Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

LATTICE SEMICONDUCTOR

LatticeCORE

Video Frame Buffer IP Core User Guide

March 2015
IPUG107_2.0

Table of Contents

Chapter 1. Introduction 4
- Quick Facts 4
- Features 5
- Release Information 5
- Device Support 5

Chapter 2. Functional Description 6
- Key Concepts 6
- Block Diagram 6
- Frame Rate Conversion 6
- Dynamic Parameter Updating 7
- Memory Bandwidth and Size 7
- Primary I/O 8
- Interface Descriptions 9
  - Video Input/Output 9
  - Memory interface 9
  - Parameter Register Read/Write Interface 11
- Timing Description 11
  - Video Input/Output Timing 11
  - Video Frame Timing 13
  - Memory Interface Timing 14
  - Dynamic Parameter Updating 14

Chapter 3. Parameter Settings 15
- Architecture 16
  - Frame Dimensions 16
- I/O Specification 17
- Implementation 18

Chapter 4. IP Core Generation 19
- Licensing the IP Core 19
- Getting Started 19
  - Configuring Video Frame Buffer Core in IPexpress 19
  - Configuring Video Frame Buffer Core in Clarity Designer 20
  - IPexpress-Created Files and Top Level Directory Structure 23
  - Clarity Designer-Created Files and IP Top Level Directory Structure 24
  - Instantiating the Core 25
  - Running Functional Simulation 25
  - Synthesizing and Implementing the Core in a Top-Level Design 26
- Hardware Evaluation 26
- Updating/Regenerating the IP Core 26
  - Regenerating an IP Core in IPexpress 26
  - Regenerating/Recreating the IP Core in Clarity Designer 27

Chapter 5. Support Resources 29
- Lattice Technical Support 29
  - E-mail Support 29
  - Local Support 29
  - Internet 29
- References 29
- .Revision History 29

Appendix A. Resource Utilization 30
- ECP5 Devices 30

Ordering Part Number 30
LatticeECP3 Devices 30
Ordering Part Number 30
LatticeXP2 Devices 31
Ordering Part Number 31

The Video Frame Buffer IP core buffers video data in external memory to be displayed on output devices such as computer monitors, projectors, etc. The Video Frame Buffer IP core supports image sizes up to 4K x 4K with YCbCr 4:2:2, 4:4:4/RGB video formats. It 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. Video Frame Buffer IP Core Quick Facts

		Video Frame Buffer IP Core Configuration		
Core Requirements	FPGA Families Supported	**LatticeECP3™**	**ECP5™**	**LatticeXP2™**
	Minimum Device Required	LFE3-17EA	LFE5-25F	LFXP2-5E
	Targeted Device	LFE3-35EA-8FN672C	LFE5UM-85F-8BG554CES	LFXP2-30E-7F672C
Resource Utilization	Configuration	**NTSC720x480 8-bit Serial YCbCr4:2:2**		
	Registers	794	789	796
	LUTs	940	899	953
	EBRs	2	2	2
Resource Utilization	Configuration	**XGA1024x768 8-bit Parallel RGB**		
	Registers	859	861	857
	LUTs	1103	1071	1116
	EBRs	2	2	2
Resource Utilization	Configuration	**HD1920x1080 8-bit Parallel YCbCr4:2:2**		
	Registers	869	860	867
	LUTs	1012	990	1028
	EBRs	2	2	2
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.3		
	Synthesis	Synopsys® Synplify™ Pro for Lattice I-2014.03L-SP1, Lattice Synthesis Engine[1]		
	Simulation	Aldec® Active-HDL™ 9.3 Lattice Edition II Mentor Graphics® ModelSim™ SE 6.3		

1. LatticeXP2 LSE flow requires Diamond 3.5 or later.

Features

- Supports single color, YCbCr 4:2:2, YcbCr 4:4:4/RGB video formats
- Supports input and output resolutions of 64 x 64 to 4K x 4K pixels
- Supports serial and parallel pixel processing
- Supports frame rate conversion
- Supports dynamic parameter update of frame size and Keep mode
- Supports configurable parameter bus width
- Supports configurable parameter bus clock
- Supports configurable memory bus width and base address
- Supports configurable memory burst length and burst count
- Configurable internal FIFO type and depth
- Supports 8, 10 or 12-bit color depth per plane

Release Information

- Video Frame Buffer IP core version 2.0
- Last updated March 2015

Device Support

- ECP5, LatticeECP3, LatticeXP2

Key Concepts

The Video Frame Buffer IP core receives input video data, stores it in the external memory and outputs it based on the timing controlled by the dout_enable signal. The core stores data into memory in a different width format as required by the application. It also provides synchronization of data across different clock domains as well as different format domains.

The core provides a simple parameter bus for dynamic frame size updating. It also implements a flexible memory interface that can be connected to Lattice memory controller IP cores.

The core also supports interlaced video stream by combining two fields into one frame outside.

Block Diagram

Figure 2-1. Video Frame Buffer IP Core Block Diagram

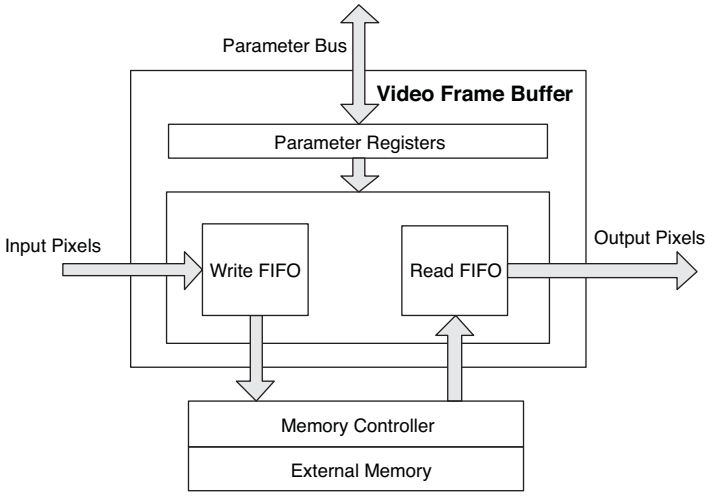


Figure 2-1 shows the block diagram of the Video Frame Buffer IP core. The core supports continuous data streams from/to external interfaces in different clock domains using asynchronous Write and Read FIFOs. Input pixels are packed and stored into the asynchronous double clock Write FIFO first. Then pixels are sent to an external memory controller to be written to the memory. After an entire video frame has been stored in the external memory, frame reading starts. The pixels read from external memory are stored in the asynchronous Read FIFO and transferred to output interface clock domain. After unpacking, pixels are output from the Video Frame Buffer IP core.

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

Frame Rate Conversion

The Video Frame Buffer IP core provides a simple frame rate conversion by repeating or dropping frames.

When frame rate conversion is enabled, the core's output data path runs at the output pixel clock rate. 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 video frame, the core will output the last 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 video stream. If there is no new input video frame, the core stops generating output data.

Dynamic Parameter Updating

The Video Frame Buffer IP core provides a parameter bus port for internal parameter update at run time. The parameters are double-buffered to avoid adverse effect to the core 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.

Table 2-1. Parameter Register Maps

Address	Register	Size	W/R	Default	Description
0x00	FRMWIDTH	32	W/R	719	**Frame width register**. The FRMWIDTH value must be (frame width – 1). The minimum value is 63, and the maximum value is the max frame width specified on the IP GUI minus 1. The default value is the maximum value.
0x04	FRMHEIGHT	32	W/R	479	**Frame height register**. The FRMHEIGHT must be (frame height – 1). The minimum value is 63, and the maximum value is the max frame height specified on the IP GUI minus 1. The default value is the maximum value.
0x08	KEEP	1	W/R	0	**Keep mode register**. The value can be 0 or 1. When this bit is 1, the core is in an output locked mode. The same output frame is sent out repeatedly if frame rate conversion is active; otherwise there is no output frame.
0x0C	UPDATE	1	W/R	0	**Update parameter enable register**. The value can be 0 or 1. Setting this bit to 1 triggers the update of the above parameters. The core resets it to 0 after the parameters have been updated.

All the parameter registers can be written to only when the UPDATE register bit is 0. The parameter KEEP will take effect at the next output frame. When the UPDATE bit is set to 1, the parameters FRMWIDTH and FRMHEIGHT 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 Video Frame Buffer IP core stores and retrieves pixels to/from external memory using memory burst write and read commands. When DDR2 memory 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 through multiple burst write/read transactions internally.

When frame rate conversion is inactive the video frame buffer core needs a 2-frame memory storage space; when frame rate conversion is active, a 3-frame storage space is required. The total external memory size the core requires can be viewed on the video frame buffer IP GUI.

The required memory bandwidth is input pixel data rate plus output pixel data rate.

For example, for parallel 8-bit YCbCr 4:2:2 pixels, if the input pixel sample clock is 74.25 MHz and output pixel sample clock is 148.5 MHz, the required bandwidth is 2*8*(74.25+148.5) = 3564 bit*MHz. If the memory data width is 32, the required memory clock will be (3564/32) = 111.375 MHz.

Primary I/O

Figure 2-2. Video Frame Buffer IP Core I/O

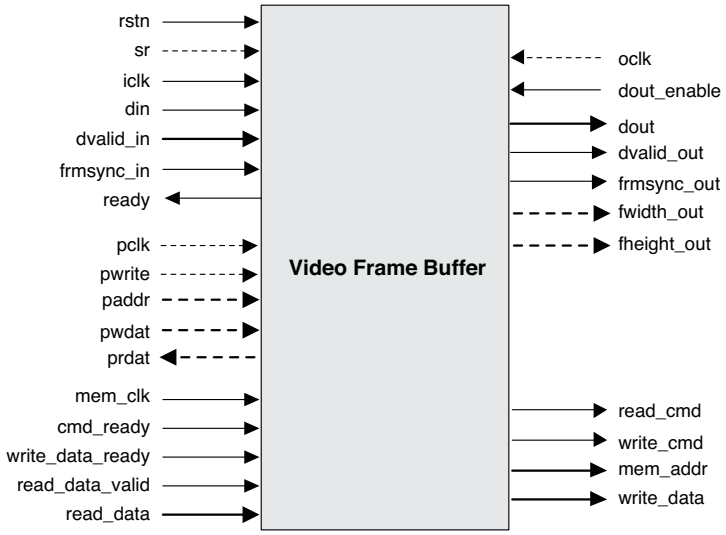


Table 2-2. Primary I/O

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 - 48	I	Input video data in frame format.
ready	1	O	When high, indicates the Video Frame Buffer 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 output data, active high.
dout	8 - 48	O	Output video pixel in 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's 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's ready to accept new write data, active high.

Table 2-2. Primary I/O (Continued)

Port	Size	I/O	Description
Optional I/Os			
fwidth_out	16	O	Current output frame width, only for dynamic mode.
fheight_out	16	O	Current output frame height, only for dynamic mode.
frm_drop	1	O	Input frame drop flag
frm_repeat	1	O	Output frame repeat flag
frm_resync	1	O	Input frame re-sync flag
pclk	1	I	Parameter bus clock, configurable.
pwrite	1	I	Parameter bus write enable, active-high.
paddr	5	I	Parameter bus address.
pwdat	8/16/32	I	Parameter bus write data.
prdat	8/16/32	O	Parameter bus read data.

Interface Descriptions

Video Input/Output

The Video Frame Buffer 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 video frame buffer, it asserts the core's dvalid_in port and at the same time placing 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 active pixel is placed on the din bus.

Similarly, dvalid_out is active when valid output pixels are available on dout, and frmsync_out marks the first pixel in an output frame.

The output ports fwidth_out and fheight_out indicate the current output frame's width and height respectively, which are valid only when frmsync_out is asserted.The input signal dout_enable enables the core to generate output pixels. When dout_enable is low (inactive), the core stops generating output pixels.

The IP core also provides several optional flags that indicate the frame status. The frm_resync output is driven to 1 when a new frmsync_in is asserted indicating a new frame is started before the current input frame is completed. The frm_drop and frm_repeat outputs indicate that the current frame is being dropped or repeated for frame rate conversion, respectively.

Memory interface

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

When connecting to a DDR memory controller that has 1:1 clock ratio between the DDR and controller local buses (such as DDR2 memory controller for LatticeECP3), the burst length and burst count of video frame buffer should have the same values as those of the DDR memory controller. When connecting to a DDR memory controller that has 2:1 clock ratio between the DDR and controller local buses (such as DDR3 memory controller for LatticeECP3 or ECP5), the burst length of video frame buffer should have the same value with DDR3 memory controller, while the burst count of video frame buffer should be half as that of the DDR memory controller.

The video frame buffer 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);
```

In order to get the best throughput (about 5% increase compared to normal connection), the users should fine tune the combination of row/bank/column/ address to get the best throughput.

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 arbitrates between memory write and read operations, ensuring only a *write_cmd* or a *read_cmd* is asserted at any given 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 Video Frame Buffer 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 has not 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 parameter bus data width equals to 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 Description

Video Input/Output Timing

The Video Frame Buffer 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 processing and combined on the din and dout ports for parallel processing. Figures 2-3 and 2-4 show the timing of RGB serial processing and parallel processing.

Figure 2-3. RGB Serial Processing (8-bit pixel)

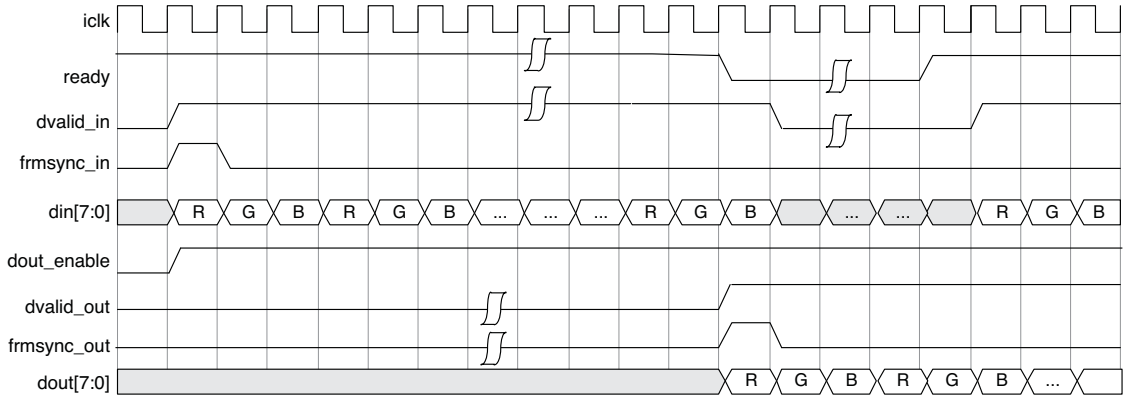
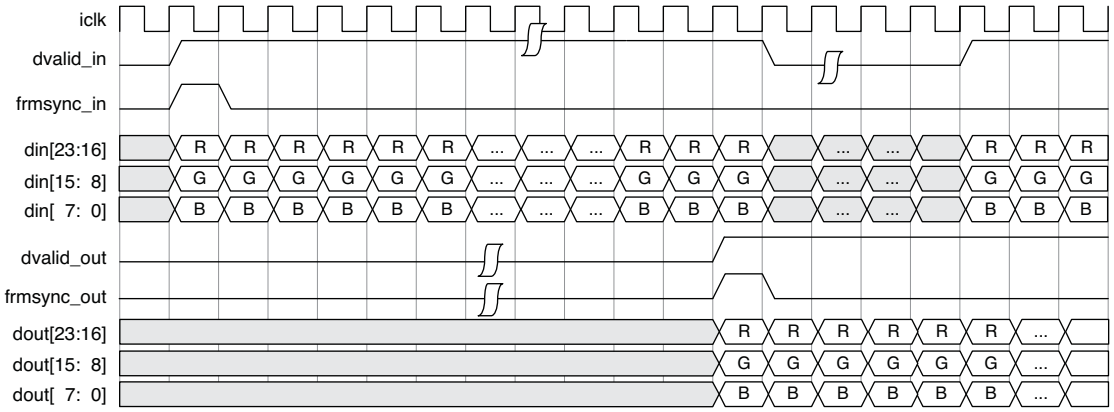


Figure 2-4. RGB Parallel Processing (8-bit pixel)



For YCbCr 4:2:2 video serial processing, the input and output sequence should be Cb, Y, Cr, Y, …. For parallel processing, 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 2-5 and 2-6 show the timing of YCbCr 4:2:2 serial processing and parallel processing.

Figure 2-5. YCbCr 4:2:2 Serial Processing (8-bit pixel)

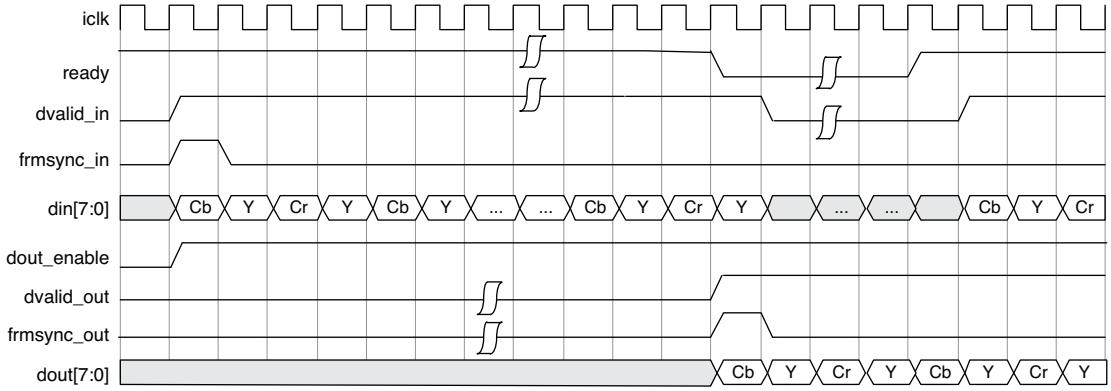


Figure 2-6. YCbCr 4:2:2 Parallel Processing (8-bit pixel)

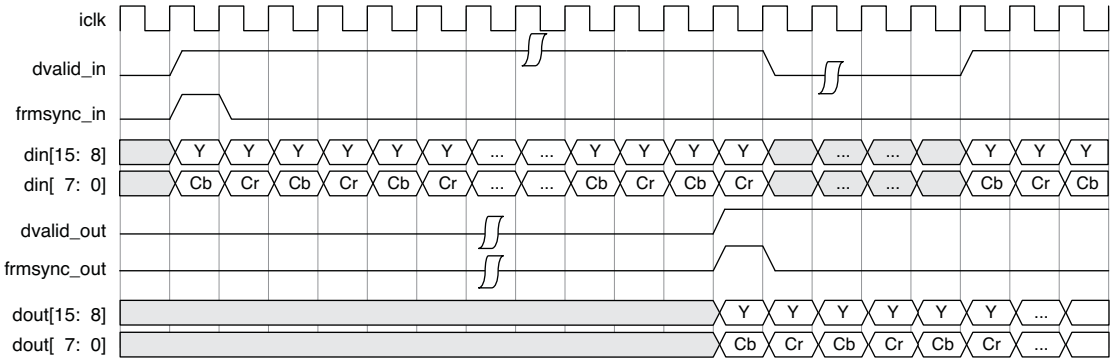
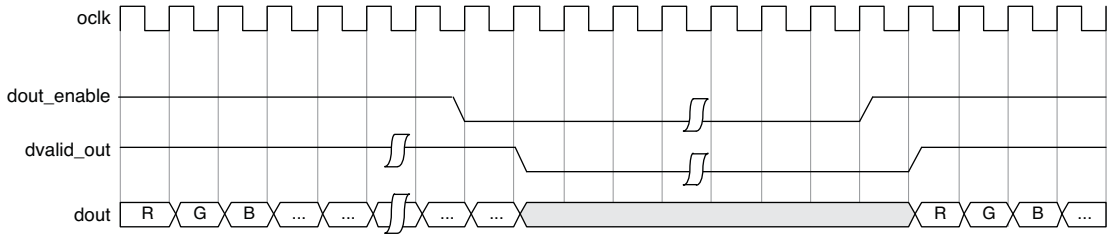


Figure 2-7 shows the dout_enable control timing. When dout_enable is de-asserted, the core will stop outputting data. Similarly, when dout_enable is asserted, the core will begin outputting data. The assertion and de-assertion of dout_enable can be used to generate a horizontal blank and a vertical blank depending on the output video format.

Figure 2-7. dout_enable Control Timing



Video Frame Timing

Figure 2-8. Output Frame Rate is the Same as the Input Frame Rate

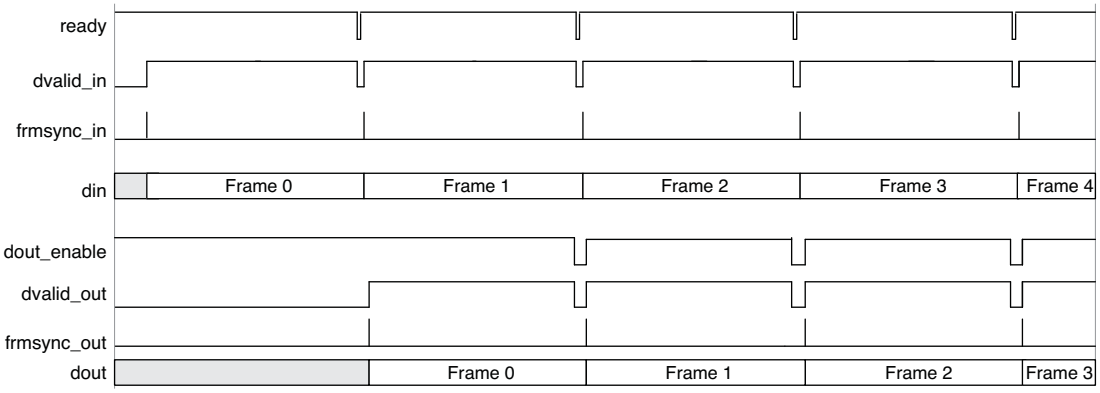


Figure 2-8 shows the frame timing when frame rate conversion is disabled. After accepting one input frame, the core starts generating frames. The output frame is driven by the input frame. When a new input frame arrives before the current output frame is finished, the current output frame is dropped and a new output frame is started. After the last input frame, the video frame buffer stops generating output frames.

Figure 2-9. Output Frame Rate is Twice the Input Frame Rate

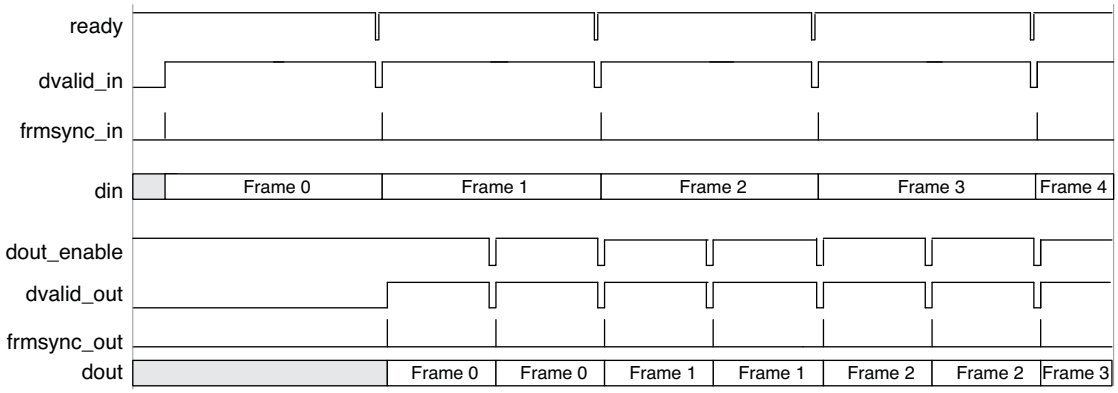


Figure 2-9 shows the frame timing when the output frame rate is twice the input frame rate. Output frames run on a separate output pixel clock. The output frame rate is determined by the output pixel clock and dout_enable signal. The core generates output frames based on the oldest pending input frame(s). 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.

Memory Interface Timing

Figure 2-10 shows the timing of the memory write operation. When the internal write FIFO is half full, it triggers memory write operation cycles. The memory write operation will continue until the write FIFO is empty.

Figure 2-10. Timing Diagram for Memory Write

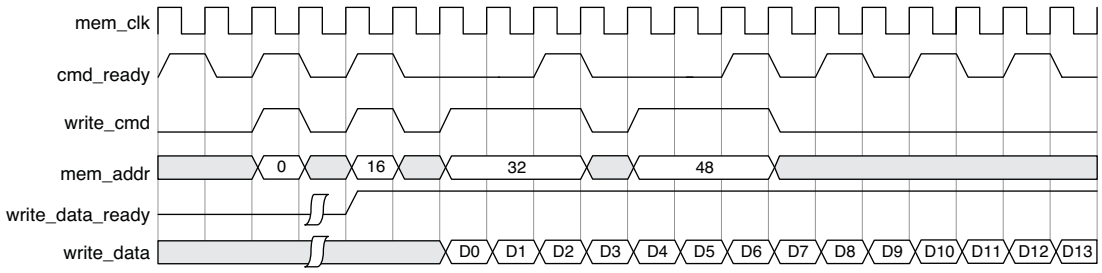
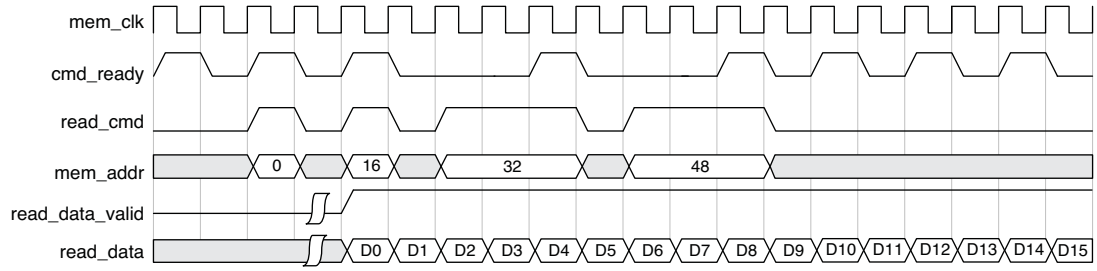


Figure 2-11 shows the timing of memory read operation. When the internal read FIFO is less than half full, and there is no ongoing write operation, the memory read burst operation is started. The read burst operation stops once the read burst makes the read FIFO half full.

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

Figure 2-11. Timing Diagram for Memory Read

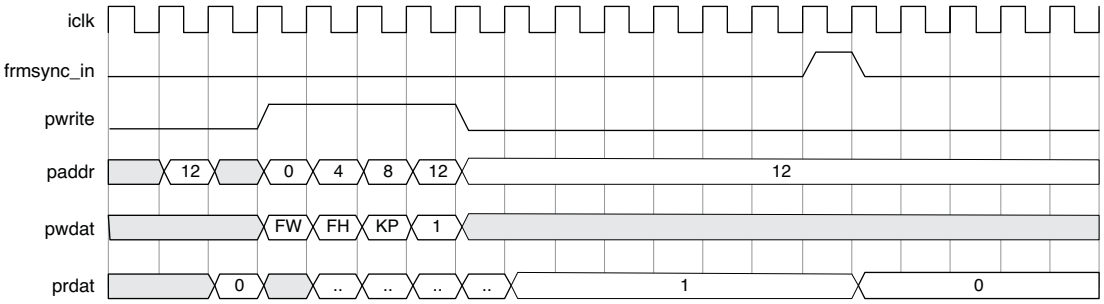


Dynamic Parameter Updating

Figure 2-12 shows the timing of dynamic parameter updating. FW and FH are the video frame width and height. KP is the KEEP register value.

The parameter bus can be configured to operate on a separate clock. By default, it operates at the 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 2-12. Timing Diagram for Dynamic Parameter Updating (parameter bus width = 32)



Parameter Settings

This section describes how to generate the Lattice Video Frame Buffer IP core using the Diamond IPexpress™ tool. Refer to the IP Core Generation section for a description of how to generate the IP.

The Video Frame Buffer 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 Video Frame Buffer IP core.

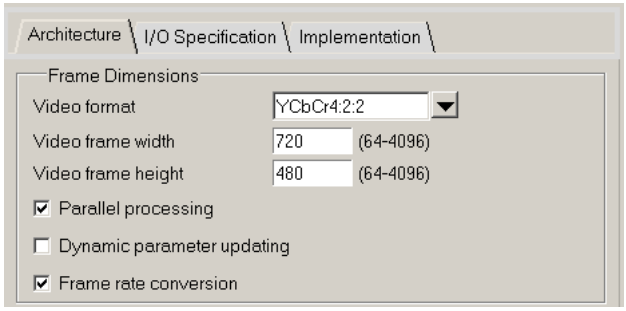
Table 3-1. Video Frame Buffer IP Core Parameters

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 (Even numbers only)	720
(Max) Video frame height	64-4096 (Even numbers only)	480
Parallel processing	Yes, No	Yes
Dynamic parameter updating	Yes, No	No
Frame rate conversion	Yes, No	Yes
I/O Specifications		
Input pixel width	8, 10, 12, 16	8
Memory bus width	8, 16, 32, 64, 128	32
Memory base address	0x00000000-0xFFFFFFFF	0x00000000
Memory address width	Read only	21
Memory size needed	Read only	2073600 bytes
Parameter bus width	8, 16, 32	32
Separate parameter bus clock	Yes, No	No
Input frame re-sync flag	Yes, No	No
Frame drop and repeat flag	Yes, No	No
Synchronous reset	Yes, No	No
Output frame size ports	Yes, No	No
Memory Type		
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
Fanout limit	1-200	100
Resource sharing	Yes, No	Yes
Pepelining 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 parameter updating.

Video format defines the format of 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 buffering. **Video frame width** and **Video frame height** parameters must be multiple of 2 (even numbers).

The **Parallel processing** checkbox determines whether the core processes video color planes in parallel.

The **Dynamic parameter updating** checkbox determines whether the core supports parameters updating at run-time. Refer to the Dynamic Parameter Updating section for more information.

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

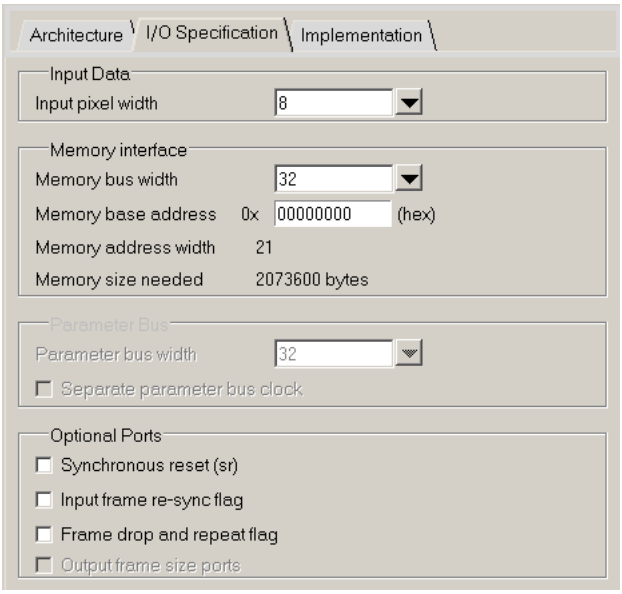
When dynamic parameter updating is enabled, the **Max video frame width** and **Max video frame height** specify the largest frame size the core needs to support.

The range for the frame dimensions is 64 to 4096.

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, 12 and 16. Default value is 8.

Memory bus width sets the data width of 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 size 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 parameter 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 parameter updating is selected.

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

The **Input frame re-sync flag** determines whether the core provides an output flag that indicates re-synchronization of the input frame.

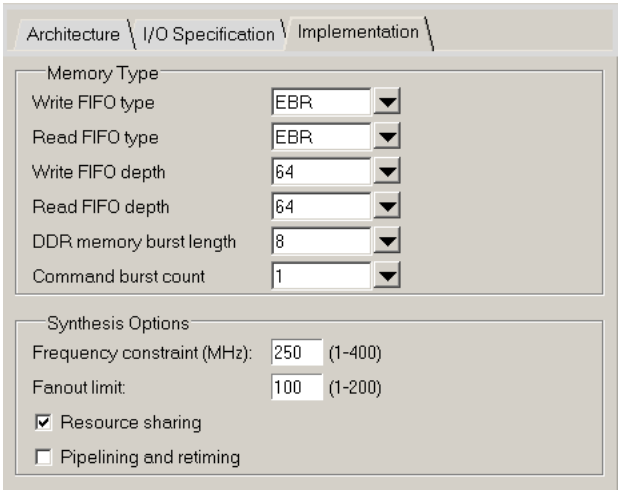
The **Frame drop and repeat flag** provides the frm_drop and frm_repeat flag signals to the core when checked.

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

Implementation

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

Figure 3-3. Implementation Tab



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 and **Read FIFO depth** selects the depth of internal read FIFO. The FIFO depth can be 32, 64, 128, 256 and 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 for the synthesis tool in MHz. This option applies to all the clocks 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

This section provides information on how to generate the Lattice Video Frame Buffer IP core using the Diamond 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 Lattice Video Frame Buffer IP core in a complete, top-level design.

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

Getting Started

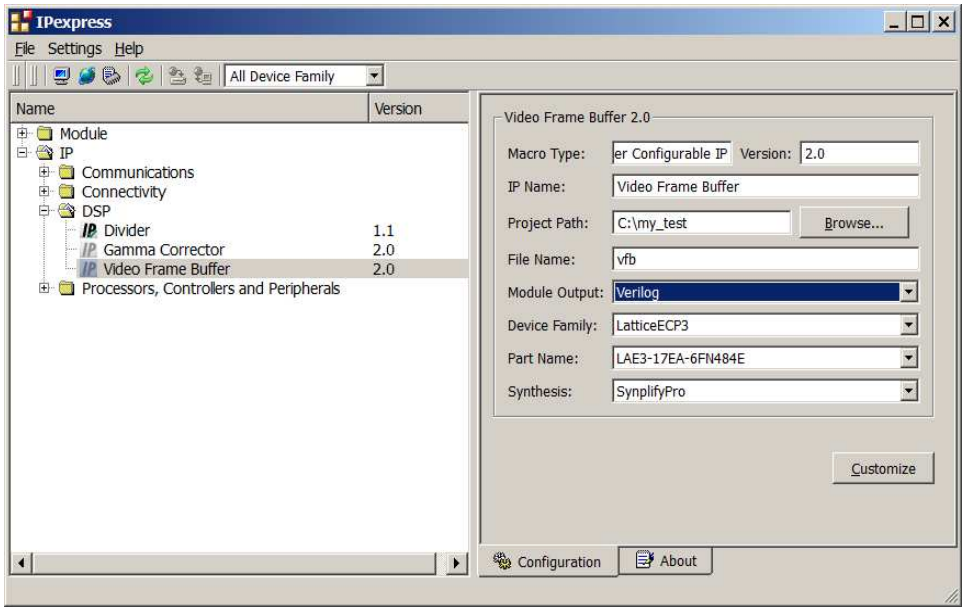
The Video Frame Buffer IP core is available for download and installation from the Lattice IP Server using the IPexpress or Clarity Designer tool. After the IP core has been installed, the IP core will be available in the IPexpress or Clarity Designer GUI dialog box.

Configuring Video Frame Buffer Core in IPexpress

- **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.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted. Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

The Video Frame Buffer configuration GUI is accessed by clicking the **Customize** button, and provides an interface for setting the desired parameters and invoking the IP core generator.

Figure 4-1. Video Frame Buffer Configuration Interface



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

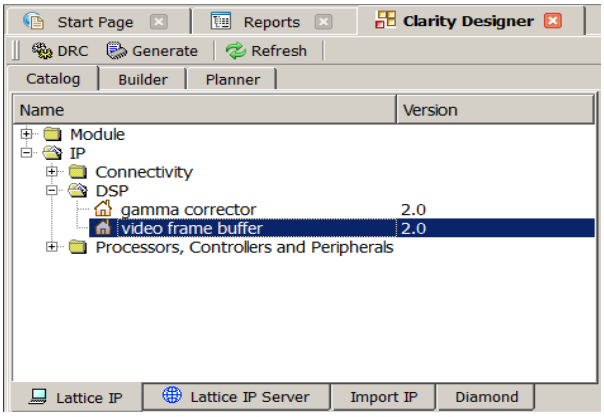
Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the Video Frame Buffer IP core Configuration GUI, as shown in Figure 4-1. From this dialog box, the user can select the IP parameter options specific to their application. Refer to Parameter Settings section for more information on Video Frame Buffer IP core parameter settings.

Configuring Video Frame Buffer Core in Clarity Designer

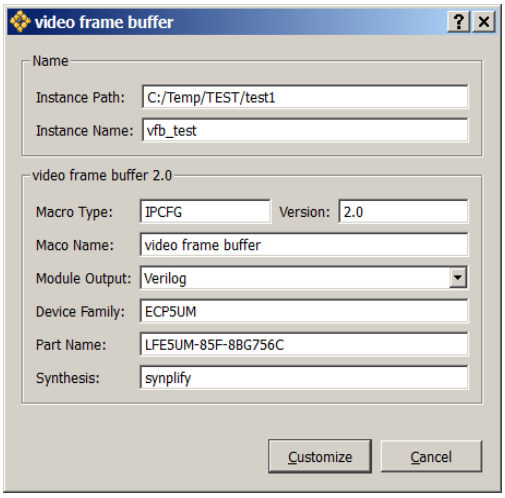
The Video Frame Buffer IP core is found under IP > DSP in the Clarity Designer IP Catalog Window as shown in Figure 4-2.

Figure 4-2. Clarity Designer Catalog Window



To configure the IP core, double click the **Video Frame Buffer IP** shown in the Catalog tab of the interface. This step opens the IP Project Dialog box as shown in Figure 4-3.

Figure 4-3. Clarity Designer IP Project Dialog Box



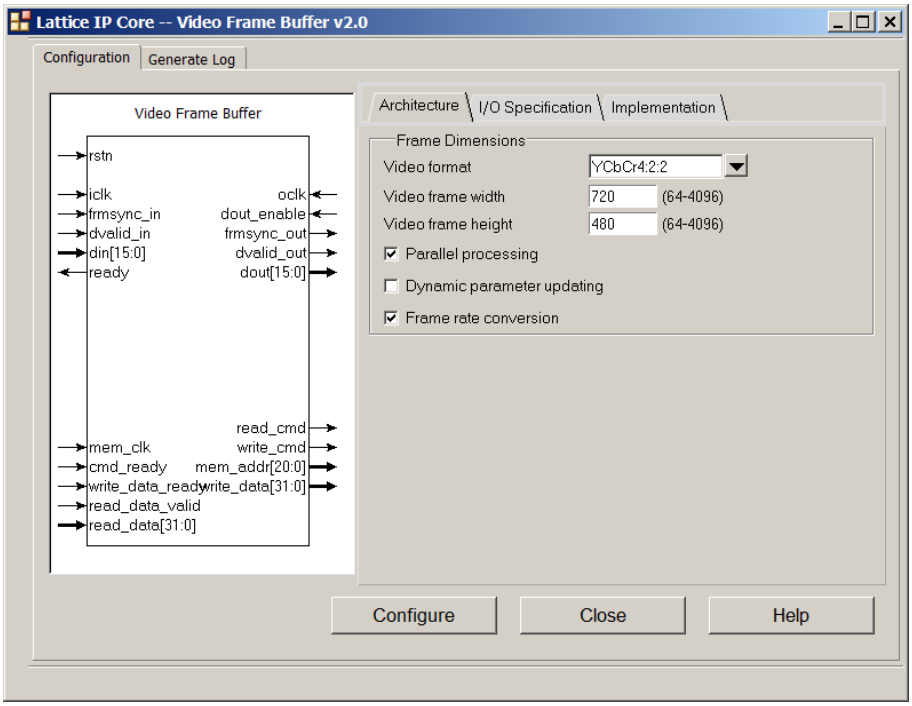
Enter the following information as described below.

- **Instance Path** – Path to the directory where the generated IP files will be loaded.
- **Instance Name** – "username" designation given to the generated IP core and corresponding folders and files. (Caution: Instance Name cannot be "frame_buffer_core," as this name has been used in the internal design of the core.)
- **Module Output** – Verilog or VHDL.
- **Device Family** – Shows the selected device family.
- **Part Name** – Shows the selected part within the selected device family.

To create a custom configuration:

1. Click the **Customize** button in the IP Project dialog box to display the Video Frame Buffer IP core Configuration interface, as shown in Figure 4-4.
2. Select the IP parameter options specific to their application. Refer to Parameter Settings section for more information on the Video Frame Buffer parameter settings.
3. Click **Configure** then **Close** to create the IP core with the selected configuration.

Figure 4-4. Configuration GUI



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 4-5. This example shows the directory structure generated with the Video Frame Buffer for a LatticeECP3 device.

Figure 4-5. Video Frame Buffer IP Core Directory Structure

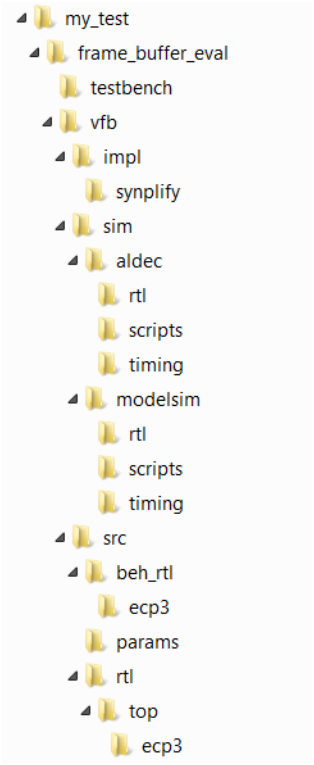


Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List (IPexpress)

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. 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 Video Frame Buffer IP core.
<username>_beh.v	This file provides the front-end simulation library for Video Frame Buffer IP core.

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

Table 4-2. Additional Files (IPexpress)

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

Clarity Designer-Created Files and IP Top Level Directory Structure

When the user clicks the Configure button then the Close button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified “Instance Path” directory. An example of the directory structure of the generated files is shown in Figure 4-6.

Figure 4-6. IP Top Level Directory Structure

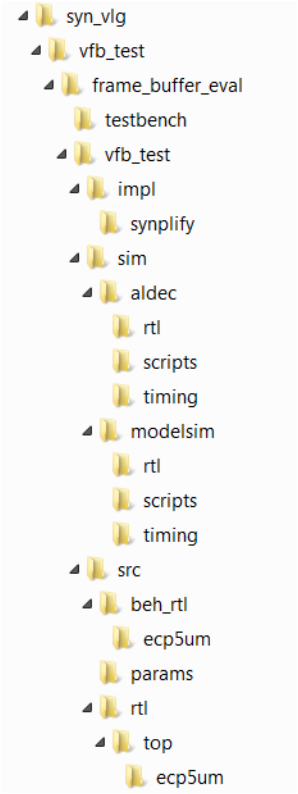


Table 4-3 provides a list of key files and directories created by the Clarity Designer tool and how they are used. The Clarity Designer tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user’s module name specified in the Clarity Designer tool.

Table 4-3. File List (Clarity Designer)

File	Description
<username>.lpc	This file contains the Clarity Designer tool options used to recreate or modify the core in the Clarity Designer tool.
<username>.ngo	This file provides the synthesized IP core.
<username>.ngd	This file is used by the Clarity Designer tool to generate the 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 Video Frame Buffer IP core.
<username>_beh.v	This file provides the front-end simulation library for Video Frame Buffer IP core.

Table 4-4. Additional Files (Clarity Designer)

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

Instantiating the Core

The generated Video Frame Buffer 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>\frame_buffer_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 Video Frame Buffer 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 Video Frame Buffer 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>\frame_buffer_eval\<username>\sim\modelsim\scripts. The simulation script supporting Active-HDL evaluation simulation is provided in \<project_dir>\frame_buffer_eval\<username>\sim\aldec\scripts. Both Modelsim and Active-HDL simulation is supported via test bench files provided in \<project_dir>\frame_buffer_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>\frame_buffer_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>\frame_buffer_eval\<username>\sim\modelsim\scripts**.
3. Under the **Tools** tab, select **Execute Macro** and execute **..\scripts\<username>_rtl_se.do**.