APV21B - RTL Module Design

Deng LiWei
Nijigasaki IC Design Club

May 30, 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).

This document is part of the IP user manual and is intended to describe the detailed hardware design in this IP. Complete technical documentation can be found in the User Manual for this IP.

1 Overview

Figure 1 illustrates the Real-time Bicubic Super-resolution IP Block Diagram.

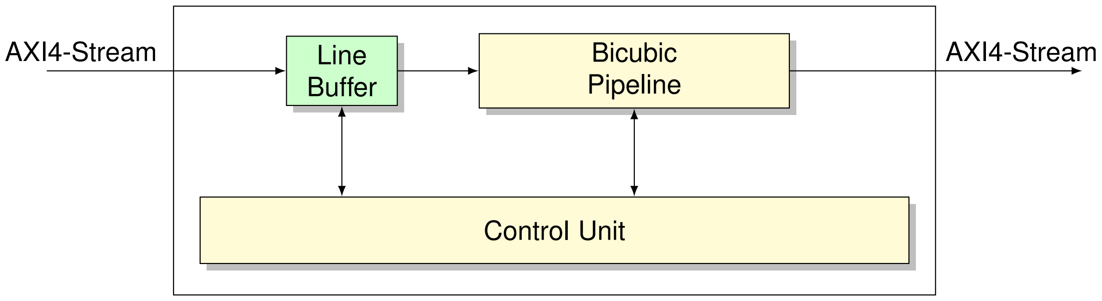

Figure 1: Real-time Bicubic Super-resolution IP Block Diagram

2 Port Descriptions

This section describes the details for each interface. The Real-time Video Bicubic Super-resolution signals are described in Table 1

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 1: I/O Signal Description

3 Detailed Design Structure

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.

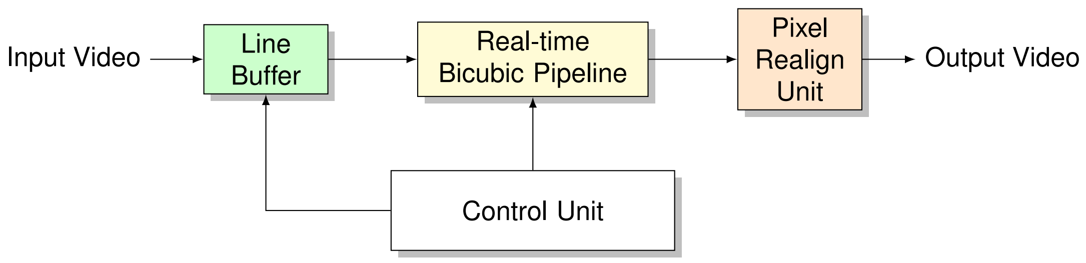

Figure 2: Real-time Video Bicubic Super-resolution IP Processing Units

3.1 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.

3.2 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 3.

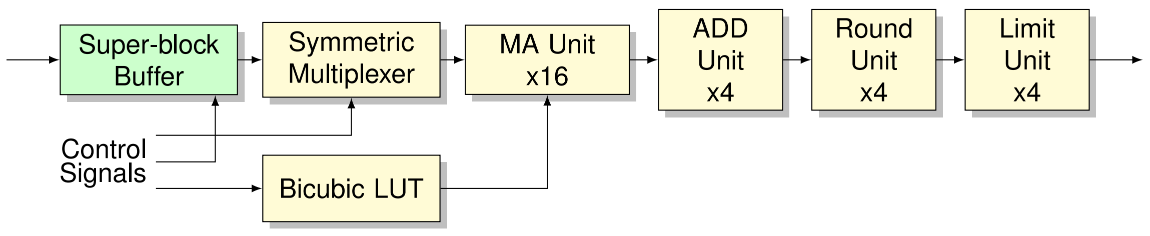

Figure 3: Real-time Bicubic Pipeline

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.

3.2.1 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 2, 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 2: Super-block Buffer Output Multiplexer Modes

Figure 4 shows the structure of the *Super-block* Buffer.

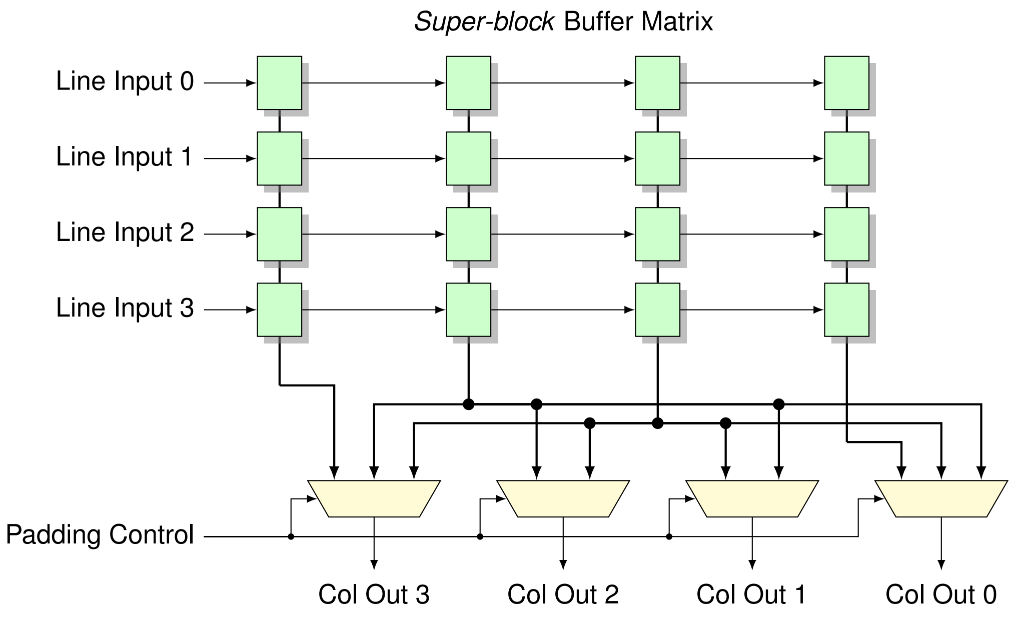

Figure 4: Diagram of the Super-block Buffer

3.2.2 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 5, $d[(1,2) \rightarrow (0,3)] = d[(1,1) \rightarrow (0,0)]$(Red arrows) and $d[(2,3) \rightarrow (2,1)] = d[(2,0) \rightarrow (2,2)]$(Blue arrows).

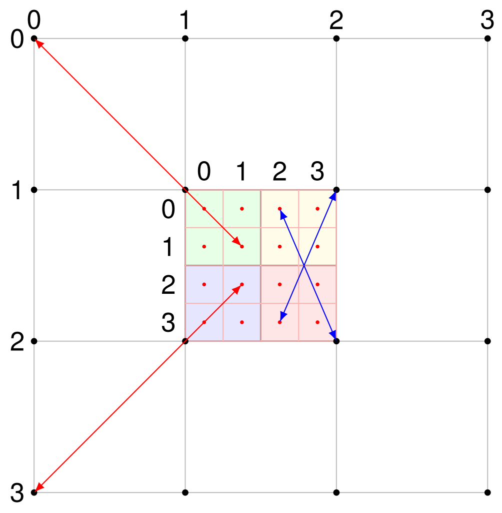

Figure 5: Example of Vertical Symmetric in a Super-block

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.

3.2.3 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.

3.2.4 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 6.

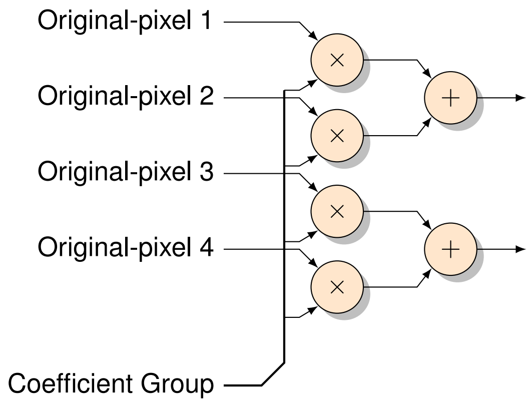

Figure 6: Mathematical Structure of MA Unit

Using the horizontal symmetry, 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.

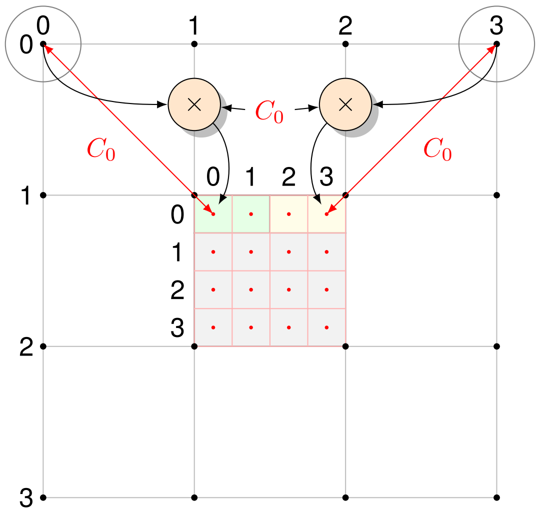

Figure 7: Example of Multiplication Operation Using Horizontal Symmetry

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 8.

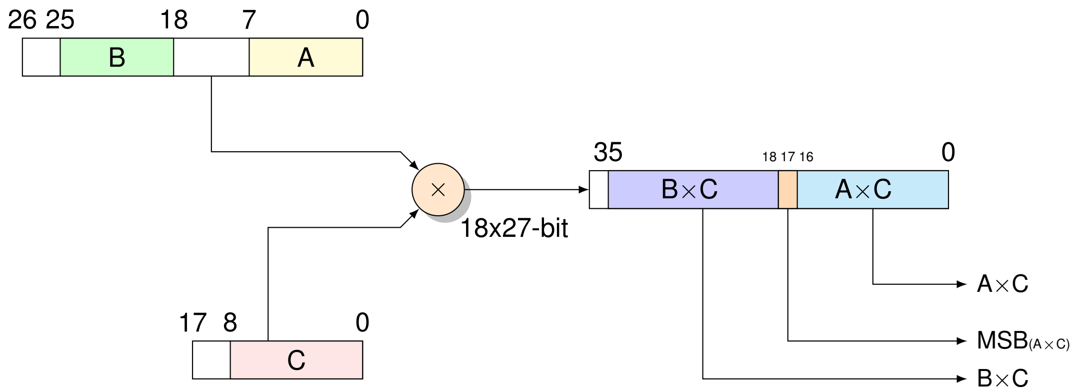

Figure 8: Bicubic Computation Process of SMS Structure

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 9.

The MA unit uses the equations are

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

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

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

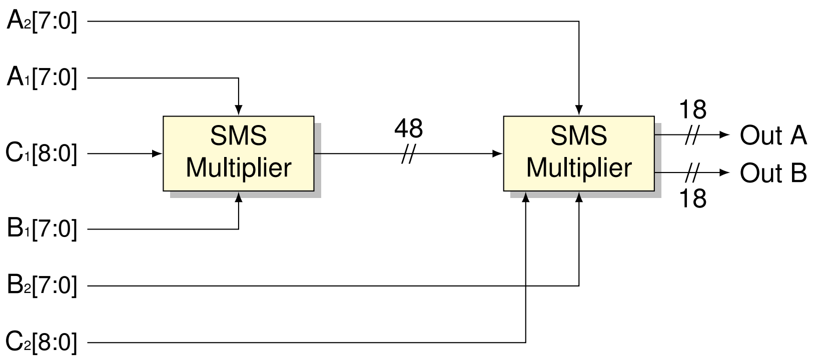

Figure 9: Structure and Data Flow of MA Unit

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 3. 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 3: 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 3: Multiplier Adder Unit Coefficient Assignments (Continued)

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

3.2.5 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 10 and Figure 11.

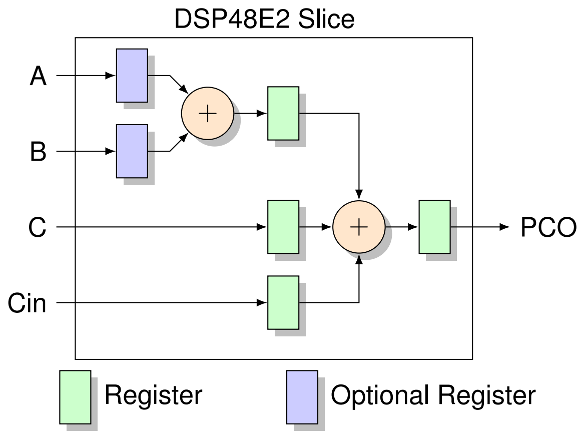

Figure 10: 3-input Adder Implemented by DSP48E2 Slice

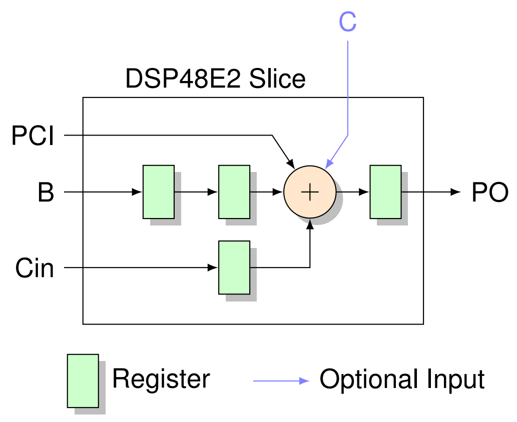

Figure 11: DSP Cascade Adder Implemented by DSP48E2 Slice

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 12.

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

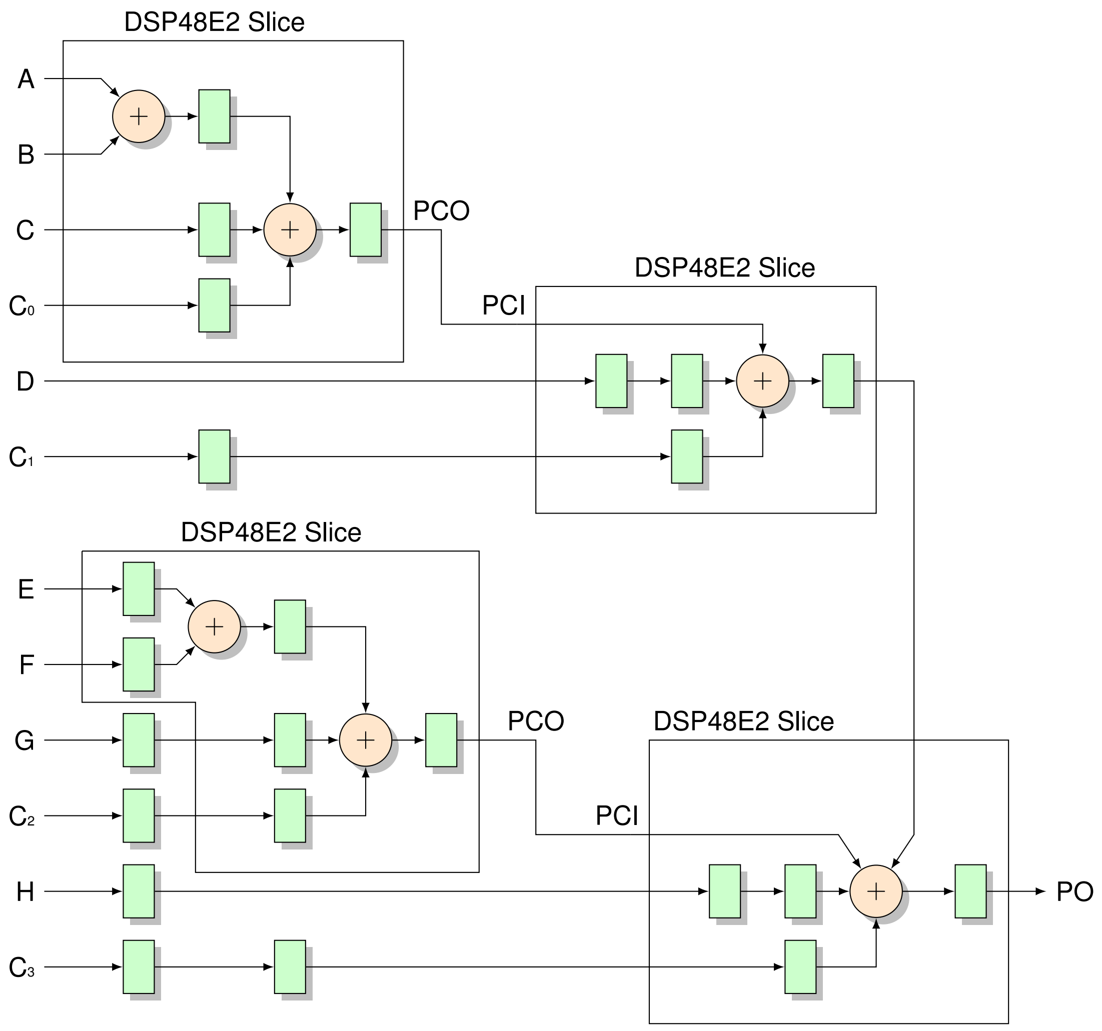

Figure 12: Structure of 8-Input 4-Carry Adder

3.2.6 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.

3.3 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.

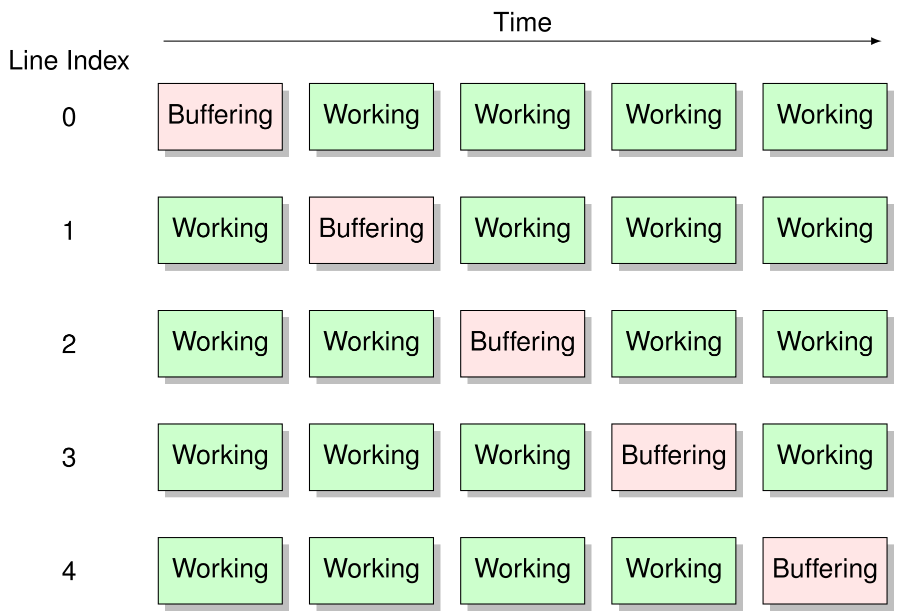

Figure 13: Working Lines and Buffering Line in Line Buffer

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 14;

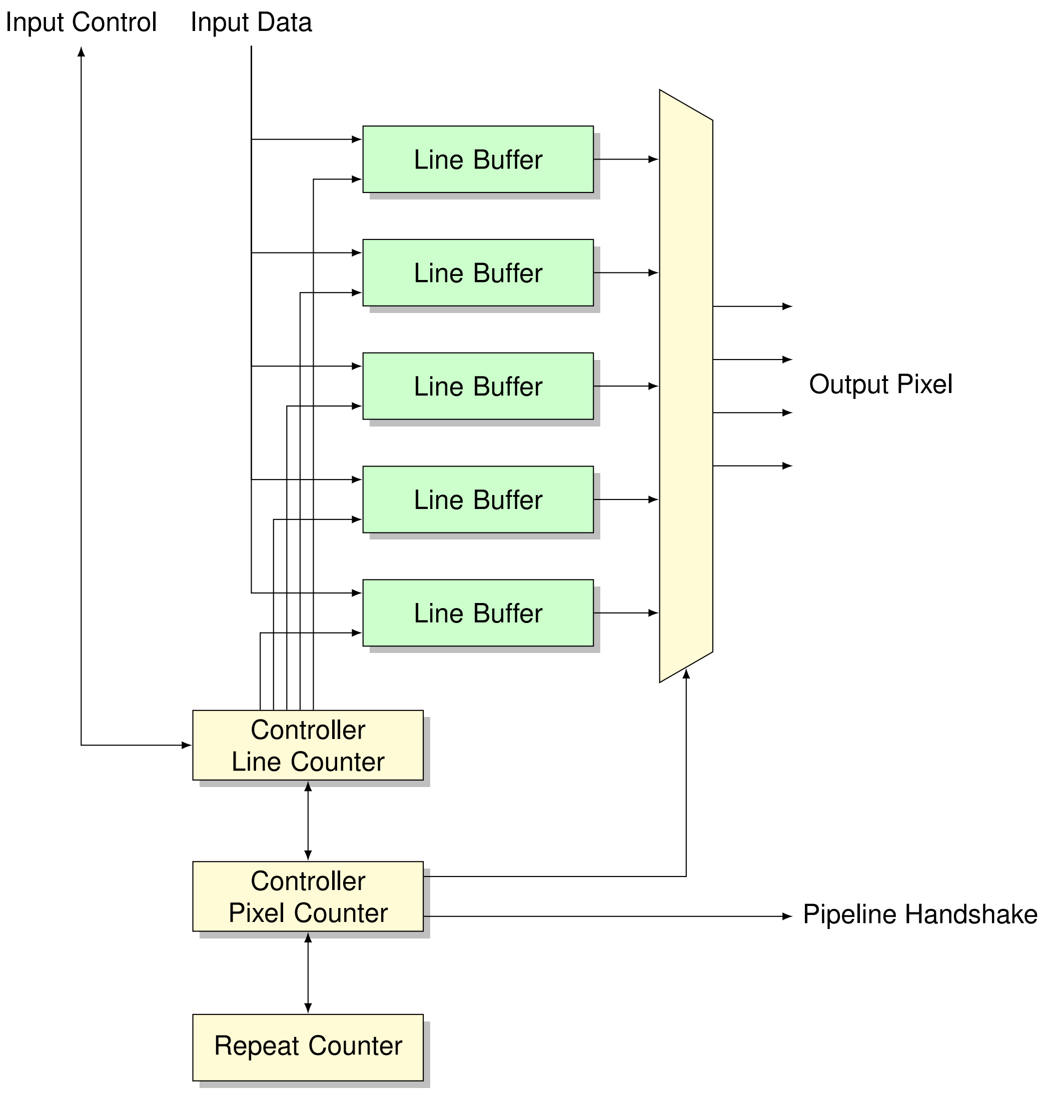

Figure 14: Diagram of Line Buffer

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 15 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.

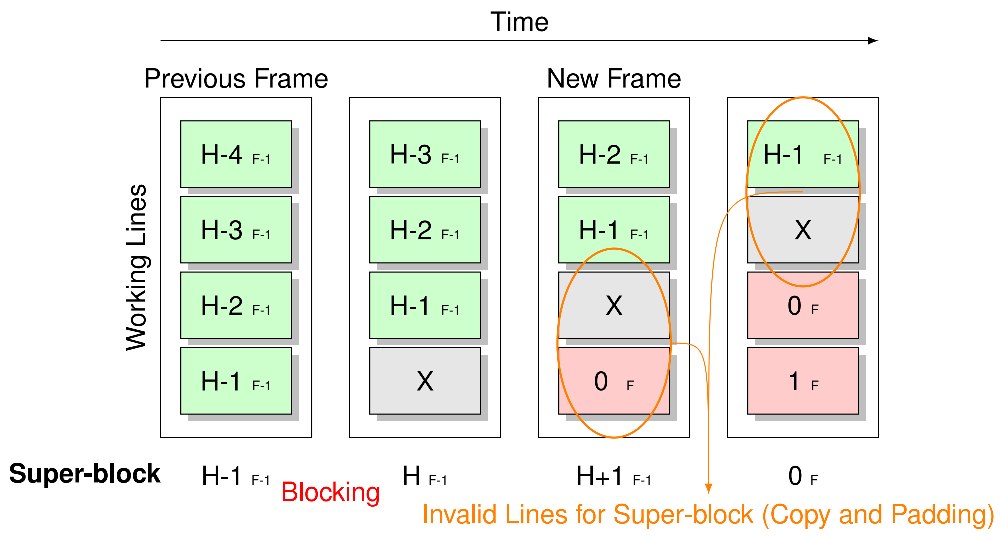

Figure 15: Line Buffer Line Padding (Top and Bottom) Between Frames

3.4 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.

3.4.1 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 16 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.

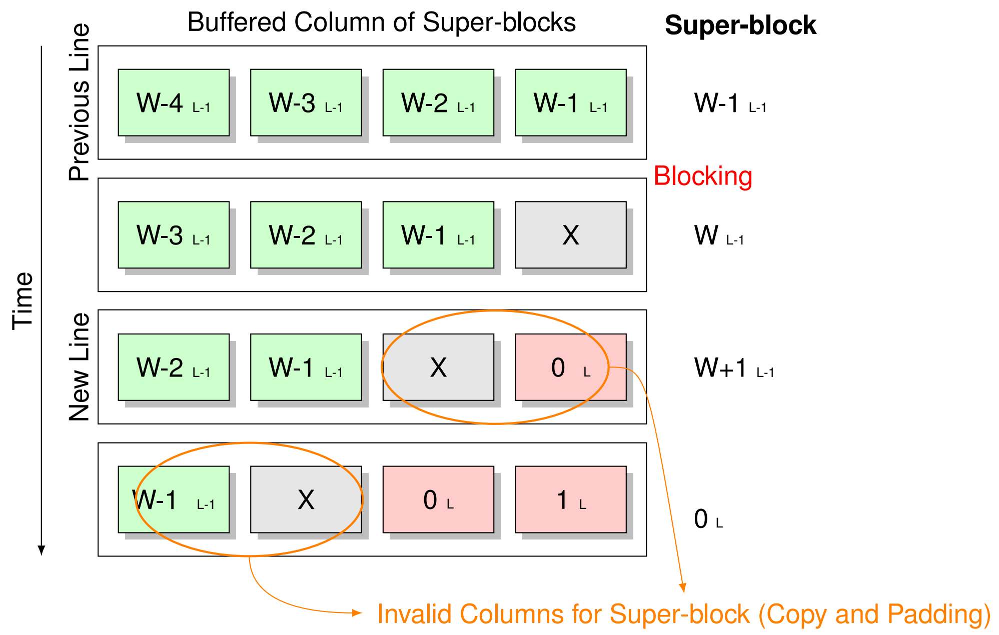

Figure 16: Control Unit Column Padding (Left and Right) Between Super-block Lines

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

3.4.2 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.

3.4.3 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.

3.4.4 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.

3.5 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.

NOTICE OF DISCLAIMER