H.264 Deblocker Core v1.0

DS592 (v1.0) May 31, 2007 **Product Specification**

Introduction

The H.264 Deblocker Core Version 1.0 is a fully functional VHDL design implemented on a Xilinx FPGA and delivered in netlist form. The Deblocker core accepts input parameters and macroblocks to deblock and generates output macroblocks as specified by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/EC Moving Picture Experts Group (MPEG) as the product of a collective partnership known as the Joint Video Team (JVT). The collaborative effort is also known as H.264/AVC/MPEG4 Part 10.

The system designer should use this document in conjunction with the User Guide. The MPEG4 Part 10 specification (ref. [1]) is also recommended documentation along with the JM10.2 H.264 Codec reference C code.

Features

- H.264/MPEG-4 Part10 Baseline/Main/High Profiles at Level 4.2
- Compliant with International Standard ISO/IEC 14496-10:2005 (E) Rec. H.264 (E)
- Supports up to 300,000 Macroblocks/s operation
- Boundary Strength calculation internal to core
- Passthrough mode
- FIFO reset facility for loss-of sync robustness
- Frame or Field processing
- 4:2:0 supported
- Two core types:
  - IF 0 Core (Economy)
  - IF 1 Core (Ease-of-Use)

LogiCORE Facts				
Core Specifics				
Supported Device Families	Virtex™-5, Virtex-4, Spartan™-3E			
Resources Used	I/O	LUTs	FFs	Block RAMs
	See Table 1 for details.			
Provided with Core				
Documentation	Preliminary Data Sheet			
Design File Formats	VHDL, EDIF			
Constraints File	.ucf (user constraints file)			
Verification	JM10 Reference C-Code vs. VHDL Testbench and HW-in-the-Loop			
Instantiation Template	VHDL Wrapper			
Reference Designs /Application Notes	None			
Design Tool Requirements				
Xilinx Implementation Tools	Xilinx ISE 8.2.03i (from ngd_build)			
Verification	ModelSim 6.1c SE, MicroSoft Visual C++ V6.0, ActivePerl 5.8.3. Annapolis Micro Systems WildCard4 Platform API 3.0 Driver 4.0 Firmware 1.0			
Simulation	ModelSim 6.1c SE			
Synthesis	Synplicity Synplify_Pro 8.6.2			
Support				
Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing functionality or support of product, if implemented in devices not listed in the documentation, or if customized beyond that allowed in the product documentation, or if any changes are made in sections of the design marked “DO NOT MODIFY.”				

Table 1: **Resource Summary**

Resources Used			
IF 0 (Economy)	Virtex-5	1080	1634 LUTs, 1565 flops 9 RAMB18x2, 3 RAMB36
		720P	1639 LUTs, 1563 flops 5 RAMB18x2, 3 RAMB36
		ITU-R-BT.601	1595 LUTs, 1550 flops 1 RAMB18x2, 7 RAMB36
		CIF	1575 LUTs, 1539 flops 1 RAMB18x2, 7 RAMB36
	Virtex-4	1080	21 block RAMs, 1438 slices
		720P	13 block RAMs, 1437 slices
		ITU-R-BT.601	9 block RAMs, 1382 slices
		CIF	9 block RAMs, 1358 slices
	Spartan-3E	1080	19 block RAMs, 1598 slices
		720P	11 block RAMs, 1577 slices
		ITU-R-BT.601	7 block RAMs, 1523 slices
		CIF	7 block RAMs, 1488 slices
IF 1 (Ease of Use)	Virtex-5	1080	1968 LUTs, 1948 flops 9 RAMB18x2, 6 RAMB36
		720P	1968 LUTs, 1946 flops 5 RAMB18x2, 6 RAMB36
		ITU-R-BT.601	1915 LUTs, 1918 flops 1 RAMB18x2, 10 RAMB36
		CIF	1933 LUTs, 1931 flops 1 RAMB18x2, 10 RAMB36
	Virtex-4	1080	23 block RAMs, 1764 slices
		720P	16 block RAMs, 1764 slices
		ITU-R-BT.601	12 block RAMs, 1715 slices
		CIF	12 block RAMs, 1685 slices
	Spartan-3E	1080	22 block RAMs, 1906 slices
		720P	14 block RAMs, 1914 slices
		ITU-R-BT.601	10 block RAMs, 1856 slices
		CIF	10 block RAMs, 1824 slices

Applications

The Deblocker core can be utilized in H.264 standard applications where hardware acceleration is needed to achieve real time operation. Typical video applications are video surveillance, video conferencing and video broadcast. The DD can also be used as a post processing engine for previous standards such as MPEG-2 decoder.

Functional Overview

This document describes the core functionality, its input/output structure, its preliminary FPGA resource estimate and how to use it in an H.264 encoding system. The aim of this core is to filter across the edges of the 4x4-pixel blocks inside and on the edge of a current input macroblocks dependent upon a number of parameters and content-based criteria (see edges V1, V2, V3, H1, H2, and H3 in Figure 1. Also processed are the macroblocks boundaries at the top and left edges (edges V0 and H0) of the current macroblocks, requiring the availability of these adjacent macroblocks. It is important to note that this process also modifies the content of these adjacent macroblocks.

To clarify some nomenclature, filtering across the **Vertical** edges (V0 – V3) is equivalent to **Horizontal filtering**, and filtering across **Horizontal** edges (H0 – H3) is equivalent to **Vertical filtering.**

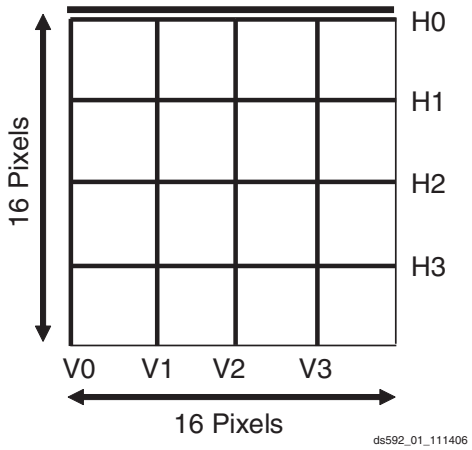

Figure 1: **Macroblock Edges**

Performance

Target clock FMax and subsequent macroblock throughput are summarized in Table 2. The throughput figures apply for IF 0 and IF 1 cores.

Table 2: **Performance Summary**

Family	Clock FMax	Approximate 4:2:0 Macroblock Throughput (Macroblocks/s)	Notes
Spartan-3E	150 MHz	200,000	Supports up to 601, 4CIF, 1080P/25, 720P/50
Virtex-4	225 MHz	300,000	Supports up to 1080P/30, 1080i/60, 720P/60
Virtex-5	225 MHz	300,000	Supports up to 1080P/30, 1080i/60, 720P/60

System Overview

Xilinx provides two interface types to satisfy customers who may want to integrate the Deblocker at different levels in their system. It is necessary to describe the two interface types available to the user. Figure 2 shows the delineation between the two interface types and shows the basic functional blocks of the system.

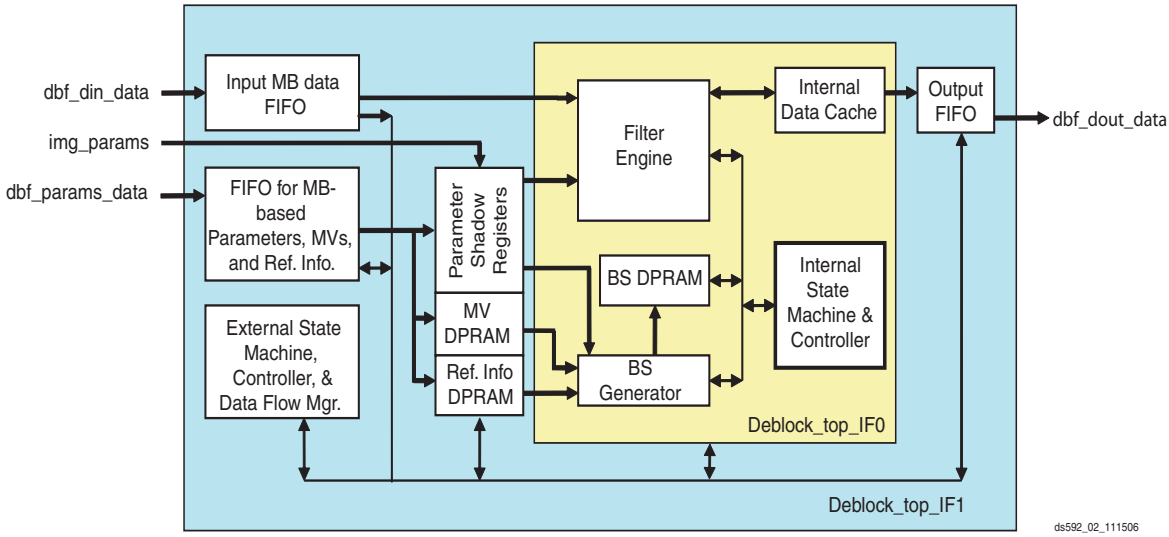

Figure 2: **Deblocker Functional Block Diagram**

IF 0 Deblocker Core

The IF 0 core is a bare-bones deblocker which relies on the user providing many inputs according to stringent conditions. It is intended for users who may already have the memory blocks and registers already in place in the host system. Using this core saves some block RAMs associated with the I/O FIFOs and also a significant amount of fabric registers and logic (see Table 1 for comparative data). Usage is relatively complex, requiring adherence to strict timing definitions of many interface signals.

IF 0 Core I/O Description

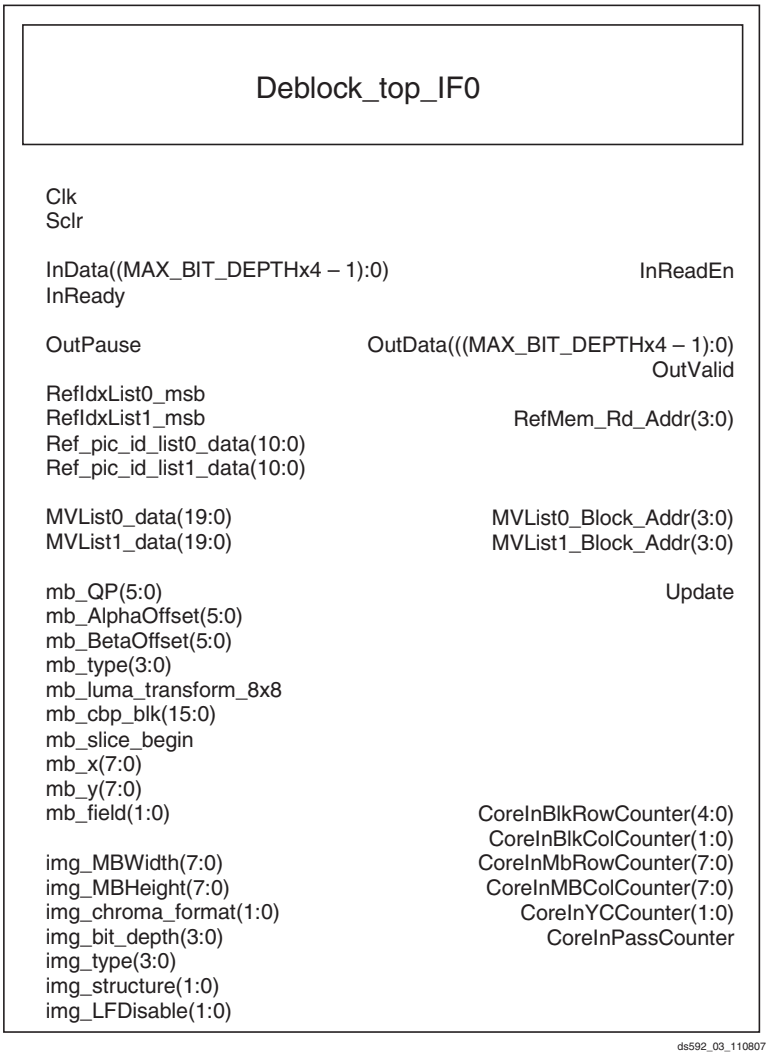

Figure 3: **Deblock_top_IF0**

Table 3 presents the IF 0 interface signals.

Table 3: **IF 0 Interface Signals**

Signal Name	Direction	Description
Clk	Input	All systems and interface operations are synchronous to this clock.
Sclr	Input	Active high clear signal that resets all internal states to a known state.
Data IO interface		
InData(DataWidthx4 – 1:0)	Input	Data input: Format described in "Input Data Format and Timing - Deblocker IF 0" on page 8
InReady	Input	Active High to indicate availability of 1 macroblock of data at the input
InReadEn	Output	Given as read-enable for macroblock memory outside this block.
Update	Output	To be used to update external shadow registers for all input parameters on an input Macroblock basis.
OutData(DataWidthx4 – 1:0)	Output	Data output: Format described below in "Output Data Format and Timing - Deblocker IF 0" on page 11
OutValid	Output	Output valid flag
OutPause	Input	Stalls core from delivering the next deblocked Macroblock.
Reference Information Interface (see "IF 0 Reference Information DPRAM Interface" on page 13 for detailed information)		
RefMem_Rd_Addr(3:0)	Output	Read address for memory block containing Ref_pic_id, Refidx.
RefIdx_List0_msb	Input	Indicates validity of Ref_pic_id_list0 as reference frame pointer.
RefIdx_List1_msb	Input	Indicates validity of Ref_pic_id_list1 as reference frame pointer.
Ref_pic_id_list0_data(10:0)	Input	List 0 reference frame pointer.
Ref_pic_id_list1_data(10:0)	Input	List 1 reference frame pointer.
Motion-Vector Memory Interface (see "IF 0 Motion Vector DPRAM Interface" on page 13 for detailed information)		
MVList0_Block_Addr	Output	Read address for list 0 motion vectors.
MVList1_Block_Addr	Output	Read address for list 1 motion vectors.
MVList0_Data	Input	List 0 motion vectors.
MVList1_Data	Input	List 1 motion vectors.
General-Purpose Counter Outputs		

Table 3: **IF 0 Interface Signals *(Continued)***

Signal Name	Direction	Description
CoreInBlkRowCounter(4:0)	Output	Indicates the current row being deblocked in the current macroblock.
CoreInBlkColCounter(1:0)	Output	Indicates the current column (of 4 pixels) being deblocked in the current macroblock.
CoreInYCCounter(4:0)	Output	Indicates the current component being deblocked. 00 = Y 01 = Cr 10 = Cb
CoreInPassCounter	Output	Indicates the current deblocking pass: 0 = Horizontal pass 1 = Vertical pass
CoreInMBColCounter(7:0)	Output	Indicates the x coordinate of the macroblock being deblocked.
CoreInMBRowCounter(7:0)	Output	Indicates the y coordinate of the macroblock being deblocked
Macroblock-Based Parameter Inputs (see "Macroblock-Based Parameters" on page 19 for detailed information)		
mb_QP(5:0)	Input	Quantization Parameter – value between 0 and 51 inclusive.
mb_AlphaOffset(5:0)	Input	Alpha QP offset. Between -12 and 12, equal to 2*slice_alpha_c0_offset_div2 from H.264 specification Ref.[1].
mb_BetaOffset(5:0)	Input	Beta QP offset. Between -12 and 12, equal to 2*slice_beta_offset_div2 from H.264 specification Ref. [1].
mb_type(3:0)	Input	Mb_type as tabulated in section 7 of the H.264 specification Ref. [1].
mb_luma_transform_8x8	Input	Transform_size_8x8 flag – active High indicates that only half of the edges within the current macroblock should be filtered.
mb_cbp_blk(15:0)	Input	One bit per 4x4 block in the current macroblock – High indicates non-zero transform coefficient levels for this 4x4 block.
mb_slice_begin	Input	High when current macroblock is first in new slice.
mb_x(7:0)	Input	x-coordinate of current macroblock using luma pixel coordinates/16: first macroblock (top left) is (0,0) next macroblock to right is (1, 0)… etc.
mb_y(7:0)	Input	y-coordinate of current macroblock using luma pixel coordinates/16: first macroblock (top left) is (0,0) next macroblock below is (0, 1)… etc.
mb_field(1:0)	Input	Not used currently.

Table 3: **IF 0 Interface Signals *(Continued)***

Signal Name	Direction	Description
Image-Based Parameter Inputs (see Image-Based Parameters for detailed information)		
img_MB_Width(7:0)	Input	Number of macroblocks across one frame of current image.
img_MB_Height(7:0)	Input	Number of macroblocks from top to bottom of one frame of current image.
img_chroma_format(1:0)	Input	00 = Monochrome 01 = 4:2:0 10 = 4:2:2 11 = 4:4:4 Currently only 4:2:0 supported.
img_bit_depth(3:0)	Input	Currently supports up to 8 bits (“1000”).
img_type(3:0)	Input	0000 = P_SLICE 0001 = B_SLICE 0010 = I_SLICE 0011 = SP_SLICE 0100 = SI_SLICE
img_structure(1:0)	Input	00 = FRAME 01 = TOP_FIELD 10 = BOTTOM_FIELD
img_LFDisable(1:0)	Input	Disable deblock 00 = not disabled 01 = disabled for this image 10 = disable on slice-boundaries
img_chroma_qp_offset(4:0)	Input	QP offset for chroma data. Value -12 to +12

Input Data Format and Timing - Deblocker IF 0

Successive macroblocks must be provided in raster order.

When an entire macroblock of data is available in a memory or FIFO at the inputs to the deblocker IF 0, the parameter inputs (see Parameters) are stable at the inputs, and reference information (see IF 0 Reference Information DPRAM Interface) and motion vectors (see IF 0 Motion Vector DPRAM Interface) for that macroblock are also stable in their corresponding memories, the InReady input should be asserted High. In response to this, when the deblocker is ready, InReadEn will be asserted High. Valid data must be present at the deblocker IF 0 data inputs (InData) exactly 3 clock cycles after InReadEn goes High.

The luma (Y) component of any macroblock must be entirely passed to the deblocker before moving to the next component. This must be followed by the Cr component of the same macroblock and finally Cb, as indicated in Figure 4. Figure 5 shows this spatially as Macroblock Linear Addressing. Four adjacent input pixel samples are required to be present on the input bus in parallel, and held for 2 clock cycles. For 4:2:0, one macroblock requires ((256 + 64 + 64)/4) * 2 = 192 clock cycles. InReadEn will reflect this as shown in Figure 4.

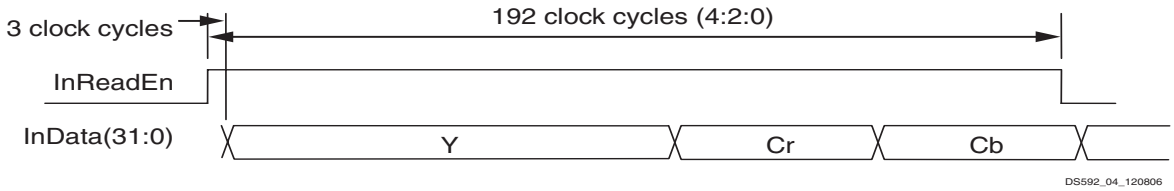

Figure 4: **Input Timing for One Macroblock Showing YCrCb for 4:2:0**

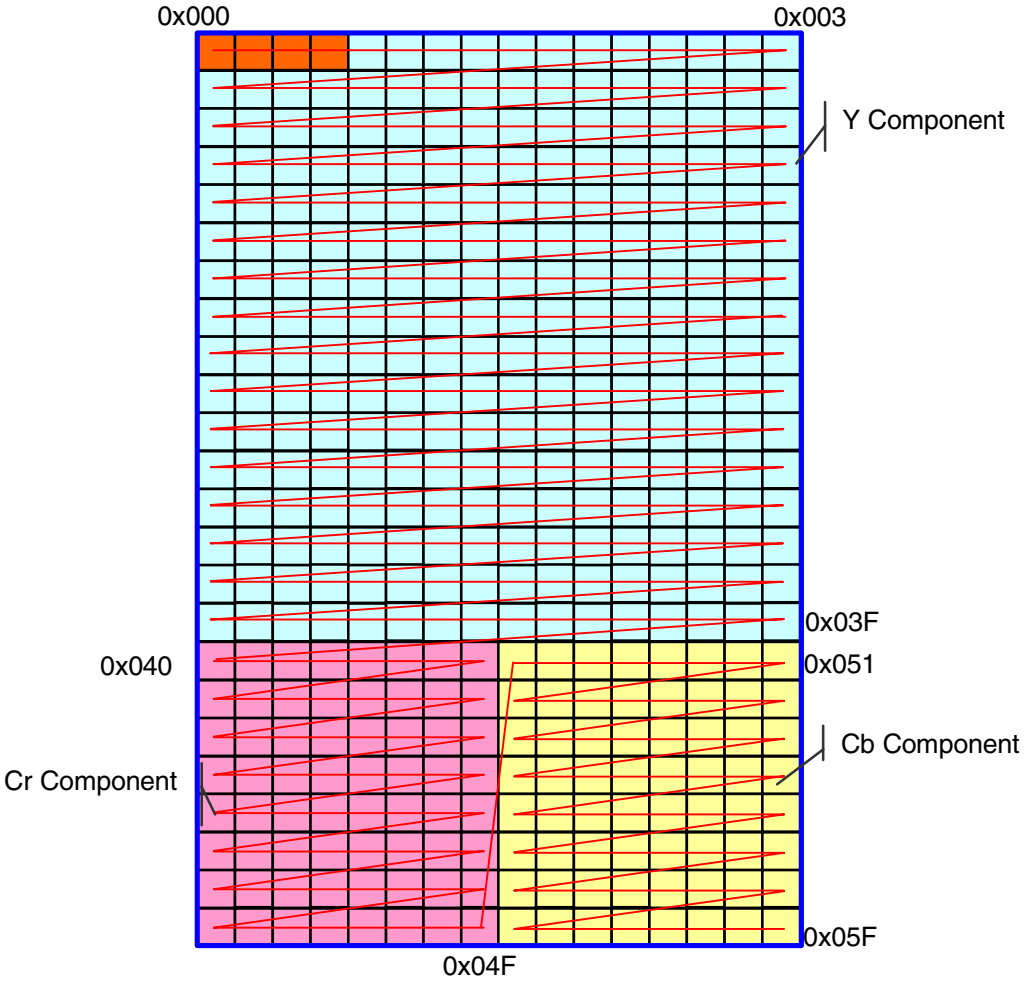

Figure 5: **Macroblock Linear Addressing for 4:2:0**

For a macroblock with the data named as in Figure 6, the data input must follow the formatting shown in Figure 7 and Figure 8. These diagrams assume an 8-bit input.

A1	B1	C1	D1	A2	B2	C2	D2	A3	B3	C3	D3	A4	B4	C4	D4
E1	F1	G1	H1	E2	F2	G2	H2	E3	F3	G3	H3	E4	F4	G4	H4
I1	J1	K1	L1	I2	K2	K2	L2	I3	J3	K3	L3	I4	J4	K4	L4
M1	N1	O1	P1	M2	N2	O2	P2	M3	N3	O3	P3	M4	N4	O4	P4
A5	B5	C5	D5	A6	B6	C6	D6	A7	B7	C7	D7	A8	B8	C8	D8
E5	F5	G5	H5	E6	F6	G6	H6	E7	F7	G7	H7	E8	F8	G8	H8
I5	J5	K5	L5	I6	K6	K6	L6	I7	J7	K7	L7	I8	J8	K8	L8
M5	N5	O5	P5	M6	N6	O6	P6	M7	N7	O7	P7	M8	N8	O8	P8
A9	B9	C9	D9	A10	B10	C10	D10	A11	B11	C11	D11	A12	B12	C12	D12
E9	F9	G9	H9	E10	F10	G10	H10	E11	F11	G11	H11	E12	F12	G12	H12
I9	J9	K9	L9	I10	K10	K10	L10	I11	J11	K11	L11	I12	J12	K12	L12
M9	N9	O9	P9	M10	N10	O10	P10	M11	N11	O11	P11	M12	N12	O12	P12
A13	B13	C13	D13	A14	B14	C14	D14	A15	B15	C15	D15	A16	B16	C16	D16
E13	F13	G13	H13	E14	F14	G14	H14	E15	F15	G15	H15	E16	F16	G16	H16
I13	J13	K13	L13	I14	K14	K14	L14	I15	J15	K15	L15	I16	J16	K16	L16
M13	N13	O13	P13	M14	N14	O14	P14	M15	N15	O15	P15	M16	N16	O16	P16

Figure 6: **Arbitrary Naming of Data within a Macroblock (for following document reference)**

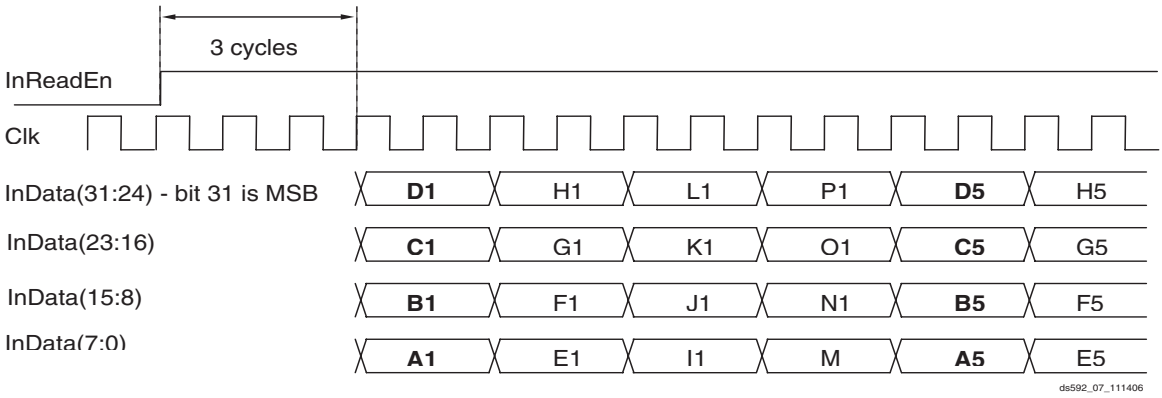

Figure 7: **Input Data Ordering and Timing (1)**

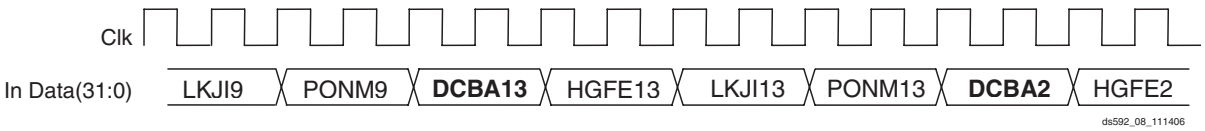

Figure 8: **Input Data Ordering and Timing (2)**

Further Operation Requirements

- InReady should be deasserted at any time while the InReadEn is active High.
- The parameters must be held stable at the deblocker IF 0 inputs when InReady is asserted.
- The motion-vector sets and reference information must also be held stable in their respective Dual-Port RAMs when InReady is asserted.
- The deblocking process involves filtering in two passes: first, a horizontal pass, and second, a vertical pass. The H-pass occurs while the data is active at the inputs to the deblocker IF 0 (InReadEn=1) as described above. However, the V-pass occurs following this period and requires the same amount of time as the H-pass. The parameters must be held stable at the deblocker IF 0 inputs during both passes. The motion-vector sets and reference information must be also be held stable in their respective Dual-Port RAMs during both passes. Hence, all these values must remain stable for 384 + 3 clock cycles (4:2:0) following the assertion of InReadEn.

Output Data Format and Timing - Deblocker IF 0

The output from the deblocker IF 0 is simple. When OutValid is High, the data output on the OutData pins is valid. For any line of macroblocks, valid data will not appear at the outputs until the input of the third macroblock has commenced for that line as shown in Figure 9. Also, following the input of the final macroblock in any line, two further macroblocks will be given before the InReadEn output is asserted for the following line of macroblocks; this is also shown in Figure 9.

The luma (Y) component of the current output macroblock will be entirely passed out from the deblocker before moving to the next component. This is followed by the Cr component of the same macroblock and finally Cb. Four adjacent input pixel samples are presented on the output bus in parallel and are valid for just one clock cycle, indicated by OutValid. Outvalid validates output values an alternate clock cycles as shown in Figure 10. For 4:2:0, one complete macroblock requires ((256 + 64 + 64)/4) * 2 = 192 clock cycles to be given completely at the output.

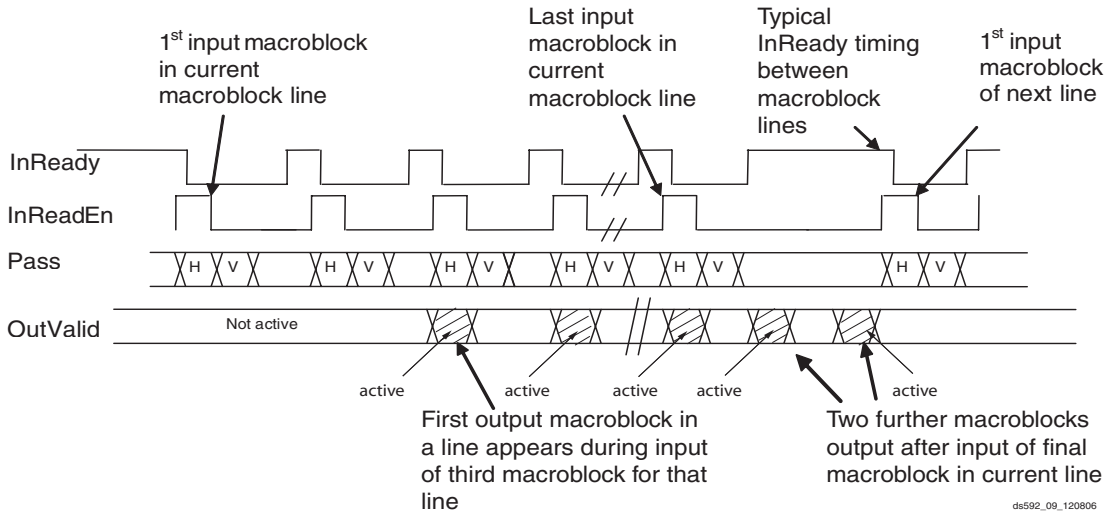

Figure 9: **Output Timing (Macroblock line-based)**

The order of the data output is partially transposed in comparison to the input. For an output macroblock of the format given in Figure 6, the output data schedule is shown in Figure 10.

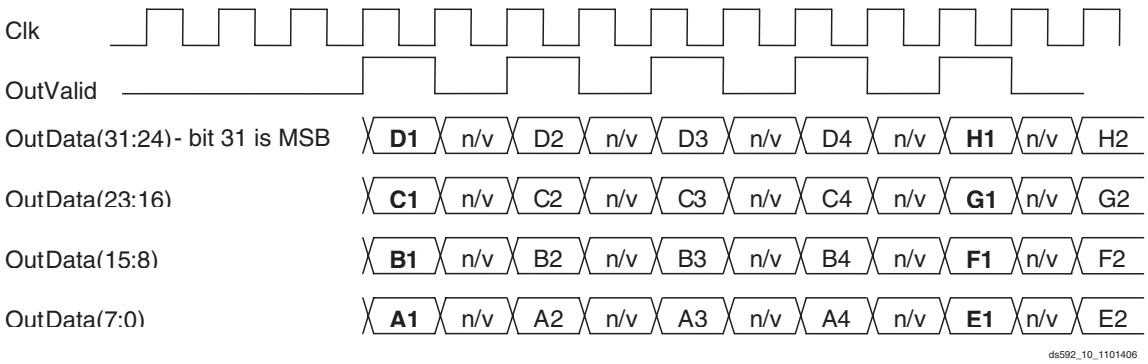

Figure 10: **Output Timing (pixel-by-pixel)**

Most macroblocks output from the deblocker IF 0 are 16 lines in height. However, because of operational simplicity when filtering the horizontal edges between macroblocks, the top line of output macroblocks is only 12 lines in height. Consequently, an additional output line of 4-line "macroblocks" is given: for an image of N luma lines in height, the number of output macroblock lines is (N/16)+1. Table 4 summarizes the heights of all macroblocks in the output data for any image size.

Table 4: **Number of Lines per Output Macroblock Line for an Image of N Luma Lines**

Macroblock line # (luma)	Height
1	12 lines
2 through (N/16)	16 lines
(N/16)+1	4 lines

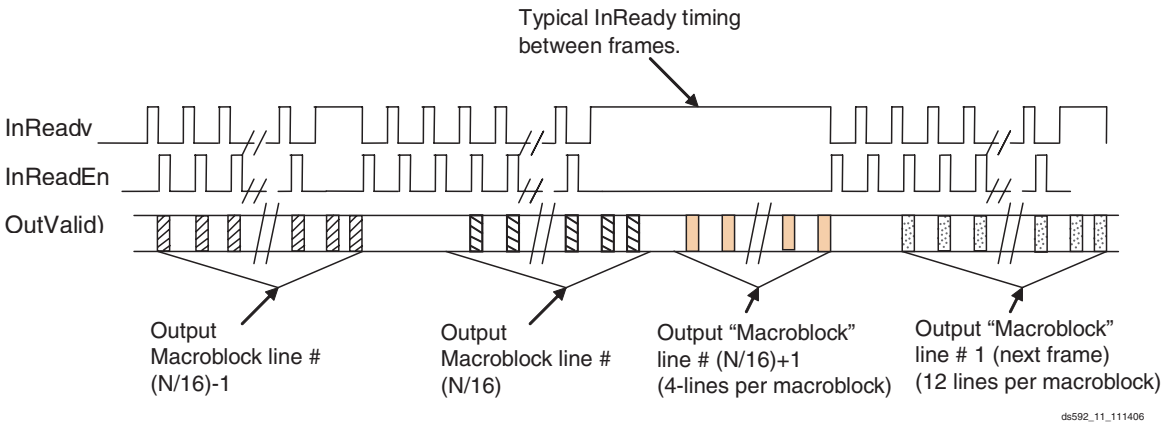

Figure 11: **Output Timing (line-by-line)**

Further Operation Features

- The Update signal is given as an indication that the current parameters at the Deblock IF 0 inputs have been used fully and may be updated. The flag lasts just one clock cycle. Use of this flag may enable the user to speed up the controlling state-machine.

- CoreInxxx counter outputs have been provided for use by the user if needed. They are all synchronized with the input data and filtering operations.

IF 0 Reference Information DPRAM Interface

The IF 0 core provides an interface to a memory block where reference information for the current macroblock is stored. This is referred to as the *Ref Info DPRAM* in Figure 2 when instantiated in the wrapper. The core provides the address RefMem_Rd_Addr(3:0) as a read-address to this memory. The memory map is as shown in macroblock form in Figure 12. The reference information is referred to on a 4x4-pixel block basis and must be stored in the DPRAM thus. The data must be provided one clock-cycle after the address changes. This is ideally suited to SelectRAM usage.

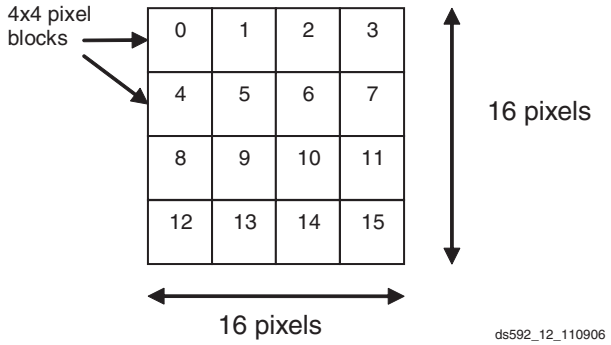

Figure 12: **Reference Information and Motion Vector DPRAM Address map (in macroblock form)**

Each location must hold and provide to the core:

- RefIdx_List0_msb: List 0 reference frame pointer valid (1 bit)
- ref_pic_id_list0_data: List 0 reference frame pointer (11 bits)
- RefIdx_List1_msb: List 1 reference frame pointer valid (1 bit)
- ref_pic_id_list1_data: List 1 reference frame pointer (11 bits)

for the 4x4 pixel block indicated in Figure 12.

For more detailed information on the derivation of these values, it is important to read the deblocking section of the H.264 specification (ref [1]) and also be familiar with the JM10.2 reference code.

IF 0 Motion Vector DPRAM Interface

The IF 0 core provides an interface to memory blocks where motion vectors for the current macroblock are stored. This is referred to as the *MV DPRAM* in Figure 2 when instantiated in the IF 1 core. In fact, there are two of these memory blocks. One for List 0 and one for List 1. They require separate addressing. The read address buses MVList0_Block_Addr(3:0) and MVList1_Block_Addr(3:0) are provided as outputs from the core. The memory map is as shown in macroblock form in Figure 12. MVs are referred to on a 4x4-pixel block basis and must be stored in the DPRAM thus. How the user packs the data into this memory is up to the user. However, the data must be provided one clock-cycle after the address changes. This is ideally suited to Select-Ram usage.

Each location must hold a pair of 10-bit motion vectors associated with that 4x4-pixel block. The two values are concatenated:

- **Bits 19:10:** MV_Y (vertical component)

- **Bits 9:0:** MV_X (horizontal component)

Deblocker IF 1

The IF 1 is a wrapper around the IF 0 core (see Figure 2) which removes several potentially awkward IF 0 issues:

- Reference information DPRAM is instantiated in the IF 1 wrapper
- MV DPRAMs are instantiated in the IF 1 wrapper
- Complex I/O data formatting issues are simplified by introducing FIFO-type I/O interfaces (dbf_din, dbf_dout).
- High-bandwidth macroblock-based parameters, MVs and Reference information, are streamlined into a single parameters FIFO interface (dbf_params)
- Internal *shadow registers* are introduced for image- and macroblock-based parameters in the wrapper allowing external values to vary freely while deblocking is active

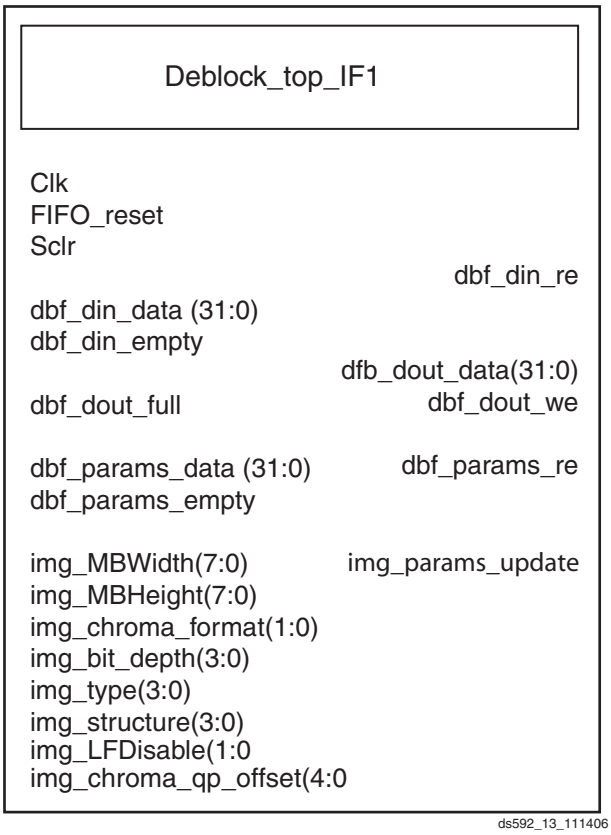

Figure 13: **IF 1 Deblocker Core**

Table 5: **IF 1 Core Interface Signals**

Signal Name	Direction	Description
Clk	Input	All systems and interface operations are synchronous to this clock.
FIFO_reset	Input	Active High - Resets input and output FIFO pointers to zero.
Sclr	Input	Active High clears signal that resets all internal states to a known state.
Data Input Interface (see "Timing for Input Data and Parameters - Deblocker IF 1" on page 16& "Input Data Format" on page 17 for more details)		
dbf_din_data(31:0)	Input	Data input - four pixels in parallel.
dbf_din_empty	Input	Assert High when no data is available for delivery into the deblocking filter IF 1 interface.
dbf_din_re	Output	Given as read-enable for the input data-holding memory structure outside the deblocking filter IF 1 interface.
Data Output Interface (see "Timing for Output Data - Deblocker IF 1" on page 17 & "Output Data Format" on page 17 for more details)		
dbf_dout_data(31:0)	Output	Data output – four pixels in parallel.
dbf_dout_full	Input	Assert High when the output data-receiving structure/memory outside the deblocking filter IF 1 interface is full and cannot receive any more data. **Only deassert this signal when the data-receiving structure/ memory can receive an entire macroblock of data.**
dbf_dout_we	Output	Given as write-enable to the output data-receiving structure/memory outside the deblocking filter IF 1 interface.
Parameters Input Interface (see "Parameter Ordering" on page 18 &"Parameters" on page 19 for more details)		
dbf_params_data(31:0)	Input	Parameters input.
dbf_params_empty	Input	Assert High when no parameters are available for delivery into the deblocking filter IF 1 interface
dbf_params_re	Output	Given as read-enable for parameters memory structure outside this block.
Image-Based Parameter Inputs (see "Parameters" on page 19 for detailed information)		
img_params_update	Output	If the most recent mb_x and mb_y parameters sent to the core are (0,0), then when **all** parameters are registered internally into their shadow registers for direct core usage, img_params_update is asserted High for one clock cycle. This indicates that it is safe to change the image-based parameters until the next time that mb_x, mb_y = (0,0).
img_MB_Width(7:0)	Input	Number of macroblocks across one frame of current image.

Table 5: **IF 1 Core Interface Signals *(Continued)***

Signal Name	Direction	Description
img_MB_Height(7:0)	Input	Number of macroblocks from top to bottom of one frame of current image.
img_chroma_format(1:0)	Input	00 = Monochrome; 01 = 4:2:0; 10 = 4:2:2; 11 = 4:4:4; Currently only 4:2:0 supported.
img_bit_depth(3:0)	Input	Currently supports up to 8 bits (“1000”)
img_type(3:0)	Input	0000 = P_SLICE; 0001 = B_SLICE; 0010 = I_SLICE; 0011 = SP_SLICE; 0100 = SI_SLICE;
img_structure(1:0)	Input	00 = FRAME; 01 = TOP_FIELD; 10 = BOTTOM_FIELD
img_LFDisable(1:0)	Input	Disable deblock 00 = not disabled; 01 = disabled for this image 10 = disable on slice-boundaries
img_chroma_qp_offset(4:0)	Input	QP offset for chroma data. Value -12 to +12.

Timing for Input Data and Parameters - Deblocker IF 1

Successive macroblocks must be provided in raster order.

It is envisaged that the user will connect this core to two input FIFOs: one containing a contiguous stream of input data and the other containing parameters in an order defined in Parameter Ordering. Separate FIFO interfaces are provided by the Deblocker IF 1 core for these two FIFOs (dbf_din & dbf_params). These interfaces consist of:

- **dbf_xxx_data(31:0)**: input bus
- **dbf_xxx_empty**: *FIFO empty* input
- **dbf_xxx_re**: read-enable output

If the user does not assert the dbf_xxx_empty input, then it is assumed that data is available in the FIFO and the core will assert dbf_xxx_re when it is ready to do so. When dbf_xxx_re is asserted, data must be present at the dbf_xxx_data inputs one clock cycle later. This is ideally suited to a SelectRAM based FIFO.

For image-based parameter inputs, when the macroblock-based parameters mb_x and mb_y are both equal to 0, and these values are in use by the core, the values at the inputs to the Deblocker IF 1 core are registered internally into some shadow registers in the wrapper for direct usage by the IF 0 core. Consequently, the single clock-pulse flag will be given on the image_params_update output to signify that it is now safe to modify the image-based parameters at the inputs to the IF 1 core.

Timing for Output Data - Deblocker IF 1

It is envisaged that the user will connect this core to an output data FIFO. A FIFO interface is provided by the Deblocker IF core for this FIFO (dbf_dout).

The interface consists of:

- **dbf_dout_data(31:0)**: output bus
- **dbf_dout_full**: FIFO full input indicator
- **dbf_dout_we**: write-enable output

If the user does not assert the dbf_dout_full input, then it is assumed that space is available in the FIFO and the core will assert dbf_dout_we when it is ready to do so. When dbf_dout_we is asserted, valid data is simultaneously present at the dbf_dout_data outputs. This is ideally suited to a SelectRAM based FIFO.

Input Data Format

For an input macroblock as shown in Figure 6, data must be fed into the Deblocker IF 1 core interface in the order shown in Figure 5, Figure 7, and Figure 8.

Output Data Format

Output data will be read from the output FIFO interface in the order shown in Figure 10 and Figure 11. A full macroblock for 4:2:0 requires (64 Y + 16 Cr + 16 Cb=) 96 read operations on this FIFO interface.

Important:

All macroblocks output from the deblocker IF 1 are 16 lines in height. However, because of operational simplicity when filtering the horizontal edges between macroblocks some apparent anomalies are introduced:

- the number of output macroblock lines is equal to img_MBHeight + 1
- only the first 12 lines of the top line of output macroblocks is valid
- only the first 4 lines the bottom line of output macroblocks is valid

For example, for an image of N luma lines in height, the number of output macroblock lines is (N/16)+1 (= img_MBHeight+1). The number of values that **must be read** from the output FIFO interface per 4:2:0 image is, therefore, equal to ((N/16)+1) x 96. However, postprocessing will be required for removal of the invalid output values. It is also important to note that the spatial location for the output “macroblocks” is not the standard location as in the input macroblocks. They have been shifted up by 4 lines. The validity and meaning of the output pixels is summarized in Table 6, numbering the output macroblocks arbitrarily from 0 to ((img_MBWidth+1)*img_MBHeight)-1 for any given image.

Table 6: **Selective Validity of Output Data on a per-Macroblock Basis**

Output Macroblock Number	Valid samples	Notes
0 through (img_MBWidth-1)	0 – 47 (48 – 63 invalid)	Top 12 lines of top macroblock line for Y
	64 – 72 (73 – 79 invalid)	Top 4 lines of top macroblock line for Cr
	80-87 (88 – 95 invalid)	Top 4 lines of top macroblock line for Cb

Table 6: **Selective Validity of Output Data on a per-Macroblock Basis *(Continued)***

Output Macroblock Number	Valid samples	Notes
img_MBWidth through (img_MBWidth*img_MBHeight – 1)	0 – 63	16 lines of Y
	64 – 79	8 lines of Cr
	80 – 95	8 lines of Cb
(img_MBWidth*img_MBHeight) through ((img_MBWidth+1)*img_MBHeight)-1	0 – 15 (16 – 63 invalid)	Bottom 4 lines of image for Y
	64 – 72 (73 – 79 invalid)	Bottom 4 lines of image for Cr
	80 – 87 (88 – 95 invalid)	Bottom 4 lines of image for Cb

Parameter Ordering

For the Deblocker IF 1 core, the following information shown in Table 7 must be passed to the core via the dbf_params input FIFO interface for each macroblock. It is vital that for every macroblock, the ordering from 0 through 41be observed in ascending order.

Table 7: **Parameter Ordering for dbf_params Interface**

Parameter #	Parameter
0-15 (bits (31:21))	**List0_ref_pic_id**: Address map as Figure 12.
0-15 (bit 20)	**List0_RefIdx_msb**: Address map as Figure 12.
0-15 (bits (19:10))	**List0MV[1]**: Motion vector Vertical component (mv_y) for List 0. Address map as Figure 12.
0-15 (bits (9:0))	**List0MV[0]**: Motion vector Horizontal component (mv_x) for List 0. Address map as Figure 12.
16-31 (bits (31:21))	**List1_ref_pic_id**: Address map as Figure 12 (add 16 to give actual addresses).
16-31 (bit 20)	**List1_RefIdx_msb**: Address map as Figure 12 (add 16 to give actual addresses).
16-31 (bits (19:10))	**List1MV[1]**: Motion vector Vertical component (mv_y) for List 1. Address map as Figure 12(add 16 to give actual addresses).
16-31 (bits (9:0))	**List1MV[0]**: Motion vector Horizontal component (mv_x) for List 1. Address map as Figure 12 (add 16 to give actual addresses).
32	mb_QP
33 (bits (11:6))	mb_BetaOffset
33 (bits (5:0))	mb_AlphaOffset
34	mb_type
35	mb_luma_transform_8x8
36	mb_field
37	mb_cbp_blk
38	mb_slice_begin

Table 7: **Parameter Ordering for dbf_params Interface *(Continued)***

Parameter #	Parameter
39	reserved – set to zero
40 (bits (15:8))	mb_x
40 (bits (7:0))	mb_y
41	reserved – set to zero

Video Synchronization and Sync Loss

Video Synchronization and Sync Loss is an important subject for broadcast users. The capability to recover from loss of synchronization in video is very useful, but should not be an overhead when it is not required. Should the transmission of a macroblock to the IF 1 core ever be incomplete, then it is likely that the following macroblock, or other (maybe corrupted) data, be enabled into the input data FIFO interface out of sync. Essentially, the first sample from a macroblock can become written in as one of the samples in the middle of the macroblock. When this occurs, clearly the deblocker will malfunction. There is no simple way of detecting whether this loss of synchronization has taken place at the deblocker level. Hence, it is necessary to provide a means of synchronization to FIFO pointers if the customer requires.

Within the deblocker IF 0 core itself, synchronization is simple. If a deblocking operation starts on any given macroblock, then that deblocking operation will complete. The deblocked macroblock will always be generated at the core outputs whether the output FIFO is ready or not, and regardless of whether or not the correct data is available at the inputs.

For the IF 1 core, the input data stream is partitioned into macroblock-sized chunks for operation. It is important that these chunks are aligned to macroblock edges, so that data from one and only one macroblock is held within them. This is part of the function of the input FIFO. Synchronization allows the user to periodically reset the read- and write-pointers to zero. For this purpose, the FIFO_reset input is provided.

It is envisaged that if the user should require the robustness introduced by this feature the user should tie this input to a frame-pulse. Internally, the falling edge of the input signal is detected to reset the FIFO pointers. This signal resets the R/W pointers of all three I/O FIFOs. In doing this, it is necessary to assert the FIFO_reset signal for at least one clock cycle following the output of the final macroblock from the deblocker. If synchronization of the deblocker is not an issue for the user, tie the FIFO_reset input to the active High system reset signal.

Parameters

Macroblock-Based Parameters

Macroblock-based parameters are the set of parameters that change the operation of the deblocker on a macroblock-by-macroblock basis. They are common to both IF 0 and IF 1 cores. IF 0 core takes all macroblock-based parameters in as direct inputs. IF 1 core takes them in via a common interface and stores them internally to the wrapper, subsequently, providing them as direct inputs to the internal IF 0 core, and as contents of the Reference Information DPRAM and MV DPRAMs.

These values are described in the H.264 spec document (Ref. [1]) and also derived from values used in the JM10 reference code. Here is a brief summary:

- **mb_QP(5:0)**:
  Quantization parameter: value between 0 and 51 inclusive.
- **mb_AlphaOffset(5:0)**:
  Alpha QP offset. Between -12 and 12, equal to 2*slice_alpha_c0_offset_div2 from H.264 specification (Ref. [1])
- **mb_BetaOffset(5:0)**:
  Beta QP offset. Between -12 and 12, equal to 2*slice_beta_offset_div2 from H.264 specification (Ref. [1])
- **mb_type(3:0)**:
  Mb_type as tabulated in section 7 of the H.264 specification (Ref. [1]).
- **mb_luma_transform_8x8**:
  Transform_size_8x8 flag – Active High indicates that only half of the edges within the current macroblock should be filtered.
- **mb_cbp_blk(15:0)**:
  One bit per 4x4 block in the current macroblock – High indicates non-zero transform coefficient levels for this 4x4 pixel block. Bit assignments are for the 4x4 blocks shown in Figure 14.

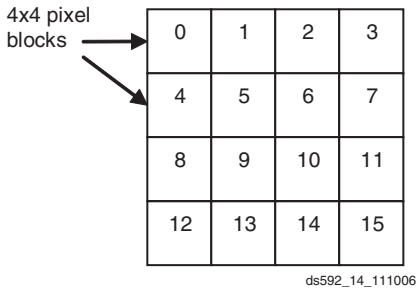

Figure 14: **mb_cbp_blk Bit Mapping for One Macroblock**

- **mb_slice_begin**:
  High when current macroblock is first in new slice. Slices can start at any macroblock. Any transition from one slice to another must be to a new slice; returning to an old slice is not permitted. Each slice must contain a number of macroblocks equal to or greater than img_mb_width.
- **mb_x(7:0)**:
  x-coordinate of current macroblock using luma pixel coordinates/16:

  first macroblock (top left) is (0, 0)
  next macroblock to right is (1, 0)… etc.
- **mb_y(7:0)**:
  y-coordinate of current macroblock using luma pixel coordinates/16:

  first macroblock (top left) is (0, 0)
  next macroblock below is (1, 0)… etc.
- **mb_field(1:0)**:
  Not used currently

Image-Based Parameters

Image-based parameters are the set of parameters that change the operation of the deblocker on an image-by-image basis. They are common to both IF 0 and IF 1 cores. Both interfaces take all image-based parameters in as direct inputs. The IF 1 wrapper takes them in and stores them internally to the wrapper in shadow registers which hold them stable during the deblocking operation. These values are described in the H.264 spec document (Ref. [1]) and also derived from values used in the JM10 reference code. Here is a brief summary:

- **img_MB_Width(7:0)**:
  Number of macroblocks across one frame of current image
- **img_MB_Height(7:0)**:
  Number of macroblocks from top to bottom of one frame of current image
- **img_chroma_format(1:0)**:
  00 = Monochrome;
  01 = 4:2:0;
  10 = 4:2:2;
  11 = 4:4:4;
  Currently only 4:2:0 supported
- **img_bit_depth(3:0)**:
  Currently supports up to 8 bits; set to “1000”
- **img_type(3:0)**:
  0000 = P_SLICE
  0001 = B_SLICE
  0010 = I_SLICE
  0011 = SP_SLICE
  0100 = SI_SLICE
- **img_structure(1:0)**:
  00 = FRAME
  01 = TOP_FIELD
  10 = BOTTOM_FIELD
- **img_LFDisable(1:0)**:
  00 = not disabled
  01 = disabled for this image
  10 = disable on slice-boundaries
- **img_chroma_qp_offset(4:0)**:
  QP offset for chroma data. Value -12 to +12

If the most recent mb_x and mb_y parameters sent to the core are (0, 0), then when these and all other parameters associated with the same macroblock are registered internally into their shadow registers for direct core usage, img_params_update is asserted High for one clock cycle. This indicates that it is safe to change the image-based parameters until the next time that mb_x, mb_y = (0,0). Note that this does not suffice as a mechanism for recovering from macroblock desynchronization. This is described in Video Synchronization and Sync Loss.

Core Module Functional Descriptions

Filter Core

The main computational engine for the Deblocker core utilizes four pixels per clock cycle to meet the performance requirements. The input format for the pixel packing is described in the Input Data Format of this document. The filter performs all filter operations along an entire single edge before moving to the next edge. This directly emulates the approach taken by the JM 10.2 reference code.

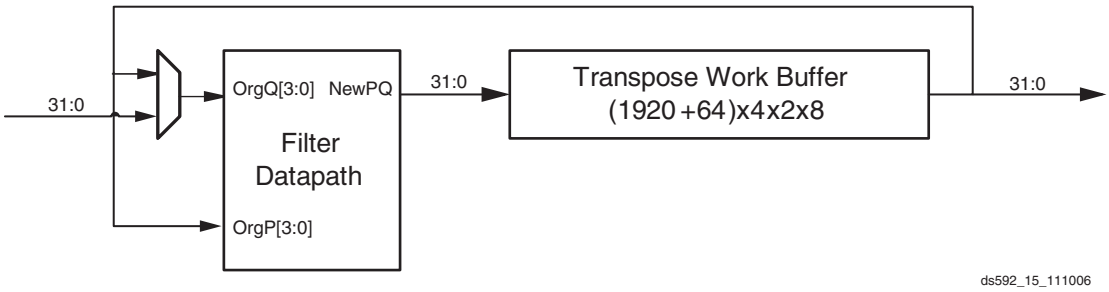

Figure 15: **Deblocker Filter Engine**

The cache in this architecture is used to store the partially filtered L input samples (previous Q output samples) and is shown as the *Work Buffer* in Figure 3. Additionally, it requires the intermediate buffer to transpose the output so that the same data format can be applied to the single processing element in the vertical domain following horizontal filtering.

This architecture requires the core running in three phases for each macroblock:

Phase 1: Horizontal Pass

The filter generates new horizontal P and Q outputs every two clocks.Horizontally filtered Q samples are stored in the work buffer. Horizontally filtered P output samples are stored in the Transpose buffer. The data should be arranged in the input object FIFO such that all four R samples are read from one memory location in the input FIFO, reducing access times by a factor of 4. The L samples are fed from the Work Buffer and are provided in parallel with the new R inputs. All 4xP and 4xQ outputs are generated in parallel in two clocks. This represents two clock cycles to create eight filter output phases.

Starting with edge V0 (see Figure 1), a 4-pixel column of 16 lines is sent in and filtered with the four equivalent L samples that are currently sitting in the Work Buffer from the last edge filter operation in the previous macroblock. This updates the Work Buffer content in preparation for the next edge filter operation. After this column has reached the base of the macroblock, the process is repeated for the next 4-pixel column.

By the end of filtering of edge V3, the Transpose buffer should contain the whole macroblock result of the horizontal pass. Assuming 8-bit input data and noting that the reference code performs clipping prior to storage on an edge-by-edge basis, this buffer needs to be capable of holding 16x16x8-bit data.

Phase 2: Vertical Pass

Data comes in from the Transpose buffer instead of the input buffer and is transposed. This means that the incoming data represents a column of four pixels. Starting with edge H0, the filter works along the edge until it reaches the right-most edge of the macroblock. This scheme is identical to the horizontal

pass. After this phase, the Transpose buffer should contain the whole macroblock result of the vertical pass.

Phase 3: De-Transpose and Output Pass

The third phase is the de-transposing of the data from the transpose buffer and preparing to write the data out in macroblock linear format as described in Figure 5.

Boundary Strength Generator

Figure 1 shows the intra- and inter-macroblock edges across which a filter operation must occur to deblock a macroblock. The extent to which the filter modifies the data across these boundaries is embodied in the Boundary Strength (BS) values. Any edge between two adjacent 4x4-pixel blocks has an associated BS value. Hence, there are 32 BS values per macroblock, that is, 16 each on Vertical and Horizontal edges.

The Boundary Strength generator (BS_Gen) uses many of the input parameters, the 4x4-pixel block-based Motion Vectors and the Reference information to generate the Boundary Strength values. The process requires around 128 clock cycles per macroblock. For any given macroblock, filtering does not commence before the BS values are available for use.

Unsupported Options

The main H.264 deblocking features not supported by either the IF 0 or IF 1 cores in version 1.0 are:

- MBAFF mode (Macroblock Adaptive Field/Frame mode)
- 4:2:2
- 4:4:4
- Monochrome
- > 8-bit video
- 1080P/60

Ordering Information

The H.264 Deblocker Core is provided under the terms of the Xilinx LogiCORE Site License Agreement which conforms to the standard licensing terms of the SignOnce program.

For part number and ordering information details, as well as instructions on how to obtain a free evaluation version of this core, visit the H.264 Deblocker product page, www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-H264-DEBLOCK
To license this core, contact your local Xilinx sales representative.

References

1. ITU-T/ISO/IEC, *Advanced Video Coding for Generic Audiovisual Services*, H.264 03/2005.

Revision History

Date	Version	Revision
05/31/07	1.0	Initial Xilinx release.