Altera Video System

Building / implementing video systems
Agenda

- Altera Video Design Framework
- Exercise #0 : VIP components & Introduction
- Exercise #1 : Avalon Video Streaming Interface
- Verification
- Exercise #2 : Run-time configuration
- Exercise #3 : Switch & clock/frame locking
- Exercise #4 : Frame buffer, FRC and Calculating Memory Bandwidth
- Exercise #5 : Deinterlacing, Scaling
- Exercise #6 : Mixer, OSD, Ctrl Sync, and CRS IP
- Exercise #7 : VIP reference design & Demo
- Summary
Altera Video Framework

- **Building block IP** cores to speed up development
- Open, low-overhead **interface standard** to mix-and-match custom or off-the-shelf IP
- **Format conversion reference designs** that showcase silicon capability, provide a starting point for design
- **System Level Tools** and design methodology
- Variety of **development kits** for rapid design prototyping
Video Image Processing (VIP) Suite

Several **hardware verified** video processing reference designs using these functions

Available Now!
Video Development Kits

- Cyclone® IV FPGA
- Stratix® IV FPGA
- Arria® V FPGA
- Arria II GX FPGA
- Cyclone III FPGA
- OmniTek
- VEEK
- NEEK
Components in system use different interfaces to communicate (some standard, some non-standard)

Typical system requires significant engineering work to design custom interface logic

Integrating design blocks and intellectual property (IP) is tedious and error-prone
Automatic Interconnect Generation

- Avoids error-prone integration
- Saves development time with automatic logic & HDL generation
- Enables you to focus on value-add blocks

*Qsys improves productivity by automatically generating the system interconnect logic*
Example Design in Qsys

M: Master
S: Slave
Qsys-Supported Standard Interfaces

- **Avalon-MM (memory mapped)**
  - Little Endian
  - Control plane
  - Master interface makes read and write requests to slave interface

- **Avalon-ST (streaming)**
  - Big Endian
  - Data plane
  - Source interface sends data to sink interface (point-to-point)

- **ARM AMBA™ AXI™ 3.0**
  - Some initial support for AXI 4.0 interface
Open, Low-overhead Interface Standard: Avalon Streaming (ST) Video

Open interface protocol for streaming video data paths and memory-mapped control paths
Standard Interface Example

Any master interface can communicate with any slave interface
Qsys Design Flow

- Create Quartus® II project
- Create Qsys project from Quartus II
- Add IP blocks to the system
- Create and add custom components
- Interconnnnect components
- Compile the Qsys system
- Instantiate the Qsys system in your Quartus II project
- Compile the Quartus II design
- Write appropriate software to control system
Use Qsys to Build VIP Systems

- Start with a New Quartus II Project
Start a New System in Qsys
Add the VIP Components
Add the VIP Components
Add Custom Components

- Map component signals to Qsys Interface types and Signal Names on the Signals tab
Connect the Components

<table>
<thead>
<tr>
<th>Use</th>
<th>Conn...</th>
<th>Name</th>
<th>Description</th>
<th>Export</th>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>alt_vip_crs_0</td>
<td>Chroma Resampler</td>
<td>unconnected</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>clock</td>
<td>Clock Input</td>
<td>Click to export</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset</td>
<td>Reset Input</td>
<td>[clock]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>din</td>
<td>Avalon Streaming Sink</td>
<td>Click to export</td>
<td>[clock]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dout Avalon Streaming Source</td>
<td>[clock]</td>
<td></td>
</tr>
</tbody>
</table>

Click the open dot to make connection

© 2012 Altera Corporation—Confidential
Connect the Components
Compile & Integrate Design

- Quartus II
- Qsys
- Development Kits
- Eclipse
Summary

- Altera® video design framework enables rapid development
  - Mix & match existing IP - Leverage Altera’s open interface standard
  - Automatically integrate embedded processors & arbitration logic
  - Leverage building block IP provided by Altera
  - Use existing reference designs as starting points

- And rapid prototyping
  - Implement design using the appropriate development boards
  - Test the design with actual video signals
Exercise #0

VIP components & Introduction
VIP Components Required for Lab 1

- Test Pattern Generator
- Clocked Video Output
- Video Trace Monitor
  - Trace System
  - Video Monitor
Lab 1: Video output and debug

- Build basic VIP system in Qsys and debug
VIP Function Details
Generate a color bar or solid color test pattern
VIP Function Details
Clock Video Output

- The Clocked Video **Output** MegaCore function converts from Avalon-ST Video to clocked video formats (such as BT656 and DVI).

- It formats Avalon-ST Video into clocked video by inserting horizontal and vertical blanking and generating horizontal and vertical sync information.

- No conversion is done to the active picture data.
  - The color plane information remains the same as in the Avalon-ST Video format.
VIP Function Details
Video Trace Monitor

Monitor Capture and Control ports connected to host via Trace System and debug pipe
- Control port enables configuration from System Console
- Captured data output via capture port

FPGA

Video IP
Av-ST Video Monitor IP
Video IP
Av-ST Video Monitor IP
Video IP

Capture
Control
Control
Capture

Trace System IP

Debug pipe (JTAG or USB), connecting to host

System Console visualisation

© 2012 Altera Corporation—Confidential
The SystemConsole Framework

- Extensible framework for moving data between the FPGA and the host
  - Trace feature plugs in to the framework

- Debug pipe agnostic
  - Now: JTAG, USB 2.0
  - Next: Ethernet, PCIe
QSys Debug System: Insert Monitors

- Insert Avalon-ST Video Monitor(s) in the video datapath
  - Connect Monitor Av-ST Video sink (din) to Video IP source
  - Connect Monitor Av-ST Video source (dout) to Video IP sink
  - Connect capture and control ports to Trace System

- The Avalon-ST Video Monitor is non-intrusive
  - Data passes through the monitor without manipulation or stall

Specify data format to match video source and sink

Specify bit width of capture port to output data to host.
- Match width with trace system capture port width

Capture video pixel data in addition to video format, resolution and statistics
- For best results use high bandwidth debug connection (eg. USBII)
  [Beta feature]
Trace Table View

- Live trace of selectable video control and data packets
- Multi-level view
- Av-ST Video protocol
  - Video control or data packet information
- Data flow statistics
  - Av-ST standard
- Raw packet data
  - Up to first 6 beats of data in packet
- Global timestamp with interleaved captured packets
Trace Table View – Pixel Data [Beta feature]

- **Av-ST Video protocol**
  - Video data packet information

- **Data flow statistics**
  - Av-ST standard

- **Raw packet data**
  - Up to first 6 beats of data in packet

- Multi-level view
Exercise #1

Avalon Video Streaming Interface
Interface Types

- Avalon Streaming (ST) Video interface
- Avalon Memory-Mapped (MM) interface
Avalon Streaming (ST) Video

Avalon-ST Video protocol is a packet-oriented way to send video and control data
- First packet contains a **frame** of uncompressed video data
- Second packet contains control information that applies to the **subsequent** video packet/frame

Structure of packet
- Packets are split into symbols (data)
- # symbols sent in parallel and bit width of symbols is fixed
- Type is defined by 4 bits

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Video Data Packet</td>
</tr>
<tr>
<td>1-8</td>
<td>User Packet Types</td>
</tr>
<tr>
<td>9-12</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>Ancillary Data Packet</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>Control Data Packet</td>
</tr>
</tbody>
</table>
Video Data Packet

- Sent per pixel in raster scan order

![Diagram of raster scan order]

- Static Parameters:
  - Bit per pixel per color plane
    - Ex. RGB data of 8bits/sample (24 bits/pixel) uses 8bits per pixel per color plane
    - Symbol Width (min. 4)
  - Color Pattern
    - Organization of color plane samples within video data packet
    - Cannot change within a packet
Color Pattern

- Each color plane sample maps to Avalon-ST symbol
  - Symbol Transmission Order

  - Horizontally Subsampled YCbCr
  - Vertical Subsampled YCbCr
## Recommended Color Patterns

<table>
<thead>
<tr>
<th>Color Space</th>
<th>RGB</th>
<th>4:4:4 YCbCr</th>
<th>4:2:2 YCbCr</th>
<th>4:2:0 YCbCr</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parallel</strong></td>
<td><img src="image1" alt="RGB Parallel" /></td>
<td><img src="image2" alt="4:4:4 YCbCr Parallel" /></td>
<td><img src="image3" alt="4:2:2 YCbCr Parallel" /></td>
<td><img src="image4" alt="4:2:0 YCbCr Parallel" /></td>
</tr>
<tr>
<td><strong>Sequence</strong></td>
<td><img src="image5" alt="RGB Sequence" /></td>
<td><img src="image6" alt="4:4:4 YCbCr Sequence" /></td>
<td><img src="image7" alt="4:2:2 YCbCr Sequence" /></td>
<td><img src="image8" alt="4:2:0 YCbCr Sequence" /></td>
</tr>
</tbody>
</table>
# Control Data Packet - Definitions

<table>
<thead>
<tr>
<th>Component</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>16</td>
<td>Pixels of the lines of a frame</td>
</tr>
<tr>
<td>Height</td>
<td>16</td>
<td>Number of lines in a frame or field</td>
</tr>
<tr>
<td>Interlacing</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 MSBs</th>
<th>Next video packet</th>
<th>2 LSBs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Progressive</td>
<td>00</td>
<td>Progressive frame last deinterlaced from:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>- Interlaced field 0 (f0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>- Interlaced field 1 (f1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>- Unknown</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Not interlaced</td>
</tr>
<tr>
<td>10</td>
<td>Interlaced field 0 (f0)</td>
<td>00</td>
<td>Frame should be constructed from:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>- f1 followed by f0 (Synchronize on f0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>- f0 followed by f1 (Synchronize on f1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>- Unknown</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Don’t care</td>
</tr>
<tr>
<td>11</td>
<td>Interlaced field 1 (f1)</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
## Control Data Packets - Examples

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td><strong>Width</strong></td>
</tr>
<tr>
<td>15</td>
<td>1920</td>
</tr>
<tr>
<td>15</td>
<td>640</td>
</tr>
<tr>
<td>15</td>
<td>640</td>
</tr>
<tr>
<td>15</td>
<td>640</td>
</tr>
<tr>
<td>15</td>
<td>640</td>
</tr>
<tr>
<td>15</td>
<td>1920</td>
</tr>
<tr>
<td>15</td>
<td>1920</td>
</tr>
<tr>
<td>15</td>
<td>1920</td>
</tr>
<tr>
<td>15</td>
<td>1920</td>
</tr>
</tbody>
</table>
Avalon-ST Video Verification IP suite – What is it?

- A SystemVerilog class library
- Released as part of Quartus 12.0
- Built on top of existing Avalon-ST BFMs
- Fully documented, open and extensible

Simplifies:
- Conformance checking against Avalon-ST Video protocol
- Testing against a DUT golden simulation model
- Running of video files into and out of the DUT
- Generation of corner cases

Supports:
- All Avalon-ST Video formats
Avalon Streaming Verification Test Environment

method calls to file reader object (1):
open_file()
read_file()
set_readiness_probability()

method calls to file writer object (1):
open_file()
wait_for_and_write
video_packet_to_file()
set_readiness_probability()
Three types of classes

1 Video “items”
   - Pixels
   - Video data packets
   - Avalon-ST Video Control packets
   - Avalon-ST User packets

2 Bus functional models
   - Sink
   - Source

3 File I/O
   - Read/write video data to file I/O
   - Assembles video items for use with BFMs or other
Class type 1: Avalon-ST Video packet items

- **Easy to use:**
  - function c_pixel pop_pixel();
  - function c_pixel push_pixel();
  - function bit compare (c_av_st_video_data r);
  - function void copy(c_av_st_video_data c);

- **Easy to constrain:**
  - Novice user leaves unconstrained
    - rand int video_length;
    - constraint c1 { video_length > 0; video_length < video_max_length;}
  - Intermediate user defines limits
    - function void set_max_length(int length);
    - function void set_garbage_probability (int i);
  - Expert user extends the class
    - class c_camera_data extends c_av_st_video_data;
    - constraint c2 { video_length inside {22,44,66,101}; }

- **Derived from common base class**
  - Enables a mailbox to handle any legal Avalon-ST video packet
Class type 2: Bus Functional Models - Source & sink

- Randomized Avalon-ST Video at pin level
  - Sink "readiness" and source "data valid" delay
    - function void set_readiness_probability(int percentage);
    - function void set_long_delay_duration_max_beats(int percentage);

- Randomized Avalon-ST Video at transaction level
  - Control packets before/after data packets
    - rand t_packet_control append_garbage = off;
  - Size of video lines/fields
    - rand bit [15:0] width;
    - rand bit [15:0] height;
    - rand bit [3:0] interlace;
File I/O test

- Tests a user’s DUT with their own video files
- Enables easy test of Avalon-ST video compliance

Flow:
- User’s .avi file converted to a .raw file and .spc file
- Test is run, simulating DUT with class library
- Output .raw file is converted back to .avi for viewing

```
>raw2avi.exe car.avi car.raw
>make –f Makefile.mk
>avi2raw.exe car.raw car.avi
```

- Note : Conversion utilities require Windows OS
Exercise #2

Run-time Configuration
Lab 2: Custom IP Verification and Insertion

- Part 1:
- Part 2:
Run-Time Control

Nios II Processor can change scaler output resolution at runtime, via the slave port.

© 2012 Altera Corporation—Confidential
# Control Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0       | Go       | - Bit zero of this register is the Go bit, all other bits are unused.  
- Setting Go bit to 0 will cause the core to pause at the end of the frame.  
- Setting Go bit to 1 will resume the core. |
| 1       | Status   | - Bit zero of this register is the Status bit, all other bits are unused.  
- The core sets the Status bit to 1 when it is running, and zero otherwise. |
| 2       | reg 0    | Function dependent (check [VIP user guide](#)) |
| ...     | ...      | ...         |
| n+1     | reg n-1  | Function dependent (check VIP user guide) |
Updating Control Registers

- Custom logic or a Nios II processor can be used to run-time configure the VIP cores
- Most VIP cores double-buffer their control data
- To update control registers:

**Without double-buffer:**

1. Set Go bit to 0
2. Poll the Status bit until it is 0
3. Update control data
4. Set Go bit to 1
5. Poll the Status bit until it is 1
6. Repeat 1-5 as needed

**With double-buffer:**

1. Set Go bit to 0
2. Update control data
3. Set Go bit to 1
Software API

- Altera provides a set of VIP APIs (C++) in the following reference designs:
  - Video Processing Reference Design (AN427)
  - Format Conversion Designs

- Pre-defined functions to access and configure control data of each VIP core
  - VIPCORE.hpp: Functions to stop, start, get status and enable interrupts
  - <VIP_function>.hpp: Functions to access the control data for that <VIP_function> core
Example – Run Time Configuration

- Detects input format
- Configures clipping and scaler resolution
- Enables or disables pass through mode for interlacer
- Configures CVO output resolution
Example – Main program

/* From main.cpp - main program*/
switch(output1) {
  case 1: // 720p60
    the_sdi_out_1.set_genlock(false);
    the_sdi_out_1.configure_mode(1, sdi_mode_definition_720p);
    genlock_1 = setup_genlock(the_cti_1, the_sdi_out_1, ... 
    interlacer_1.do_write(2, 1);
    the_scaler_1.set_output_size(1280, 720);
    break;
Run-time Control of the Video Chain

- Build video signal chains that can be updated **on-the-fly** without changing the HDL or bit-stream
  - Using the streaming interface protocol: Avalon-ST Video
  - Using an embedded processor
Run-time Control of the Video Chain

- Most video functions permit run-time control of some aspects of their behavior, using a common type of Avalon-MM slave interface.
- Each slave interface provides access to a set of control registers which must be set by an embedded processor/logic.
- Generally, a function can be updated on a frame boundary.
/**
 * void move_the_window(Mixer& the_mixer, unsigned int layer, unsigned int size_of_square_path)
 * Move layer in a square starting from (0,0)offset from background image
 *
 */

void move_the_window(Mixer& the_mixer, unsigned int layer)
{
    unsigned int x = 0;
    unsigned int y = 0;
    unsigned int size_of_square_path = 100;

    for(x = x ; x <= size_of_square_path ; x++){
        the_mixer.set_layer_position(layer,x,y);
        delay(10000);
    }

    for(y = y ; y <= size_of_square_path ; y++){
        the_mixer.set_layer_position(layer,x,y);
        delay(10000);
    }

    for(x = x ; x > 0 ; x--){
        the_mixer.set_layer_position(layer,x,y);
        delay(10000);
    }

    for(y = y ; y > 0 ; y--){
        the_mixer.set_layer_position(layer,x,y);
        delay(10000);
    }
}
Exercise #3

Switch & clock/frame locking
Lab 3: Switch IP, clock locking and frame locking
Additional Components Required for Lab 3

- Nios II Processor-Controlled Sub-System
- Clocked Video Input VIP Core
- (Custom) Terminator Block
VIP Function Details
Clock Video Input

- The Clocked Video Input MegaCore function converts from clocked video formats (such as BT656 and DVI) to Avalon-ST Video.

- It strips the incoming clocked video of horizontal and vertical blanking, leaving only active picture data.

- No conversion is done to the active picture data:
  - The color plane information remains the same as in the clocked video format.
Lab3: Video input (and debug) and NIOS run-time control
Switch

- Allows flexible routing of Avalon-ST Video
  - Can be used for muxing, de-muxing and crossbar switches
  - Dynamic switching on frame boundaries

- Configuration
  - Each output has its own one-hot register
  - Writing a 0 to the register disables the output
  - Writing a one-hot value connects that input to the output
    - $1 = \text{din}_0$, $2 = \text{din}_1$, $4 = \text{din}_2$, ....
  - The switch does not support connecting 2+ inputs to 1 output
    - Writing anything other than 0 or one-hot values will break the switch
  - The switch does not support connecting 1 input to 2+ outputs
    - Setting 2 or more outputs to the same one-hot value will break the switch
Bypassing An IP Core

- **Starting state**
  - **Video Input** is connected to **Out 1**
  - **In 1** is connected to **Video Output**

- **Using Switch 1** connect **Video Input** to **Out 2**
  - Poll **Output Switch** register until it is 0 (switch has occurred)

- **Using Switch 2** connect **In 2** to **Video Output**
Avoiding Deadlock

- Changing Switch 2, too early could result in deadlock
  - Solution is to poll until Switch 1 change is complete
Genlock

“Sync generator locking” is the process of aligning video outputs to the timing of a reference input

Typically this involves two processes:

- Clock locking
  - Aligning the phase and frequency of the output video clock to the reference clock and maintaining that alignment as the reference clock drifts
  - For clocks of a different frequency the alignment will take place at some common division of the two clock periods

- Frame locking
  - Aligning the start of frame of the output video to the reference start of frame
Clock Locking – RX Clock

- Poor jitter performance
  - Quartus may not allow vid_clk to be connected to a transceiver for this reason
  - May be OK for sending to an external transmitter, if it can clean up the clock
Clock Locking – VCXO

Detracts phase difference between reference and output clocks

Typically H pulse extraction from horizontal sync

74.25 MHz

148.5 MHz
Frame Locking

- Input and output frames must have the same **frame rate**
  - **Frame rate** is the number of frames per second
  - If clocks are not locked then input and output frame rates will be subtly different

- Alignment of **start of frame** allows clean frame switching
  - Switching between two frames from different video streams without any glitches
Frame Locking - Example

- **NTSC**
  - Interlaced @ 29.97 fps
  - Deinterlace to progressive 480p @ 59.94 fps
  - 13.5 MHz pixel clock becomes 27MHz after deinterlacing
  - 29.7 fps becomes 59.94 fps after deinterlacing

- **720p**
  - Progressive @ 59.94 fps
  - 74.176 MHz pixel clock

<table>
<thead>
<tr>
<th>Format</th>
<th>Total Width (inc blanking)</th>
<th>Total Height (inc blanking)</th>
<th>Pixel Clock</th>
<th>Period (ns)</th>
<th>Frame Period (ms)</th>
<th>Frames per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTSC</td>
<td>858</td>
<td>525</td>
<td>13.5</td>
<td>74.07407407</td>
<td>33.36666667</td>
<td>29.97002997</td>
</tr>
<tr>
<td>480p</td>
<td>858</td>
<td>525</td>
<td>27</td>
<td>37.03703704</td>
<td>16.68333333</td>
<td>59.94005994</td>
</tr>
<tr>
<td>720p</td>
<td>1650</td>
<td>750</td>
<td>74.1758</td>
<td>13.48148588</td>
<td>16.68333877</td>
<td>59.94004040</td>
</tr>
<tr>
<td>640x480</td>
<td>800</td>
<td>494</td>
<td>23.5</td>
<td>42.55319149</td>
<td>16.81702128</td>
<td>59.46356275</td>
</tr>
<tr>
<td>800x600</td>
<td>960</td>
<td>618</td>
<td>35</td>
<td>28.57142857</td>
<td>16.95085714</td>
<td>58.99406688</td>
</tr>
</tbody>
</table>
Clocked Video Output - Locking

- **Locking procedure**
  - Compares sof signal to its internal sof
    - **post-sof gap** and **pre-sof gap** must be constant over 2 frames
    - If the sof signal comes first then that many samples and lines are removed
      - Internal sof jumps forwards
    - If the sof signal comes last then that many samples and lines are added
      - Internal sof jumps backwards

![Diagram of sof and internal sof signals with post-sof gap and pre-sof gap](image-url)
Exercise #4

Frame buffer, FRC and Calculating Memory Bandwidth
Lab 4: Ext Memory, Frame Buffer IP and Frame Rate Conversion

Diagram:
- SDI Input 1
- Clocked Video Input
- Frame Buffer
- DDR3
- NIOS II
- Test Pattern Generator 1
- Switch
- Clocked Video Output
- SDI Output 1

Annotation:
- Doing frame rate conversion
Frame Buffer

Maximum image width: 1,024 Pixels
Maximum image height: 768 Pixels
Bits per pixel per color plane: 8 Bits
Number of color planes in sequence: 1 Planes
Number of color planes in parallel: 3 Planes

Number of packets buffered per frame: 0 Packets
Maximum packet length: 10 Symbols

Avalon Memory-Mapped Interfaces

Base address of frame buffers: 0x000000

3 frame buffers are required, a total of 9216 kilobytes.
Frame Buffer

Double Buffering:
- Used when frame has to be received in a short period of time relative to overall frame rate
- Used with constant input and output frame rate, but highly irregular pixel rates

Triple Buffering
- Spare buffer enables asynchronous input and output swap
- Frame rate conversion via frame dropping and/or repeating
- Enable Run-time control for locked frame rate conversion

Reduce burstiness in video data path
Frame Rate Conversion

- The Frame Buffer MegaCore performs frame rate conversion

- It has two modes:
  - Triple Buffer mode
    - Can drop and repeat frames to convert any frame rate to any other frame rate
    - Or perform locked frame rate conversion
  - Double Buffer mode
    - Frame rates remain unchanged
    - Allows line rate conversion
Triple Buffer Mode

- Output 1 shows frame repeating
- Output 2 shows frame dropping
- Output 3 shows locked frame rate conversion
External Memory Calculation

- Calculate worst case external memory requirements of the system:
  - Compute worst case requirement for each Avalon MM port
  - Compute worst case requirement for each IP
  - Compute worst case requirement for the system

<table>
<thead>
<tr>
<th>Worst-case external memory requirement for:</th>
<th>Calculation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each Avalon MM port</td>
<td>max image height x max image width x max frame or field rate x max bits per pixel</td>
</tr>
<tr>
<td>Each IP function:</td>
<td>Sum the Avalon-MM Master bandwidth requirements</td>
</tr>
<tr>
<td>Overall system:</td>
<td>Typically, sum the IP function bandwidth requirements</td>
</tr>
</tbody>
</table>
Components that access external memory:
• 2x Motion Adaptive Deinterlacer
• 2x Frame Buffer
All have one or more memory-mapped interfaces
Understanding External Memory Requirement
Deinterlacer (motion adaptive)

- Motion adaptive deinterlacer requires five (master) accesses to the DDR memory
  - 1 field write (@input rate)
  - 2 field reads (@output rate) – 4 fields in 2 accesses
  - 1 motion vector write
  - 1 motion vector read

- Calculating DDR memory bandwidth
  - Input format: 1080i, 10-bit color, 60 fields/sec
    - $1920 \times 1080 \times 30\text{bits} \times 60/2 = 1.866\text{Gbit/s}$
  - Output format: 1080p, 60 frames/sec, 10-bit color
    - $1920 \times 1080 \times 30\text{bits} \times 60 = 3.732\text{Gbit/s}$
  - Motion format: Only use 10bits for the motion values
    - $1920 \times 1080 \times 10\text{bits} \times 60/2 = 0.622\text{Gbit/s}$
  - Memory access:
    - 1 × write at input rate: 1.866Gbit/s
    - 2 × read at output rate: 7.464Gbit/s
    - 1 × write at motion rate: 0.622Gbit/s
    - 1 × read at motion rate: 0.622Gbit/s

- Total: 10.574Gbit/s
Understanding External Memory Requirement
Deinterlacer (motion adaptive)

- Motion adaptive deinterlacer requires five (master) accesses to the DDR memory
  - 1 field write (@input rate)
  - 2 field reads (@output rate) – 4 fields in 2 accesses
  - 1 motion vector write
  - 1 motion vector read

- Calculating DDR memory bandwidth
  - Input format: 1080i, 60 fields/sec, 10-bit color
    - 1920 × 1080 × 20bits × 60/2 = 1.24Gbit/s
  - Output format: 1080p, 60 frames/sec, 10-bit color
    - 1920 × 1080 × 20bits × 60 = 2.48Gbit/s
  - Motion format: Only use 10bits for the motion values
    - 1920 × 1080 × 10bits × 60/2 = 0.622Gbit/s
  - Memory access:
    - 1 × write at input rate: 1.24Gbit/s
    - 2 × read at output rate: 4.96Gbit/s
    - 1 × write at motion rate: 0.622Gbit/s
    - 1 × read at motion rate: 0.622Gbit/s

- Total: 7.44Gbit/s
External Memory Bandwidth Calculation Data Sample Packing

Limiting factors:
- 20 bit YCbCr 4:2:2 pixels
- 10 bit motion values
- External memory: 256 bit data words

Video data:
- 12 pixels per transfer (240 bits)
- 16 unused bits per transfer
- Wasted bits per pixel:
  16 bits/12 pixels = \textbf{1.33 bits/pixel}

Motion data:
- 25 MVs per transfer (250 bits)
- 6 unused bits per transfer
- Wasted bits per motion value:
  6 bits/25 MV = \textbf{0.24 bit/MV}
## External Memory Bandwidth Calculation
### Worst-Case Bandwidth for Individual IPs

<table>
<thead>
<tr>
<th>Function</th>
<th>Worst Case Input Format</th>
<th>Worst Case Output Format</th>
<th>Worst Case Motion Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deinterlacer</td>
<td>(1080i60) 1920 x 1080 x 21.3 bits x 60/2 fps</td>
<td>(1080p60) 1920 x 1080 x 21.3 bits x 60 fps</td>
<td>(1 motion value per sample) 1920 x 1080 x 10.24 bits x 60/2 fps</td>
</tr>
<tr>
<td></td>
<td>1.327 GBit/s</td>
<td>2.654 GBit/s</td>
<td>0.637 Gbit/s</td>
</tr>
<tr>
<td>Frame Buffer</td>
<td>(1080p60) 1920 x 1080 x 21.3 bits x 60 fps</td>
<td>(1080p60) 1920 x 1080 x 21.3 bits x 60 fps</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>2.654 Gbit/s</td>
<td>2.654 GBit/s</td>
<td></td>
</tr>
</tbody>
</table>

### Memory Access

<table>
<thead>
<tr>
<th>Function</th>
<th>Memory Access</th>
<th>Bandwidth (GBit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deinterlacer</td>
<td>1 read at input rate, 1 read at motion rate, 1 write at motion rate, 2 accesses at output rate</td>
<td>1 x 1.327 = 1.327, 1 x 0.637 = 0.637, 1 x 0.637 = 0.637, 2 x 2.654 = 5.308</td>
</tr>
<tr>
<td></td>
<td><strong>Total Bandwidth</strong></td>
<td><strong>7.909</strong></td>
</tr>
<tr>
<td>Frame Buffer</td>
<td>1 write at input rate, 1 read at output rate</td>
<td>1 x 2.654 = 2.654, 1 x 2.654 = 2.654</td>
</tr>
<tr>
<td></td>
<td><strong>Total Bandwidth</strong></td>
<td><strong>5.308</strong></td>
</tr>
</tbody>
</table>
External Memory Bandwidth Calculation Worst-Case Bandwidth for Total System

Total System Bandwidth Requirement

<table>
<thead>
<tr>
<th>Function</th>
<th>Bandwidth (GBit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x Deinterlacer</td>
<td>2 x 7.909 = 15.818</td>
</tr>
<tr>
<td>2 x Frame Buffer</td>
<td>2 x 5.308 = 10.616</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>26.434</strong></td>
</tr>
</tbody>
</table>

The Arria V GX starter board when using the soft memory controller and the Micron MT41J64M16LA-15E DDR3 SDRAM provides a maximum theoretical bandwidth of:

400 MHz x 32 bits x 2 (both clock edges used) = 25.6 Gbit/s

*Not Enough memory bandwidth for 2 channel 1080i Input

*Use SD for 2nd channel input*
Efficient Memory Subsystem Architecture

Higher memory access efficiency is achieved by using:
- Long bursts – transaction size
- Large on-chip buffers
- Bank-interleaving
Multi-Port Front End Qsys Component (UDX Reference Design)

Design Requirements:
- Adding Nios II and On Screen Display overlay will increase the memory band width requirement on top of the 2 channel processing
- Replaces system interconnect fabric
- Features includes:
  - Time critical and non-critical prioritization
  - Weighted round robin arbitration scheme
High Performance Controller II Features

- Supports up to 1066 MHz DDR3 memory
- Power management
- Advanced bank management w/ command reordering
- Inter-bank data reordering
- Five cycle controller latency (6 w/ ECC)
- ECC with sub-word write
- Flexible system interface
- Run time programmable
- Efficiency Monitor and Protocol Checker
- Multi-cast writes
- Quarter-rate support
Exercise #5

Deinterlacing, Scaling
Lab 5: Format Conversion (w/ Deinterlacer and Scaler IP) and CSC IP
Deinterlacing: The Basics

**Interlace**
First all odd lines scanned (1/60sec)
then all even lines (1/60sec)
presenting a full picture (1/30sec)

**Progressive**
All lines scanned in single pass
presenting a full picture (1/60sec)
VIP Function Details
Deinterlacer II (motion adaptive)

NEW

- Deinterlacer II
  - Supports 3:2 cadence detection and low-angle edge detection
  - Also supports 2:2 cadence detection
  - Supports lower latency
  - Does not support bob or weave methods
  - Different frame buffering method
Deinterlacing
The problem - I

- Moving images in interlaced video streams cannot be simply woven together to produce a high quality image.
Deinterlacing
The problem - II

- They can be better de-interlaced by interpolating the missing pixels using information in the lines in just one field. A naive scheme produces poor results however.
VIP Suite IP: Deinterlacing Algorithms

**Bob Interlacing**
- Scan line interpolation
- Scan line duplication

Preferred when motion in the video

**Weave Interlacing:**
- merge two neighboring field to a frame

Preferred when no motion in the video

**With-Motion**
- Preferred when motion in the video
  - BOB
  - WEAVE

**Without-Motion**
- Preferred when no motion in the video
  - BOB
  - WEAVE
De-interlacing: Motion Adaptive
Switches between Bob and Weave

- Pixels are collected from the current field and the three preceding it

- These pixels are assembled into two 3x3 groups of pixels and the minimum absolute difference (MAD) of the two groups is calculated

- The new pixel values are function of the Motion detected

\[
\text{Output Pixel} = M \cdot \frac{\text{Upper Pixel} + \text{Lower Pixel}}{2} + (1 - M) \cdot \text{Still Pixel}
\]
Understanding External Memory Requirement
Deinterlacer (motion adaptive)

- ‘Motion bleed’ is optional
  - If selected, the computed motion value is compared with a recorded motion value for the same location in the previous frame
  - If NOT selected, comparison is skipped

- ‘Motion bleed’ reduces unpleasant flickering artifacts
  - Needs one extra DDR access
Deinterlacer in the Frame Buffer Mode

Deinterlacer can provide double or triple frame buffering in external RAM

- Frame buffering is required for motion-adaptive and weave de-interlacing
- Frame buffering can be selected for bob de-interlacing
Memory Access for Deinterlacer

Motion Bleed ON, 10bit color depth, 10bit motion value

1 field write

4 field reads over 2 ports

Arbitration Logic

Motion Values

1920 × 1080 × 30bits × 60/2
= 1.866Gbit/s

1920 × 1080 × 30bits × 60/2
= 1.866Gbit/s
x4
= 7.464Gbits/s

1920 × 1080 × 10bits × 60/2
= 0.622Gbit/s
x2
= 1.3Gbits/s

Total: 10.574Gbit/s
Deinterlacing
The problem - III

- For best results, the interpolation must be performed in the correct direction by accurately detecting the edge.
Maths to the rescue!

- Edge direction information may be gained in *progressive* video frames by using well known operators such as Sobel, Canny and Harris.
- A *modified* Sobel operator is proposed in academic literature that allows edge direction information to be gained in *interlaced* video fields.
“calendar” sequence, bucket detail

Dil II UDX 4 (ACDS 11.1)  Dil II UDX 5 (ACDS 12.1)

Incorrect edge direction creates unsightly artefacts

Correct edge direction creates smooth edges
“flag” sequence

Dil II UDX 4 (ACDS 11.1)  Dil II UDX 5 (ACDS 12.1)

Incorrect edge direction at low angles causes *dashing* effect in stripes

Correct edge direction at low angles gives smooth stripes
moving zoneplate detail

Dil II UDX 4 (ACDS 11.1)  Dil II UDX 5 (ACDS 12.1)

Many incorrect edge direction decisions make for terrible overall quality

Correct edge direction decisions make for very high overall quality
What’s the cost?
Finally - ACDS 12.1 Release

- Since the UDX5 release additional pipelining has been included to allow 1080p60 deinterlacing on an ArriaVc5 class device.
- Therefore ALM usage in ACDS 12.1 will be somewhat higher (10-20%)?
VIP Suite IP: Scaling Algorithms

### Nearest Neighbor

\[
\begin{align*}
A & \quad A & \quad B & \quad B \\
A & \quad A & \quad B & \quad B \\
E & \quad E & \quad F & \quad F \\
E & \quad E & \quad F & \quad F \\
\end{align*}
\]

- \( f = \frac{A+B}{2} \)
- \( g = \frac{B+D}{2} \)
- \( h = \frac{C+D}{2} \)
- \( e = \frac{A+C}{2} \)

### Bi-Linear scaling

\[
\begin{align*}
A & \quad f & \quad B \\
e & \quad i & \quad g \\
C & \quad h & \quad D \\
\end{align*}
\]

### Multi-tap, poly-phase

<table>
<thead>
<tr>
<th>A</th>
<th>e</th>
<th>f</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
</tr>
<tr>
<td>C</td>
<td>k</td>
<td>l</td>
<td>D</td>
</tr>
</tbody>
</table>

- New pixel = weighted average of multiple original pixels
- Example:
  - \( e = (0.75A+0.25B) \)
  - \( f = (0.25A+0.75B) \)

Lower cost (FPGA resources)  Higher Quality
VIP Function Details
Scaler Polyphase Algorithm

- Data from multiple lines of the input image are assembled into line buffers – one for each vertical tap
- These data are then fed into parallel multipliers, before summation and possible loss of precision
- The results are gathered into registers – one for each horizontal tap
- These are again multiplied and summed before precision loss down to the output data bit width
Understanding Resource Tradeoff
Scaler Polyphase Algorithm

- Resource usage of a $N_v \times N_h$ taps scaling engine
  - Total number of multipliers $N_v + N_h$
  - Total number of line buffers $N_v$

- Recommended parameters
  - **Scaling Up**
    - 4 Taps, 16 Phases, Lanczos-2
  - **Scaling Down** – M pixels to N pixels
    - $(4 \times M)/N$ Taps, 16 Phases, Lanczos-2
Scaling: Not So Simple In Implementation

- Video scaling is also subject to artifacts
  - Simple “stretching” is pixelated
  - Could improve by interpolating

- Interpolation is not perfect
VIP Suite – Scaler II

- Edge-adaptive scaling \((v12.0)\)
  - Reduces blurriness while maintaining realism

- Polyphase scaling \((Existing)\)
  - Capable to parameterize 16 taps / 256 phases
  - Enables sharper images from any source such as video acquired through phone

VIP Quality to Rival Leading ASSPs
Scaler II – Edge Adaptive

- Available in ACDS 12.0

- Standard polyphase Scaler II quality is ‘not bad’ so significant improvements are not easy to achieve

- Goal is to improve sharpness and reduce stepping on diagonal/curved edges
  - Sharpening will tend to increase step artefacts

- Reducing stepping without increasing blur can be done with iterative algorithms (and maybe wavelets), but this would come at significant cost
  - Our improvements have been made within the constraints of the current scaler architecture – separable vertical & horizontal FIR filters
Scaler II – Edge Adaptive

■ 2 additions to polyphase algorithm

− Use two sets of coefficients, one for pixels around edges and the other for non-edge pixels
  ● Runtime adjustable edge threshold
  ● Use ‘softer’ coefficients in flat areas and ‘sharper’ coefficients around edges

− Optional edge adaptive sharpening filter (FIR filter) applied post scaling
  ● This is cheating (distortion) – but can make the downscale look sharper
  ● Detects ‘blurred edges’ and only applies filter in these regions
  ● Runtime adjustable upper and lower ‘blur’ thresholds
Upscale (1)

- For upscale
  - Some improvement in edge sharpness with reduced stepping

v11.1  

v12.0
Upscale (2)

- For upscale
  - Some improvement in edge sharpness with reduced stepping
For downscale
- General improvement in the sharpness of the output image
Downscale (2)

- For downscale
  - General improvement in the sharpness of the output image
### Scaler II – Edge Adaptive Resource Usage

<table>
<thead>
<tr>
<th>Resource</th>
<th>Usage on Arria V</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALMs</td>
<td>3090</td>
</tr>
<tr>
<td>Memory Bits</td>
<td>313141</td>
</tr>
<tr>
<td>M10Ks</td>
<td>48</td>
</tr>
<tr>
<td>DSPs</td>
<td>16</td>
</tr>
</tbody>
</table>
Exercise #6

Mixer, OSD, Ctrl Sync, and CRS IP
Lab 6: Mixer IP, OSD (and transparency), Ctrl Sync IP (for layer switching) and CRS IP
VIP Suite IP: Alpha Blending Mixer

Select the size of the image

The level of blending is controlled by the number of alpha bits (up to 8)

Select the number of layers to be mixed
Control Synchronizer

- When the control data packet of the next video frame changes
  - Control Synchronizer detects it
  - Triggers a write to the scaler
  - Stalls the video data pipeline

- Nios II provides the new value for the scaler

- Control Synchronizer writes that data in the scaler registers

- After the scaler has been reconfigured
  - Control Synchronizer resumes the video data pipeline
Control Synchronizer … in our case

- The control synchronizer is used to ensure that register accesses to the switch and mixer occur during the same video frame.
- Both the switch and mixer only apply register updates at the start of a video frame.
VIP Suite IP: Chroma Resampler

- This function allows you to change between 4:4:4, 4:2:2 and 4:2:0 sampling rates

- For 4:2:2 → 4:4:4, the filtered algorithm uses a 4-tap filter with fixed Lanczos-2 coefficients

- For 4:4:4 → 4:2:2, the filtered algorithm uses a 9-tap filter with fixed Lanczos-2 coefficients
Exercise #7

VIP Reference Designs
Reference Design: UDX6 on Arria-V starter kit

- DisplayPort
- SDI
- Qsys
- ISO
- DDR3 HP2 Memory Controller
- Video Trace Monitor

New in UDX 5
New in UDX 6
VIP Core

Run-time configurable via register map (Avalon-MM Slave interface).
Configuration changes (e.g., viewing mode changes, scaler coefficient reload) performed by software executing on Nios II processor.

© 2012 Altera Corporation—Confidential
UDX 5.1.3 (available now)

- 2 channels of 1080p60 processing featuring:
  - Motion Adaptive Deinterlacing
  - Edge Adaptive Scaling
  - Supports SDI, Displayport and HDMI

- Runs on SIVGX dev kit
  - HDMI output available on the board
  - SDI I/O supported by HSMC daughtercard (required)
  - DisplayPort I/O supported through HSMC daughtercard (optional)

- Available now
4K Upscale 3

- Deinterlace and up-scale from SD to 3840x2160p60 (QFHD)
  - 1x 3G SDI Input, 4 x 3G SDI Outputs

- Runs on SVGX dev kit
  - SDI I/O supported by 2 HSMC daughtercards (required)

- VCXOs removed
  - All clock locking performed by FPGA (fPLLs)
  - SDI II megacore supports (ACDS 12.1)
4K Upscale 3 Block Diagram

Each frame reader reads a quarter of the image (1080p60)

Avalon-MM
Avalon-ST Video
Avalon-ST Message (data)
Avalon-ST Message (command)
Avalon-ST Message (response)

Video Input Component
Video Pipeline Component
Video Control Component
Video Output Component
Memory Component

© 2012 Altera Corporation—Confidential
4K Upscale 2.0 (available now)

- 1 channel of QFHD processing featuring:
  - Motion Adaptive Deinterlacing
  - Edge Adaptive Scaling

- Runs on SIVGX dev kit
  - SDI I/O supported by 2 HSMC daughtercards (required)

- Available now
Introducing: A larger & powerful NEEK with 5 Mega Digital Camera

VEEK

Terasic’s Video & Embedded Eval. Kit
How to Equip with VEEK

- Download VIP Suite designs
  - Two type of files
    - Design files
    - SD card files
    - [www.alterawiki.com/wiki/Videoframework](http://www.alterawiki.com/wiki/Videoframework)

- Available online from Terasic
  - VEEK is Terasic’s product
  - Purchase online through Terasic’s website - $795/unit

- Equipment you may need
  - Video Source (Composite Video)
  - Camera / DVI daughter cards
Demo # 1 – Simple Format Conversion

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite</td>
<td>de-interlaced video /w txt overlay (NTSC to 800x600 with real-time resizing, zoom and panning)</td>
<td>VEEK display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VGA out</td>
</tr>
</tbody>
</table>

- Composite video input
- Key Video Functions
  - Deinterlacer, Scaler,
    Alpha blending mixer, Color Space Converter, Clipper

**NEEK VIP Suite Demo Now Ported to VEEK**
Demo # 2 – Camera Video Scaling

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera on VEEK</td>
<td>Camera Input/w txt overlay (Camera input scaling with real-time resizing, zoom and panning)</td>
<td>VEEK display</td>
</tr>
</tbody>
</table>

- Camera video input
- Key Video Functions
  - Scaler, Alpha blending mixer, Frame buffer, Clipper

Only Requires VEEK - No Additional Hardware Needed
Demo # 3 – Picture in Picture

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite</td>
<td>de-interlaced video /w txt overlay</td>
<td>VEEK display</td>
</tr>
<tr>
<td>camera on VEEK</td>
<td>PIP - (camera source)</td>
<td>VGA out</td>
</tr>
</tbody>
</table>

- 2 video inputs
  - Composite, Camera

- Key Video Functions
  - Deinterlacer, Scaler, Alpha blending mixer, Color Space Converter, Clipper

New PiP Demo on VEEK
Demo # 4 – Multi-view

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite</td>
<td>CH 1 / 3: de-interlaced video</td>
<td>DVI (HSMC) @ 800x600</td>
</tr>
<tr>
<td>camera (GPIO)</td>
<td>CH 2: camera video</td>
<td></td>
</tr>
<tr>
<td>DVI (HSMC)</td>
<td>CH 4: DVI video (800x600 down scaling)</td>
<td></td>
</tr>
</tbody>
</table>

- 3 video inputs
  - Composite, Camera, DVI
- 1 DVI output with multiview
- Key Video Functions
  - Deinterlacer, Scaler, Alpha blending mixer, Color Space Converter, Clipper
Summary
Altera Technical Support over worldwide

- Reference Quartus II software on-line help
- **Quartus II Handbook**
- Consult Altera applications (factory applications engineers)
  - Hotline: (800) 800-EPLD (7:00 a.m. - 5:00 p.m. PST)
- Field applications engineers: contact local Altera sales office
- Receive literature by mail: (888) 3-ALTERA
- Altera Forum: [www.alteraforum.com](http://www.alteraforum.com)
- FTP: [ftp.altera.com](ftp.altera.com)
- World-wide web: [http://www.altera.com](http://www.altera.com)
  - Use solutions to search for answers to technical problems
  - View design examples
- Altera device selection guide
Axios Technical Support in Korea

- FAE support: 031-776-9888
- Altera town café: http://cafe.naver.com/alteratown
- NOIS town café: http://cafe.naver.com/niosii.cafe
- Axios Web board: http://board.uniquest.co.kr
  - ID: ualtera
  - Password: altera123

- Terasic support
  - Woorimtni. Co., Ltd
  - 02-512-7661, terry@woorimtni.co.kr; terry@terasic.com
  - www.woorimtni.co.kr
Learn More Through Technical Training

<table>
<thead>
<tr>
<th>Instructor-Led Training</th>
<th>Virtual Classroom Training</th>
<th>Online Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Altera's instructor-led training courses, you can:</td>
<td>With Altera's virtual classroom training:</td>
<td>With Altera's online training courses, you can:</td>
</tr>
<tr>
<td>▪ Learn from an experienced Altera technical training engineer (instructor)</td>
<td>▪ Get the best of both worlds!</td>
<td>▪ Take a course at any time that is convenient for you</td>
</tr>
<tr>
<td>▪ Complete hands-on exercises with guidance from an Altera instructor</td>
<td>▪ All the benefits of a live, instructor-led training class from the comfort of your home or office</td>
<td>▪ Take a course from the comfort of your home or office (no need to travel as with instructor-led courses)</td>
</tr>
<tr>
<td>▪ Ask questions and receive real-time answers from an Altera instructor</td>
<td></td>
<td>▪ Each online course takes approximate one to three hours to complete</td>
</tr>
<tr>
<td>▪ Each instructor-led class is one or two days in length (8 working hours per day)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[http://www.altera.com/training](http://www.altera.com/training)

View training class schedule and register for a class
Thank You!