR) for the safety use cases.

AXI HPD

Six high-performance interfaces provide the PL bus masters access to all PS slaves. However, these are designed to provide high-bandwidth datapaths to the DDR memory

The PL-PS interfaces are designed to provide a high-throughput datapath between the PL masters and PS memories, including the DDR and OCM memories. The main features of these interfaces are list as follows.

- Support for AXI4. The conversion to AXI3 takes place in the PS.
- 32, 64, or 128-bit data-wide master interfaces that are independently programmed for read and write per port.
- Efficient dynamic upsizing for all full-width AXI INCR commands.
- Asynchronous clock frequency domain crossing for all AXI interfaces between the PL and PS. Two PL clocks per interface, one for read and one for write.

8.5 Workflow of the Design

The Zynq UltraScale+ CPU in the example design is mainly used to configure the parameters of each peripheral. At system startup, the CPU runs a program to initialize each peripheral. The flow is as follows.

1. Initialize each bus interface.
2. Initialize the I2C peripheral and configure the registers of the external HDMI-to-video signal converter chip via I2C.
3. Initialize the DP controller and DPDMA and configure it to Live Video mode to receive and display live video from the video processing pipeline.

4. Configure DP link monitoring interrupt to enable DP cable hot-plugging.

5. Configure input and output VDMA and enable multi-frame buffering and vertical synchronization.

When the CPU completes the system configuration, the system enters normal operation.

During normal system operation, the HDMI signal is converted to a standard video signal through the video converter chip and input to the FPGA. Video In to AXI-Stream IP converts the input video signal to an AXI4-Stream video stream and input to Input VDMA.

The Input VDMA saves the input video data to the DDR on the PS side via the AXI FPD bus, and sends frames to the video processing pipeline with frame synchronization.

The video processing pipeline consists of RGB-YUV conversion IP, YUV-RGB conversion IP, and three Real-time Video Bicubic Super-resolution IPs. Each Real-time Video Bicubic Super-resolution IP processes one of the Y/U/V channel independently but in parallel and synchronized.

After the Real-time Video Bicubic Super-resolution IP super-resolution processing and YUV-RGB conversion, the video stream is input to Output VDMA.

Output VDMA can save the processed video to the DDR on the PS side via AXI FPD bus, and at the same time send the video to the Live Video interface of DP controller under frame synchronization. DPDMA and DP controller sends the high-resolution video to DP port for display.

9 Licensing and Open Soruce Information

The Real-time Video Bicubic Super-resolution IP is a fully open source IP core, with all source code and detailed design materials publicly available.

The Real-time Video Bicubic Super-resolution IP is licensed under the GNU Lesser General Public License (LGPL).

REVISION HISTORY

The Table 16 shows the revision history for this document.

Date	Version	Description of Revisions
5/10/2022	1.0	Initial Nijigasaki IC Design Club release.
7/20/2022	2.0	Merge ”Design Verification” section into the documemt.

Table 16: Revision History

NOTICE OF DISCLAIMER

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Nijigasaki IC Design Club products. Tothe maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Nijigasaki IC Design Club hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOTLIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULARPURPOSE; and (2) Nijigasaki IC Design Club shall not be liable (whether in contract or tort, including negligence, or under any other theory ofliability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (includingyour use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including lossof data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if suchdamage or loss was reasonably foreseeable or Nijigasaki IC Design Club had been advised of the possibility of the same. Nijigasaki IC Design Club assumes noobligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to productspecifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. IP cores may be subject to warranty and support terms contained in a license issued toyou by Nijigasaki IC Design Club. Nijigasaki IC Design Club products are not designed or intended to be fail-safe or for use in any plication requiring fail-safeperformance; you assume sole risk and liability for use of Nijigasaki IC Design Club products in Critical.