

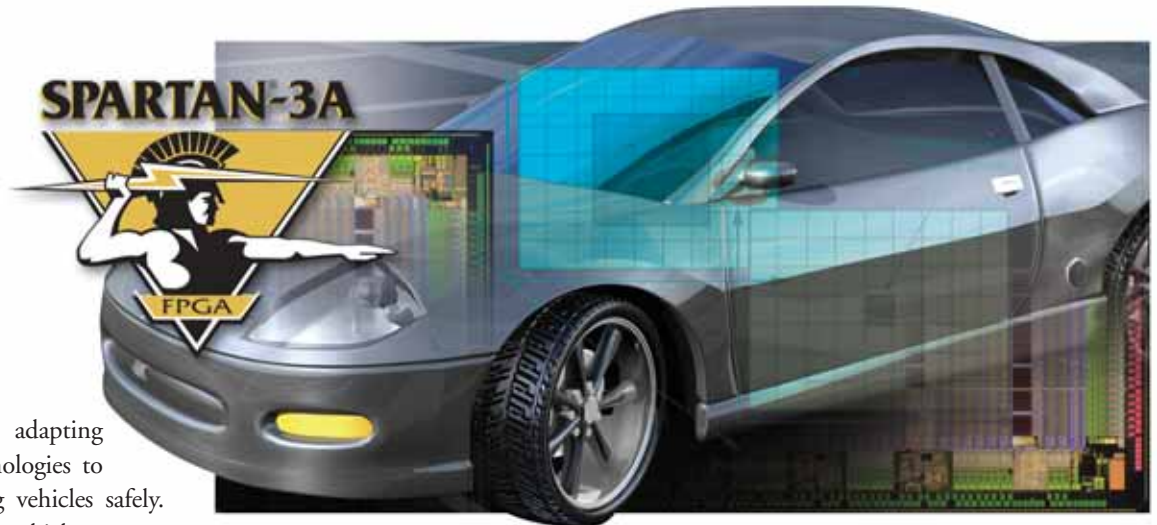


# Block Matching for Automotive Applications on Spartan-3A DSP Devices

The Spartan-3A DSP FPGA outshines VLIW DSP-CPU's in automotive driver assistance applications.

by Daniele Bagni  
DSP Specialist  
Xilinx, Inc.  
daniele.bagni@xilinx.com

Paul Zoratti  
Automotive Senior System Architect  
Xilinx, Inc.  
paul.zoratti@xilinx.com



Automotive engineers are adapting numerous intelligent technologies to assist humans in operating vehicles safely. Prominent technologies in vehicle systems include radar-, ultrasonic-, and camera/vision-based sensing. Collectively referred to as driver assistance (DA) systems, these technologies are designed to facilitate safe driving during adverse conditions and potentially dangerous roadway situations.

The first generation of camera-based DA systems is now available on a variety of production vehicle models. The majority of such systems provide drivers with a video image of the environment around the vehicle. The most prevalent systems are parking/backup aids that use a rear-facing camera to capture the scene behind the host vehicle and display the image on a radio/navigation system screen, or on a small display in the instrument cluster.

A second generation of camera-based systems is in development and testing, with some limited deployment. Rather than merely providing an image to drivers, these second-generation systems apply image processing and analytics to extract information from the video stream and characterize and evaluate the vehicle environment. If warranted, the driver receives an appropriate warning.

As engineers gain real-world experience in characterizing the vehicle environment, future DA techniques will increase in complexity, offer greater utility to consumers, and enhance the performance of other vehicle subsystems. Figure 1 summarizes the variety of current and future DA features.

## Advanced Processing Requirements

Processing requirements for DA systems can exceed the capabilities of current automotive-grade serial DSP processors. In addition, a growing need exists to bundle multiple DA features together, based on a single suite of vision sensors to drive consumer value.

For example, a forward-looking vision module may need to simultaneously support lane departure warning, intelligent headlamp control, and sign recognition functionality – all of which require different processing algorithms. Therefore, the DA market offers a real opportunity for FPGAs to provide system value through raw processing performance, configuration flexibility, and device scalability.

Image processing and analytics functionality for vision-based DA schemes can include spatial/temporal filtering, lens-distortion correction, image sharpening, contrast enhancement, edge detection, pattern matching, object recognition, object tracking, and, in some cases, graphical overlay. Of particular interest is a form of pattern-matching functionality that supports motion estimation or stereo disparity calculations.

To illustrate the performance value of FPGA processing, consider the following vision-based system: a wide-VGA resolution imaging device (752 x 480 pixels) generating video at a 30-Hz frame rate (fps), and the need to estimate the motion (or flow) of objects from one frame to another. One algorithmic approach – also suitable for stereo ranging disparity calculations – is to partition the image into blocks (say, 4 x 4 pixels in size) and evaluate a match criteria for each block in the first frame to a location in the second frame over a specified search area (say, 20 x 20 pixels).

A common match criteria is to find the minimum absolute error (MAE) of the pixel intensities between the 4 x 4

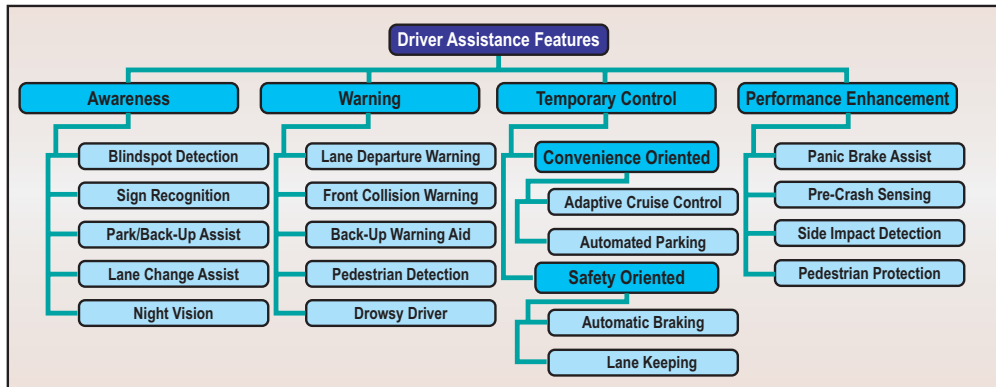


Figure 1 – Driver assistance features

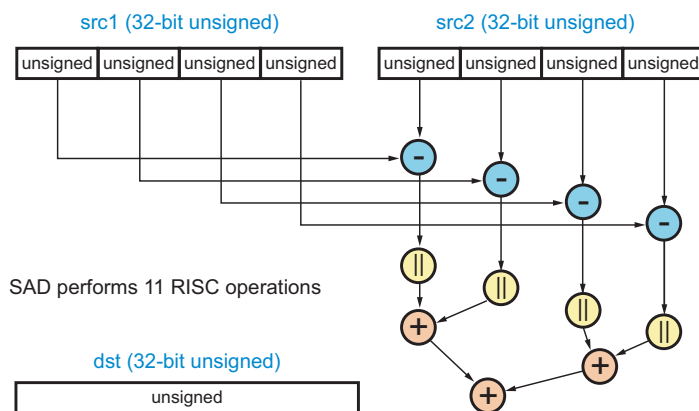


Figure 2 – An example of SIMD: SAD operation on a quadruple of 8-bit samples

block in the first image and the pixels within the search area on the second image by using an operator called SAD (sum of absolute differences).

Our 4 x 4 block matching example requires more than 250 MMAE/s (millions of MAE calculations per second), since (752 pixels) x (480 lines) x (20 x 20 pixel search area) x (30 fps) / (4 x 4 pixel block size) = 270,720,000 MAE/s. MAE indicates the final matching error of a 4 x 4 pixel block, whereas SAD refers to the sum of absolute differences calculation on four individual pairs of elements. Therefore, every MAE requires four SAD operations.

### Processing Options

Processing options at the automotive design engineer's disposal include very long instruction word (VLIW) DSP-CPU and FPGAs. FPGAs offer dramatically higher

processing capability than any existing VLIW DSP-CPU. This is because of the architecture of the FPGA: the large amount of functional units in parallel (including programmable MACs) allows the FPGA to achieve 10-30 times better performance than that offered by any DSP, depending on the application implemented, even if the clock frequency of the FPGA is much lower than the clock frequency of the DSP-CPU. Using the block-matching operation example, we will demonstrate that Xilinx® FPGAs have superior performance to any VLIW DSP-CPU processor.

### SAD and MAE Calculations in VLIW DSP-CPU Processors

A SAD operation on four elements of 8-bit pixel video data could be implemented in a single-instruction-on-multiple-data (SIMD) within a 32-bit architecture DSP-

CPU, thus efficiently executing the equivalent of 11 elementary instructions in only one cycle, as shown in Figure 2.

For example, the Nexperia PNx1500 media processor equipped with the TriMedia 32-bit VLIW-CPU can implement two quadruple SAD instructions on a single clock cycle for 8-bit pixels with two cycles of latency. Associated with every long instruction word are as many as five elementary RISC/SIMD instructions

per clock cycle, of which only two can be SAD (named "8meii" in the TriMedia databook).

Therefore, the MAE on a 4 x 4 block size would require five clock cycles, as shown in Table 1: two cycles for pipelining two quadruple SADs (cycle 1 for sad1/sad2, cycle 2 for sad3/sad4) and three cycles for accumulating the partial results (cycles 3, 4, and 5). Hence, a 300-MHz Nexperia PNx1500 processor could compute a peak of 60 MMAE/s if processing a single block.

By processing more than one 4 x 4 block at a time, the peak performance can improve a little. For example, the MAE of two 4 x 4 blocks in parallel could be computed in seven cycles, thus achieving 85.71 MMAE/s, and three blocks could be processed in nine cycles, or 100 MMAE/s.

The maximum amount of blocks that can be processed in parallel is limited first by the number of SIMD SAD operations allowable in any long instruction word; second by the number of general-purpose registers of the VLIW-CPU; and third by the scheduling algorithms of the optimizing compiler. Overall performance goes into saturation when adding more blocks, which is why we do not consider more than three MAEs processed in parallel.

The Texas Instruments (TI) TMSD320DM6437 digital media processor has a long instruction of eight elementary RISC operations per cycle through two separated data paths: each one of four slots per cycle. Its VLIW-CPU can execute as many as two SAD instructions per cycle (named "subabs4" in the TI DM6437



databook), each with a latency of one cycle. However, to accumulate the partial results, a three-cycle latency SIMD MAC operation must be performed (named “dotpsu4”), with a constant 0x01010101.

Therefore, a 600-MHz TI DM6437 DSP-CPU could compute an MAE in seven cycles (as shown in Table 2), thus achieving a peak performance of 85.71 MMAE/s for 4 x 4 pixel blocks. If two blocks are processed in parallel, we get nine cycles and 133.33 MMAE/s, whereas for three blocks we get 11 cycles and 163.64 MMAE/s – again lower than our 250 MSAD/s requirement.

### De-Rating VLIW DSP-CPU Performance

Thus far, we have assumed 8 bits per pixel, which is very suitable for 32-bit architecture DSP-CPU processors. However, new CMOS image sensors have a higher resolution range: 12 to 14 bits per pixel. For these data types, the classic quadruple 8-bit sub-words SIMD of 32-bit architectures are less effective and must be replaced by dual 16-bit half-words SIMD, in which the sub-

word parallelism is only two. Therefore, peak performance degrades significantly, because more clock cycles are necessary to compute an MAE.

Table 3 shows what could be the pseudo assembly code of the SAD computation on the TI VLIW DSP-CPU when using 16-bit sub-word instructions, considering the correct latency and the functional issue slot able to deliver such instructions. As a result, eight cycles are necessary for one 4 x 4 block while 10 and 12 cycles are required, respectively, for two and three blocks processed in parallel. Corresponding peak performance is then 75 MMAE/s, 120 MMAE/s, and 150 MMAE/s. All of these numbers are smaller than the ones achieved with 8-bit sub-word instructions.

### Spartan-3A DSP FPGA SAD and MAE Performance

To fill the processing performance gap between Spartan™-3 and Virtex™-4 devices, Xilinx introduced the Spartan 3A-DSP 1800A and 3400A FPGAs. These

devices incorporate a modified version of the DSP48 slices found in Virtex-4 devices. In addition, 3A-DSP parts include a larger number of on-chip memory (block RAMs). Both of these enhancements, along with a price point suited to high-volume applications, make 3A-DSP devices a good fit for automotive vision-based DA systems.

Figure 3 shows the scheme of SAD computation for a quadruple of 12-bit pixels on the Spartan-3A DSP 1800 (XC3SD1800A-4FG676) device. This implementation was done with the System Generator for DSP design flow (a bit-true, cycle-accurate, synthesizable library provided by Xilinx in the Simulink tool). The amount of resources required is 121 slices (236 LUTs and 140 flip-flops). By replicating this structure four times and adding the partial results, we get the scheme for a whole 4 x 4 block, which requires 508 slices (990 flip-flops and 606 LUTs) with a throughput of one (which means that at any clock cycle we can start the computation of a new MAE) and a latency of seven cycles.

Utilizing a 150-MHz clock frequency (of the maximum 250 MHz the device could achieve), we would need only two parallel structures occupying approximately 6% of the device area to achieve 300 MMAE/s and meet the 250 MMAE/s requirement of our example application. This leaves ample resources available to implement other image processing func-

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
Cycle 1	sad1=8meii(A1,B1)	nop	sad2=8meii(A2,B2)	nop	nop
Cycle 2	sad3=8meii(A3,B3)	nop	sad4=8meii(A4,B4)	nop	nop
Cycle 3	sad12=sad1+sad2	nop	nop	nop	nop
Cycle 4	sad34=sad3+sad4	nop	nop	nop	nop
Cycle 5	tot=sad12+sad34	nop	nop	nop	nop

Table 1 – Pseudo assembly code for the MAE computation on the Nexperia/TriMedia VLIW DSP-CPU, with quadruple 8-bit sub-word parallelism

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
	L1	S1	M1	D1	L2	S2	M2	D2
Cycle 1	d1=subabs4(A1,B1)	nop	nop	nop	d2=subabs4(A2,B2)	nop	nop	nop
Cycle 2	d3=subabs4(A3,B3)	nop	sad1=dotpsu4(d1, 0x01010101)	nop	d4=subabs4(A4,B4)	nop	sad2=dotpsu4(d2, 0x01010101)	nop
Cycle 3	nop	nop	sad3=dotpsu4(d3, 0x01010101)	nop	nop	nop	sad4=dotpsu4(d4, 0x01010101)	nop
Cycle 4	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 5	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 6	sad13=sad1+sad3	nop	nop	nop	sad24=sad2+sad4	nop	nop	nop
Cycle 7	tot = sad13+sad24	nop	nop	nop	nop	nop	nop	nop

Table 2 – Pseudo assembly code for the MAE computation on the TI VLIW DSP-CPU, with quadruple 8-bit sub-word parallelism

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
	L1	S1	M1	D1	L2	S2	M2	D2
Cycle 1	d1=sub2(A1,B1)	d3=sub2(A3,B3)	nop	nop	d2=sub2(A2,B2)	d4=sub2(A4,B4)	nop	nop
Cycle 2	ad1=abs2(d1)	nop	nop	nop	ad2=abs2(d2)	nop	nop	nop
Cycle 3	ad3=abs2(d3)	nop	nop	nop	ad4=abs2(d4)	nop	nop	nop
Cycle 4	z13=add2(ad1, ad3)	nop	nop	nop	z24=add2(ad2, ad4)	nop	nop	nop
Cycle 5	nop	nop	s13=dotp2(z13, 0x00010001)	nop	nop	nop	s24=dotp2(z24, 0x00010001)	nop
Cycle 6	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 7	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 8	tot = s13 + s24	nop	nop	nop	nop	nop	nop	nop

Table 3 – Pseudo assembly code for the MAE computation on the TI VLIW DSP CPU, with dual 16-bit sub-word parallelism

tions, data routing pipes, memory interface controllers, and a 32-bit MicroBlaze™ embedded processor for serial processing and external communications.

For reference, by utilizing only 70% of the whole FPGA device again at 150 MHz, the Spartan 3A-DSP 1800A device could process up to 23 blocks in parallel (70% x 16,640 slices/508 slices/block = 23 blocks). This corresponds to a peak performance of 3,529 MMAE/s, which is at least 25 times greater than the 600-MHz TI DSP-CPU's peak performance.

### Conclusion

Using an automotive vision-based application example, we've shown how to leverage the programmable parallelism of a medium-sized, low-cost Xilinx FPGA to provide required processing performance over VLIW DSP-CPU's. Table 4 summarizes the results of our analysis.

Note that for MAE calculations on 4 x 4 blocks of 12-bit pixel data, the Spartan-3A DSP outperforms the TI TMS320DM6437 by 2 times at only one-fourth the clock speed. Furthermore, resource utilization on the FPGA is only 6%, thus allowing for the implementation of other image processing functions (in parallel if necessary) on the same device.

On the other hand, the VLIW DSP-CPU is totally busy during the SAD calculations, leaving little opportunity to conduct other simultaneous functions by consuming the available slots of the serial processor long instructions.

We were quite conservative about the FPGA estimated clock frequency (150 vs. 250 MHz), as well as the motion estimation search area (the larger the search area, greater the number of MAEs to be computed). A 30 x 30 search area, for example, would require 609 MMAE/s, far beyond the VLIW DSP-CPU capability but using only 12% of the slices on the 1800A device.

Finally, in the MAE implementation we did not use at all of the DSP48 MAC units: we estimate that a 4 x 4 block of 12-bit input data MAE could take 400 slices (782 flip-flops and 400 LUTs) and four DSP48s by replacing the 100-slice adder tree with four DSP48 units.

Thus, the Spartan-3A DSP 1800A device is well suited for vision-based applications requiring significant processing horsepower,

flexibility, and scalability such as those found on future generations of automotive driver assistance applications. ●●

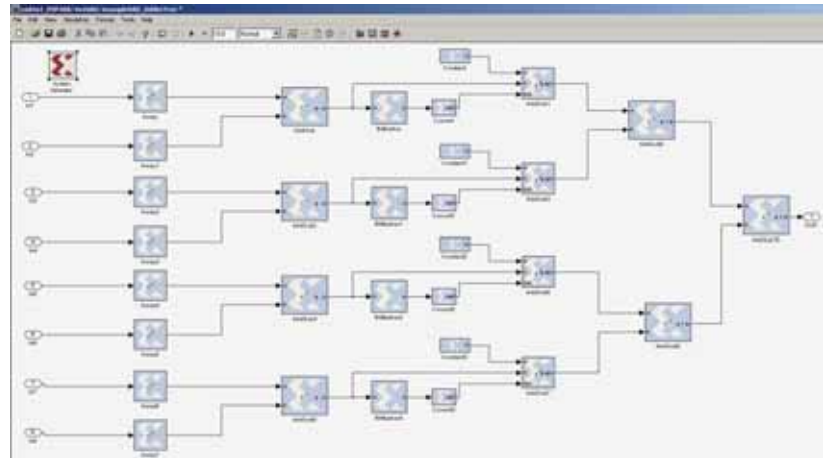


Figure 3 – System Generator for DSP scheme of the SAD computation for a quadruple of four 12-bit samples in the Spartan-3A DSP 1800 device

Device	Configuration	Clock Freq.	Performance
Example Application Requirement	752 x 480 image 4 x 4 pixel block 20 x 20 pixel search area	N/A	> 250 MMAE/s
Philips Nexperia PNX 1500 VLIW DSP-CPU	8-bit pixel depth Parallelism: Three 4 x 4 blocks	300 MHz	100 MMAE/s
TI TMSD320DM6437 VLIW DSP-CPU	8-bit pixel depth Parallelism: Three 4 x 4 blocks	600 MHz	163.64 MMAE/s
TI TMSD320DM6437 VLIW DSP-CPU	12-bit pixel depth Parallelism: Three 4 x 4 blocks	600 MHz	150 MMAE/s
Xilinx Spartan-3A DSP 1800A FPGA	12-bit pixel depth Parallelism: Two 4 x 4 blocks (approximately 6% of device resources)	150 MHz	300 MMAE/s
Xilinx Spartan-3A DSP 1800A FPGA	12-bit pixel depth Parallelism: 23 4 x 4 blocks (approximately 70% of device resources)	150 MHz	3,450 MMAE/s

Table 4 – Summary of results

#### TAKE THE NEXT STEP (Digital Edition: [www.xcellpublications.com/subscribe/](http://www.xcellpublications.com/subscribe/))

- Evaluate the System Generator for DSP design tool.
- Learn more about and purchase the Spartan-3A DSP 1800A device.
- Buy the XtremeDSP Development Platform – Spartan-3A DSP 3400A Edition.