LatticeCORE

Deinterlacer IP Core User's Guide

September 2013
IPUG97_01.1

Table of Contents

Introduction

The Lattice Deinterlacer IP core converts interlaced video into progressive video format using bob, intra or inter motion adaptive deinterlacing algorithms to reduce interline flickers and jagged edges. The Deinterlacer IP core supports image sizes up to 4kx4k with YCbCr 4:2:2, 4:4:4 and RGB video formats. The Deinterlacer IP core supports dynamic parameter updating via a parameter bus which can be configured to operate on a different clock from the core. Simple frame rate conversion is employed to support different input and output frame rates.

Quick Facts

Table 1-1 gives quick facts about the Deinterlacer IP core.

Table 1-1. Deinterlacer IP Core Quick Facts

		Deinterlacer IP Core Configuration			
		PAL720x576 YCbCr4:2:2		**1080 60I to 1080 60P YCbCR4:2:2**	
Core Requirements	FPGA Families Supported	**LatticeECP2™**			
	Minimum Device Required	LFE2-12E			
Resource Utilization	Targeted Device	LFE2-50E-7F484C			
	Algorithms	Intra	Inter	Intra	Inter
	Registers	2612	3825	2657	3916
	LUTs	3144	4702	3223	4774
	EBRs	4	7	6	11
Core Requirements	FPGA Families Supported	**LatticeECP3™**			
	Minimum Device Required	LFE3-17EA			
Resource Utilization	Targeted Device	LFE3-70EA-8FN1156C			
	Algorithms	Intra	Inter	Intra	Inter
	Registers	2609	3833	2656	3916
	LUTs	3127	4690	3167	4734
	EBRs	4	7	6	11
Core Requirements	FPGA Families Supported	**LatticeXP2™**			
	Minimum Device Required	LFXP2-8E			
Resource Utilization	Targeted Device	LFXP2-40E-7F484C			
	Algorithms	Intra	Inter	Intra	Inter
	Registers	2612	3825	2657	3916
	LUTs	3144	4702	3223	4774
	EBRs	4	7	6	11
Design Tool Support	Lattice Implementation	Lattice Diamond® 1.3 or ispLEVER® 8.1 SP1			
	Synthesis	Synopsys® Synplify™ Pro for Lattice E-2011.03L			
	Simulation	Aldec® Active-HDL™ 8.2 SP1 Lattice Edition Mentor Graphics® ModelSim™ SE 6.3F			

Features

The Deinterlacer IP core supports the following features:

- Single-color, YCbCr 4:2:2, YcbCr 4:4:4 and RGB video formats
- Serial and parallel deinterlacing
- Weave, bob, intra and inter motion adaptive deinterlacing algorithms
- Frame rate conversion
- Configurable initial field
- Configurable thresholds for inter motion adaptive deinterlacing algorithm
- Dynamic parameter update of frame size, initial field and bypass mode
- Configurable parameter bus width
- Configurable parameters bus clock
- Configurable memory bus width and base address
- Configurable memory burst length and burst count
- Configurable internal FIFO type and depth
- 8, 10 or 12-bit color depth per plane
- Configurable line buffer type

Release Information

- Deinterlacer IP core version 1.0
- Last updated August 17, 2011

Device Support

- LatticeECP2™, LatticeECP2M, LatticeECP2MS, LatticeECP2S, LatticeECP3, LatticeXP2

Key Concepts

In interlaced video, a frame is divided into two fields, with each field containing every other horizontal line in a frame. One field is transmitted at a time and thus uses only half the bandwidth. Most modern displays support progressive scan, and to display interlaced video on progressive scan displays, deinterlacing, a method of combining the two fields into a frame, is applied to the video signal.

Because each frame of interlaced video is composed of two fields that are captured at different moments in time, there will be flickers and jagged edges in the combined frame. A good deinterlacing algorithm should reduce these artifacts as much as possible and yield a good video quality in the process.

The Deinterlacer IP core provides several deinterlacing algorithms for different video quality and resource requirements: weave, bob, intra and inter motion adaptive deinterlacing algorithms.

Block Diagram

Figure 2-1. Deinterlacer IP Core Block Diagram

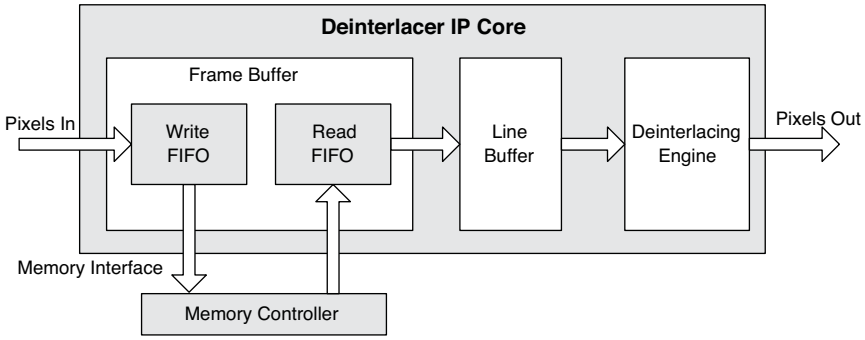


Figure 2-1 shows the block diagram of the Deinterlacer IP core. The core consists of three modules: frame buffer, line buffer and deinterlacer engine. The frame buffer module manages memory write/read and combines interlaced fields into progressive frames. The line buffer module and deinterlacing engine process the combined frames to reduce artifacts.

In the deinterlacer, several clock sources are involved. When frame rate conversion is enabled, there are two clocks in the video data path: input pixel sample clock and output pixel sample clock. The frame buffer module handles the rate conversion, and the line buffer and deinterlacing engine operate at the output pixel clock rate. When frame rate conversion is disabled, all the modules in the video data path operate at input pixel clock rate. When dynamic parameter updating is selected, the parameter bus can be configured to run on a separate clock. By default, the parameter bus runs on the input pixel sample clock. The memory interface always operates on a separate clock.

The input data must be in interlaced video format. An interlaced frame is composed of two fields. The first field in the interlaced frames can be configured to be the top field or the bottom field. This parameter can be configured at run time.

When processing YCbCr 4:2:2 video format, the core copies neighboring pixels' Cb and Cr vectors to construct the YCbCr 4:4:4 format for deinterlacing.

The Deinterlacer IP core does not use multipliers.

Deinterlacing Algorithms

The Deinterlacer IP core supports weave, bob, intra and inter motion adaptive deinterlacing algorithms.

Weave

The weave algorithm simply combines the two interlaced fields together. There is no line buffer module or deinterlacing engine instantiated in the core.

Weaving is fine when there is no change in the image between fields, but any change will result in artifacts known as “combing,” when pixels in one frame do not line up with pixels in the others, forming jagged edges.

Bob

The bob algorithm uses one single field to generate a frame by averaging up lines and down lines to generate new lines. Either even or odd fields can be selected for progressive frame generation.

Intra Motion Adaptive Deinterlacing

Figure 2-2. Intra Deinterlacing Engine Structure

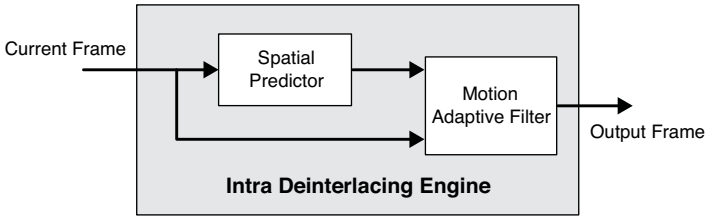


Figure 2-2 shows the structure of the intra deinterlacing engine.

The intra motion adaptive deinterlacing algorithm enhances the quality of the combined frames by using a spatial predictor with advanced ELA (Edge-based Line Average) algorithm to predict the center pixel. The intra Motion Adaptive Filter uses an enhanced multi-stage median filter with robustness control to perform motion adaptive deinterlacing.

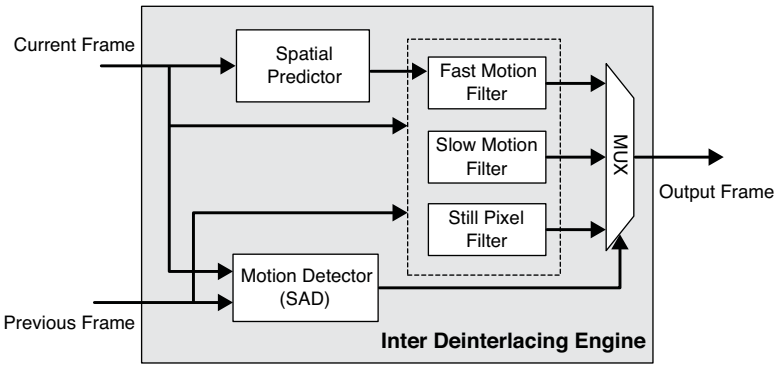
Inter Motion Adaptive Deinterlacing

Figure 2-3 shows the structure of the inter motion deinterlacing engine.

The inter motion adaptive deinterlacing algorithm uses two combined frames (four fields) to generate one progressive frame. The motion detector uses a 3x3 pixel window SAD (Sum of Absolute Differences) algorithm to determine the motion of the center pixel. The SAD value is divided by two thresholds (TH1 and TH2) into three pixel motion regions: still pixel, slow motion pixel or fast motion pixel. Each region has a separate filter with robustness control to generate the output pixel. The final output pixel value is determined by the detector value and the threshold values.

The two threshold values can be updated at run time.

Figure 2-3. Inter Deinterlacing Engine Structure



Frame Rate Conversion

The Deinterlacer IP core provides a simple frame rate conversion by copying or dropping frames.

When frame rate conversion is enabled, the core's output data path runs at the output pixel clock rate, and the core needs one more external frame memory space. The output frame rate is controlled by the output pixel sample clock and dout_enable signal. When dout_enable signal is high, the core outputs pixels continuously. When there is no new interlaced video frame, the core will output the last progressive frame repeatedly.

When frame rate conversion is disabled, the core's output data path will run on the input pixel sample clock. The output frame is generated directly from the input interlaced video stream. If there is no new input video frame, the core will stop outputting data.

Dynamic Parameters Updating

The Deinterlacer IP core provides a parameter bus port for internal parameter update at run time. The parameters are double-buffered so that the core's operation is not impacted when they are changed. The new values are buffered and transferred to the working registers when the core is ready to accept a new configuration. The UPDATE register is used to indicate when the new values are consumed and when the buffers can accept new data.

The parameter register maps are included in Table 2-1.

Table 2-1. Parameter Register Maps

Address	Register	Size	Read/Write	Description
0x00	FRMWIDTH	32	Read/Write	**Frame width register.** The FRMWIDTH value must be (frame width - 1). The minimum value is 63, and the maximum value is the maximum frame width specified on the IP GUI minus 1. The default value is the maximum value. Frame width must be an even number (i.e. FRMWIDTH must be odd).
0x04	FRMHEIGHT	32	Read/Write	**Frame height register.** The FRMHEIGHT must be (frame height - 1). The minimum value is 63, and the maximum value is the maximum frame height specified on the IP GUI minus 1. The default value is the maximum value. Frame height must be an even number (i.e. FRMHEIGHT must be odd).
0x08	INITFLD	32	Read/Write	**First field register.** Specifies the first field in the interlaced input frames. 0 for top field and 1 for bottom field. The default value is the corresponding parameter specified on the IP GUI.
0x0C	BYPASS	32	Read/Write	**BYPASS mode register.** The value can be 0 or 1. When the BYPASS register is 1, the Deinterlacer IP core passes the interlaced video through unaltered. The default value is 0.

Address	Register	Size	Read/Write	Description
0x10	KEEP	32	Read/Write	**KEEP mode register.** The value can be 0 or 1. When the KEEP register is 1, the core will be locked. The core outputs the same frame repeatedly if frame rate conversion is active; otherwise it generates no output frames. The default value is 0.
0x14	INTER_TH1	32	Read/Write	**SAD threshold for still pixel for the inter algorithm**. The maximum value is 65535. N/A for other algorithms. The default value is the corresponding parameter specified on the IP GUI.
0x18	INTER_TH2	32	Read/Write	**SAD threshold for slow motion pixel for inter algorithm**. The maximum value is 65535. N/A for other algorithms. The default value is the corresponding parameter specified on the IP GUI. The INTER_TH2 must be greater than INTER_TH1 value.
0x1C	UPDATE	32	Read/Write	**Update parameters enable register**. The value can be 0 or 1. Setting this bit to 1 triggers the update of parameters. The core resets it to 0 after the parameters have been updated. The default value is 0.

All the parameter registers can be written to only when the UPDATE register bit is 0. The parameters KEEP, INTER_TH1 and INTER_TH2 will take effect at the next frame, regardless of the UPDATE value. When the UPDATE bit is set to 1, the parameters FRMWIDTH, FRMHEIGHT, INTIFLD and BYPASS will take effect when the frmsync_in signal is active, indicating a new input frame is arriving. After updating its internal parameters, the core resets the UPDATE bit to indicate that the parameter registers are now empty and can take on new values.

Memory Bandwidth and Size

The Deinterlacer IP core stores and retrieves pixels to/from external memory using memory burst write and read commands. When DDR2 is used for external memory, one burst operation, (burst_length*burst_count/2)*memory_data_width bit, cannot exceed the size of a single video line. A single video line will be transferred though multiple burst write/read transactions internally.

For weave, bob and intra algorithms, the Deinterlacer IP core needs a 2-frame memory storage space; if frame rate conversion is active, a 3-frame storage space is required. When the output frame rate is the same as the input frame rate, the required memory bandwidth is twice the input data rate. When the output frame rate is double the input frame rate, the required memory bandwidth is triple the input data rate.

For the inter algorithm, the Deinterlacer IP core uses two interlaced frames to generate one progressive frame; it needs a 3-frame storage space. If frame rate conversion is enabled, the core needs a 4-frame memory space. When the output frame rate is the same as the input frame rate, the required memory bandwidth is triple the input data rate. When the output frame rate is double the input frame rate, the required memory bandwidth is five times the data rate of the input video stream.

For example, for 8-bit YCbCr 4:2:2 pixel, parallel deinterlacing, if the input pixel sample clock is 74.25MHz and output pixel sample clock is 148.5MHz, the required bandwidth for the intra algorithm is 2*8*(74.25+148.5) = 3564 bit*MHz. The required bandwidth for the inter algorithm is 2*8*(74.25+2*148.5) = 5940 bit*MHz. If the memory data with is 32, the require memory clock will be (3564/32) = 111.375 MHz for the intra algorithm and (5940/32) = 185.625MHz for the inter algorithms.

The total external memory size the core requires can be viewed on the Deinterlacer IP GUI.

Primary I/O

Figure 2-4. Deinterlacer IP Core I/O

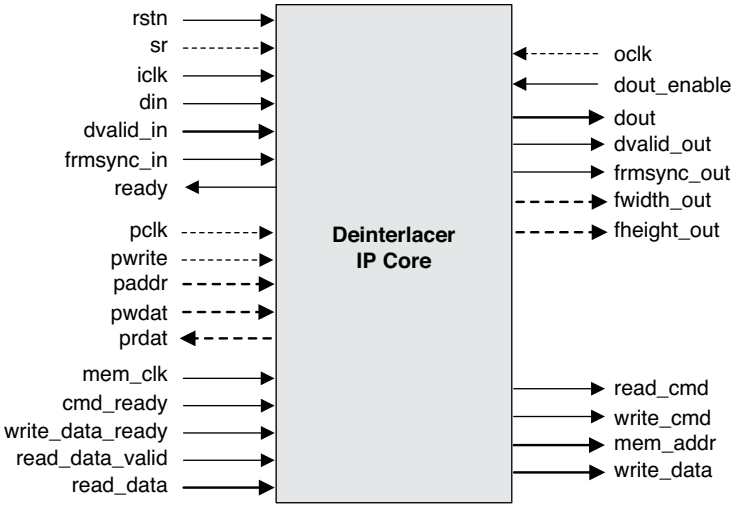


Table 2-2. Primary I/O

Port	Size	I/O	Description
General I/Os			
rstn	1	I	Asynchronous active-low reset signal
sr	1	I	Synchronous reset signal (optional)
iclk	1	I	Input pixel sample clock
frmsync_in	1	I	New input video frame indicator, active-high
dvalid_in	1	I	Input video data valid signal, active-high
din	8/10/12	I	Input video data in interlaced frame format
ready	1	O	When high, indicates that the Deinterlacer IP core can accept more input data
oclk	1	I	Output sample clock, present when frame rate conversion is active
dout_enable	1	I	Input from down-stream module to enable progressive output data, active-high
dout	8/10/12	O	Output video pixel in progressive frame format
dvalid_out	1	O	Output video pixel valid signal, active high
frmsync_out	1	O	New output frame indicator, active high
Memory Interface			
mem_clk	1	I	Memory write/read clock
cmd_ready	1	I	Input from memory controller indicating it is ready to accept a new command, active-high
mem_addr	32	O	Memory read/write address
read_cmd	1	O	Memory read command, active-high
read_data	8/16/32/ 64/128	I	Read data output from memory
read_data_valid	1	I	Read data valid indicator from memory controller, active-high
write_cmd	1	O	Memory write command, active-high
write_data	8/16/32/ 64/128	O	Write data to memory
write_data_ready	1	I	Input from memory controller indicating that it is ready to accept new write data, active-high
Optional I/Os			

Port	Size	I/O	Description
fwidth_out	16	O	Current output frame width, only for dynamic mode
fheight_out	16	O	Current output frame height, only for dynamic mode
pclk	1	I	Parameters bus clock, configurable
pwrite	1	I	Parameters bus write enable, active-high
paddr	5	I	Parameters bus address
pwdat	8/16/32	I	Parameters bus write data
prdat	8/16/32	O	Parameters bus read data

Interface Descriptions

Video Input/Output

The Deinterlacer IP core uses a simple handshaking method to pass pixel data into and out of the core. The core asserts its *ready* output when it is ready to receive data. When the driving module has data to pass to the deinterlacer, it asserts the core's *dvalid_in* port and at the same time places the input video data on the *din* port. The frmsync_in input should be driven to a '1' during the clock cycle when the very first pixel of the interlaced frame is active.

Similarly, *dvalid_out* is active when valid output pixel data is available on *dout*, and *frmsync_out* marks the first pixel in the output progressive frames.

The ports *fwidth_out* and *fheight_out* indicate the current output frame's width and height, respectively, These are valid only when frmsync_out is asserted.

When the input signal *dout_enable* is asserted, the core will output progressive video pixels. When *dout_enable* is de-asserted, the core stops generating output pixels after some delay, which is less than 16 output pixel sample clock cycles.

Memory Interface

The Deinterlacer IP core implements a flexible memory interface which operates on a separate clock from the main core.

When connecting to DDR2 memory controller, the burst length and burst count of deinterlacer should have the same values as those of DDR2 memory controller; when connecting to DDR3 memory controller, the burst length of deinterlacer should have the same value with DDR3 memory controller, while the burst count of deinterlacer should be half as that of DDR3 memory controller.

The deinterlacer assumes a memory byte addressing scheme. When connecting the memory controller, the address connection should be adjusted according to the DDR memory data width.

Normally the address connection can be (Assuming the DDR2/3 memory has 27 bits address width):

```
ddr_addr        = {1'b0,mem_addr[25:0]}; // for 8-bit DDR memory only

ddr_addr        = {2'b00,mem_addr[25:1]}; // for 16-bit DDR memory only

ddr_addr        = {3'b000,mem_addr[25:2]}; // for 32-bit DDR memory only
```

And the corresponding command should be:

```
ddr_cmd         = (write_cmd) ? 4'b0010 : 4'b0001; // (write_cmd) ? WRITE : READ

ddr_cmd_valid       = (write_cmd || read_cmd);
```

To get maximum throughput on the memory bus (about 5% increase compared to normal connection), the user needs to steer clear of write-to-precharge/read-to-precharge/precharge-to-active during row switching. That is, to access each bank in turn by using write/read with auto-precharge command to close current bank immediately after current burst write/read. In order to adopt this policy, the users should fine tune the combination of Row/Bank/Column address to get the best throughput.

For example, when DDR2 memory data width is 16, row size is 14, column size is 10, bank size is 8, burst length is 8 and burst count is 1, which means memory controller user interface data width is 32, row address is 14 bits, column address is 10 bits, bank address is log2(bank size)=3 bits and log2(burst length * burst count)= 3 bits. The memory controller address mapping inside the memory controller IP core is

```
ddr_addr = {row_addr[13:0], bank_addr[2:0], col_addr[9:0]}
```

In order to get the best throughput, we increase the col_addr[2:0] first to utilize the burst operation, then we increase the bank_addr[2:0], followed by increasing the rest of col_addr[9:3], and finally with the row_addr. So the interface to the DDR2/3 memory will look like:

```
ddr_addr = {2'b00, mem_addr[25:14], mem_addr[6:4], mem_addr[13:7], mem_addr[3:1]};
```

The corresponding command should be:

```
ddr_cmd = (write_cmd) ? 4'b0100 : 4'b0011; // (write_cmd) ? WRITEA : READA
```

As the memory controller user interface data width is 32, mem_addr[1:0] will always be zero.

The core has internal control logic to arbitrate between memory write and read operations, ensuring the *write_cmd* and *read_cmd* are not asserted at the same time.

Two clock cycles after the *write_data_ready* is asserted, data will be available on the *write_data* port. That means the parameter “Data_rdy to Write Data Delay” of memory controller must be set to 2

The *cmd_ready* can be asserted once every two clock cycles, and it should have at least a one-cycle interval, which is consistent with Lattice DDR Memory Controller IP cores.

Parameter Register Read/Write Interface

The Deinterlacer IP core implements a simple register read/write interface for run-time parameter updates. The parameter bus interface can be configured to run on a separate clock. It operates at the input pixel clock rate by default.

When *pwrite* is high, *pwdat* and *paddr* must contain valid data. The contents of all parameter registers will be transferred to the core’s internal storage when UPDATE is asserted. If a parameter hadn’t been written to before the assertion of UPDATE, its old value will be transferred into the internal storage.

prdat contains register read data corresponding to the address value placed on the *paddr* in the previous clock cycle.

When the parameter bus data width is equal to 32, *paddr[1:0]* should be fixed to 0. When the parameter bus data width equals 16, *paddr[0]* should be fixed to 0.

The parameter bus data width should be configured based on the system CPU’s data width.

Timing Specifications

Video Input/Output Timing

The Deinterlacer IP core supports single-color, YCbCr 4:2:2, YCbCr 4:4:4 or RGB video format.

For YCbCr 4:4:4 or RGB video format, the three planes are interleaved for serial deinterlacing and combined on the *din* and *dout* ports for parallel deinterlacing. Figures 2-5 and 2-6 show the timing of RGB serial deinterlacing and parallel deinterlacing.

Figure 2-5. RGB Serial Deinterlacing

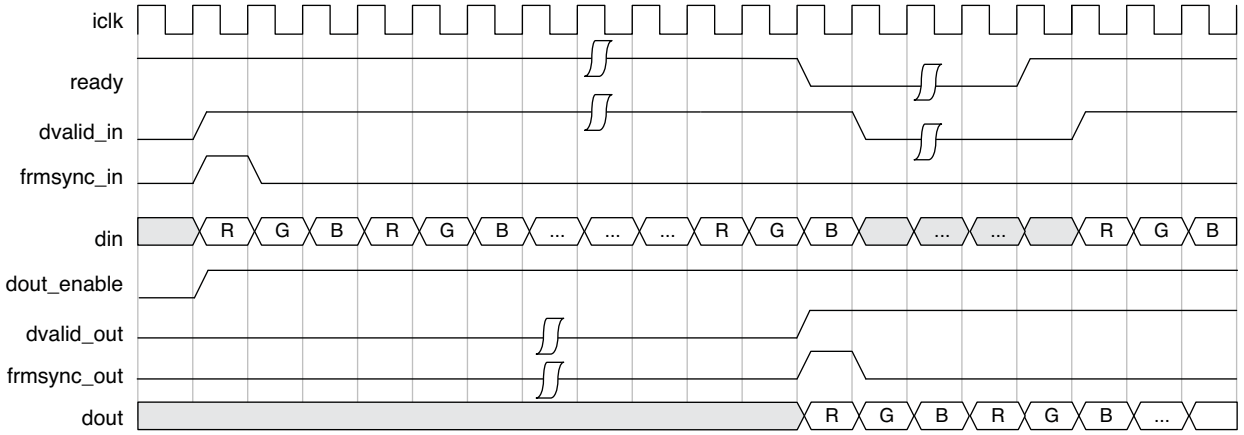
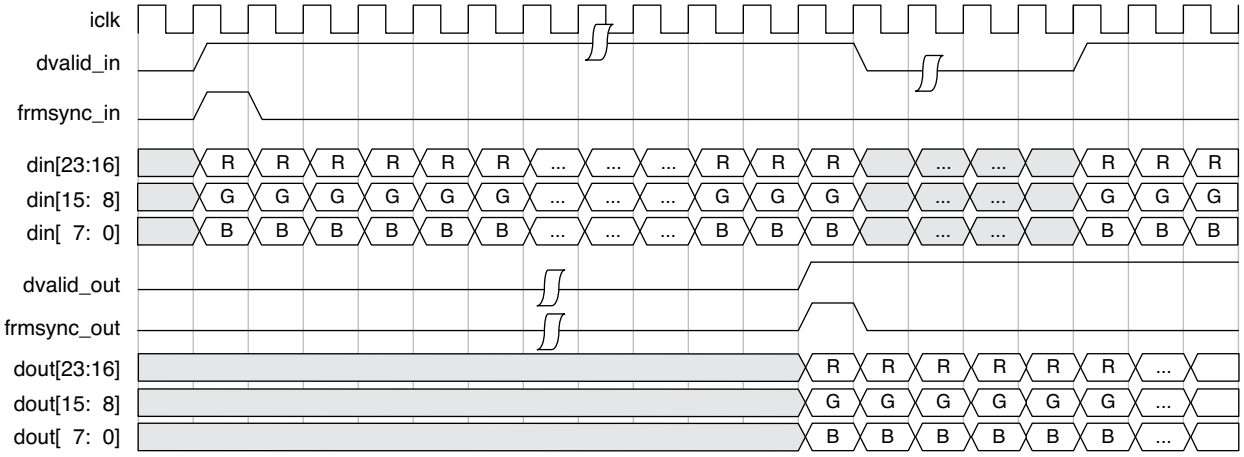


Figure 2-6. RGB Parallel Deinterlacing (8-Bit Pixel)



For YCbCr 4:2:2 video serial deinterlacing, the input and output sequence should be Cb, Y, Cr, Y, …. For parallel deinterlacing, the Y plane occupies the upper bits of the *din* and *dout* ports, and the Cb and Cr planes the lower bits. Cb and Cr planes are interleaved in the lower half, and Cb comes before Cr. Figures 2-7 and 2-8 show the timing of YCbCr 4:2:2 serial deinterlacing and parallel deinterlacing.

Figure 2-7. YCbCr 4:2:2 Serial Deinterlacing

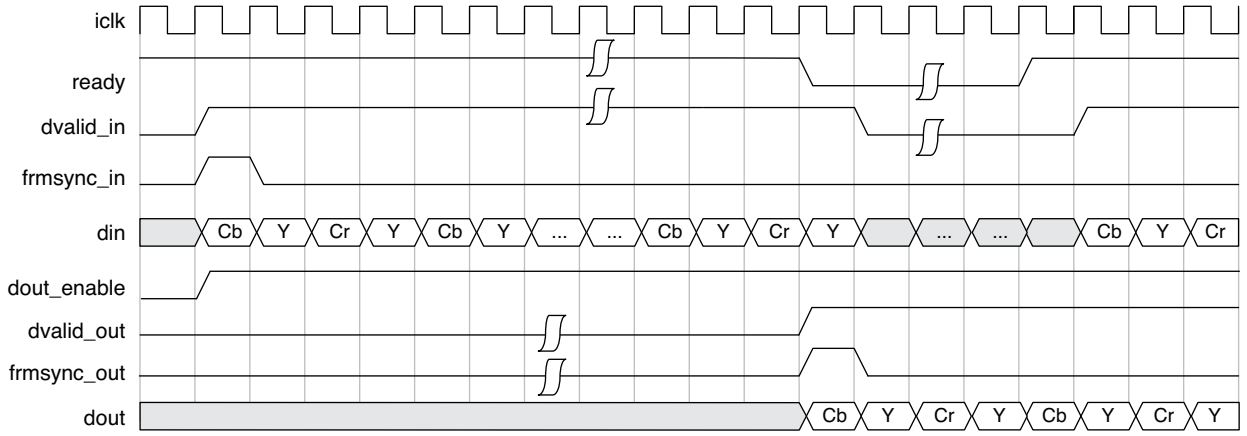


Figure 2-8. YCbCr 4:2:2 Parallel Deinterlacing (8-Bit Pixel)

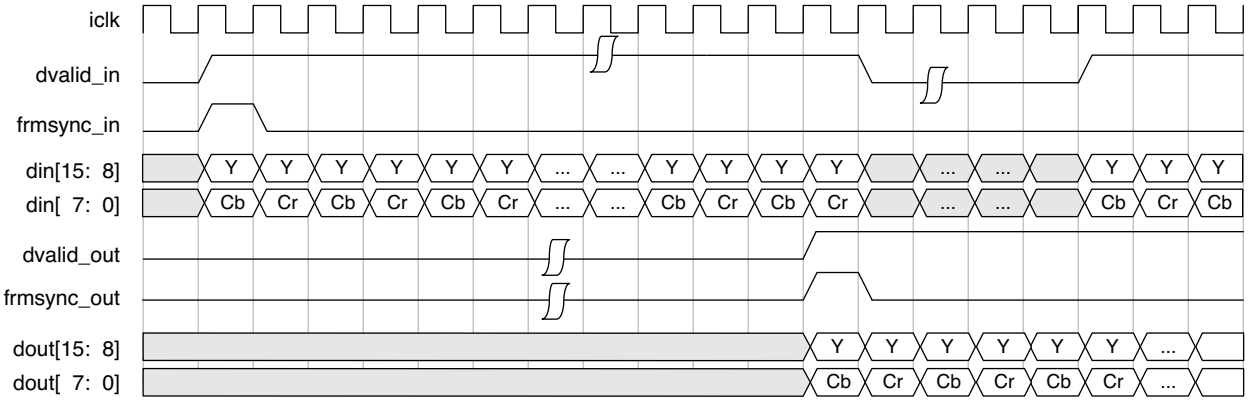
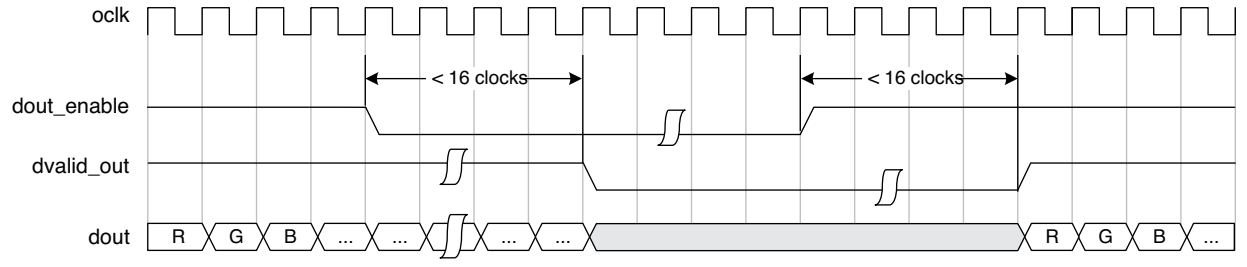


Figure 2-9 shows the *dout_enable* control timing. When the *dout_enable* is de-asserted, the core will stop outputting data after a certain delay. Similarly, when the *dout_enable* is asserted, the core will begin outputting data after a certain delay. The number of delay clocks is different for different deinterlacing algorithms and the maximum value is less than 16 clocks. The asserting and de-asserting of *dout_enable* can be used to generate horizontal blanks and vertical blanks depending on the output video format.

Figure 2-9. dout_enable Control Timing



Video Frame Timing

Figure 2-10 shows the frame timing when frame rate conversion is disabled. After accepting one input interlaced frame, the core starts generating progressive frames. The output frame is driven by the input frame. When a new input frame arrives before the current output frame is finished, the current output frame is dropped, and a new output frame is started. After the last input frame, the deinterlacer stops generating output frames.

Figure 2-10. Output Frame Rate is the Same as the Input Frame Rate

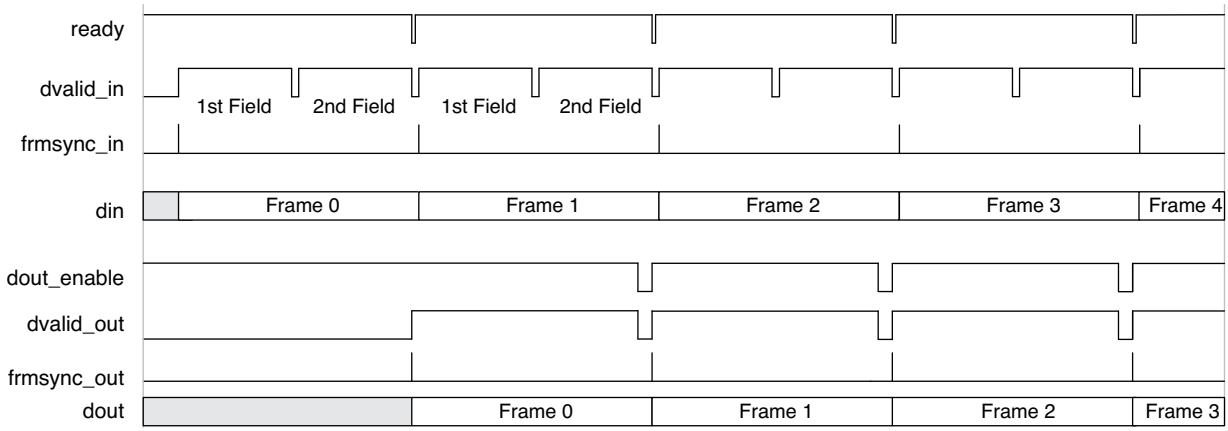
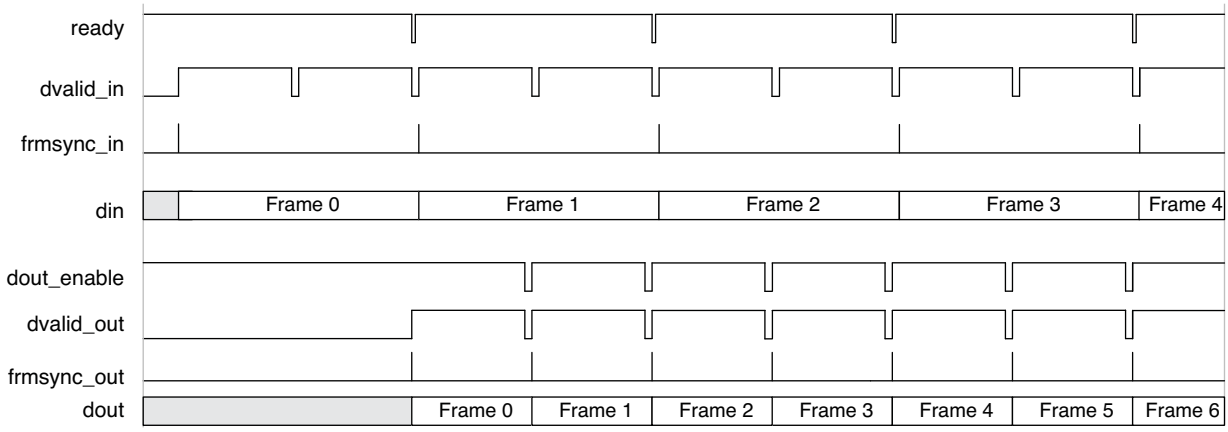


Figure 2-11 shows the frame timing when the output frame rate is twice the input frame rate. Output frames run on a separate output pixel clock. The output frame rate is determined by the output pixel clock and *dout_enable* signal. The core generates output frames based on the oldest pending input frame(s). When a new input frame is received, the core pushes out the oldest frame and exports the second oldest frame. When there is no new frame input, the core exports the stored frames and then keeps exporting the last frame repeatedly. When the external memory is fully occupied by the unconsumed frames (no more space for new frames), the core will overwrite the latest input frame.

Figure 2-11. Output Frame Rate is Twice Input Frame Rate



Memory Interface Timing

Figure 2-12 shows the timing of the memory write operation. When the internal write FIFO is half full, it triggers memory write operation cycles. Memory write operation will continue until the write FIFO is empty.

Figure 2-12. Timing Diagram for Memory Write

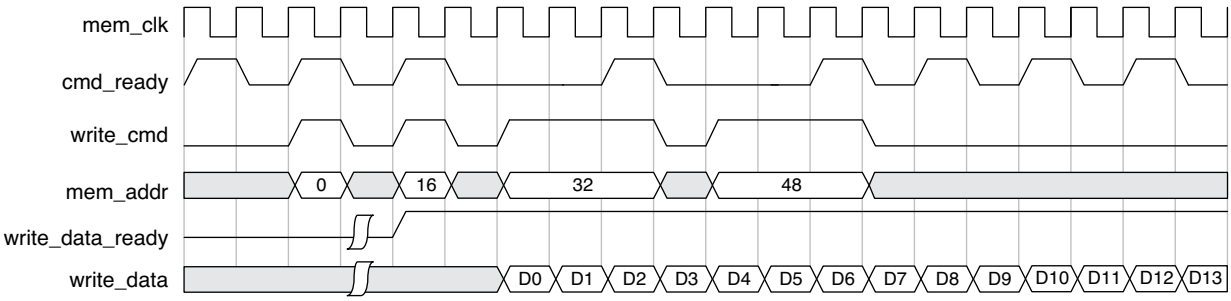
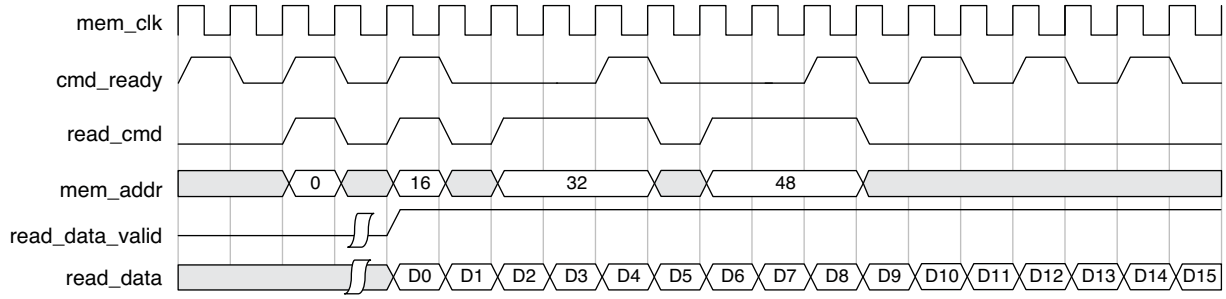


Figure 2-13 shows the timing of the memory read operation. When the internal read FIFO is less than half full, and there is no ongoing write operation, memory read burst operation is started. The length of the read burst operation and once read burst makes the read FIFO half full.

The memory write and read operations are in bursts. The memory write and read operations switch at the end of the burst cycle.

Figure 2-13. Timing Diagram for Memory Read

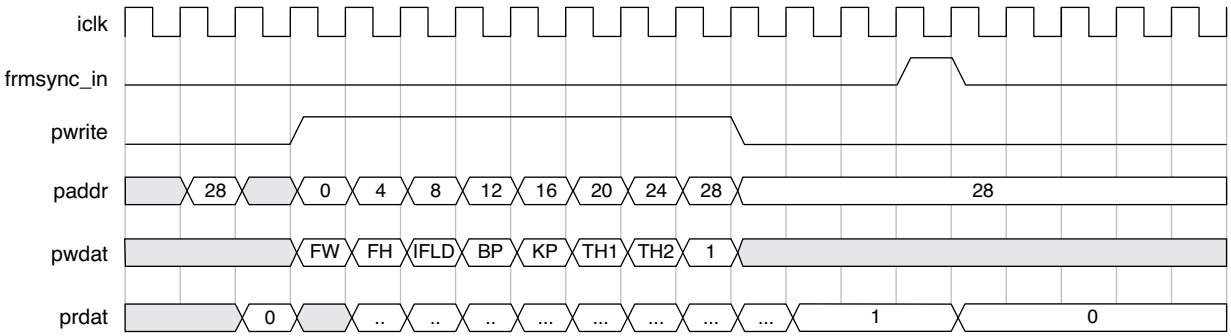


Dynamic Parameters Updating

Figure 2-14 shows the timing of dynamic parameters updating. FW and FH are the video frame width and height. IFLD is the INITFLD register value, BP is the BYPASS register value and KP is the KEEP register value.

The parameter bus can be configured to operate on a separate clock. By default, it operates at input pixel clock rate. The parameter registers are writable only when the UPDATE register is 0. When the UPDATE register bit is set to 1, the core will update its internal parameter registers with the new values at the next active *frmsync_in* and reset the UPDATE register bit.

Figure 2-14. Timing Diagram for Dynamic Parameters Updating (parameter bus width = 32)



Parameter Settings

This section describes how to generate the Deinterlacer IP core using the Diamond or ispLEVER IPexpress™ tool. Refer to "IP Core Generation and Evaluation" on page 22 for a description of how to generate the IP.

The Deinterlacer configuration GUI is accessed via the IPexpress tool and provides an interface for setting the desired parameters and invoking the IP core generator. Since the values of some parameters affect the size of the resultant core, the maximum value for these parameters may be limited by the size of the target device. Table 3-1 provides a list of user-configurable parameters for the Deinterlacer IP core.

Table 3-1. Deinterlacer IP Core Parameters

Parameter	Range/Options	Default
Video Frame In and Out		
Video format	Single color, YCbCr4:2:2, YcbCr4:4:4 or RGB	YCbCr4:2:2
(Max.) Video frame width	64-4096	720
(Max.) Video frame height	64-4096	480
First field	Top field, Bottom field	Top field
Parallel processing	Yes, No	Yes
Dynamic parameters updating	Yes, No	No
Frame rate conversion	Yes, No	Yes
Duplicate frames in frame rate conversion mode	Yes, No	Yes
Filter Specification		
Deinterlacer algorithms	weave, bobe, bobo, intra, inter	inter
SAD threshold 1	0-65535	90
SAD threshold 2	0-65535	180
Default SAD thresholds value	Yes, No	No
I/O Specifications		
Input pixel width	8, 10, 12	8
Memory bus width	8, 16, 32, 64, 128	32
Memory base address	0x00000000-0xFFFFFFFF	0x00000000
Memory address width	Read only	22
Memory size needed	Read only	2764800 bytes
Parameter bus width	8, 16, 32	32
Separate parameter bus clock	Yes, No	No
Synchronous reset	Yes, No	No
Output frame size ports	Yes, No	No
Memory Type		
Line buffer type	EBR, Distributed	EBR
Write FIFO type	EBR, Distributed	EBR
Read FIFO type	EBR, Distributed	EBR
Write FIFO depth	32, 64, 128, 256, 512	64
Read FIFO depth	32, 64, 128, 256, 512	64
DDR memory burst length	2, 4, 8	8
Command burst count	1, 2, 4, 8	1
Synthesis options		
Frequency constraint (MHz)	1-400	250

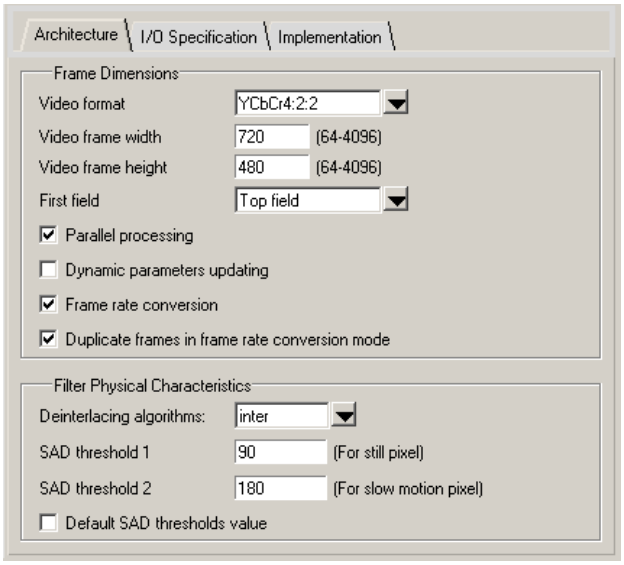
Table 3-1. Deinterlacer IP Core Parameters (Continued)

Parameter	Range/Options	Default
Fanout limit	1-200	100
Resource sharing	Yes, No	Yes
Pipelining and retiming	Yes, No	No

Architecture

The Architecture tab provides settings for video frames and algorithm options.

Figure 3-1. Architecture Tab



Frame Dimensions

This section provides settings that define the video format, input/output frame dimensions and dynamic parameters updating.

Video format defines the format of the video stream. It can be single-color, YCbCr4:2:2, YCbCr4:4:4 or RGB.

Video frame width and **Video frame height** define the video frame size for deinterlacing.

First field defines the first field in the input interlaced frames. It can be top field or bottom field.

The **Parallel processing checkbox** determines whether the core processes video stream parallelly.

The **Dynamic parameters updating** checkbox determines whether the core supports parameters updating at run-time. Refer to "Dynamic Parameters Updating" on page 8 for more information.

The **Frame rate conversion** checkbox determines whether the core supports frame rate conversion. When it is enabled, the output video stream runs on a separate clock.

The **Duplicate frames in frame rate conversion mode** checkbox determines whether the core uses duplicated frames when frame rate conversion is enabled.

When dynamic parameters updating is enabled, the **Max video frame width** and **Max video frame height** specify the largest frame size the core needs to support.

The range for the frame dimensions is 64 to 4,096.

Filter Physical Characteristics

Deinterlacing algorithms selects the deinterlacing algorithm of the core. It can be weave, bobe (bob even), bobo (bob odd), intra or inter.

SAD threshold 1 defines the threshold value for still pixel for the inter deinterlacing algorithm.

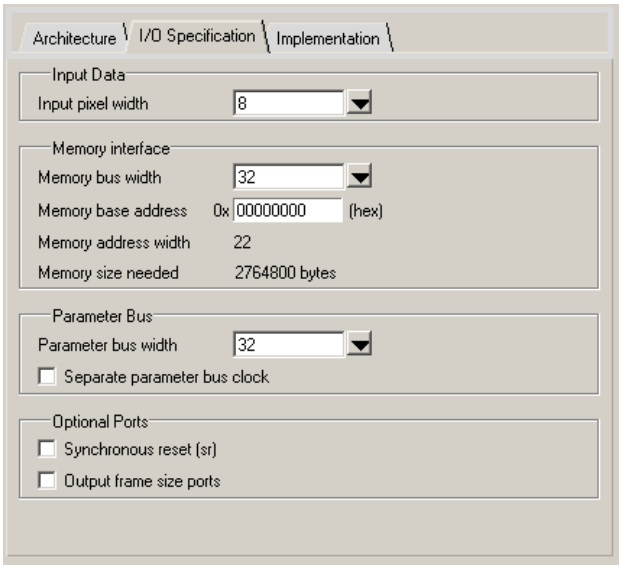
SAD threshold 2 defines the threshold value for slow motion pixel for the inter deinterlacing algorithm.

The **Default SAD thresholds value** checkbox determines whether the core uses the default SAD threshold value. When dynamic parameters updating is enabled, the two threshold values can be updated at run time.

I/O Specification

The I/O Specification tab provides settings for pixel data width, memory bus width, memory base address, parameter bus width and optional ports.

Figure 3-2. I/O Specification Tab



Input pixel width sets the bit width of the incoming pixel. Values can be 8, 10 and 12. Default value is 8.

Memory bus width sets the data width of the memory interface. Values can be 8, 16, 32, 64 and 128. Default value is 32.

Memory base address sets the base address of the external memory space used. Value must be in hex format and its bit width cannot exceed 32 bits.

Memory address width specifies the address width of the memory interface. **Memory space needed** specifies the size of the external memory space needed. Memory address width and memory size needed are generated automatically by the GUI.

Parameter bus width sets the bus width of the parameter bus interface. This parameter is only available when dynamic parameters updating is selected. Values can be 8, 16 and 32. Default value is 32.

The **Separate parameter bus clock** checkbox determines whether the core uses a separate clock to run the parameter update interface. This parameter is only available when dynamic parameters updating is selected.

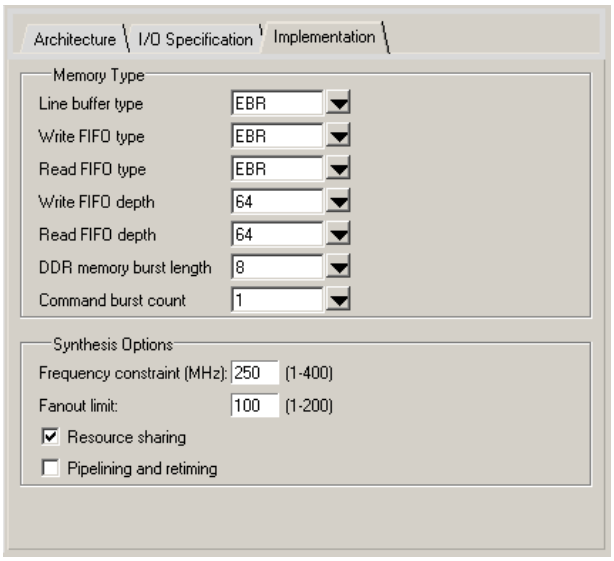
The **Synchronous reset** checkbox determines whether the core has a synchronous reset port.

The **Output frame size ports** checkbox determines whether the core provides output frame size ports. This parameter is only available when dynamic parameters updating is selected.

Implementation

The Implementation tab provides settings for the memory type and synthesis constraints.

Figure 3-3. Implementation Tab



Line buffer type selects EBR or Distributed RAM for the line buffer implementation.

Write FIFO type selects EBR or Distributed RAM for the internal write FIFO type of the frame buffer module. **Read FIFO type** selects EBR or Distributed RAM for the internal read FIFO type of the frame buffer module. **Write FIFO depth** selects the depth of the internal write FIFO. **Read FIFO depth** selects the depth of internal read FIFO. The FIFO depth can be 32, 64, 128, 256 or 512. The default value is 64.

DDR memory burst length selects the burst length value. **Command burst count** selects the burst count value for working with the memory controller. The burst length can be 2, 4 and 8. Its default value is 8. The burst count can be 1, 2, 4 and 8. Its default value is 1.

Frequency constraint sets the required clock frequency in MHz. This option is active for all the clock domains in the core. **Fanout limit** sets the fanout limit value for the synthesis tool. **Resource sharing** and **Pipelining and retiming**, if enabled, are synthesis directives that are used in the core generation. Users can adjust these options to get a better timing result.

IP Core Generation and Evaluation

This section provides information on how to generate the Deinterlacer IP core using the Diamond or ispLEVER IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the Deinterlacer IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are provided on the IP Core Purchasing and Licensing Process page of the Lattice web site.

Users may download and generate the Deinterlacer IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The Deinterlacer IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 27 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond or the ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation time-out limitation.

Getting Started

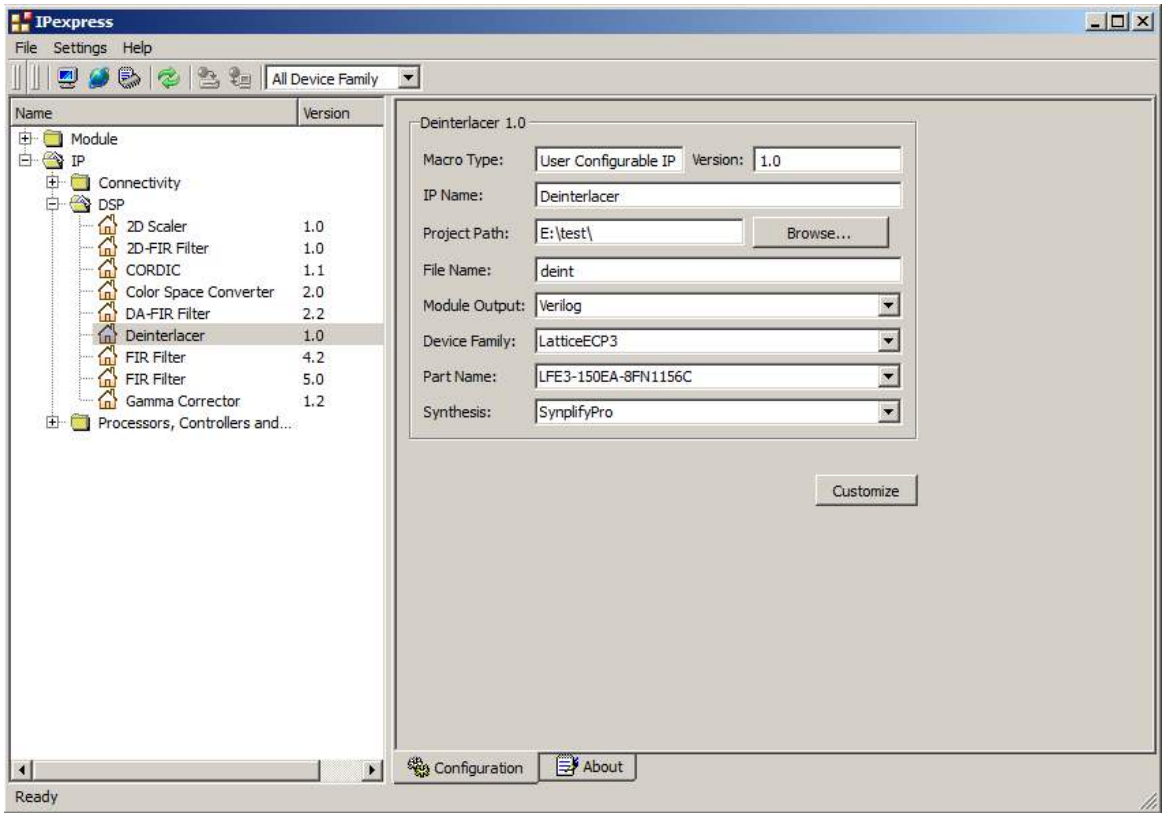
The Deinterlacer IP core is available for download from the Lattice IP server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any user-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1. To generate a specific IP core configuration, the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files
- **(Diamond) Module Output** – Verilog or VHDL
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL
- **Device Family** – Device family to which the IP core is to be targeted (e.g. LatticeECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family

Configuring the Deinterlacer IP Core in IPexpress

The Deinterlacer configuration GUI is accessed via the IPexpress tool, and provides an interface for setting the desired parameters and invoking the IP core generator. The start-up IPexpress page allows the user to select the IP to be generated, project directory, user-designated module name, design entry type, and target device. The "File Name" will be used as the username in the core generation. The Deinterlacer IP core is found under **IP > DSP**, as shown below.

Figure 4-1. IPexpress Dialog Box (Diamond Version)

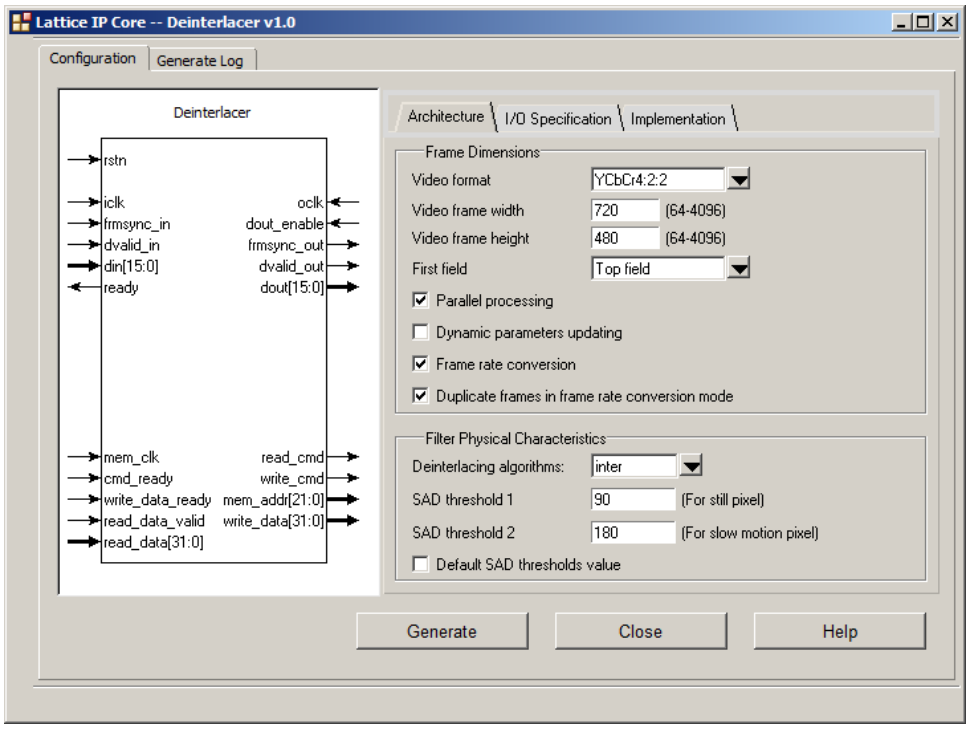


Important Note: File Name cannot be "deinterlacer_core," as this name has been used in the internal design of the core.

Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the Deinterlacer IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to See "Parameter Settings" on page 18. for more information on the Deinterlacer IP core parameter settings.

Figure 4-2. Configuration GUI (Diamond Version)



IPexpress-Created Files and Top-Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3. This example shows the directory structure generated with the Deinterlacer IP core for the LatticeECP3 device.

Figure 4-3. Deinterlacer IP Core Directory Structure

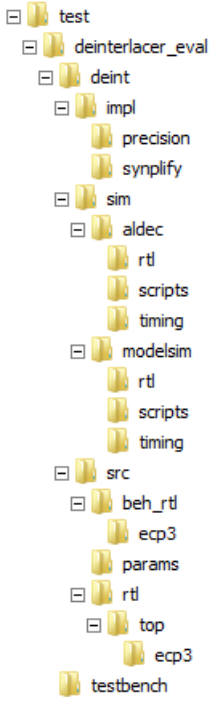


Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Description
<username>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<username>.ipx	The IPX file holds references to all of the elements of an IP or module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/module generation GUI when an IP/module is being re-generated.
<username>.ngo	This file provides the synthesized IP core.
<username>_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>_inst.v	This file provides an instance template for the Deinterlacer IP core.
<username>_beh.v	This file provides the front-end simulation library for the Deinterlacer IP core.

Table 4-2 provides a list of additional files providing IP core generation status information and command line generation capability that are generated in the user's project directory.

Table 4-2. Additional Files

File	Description
<username>_generate.tcl	This file is created when the GUI **Generate** button is pushed. This file may be run from the command line.
<username>_generate.log	This is the synthesis and map log file.
<username>_gen.log	This is the IPexpress IP generation log file.

Instantiating the Core

The generated Deinterlacer IP core package includes black-box (<username>_bb.v) and instance (<username>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in \<project_dir>\deinterlacer_eval\<username>\src\rtl\top. Users may also use this top-level reference as the starting template for the top level for their complete design.

Running Functional Simulation

Simulation support for the Deinterlacer IP core is provided for the Aldec Active-HDL (Verilog and VHDL) simulator and the Mentor Graphics ModelSim simulator. The functional simulation includes a configuration-specific behavioral model of the Deinterlacer IP core. The test bench sources stimulus to the core and monitors output from the core. The generated IP core package includes the configuration-specific behavior model (<username>_beh.v) for functional simulation in the "Project Path" root directory. The simulation scripts supporting ModelSim evaluation simulation is provided in:

\<project_dir>\deinterlacer_eval\<username>\sim\modelsim\scripts

The simulation script supporting the Active-HDL evaluation simulation is provided in:

\<project_dir>\deinterlacer_eval\<username>\sim\aldec\scripts

Both ModelSim and Active-HDL simulation is supported via test bench files provided in:

\<project_dir>\deinterlacer_eval\testbench

Models required for simulation are provided in the corresponding \models folder. Users may run the Active-HDL evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute Macro**.
3. Browse to folder \<project_dir>\deinterlacer_eval\<username>\sim\aldec\scripts and execute one of the “do” scripts shown.

Users may run the ModelSim evaluation simulation by doing the following:

1. Open ModelSim.
2. Under the **File** tab, select **Change Directory** and choose the folder <project_dir>\deinterlacer_eval\<username>\sim\modelsim\scripts.
3. Under the **Tools** tab, select **Execute Macro** and execute the ModelSim “do” script shown.

*Note: When the simulation is complete, a pop-up window will appear asking “Are you sure you want to finish?” Choose **No** to analyze the results. Choosing **Yes** closes ModelSim.*

Synthesizing and Implementing the Core in a Top-Level Design

Synthesis support for the Deinterlacer IP core is provided for Mentor Graphics Precision RTL or Synopsys Synplify. The Deinterlacer IP core itself is synthesized and is provided in NGO format when the core is generated in IPexpress. Users may synthesize the core in their own top-level design by instantiating the core in their top level as described previously and then synthesizing the entire design with either Synplify or Precision RTL synthesis.

The top-level file <username>_eval_top.v provided in:

\<project_dir>\deinterlacer_eval\<username>\src\top

supports the ability to implement the Deinterlacer IP core in isolation. Push-button implementation of this top-level design with either Synplify or Precision RTL synthesis is supported via the project files

<username>_eval.ldf (Diamond) or
.syn (ispLEVER)

located in the

\<project_dir>\deinterlacer_eval\<username>\impl\synplify and
\<project_dir>\deinterlacer_eval\<username>\impl\precision

directories, respectively.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to **\<project_dir>\deinterlacer_eval\<username>\impl\(synplify or precision)** in the Open Project dialog box.
3. Select and open **<username>.ldf**. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the **Process** tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project**.
2. Browse to **\<project_dir>\deinterlacer_eval\<username>\impl\(synplify or precision)** in the Open Project dialog box.
3. Select and open **<username>.syn**. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level entry in the left-hand GUI window.
5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The Deinterlacer IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the **Build Database** process and choose **Properties** from the drop-down menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the Target box.
4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate**. The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the **About** tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.

7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.
4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the **About** tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

Support Resources

Lattice Technical Support

There are a number of ways to receive technical support.

E-mail Support

techsupport@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

References

- HB1009, LatticeECP3 Family Handbook
- HB1003, LatticeECP2/M Family Handbook
- HB1004, LatticeXP2 Family Handbook

Revision History

Date	Document Version	IP Core Version	Change Summary
August 2011	01.0	1.0	Initial release.
September 2013	01.1	1.1	Added information in the Memory Interface section. Provided the scheme on improving data bus throughput.
			Updated corporate logo.
			Updated Lattice Technical Support information.

Appendix A:

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the Deinterlacer IP core.

IPexpress, the Lattice IP configuration utility, is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the use of IPexpress can be found in the IPexpress, Diamond or ispLEVER help systems. For more information on the Diamond or ispLEVER design tools, visit the Design Software page of the Lattice web site.

LatticeECP2 and LatticeECP2S Devices

Table A-1. Performance and Resource Utilization[1]

Video Format	Frame Width	Frame Height	Deinterlacer Algorithm	Pixel Width	Memory Bus Width	Frame Rate Conversion	Registers	LUT4s	EBRs	f_{MAX} (iclk)	f_{MAX} (mem_clk)	f_{MAX} (oclk)
YCbCr4:2:2	720	576	intra	8	32	Yes	2612	3144	4	254	278	258
YCbCr4:2:2	720	576	inter	8	32	Yes	3825	4702	7	294	260	256
YCbCr4:2:2	1920	1080	intra	8	32	Yes	2657	3223	6	248	262	250
YCbCr4:2:2	1920	1080	inter	8	32	Yes	3916	4774	11	267	244	242

1. Performance and utilization data are generated targeting a LFE2-50E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2 family.

Ordering Part Number

The Ordering Part Number (OPN) for the Deinterlacer IP core targeting LatticeECP2/S devices is DLACE-P2-U1.

LatticeECP2M and LatticeECP2MS Devices

Table A-2. Performance and Resource Utilization[1]

Video Format	Frame Width	Frame Height	Deinterlacer Algorithm	Pixel Width	Memory Bus Width	Frame Rate Conversion	Registers	LUT4s	EBRs	f_{MAX} (iclk)	f_{MAX} (mem_clk)	f_{MAX} (oclk)
YCbCr4:2:2	720	576	intra	8	32	Yes	2612	3144	4	253	291	256
YCbCr4:2:2	720	576	inter	8	32	Yes	3825	4702	7	296	237	251
YCbCr4:2:2	1920	1080	intra	8	32	Yes	2657	3223	6	287	275	263
YCbCr4:2:2	1920	1080	inter	8	32	Yes	3916	4774	11	280	233	231

1. Performance and utilization data are generated targeting a LFE2M50E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

Ordering Part Number

The Ordering Part Number (OPN) for Deinterlacer IP core targeting LatticeECP2M/S devices is DLACE-PM-U1.

LatticeECP3 Devices

Table A-3. Performance and Resource Utilization[1]

Video Format	Frame Width	Frame Height	Deinterlacer Algorithm	Pixel Width	Memory Bus Width	Frame Rate Conversion	Registers	LUT4s	EBRs	f_{MAX} (iclk)	f_{MAX} (mem_clk)	f_{MAX} (oclk)
YCbCr4:2:2	720	576	intra	8	32	Yes	2609	3127	4	281	257	249
YCbCr4:2:2	720	576	inter	8	32	Yes	3833	4690	7	288	245	251
YCbCr4:2:2	1920	1080	intra	8	32	Yes	2656	3167	6	252	252	244
YCbCr4:2:2	1920	1080	inter	8	32	Yes	3916	4734	11	243	255	243

1. Performance and utilization data are generated targeting a LFE3-70EA-8FN1156C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for Deinterlacer IP core targeting LatticeECP3 devices is DLACE-E3-U1.

LatticeXP2 Devices

Table A-4. Performance and Resource Utilization[1]

Video Format	Frame Width	Frame Height	Deinterlacer Algorithm	Pixel Width	Memory Bus Width	Frame Rate Conversion	Registers	LUT4s	EBRs	f_{MAX} (iclk)	f_{MAX} (mem_clk)	f_{MAX} (oclk)
YCbCr4:2:2	720	576	intra	8	32	Yes	2612	3144	4	232	247	214
YCbCr4:2:2	720	576	inter	8	32	Yes	3825	4702	7	228	212	225
YCbCr4:2:2	1920	1080	intra	8	32	Yes	2657	3223	6	239	233	217
YCbCr4:2:2	1920	1080	inter	8	32	Yes	3916	4774	11	252	226	221

1. Performance and utilization data are generated targeting a LFXP2-40E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.

Ordering Part Number

The Ordering Part Number (OPN) for Deinterlacer IP core targeting LatticeXP2 devices is DLACE-X2-U1.