

## Image-Based Driver Assistance Development Environment

This white paper describes a development environment for all driver assistance (DA) requirements using Altera® FPGA and HardCopy® ASIC devices. This development environment consists of a development platform, an architecture template, an implementation methodology, and tools. This white paper describes using the development environment to implement a lane departure warning (LDW) algorithm.

### Introduction

Motor vehicles are incorporating new solutions using cameras to assist drivers. First-generation features that are in production cars today include LDW, rear-view with parking aid, and night vision. Second-generation systems under development include processing of multiple cameras.

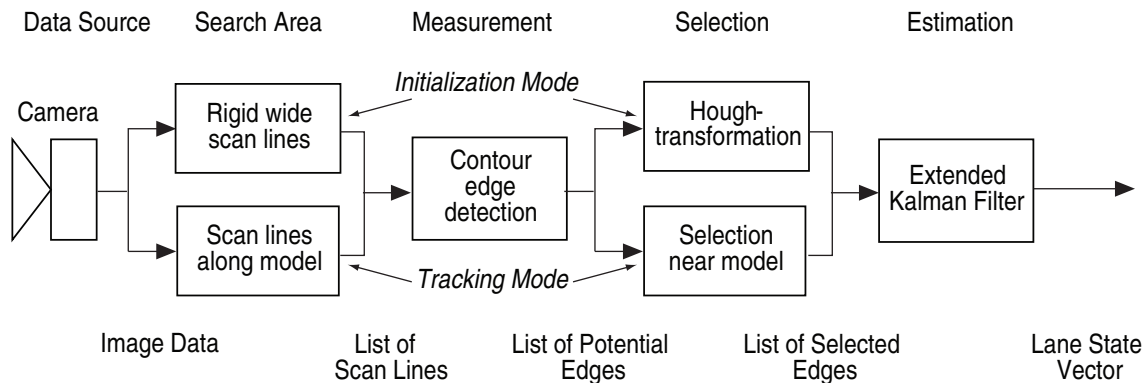
System developers face many challenges. The latest algorithms require huge processing capabilities and power constraints limit the options available. A high level of scalability is necessary to address a wide range of requirements from the most basic systems with one camera input to complex ones, such as a birds-eye and surround view, integrating up to four camera inputs. This area uses a wide range of semiconductor devices, including ASSPs, DSPs, and FPGAs or a combination of those, resulting in complex implementations that are difficult to maintain.

### LDW Algorithm

The LDW software from Elektrobit Automotive GmbH, which is based on the PReVENT SAFELANE European project, is C++ floating point source code that is not optimized for embedded implementations. The algorithm (Figure 1) processes the video by extracting relevant measurement points, identifying lane candidates and filtering results using information from previous frames. The result is the system identifying the position of the car within the lane. A warning policy can then be applied based on previously calculated data.

For more information on Elektrobit Automotive GmbH, refer to [www.elektrobit.com](http://www.elektrobit.com).

Figure 1. LDW Processing Flow



### Development Platform

The DA development environment (Figure 2) is based on the Altera platform ASSP replacement infotainment system (PARIS-1) development platform.

For more information on the PARIS-1 development platform, refer to the *Gleichman AAP-PARIS-1 Manual*.

Figure 2. Two LCDs (Left), PARIS-1 Development Platform (Center), and HDD (Right)



The PARIS-1 development platform consists of the following hardware:

- A Stratix® II FPGA module (Figure 3)
- A motherboard

Figure 3. FPGA Module



The Stratix II FPGA module contains two DDR2 memories, a flash memory, PHY devices for USB and Ethernet, and an EXM32 connector to the motherboard.

The motherboard has different bus and multimedia interfaces that can be accessed by the module. The PARIS-1 development platform uses the hard disk drive (HDD) interface, two thin-film transistor (TFT) LCD connections (one for the DA subsystem and one for the streaming subsystem) and touchscreen interface.

You can reuse the PARIS-1 development platform for demonstrations by using the motherboard's general purpose I/O (GPIO) connectors for camera inputs. Moreover, you can build simple prototypes based on the Stratix II FPGA module.

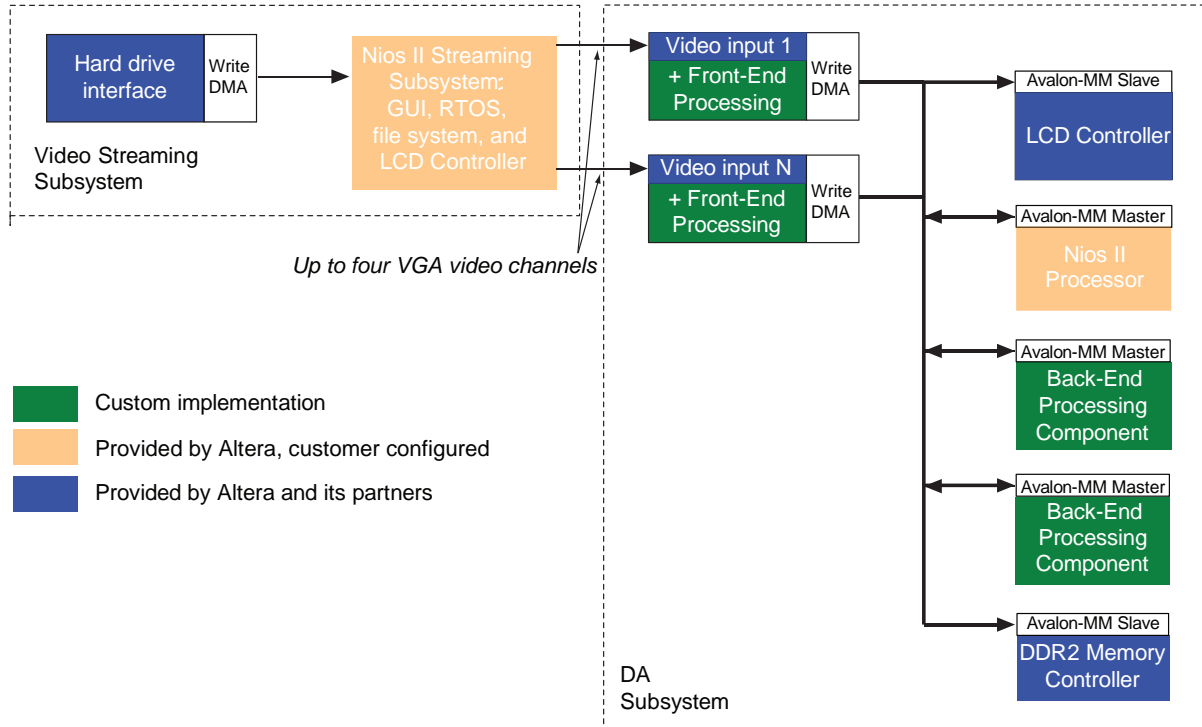
## Architecture Template

DA projects usually start with algorithm development on PCs and use test videos to evaluate the performance of the video processing. The development environment can stream real-time video recordings from an HDD. You can then apply the same video recordings for the algorithm validation stage on the final embedded solution.

The FPGA design (Figure 4) is divided into two separate entities, the streaming subsystem and the DA processing subsystem. Each subsystem has its own independent DDR2 memory interface. Because of the nature of the FPGA implementation, the two entities are working independently. Thus, the load of one subsystem does not impact the

performance of the other. There is one requirement—the streaming subsystem must provide input test data at a sufficient rate to the DA subsystem. In the final solution, the streaming subsystem is replaced by camera inputs.

Figure 4. FPGA Subsystems Overview



The streaming subsystem has a configurable touchscreen (Figure 5). Its HDD interface and software driver integrated in a file system can stream up to four uncompressed color VGA video channels to the FPGA (from the HDD to the DA subsystem).

Figure 5. Streaming Subsystem GUI



The DA processing subsystem is built around a Nios® II soft-core processor with the following basic elements:

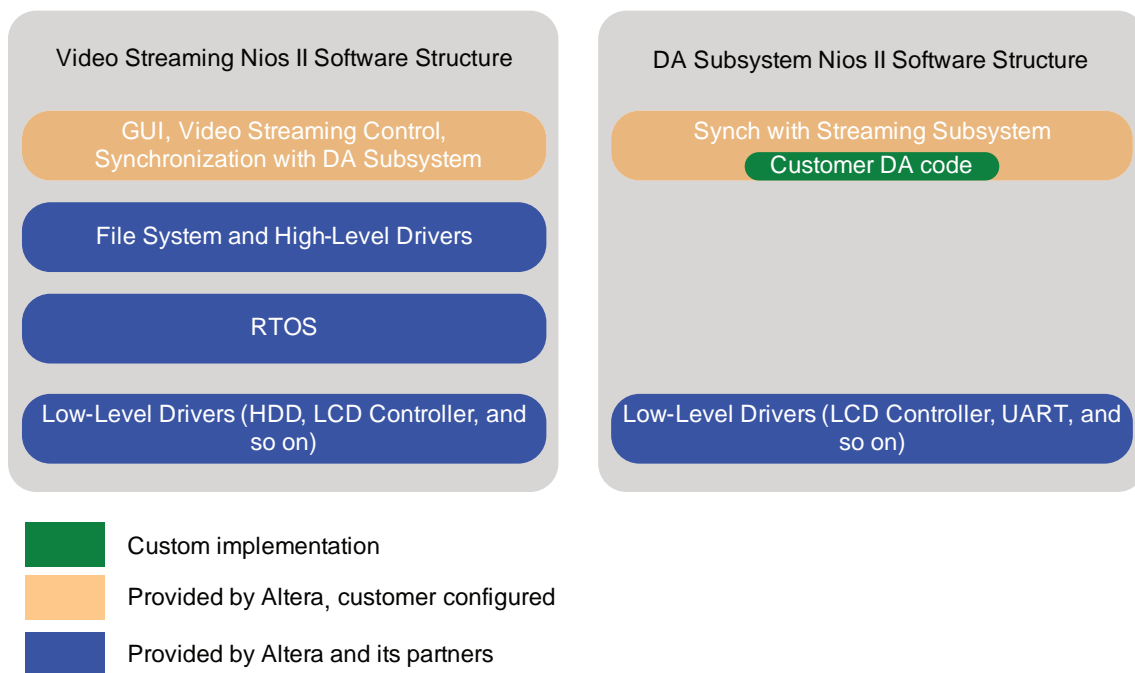
- A performance counter for code profiling
- A message buffer to interface with the streaming subsystem
- An LCD controller to display frame buffer contents

Figure 6 shows how the processing code is instantiated within the software template. After a profiling stage, you can identify the part of the code that needs hardware acceleration. The following two methodologies define code acceleration based on a simple observation of the algorithm:

- If the algorithm is processing pixels of the input video stream consecutively, implement it as a front-end SOPC Builder component. The input from such a component is a video streaming interface. Output data is stored in a frame buffer of preprocessed video data.
- If the video processing requires random access in each video frame received, implement it in a process reading input data from a frame buffer and writing the results to another buffer. This type of component is referred to as a back-end processing SOPC Builder component.

These two methodologies are not exclusive, so designs can implement different parts of a solution using one or both techniques.

Figure 6. Subsystems Software Structure



The LDW algorithm consists of C++ code. To integrate the LDW C++ code into the development environment, Altera and Elektrobitt performed the following steps:

1. Removed the PC environment part of the code that handles input data reading and post processing display settings.
2. Integrated the processing part of the code in the new environment taking input from a frame buffer and stored the output to another frame buffer.
3. Configured the synchronization between the streaming subsystem and the DA subsystem.

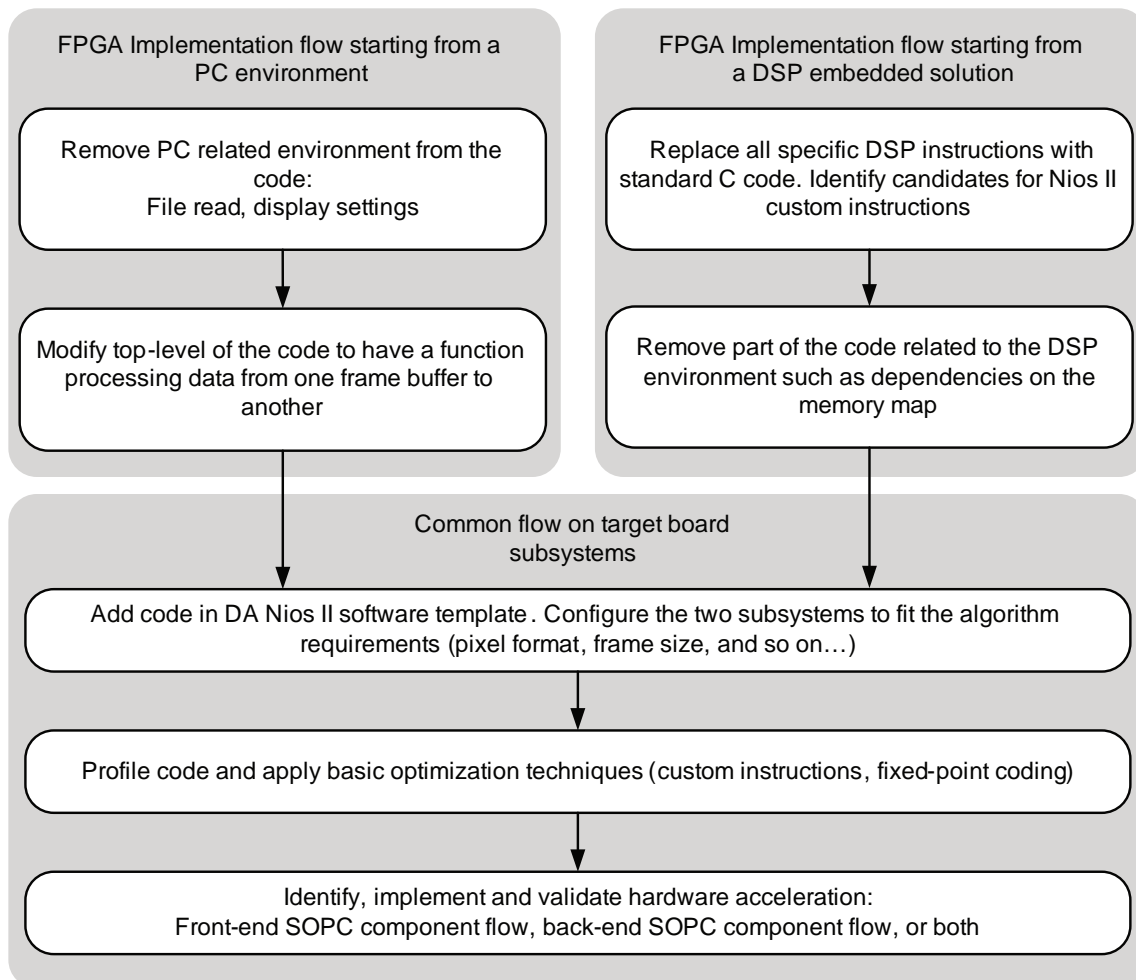
4. Converted unnecessary floating-point code to fixed-point code. Fixed-point operations are implemented much more efficiently than floating-point code.

When this set-up and profiling activity was completed, the measurement points generation stage, taking 70% of the cycles, was identified as a candidate for a front-end SOPC Builder component implementation.

## Implementation Methodology

The code implemented and accelerated in the DA development environment can have multiple sources. It can be embedded code that targets a different device, or high-level code in a PC environment. In either case, the implementation methodology is similar (Figure 7). The main task is to reduce the code to a standard C or C++ function processing one frame buffer and outputting the result in another. When this step is complete, the function can be added into the provided DA software template and is ready to be optimized using techniques and tools described in this white paper.

Figure 7. Code Implementation and Acceleration Flow

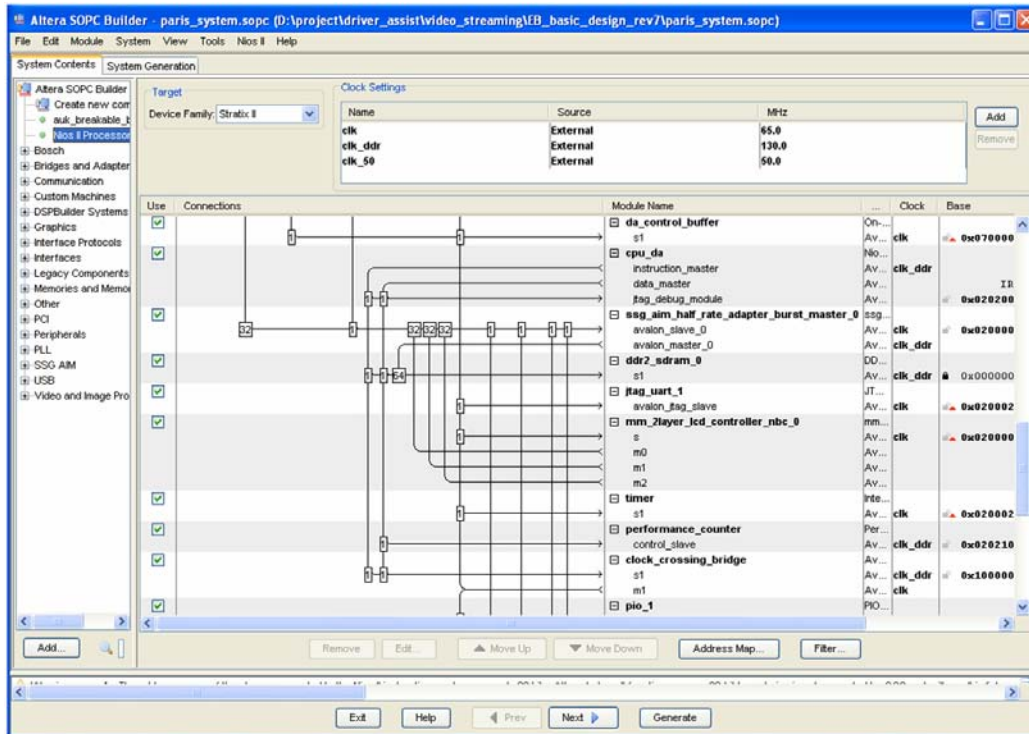


## Tools Review

Altera used SOPC Builder to build the two subsystems. The system is configured with a GUI (Figure 8) and different SOPC Builder components (DDR2 SDRAM controller, Nios II processor, and performance counter) are instantiated

around the Avalon<sup>®</sup> memory-mapped (Avalon-MM) bus. SOPC Builder provides files to the Quartus<sup>®</sup> II software for the FPGA configuration elaboration and to the Nios II software development tools to build a specific board support package (BSP) library.

Figure 8. SOPC Builder



The Nios II software development tools offer software drivers for the different SOPC Builder components instantiated in the system and support systems with multiple Nios II CPUs. Therefore, you only have to spend effort on developing your application specific code.


Many hardware generation tools can target Altera FPGAs. For driver assistance, the focus is on solutions that can generate SOPC Builder components. Altera provides DSP Builder and C2H. DSP Builder allows the developer to design SOPC Builder components using the industry-standard MATLAB and Simulink tools. C2H is integrated within the Nios II software development tools and generates hardware acceleration SOPC Builder components directly from functions within the C code.

Third-party tools are available, which integrate with Altera development tools. For example, the Impulse C compiler, from Impulse Accelerated Technologies. Starting with a C description of the hardware component's behavior, you perform the functionality validation and generation steps within the Impulse C environment, and then integrate the resulting SOPC Builder components and drivers in the DA subsystem and Nios II application code.

### Basic System Optimization

Before considering adding hardware acceleration SOPC Builder components to a design, Altera recommends some basic optimizations. For example, use the Nios II fast configuration and tuning the instruction and data cache size to support the code requirements, then add custom instructions to maximize the processor performance on repetitive multi-cycle operations that are not supported in the default arithmetic logic unit (ALU).

In the LDW solution, the filtering part of the lane information retrieved from each video frame uses a large amount of floating-point operations. To preserve the accuracy of the results, these operations are kept in floating-point code. Adding the relevant floating-point custom instructions enabled an acceleration of up to 20 times compared to the original ALU performance.

 For more information on the Nios II custom instructions, refer to *Nios II Custom Instruction User Guide*.

## Hardware Acceleration Methodology

This section discusses front-end and back-end processing SOPC Builder components.

### Front-end Processing SOPC Builder Components

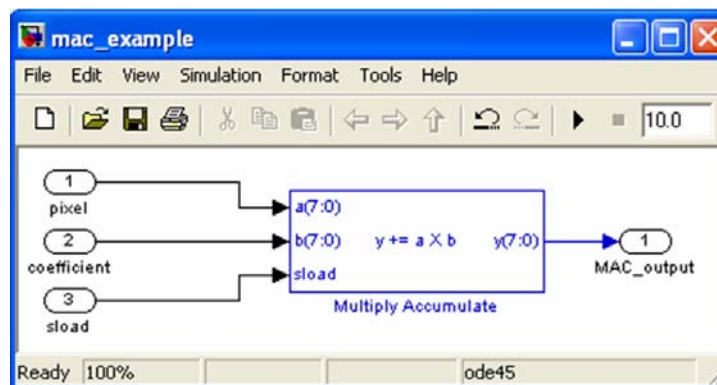
Front-end SOPC Builder components perform operations on the video stream as it is received. Such an implementation frees the CPU from performing repetitive arithmetic operations, and removes the previous delay associated with the time the CPU takes to perform these operations.

Consider a simple multiply accumulate (MAC) operation to be performed on every pixel of an image of size  $N$  pixels. For a processor that can process one MAC operation per clock-cycle, the delay is the number of pixels to process per frame:  $N$  cycles (for example 307,200 cycles for VGA). If a front-end SOPC Builder component implements this MAC operation in one cycle ([Figure 9](#)), the delay added to the video input is instead only one clock-cycle.

DSP Builder is well suited to implement such components. The MATLAB environment offers many utilities to test the implementation before generating the hardware. Once the functionality is validated, it takes only a few minutes to add the component to the rest of the SOPC Builder system.

For more information on DSP Builder, refer to the [DSP Builder](#) web page.

Figure 9. DSP Builder MAC



The LDW algorithm starts with a measurement points generation stage. To simplify the analysis, this stage is referred to as edge detection. The edge detection is a linear process in the flow. Going through each frame pixel-by-pixel, repetitive operations are performed to retrieve edge information. It is then logical that the FPGA embedded solution uses a front-end block to implement this function. The DSP Builder implementation of this front-end block exchanged 70% of the CPU load for a logic cost of approximately 2,500 logic elements (LEs) and 40  $9 \times 9$  DSP elements.

### Back-End Processing SOPC Builder Components

Back-end SOPC Builder components perform operations randomly on the video or a set of data. Using integrated master ports and optimized hardware implementations for their specific functionality, these components process data much faster than a traditional processor implementation. However, as back-end SOPC Builder components do not

process the data on the input flow, the processing operation still introduces a delay. To benefit the most from the hardware accelerator, you should modify the code to execute tasks in parallel. With such code modifications, instead of waiting for the hardware accelerator to complete its task, the Nios II processor can simultaneously process another part of the algorithm. Parallelization can also be achieved with pipelining scheduling techniques when the remaining part of the algorithm is dependant on the data provided by the hardware accelerator.

LDW did not need back-end acceleration, as the performance satisfied requirements. But using this methodology, Altera implemented fish-eye correction, another algorithm commonly found in video based DA solutions. The complete solution is the drawing of an overlay showing the car's direction on top of a corrected video stream coming from a rear-view camera. The two tasks performed are independent and, as such, can be processed in parallel. The fish eye correction function was accelerated using C2H. While the generated SOPC Builder component performs the correction, the Nios II processor draws the direction overlay.

## Conclusion

Altera FPGA and HardCopy ASIC devices are attractive for driver assistance solutions. One set of development tools support all device families, offering unique scalability possibilities. By using parallel implementation techniques, instead of increasing the system operating frequency, processing performance can be gained without compromising on power consumption.

Implementing complex DSP solutions on these devices is simplified by the recent addition of new tools and market-specific development platforms. As the DA development environment is available today, you can focus on the specifics of your own innovative solutions.

## Further Information

For further information refer to the following sources:

- Nios II C-to-Hardware acceleration compiler:  
[www.altera.com/products/ip/processors/nios2/tools/c2h/ni2-c2h.html](http://www.altera.com/products/ip/processors/nios2/tools/c2h/ni2-c2h.html)
- DSP Builder web page:  
[www.altera.com/products/software/products/dsp/dsp-builder.html](http://www.altera.com/products/software/products/dsp/dsp-builder.html)
- Reducing Power in Embedded Systems by Adding Hardware Accelerators:  
[www.embedded.com/design/embeddedfpga/207100707?\\_requestid=520986](http://www.embedded.com/design/embeddedfpga/207100707?_requestid=520986)
- Flexible Microcontroller Solutions for Automotive Applications:  
[www.altera.com/end-markets/auto/flex-micro/aut-flex-micro.html](http://www.altera.com/end-markets/auto/flex-micro/aut-flex-micro.html)
- *Creating Low-Cost Intelligent Display Modules With an FPGA and Embedded Processor White Paper*
- *Applying Graphics to FPGA-Based Solutions White Paper*

## Acknowledgment

Yann Le Hénaff, Member of Technical Staff, Automotive System Solutions, Altera Corporation.



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

Copyright © 2008 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.