

# Acceleration of Stereo Correlation in Verilog

Carlos Villalpando

Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

Email: carlos@jpl.nasa.gov

## Abstract

*To speed up vision processing in low speed, low power devices, embedding FPGA hardware is becoming an effective way to add processing capability. FPGAs offer the ability to flexibly add parallel and/or deeply pipelined computation to embedded processors without adding significantly to the mass and power requirements of an embedded system. This paper will discuss the JPL stereo vision system, and describe how a portion of that system was accelerated by using custom FPGA hardware to process the computationally intensive portions of JPL stereo. The architecture described takes full advantage of the ability of an FPGA to use many small computation elements in parallel. This resulted in a 16 times speedup in real hardware over using a simple linear processor to compute image correlation and disparity.*

## 1. Introduction

Image processing is becoming an important part of autonomous robotic behaviors for both terrestrial and space based rovers. A variety of sensors are used in order for the robot to navigate and interact with its surroundings. One such sensor Stereo ranging camera systems which use vision applications are an example of one of these sensors. Real time stereo ranging enables activities such as autonomous navigation and hazard avoidance. Currently, real time stereo ranging requires desktop class computers, but in many applications, power, size, and speed of available space qualified processors are a limiting factor. Processor speeds in the gigahertz range are unavailable; instead, only processors in the low hundreds of megahertz are space qualified, making real-time stereo processing impossible for any reasonable image size. Given those restrictions, the use of Field Programmable Gate Arrays (FPGA) with embedded processors has become an increasingly attractive technique for embedded processing. Integrating a sequential processor to do sequential tasks, and FPGA fabric to do vector and/or parallel processing enables the low power and high computation ability required for robotic applications.

Using the Mobility Avionics Module (MAM), a single board computer developed at JPL, we have developed a system that integrates hardware and software to do stereo vision along with motor and sensor control. The MAM consists of a custom board in a PMC form factor with a VirtexII Pro with an embedded pair of hard core PPC405 processors, Ethernet, PCI, Compact Flash, 1394, and serial I/O running a Linux based operating system.[2] Images are captured with Pt. Grey Dragonfly 1394 cameras and images are processed on the MAM.

Section 2 reviews the current JPL algorithm for stereo image processing and identifies which portions are done in the processor and which portions are accelerated by FPGA fabric. Section 3 will describe the architecture of the disparity module and the steps taken to compute correlation.

## 2. The JPL Stereo Vision System

Binocular stereo vision is a computationally expensive algorithm due to the large number of matrix operations. The current steps involved in the JPL stereo algorithm are: [1]

1. Digitize the stereo image pair.
2. Rectify the images
3. Compute image pyramids by laplacian filtering and downsampling.
4. Correlate, or measure image similarity by computing the Sum of Absolute Differences (SAD) for 7x7 windows over a fixed disparity search range.
5. Estimate disparity by finding the SAD minimum independently for each pixel.
6. Filter out bad matches by using the left-right-line-of-sight consistency check.
7. Estimate sub-pixel disparity by fitting parabolas to the three SAD values surrounding the SAD minimum and taking the disparity estimate to be the minimum of the parabola.

8. Smooth the disparity map with a 3x3 low-pass filter to reduce noise and artifacts from the sub-pixel estimation process
9. Filter out small regions (likely bad matches) by applying a blob filter that uses a threshold on the disparity gradient as the connectivity criterion.
10. Triangulate to produce the X-Y-Z coordinates at each pixel and transform to the vehicle co-ordinate frame.

Rectification determines a transformation of each image plane such that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes. Rectification reduces the correlation search space to a simple 1-D line based search. [1]

The image is filtered in order to remove the DC offset of each image. Filtering maximizes the correlation scores, producing better matches. The image is then downsampled to the working size.

Correlation is the area based search for matching features between the two stereo pairs. One of the images in the stereo pair is defined as the reference image. For each pixel in the reference image, the other image is searched along the same scan line to find a match to the reference image. As a rule of thumb, the search space is chosen to be 10% of the image width and is called the disparity search space. The difference in pixel location of the best match between the stereo pairs is called the disparity for the reference image pixel. [1]

The LRLOS consistency check is a procedure that ensures that disparities obtained by choosing best matches along the left camera lines of sight agree with those obtained by choosing matches along the right camera lines of sight. If, for a given pixel, the left and right lines of sight do not give the same match, then the match is suspect, and is discarded. [1]

In the MAM, the main processor does all steps other than steps 4, 5, 6, and 7, which are given to the FPGA. Digitizing the frames are done by the Dragonfly 1394 cameras, and fed to the main processor via the PCI bus. Future work

will migrate other steps to the FPGA portion of the MAM. For example, image downsampling and filtering in the FPGA fabric is currently complete, but not yet integrated at the time of writing this paper.

### 3. Correlation Accelerator

The most demanding computations in Stereo come in the correlation and disparity search functions. This step involves performing a Sum of Absolute Difference (SAD) of a 7x7 window in one image against multiple 7x7 windows in the other image for each pixel. For example, for 320x240 image, and searching 10% of the image width, disparity requires 32 7x7 SADs per image pixel. The next step is to find the minimum value of the 32 scores in both the Left and Right eye line of sights (LRLOS). In a sequential, non-vector processor, representative of many embedded processors, all these steps take a large amount of time to complete.

The hardware is broken up into 3 major parts as illustrated in Figure 1. They are the score generator, a score delaying function necessary for LRLOS, and a disparity computer and checker. The overall hardware architecture emphasizes parallelism and pipelining in order to complete computations quickly. Each of these modules is fully pipelined with a defined latency and an issue rate of one data value per clock. Thus, the FPGA can produce one disparity output pixel per clock.

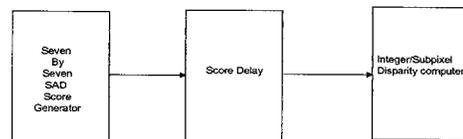


Figure 1: Disparity Engine

The SAD Score generator consists of multiple absolute difference modules in running in parallel. For each stereo image pixel fed in, the left image pixel is applied against the 32 previous right image pixels. This result is fed to 32 parallel 7 pixel rolling sum calculators,

which results in thirty two 7 column by 1 row SAD scores representing the partial sum of the disparity search space for that pixel. A rolling sum of the previous seven values is used in order to cut down on the need to re-compute score values already computed. This structure is illustrated by Figure 2.

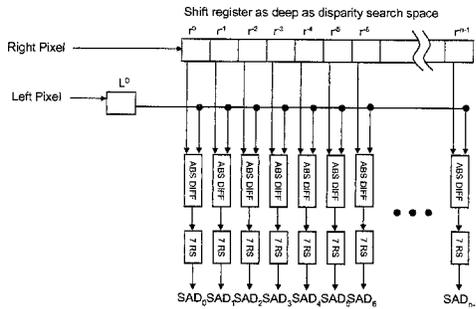


Figure 2: Seven Column by One Row Sum of Absolute Difference Calculator

The output of the 7x1 SAD is fed into the 7x7 SAD score generator. The 7x7 SAD score generator operates on a similar principle as the 7x1 SAD calculator using a 7 value rolling sum. Instead of using shift registers, the 7x7 SAD computers use the Xilinx on chip memory primitives to store the previous seven line's worth of intermediate results and the current line's final result. Thirty two values are done in parallel as illustrated in Figure 3. The result of this module is a vector of 32 values representing the SAD scores for all 32 disparities for one pixel.

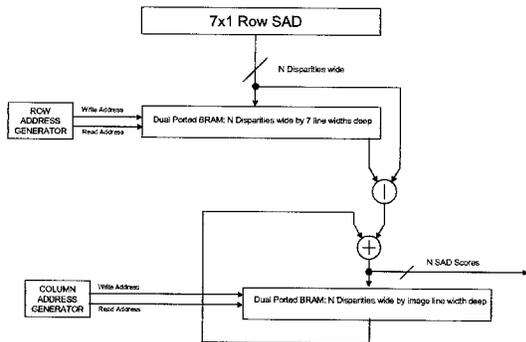


Figure 3: Seven by Seven SAD Score Generator

At this stage, we now have enough data to compute the disparity value for the current pixel; however, there are certain edge obscuration artifacts that give either degraded or

false disparity data. We would also like to have a consistency check to determine if the computed disparity is valid. In order to validate disparity, a check called the "left/right line of sight test" (LRLOS) test is applied. The primary disparity path is to check the left image pixel against 32 right image pixels. To perform the LRLOS check, instead of the left image pixel being checked against 32 right pixels, the right image pixel is checked against 32 left image pixels. In order to prevent duplicating resources and computation, it is noted that if the results of the left image disparity computation are re-arranged, the values of the right image disparity computation can be obtained as illustrated in Figure 4. The implementation of the rearranging is a simple delay based on the corresponding disparity level. The left image score set is delayed by the latency of the rearranger to keep left data synchronized with right data. The data is delayed as follows. If  $L(i)$  is the left pixel at position  $i$ , and  $D(x)$  is the SAD score at disparity  $x$  for pixel  $L(i)$ , then the score at  $L(i)D(x)$  is equivalent to  $R(i-x)D(x)$ .

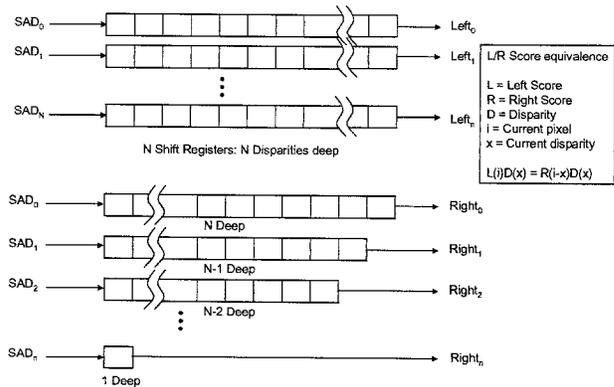


Figure 4: L/R Line of Sight Score Delay

We now have two vectors representing the left image SAD scores and the right image SAD scores. We can now compute integer and sub-pixel disparities. The left image is the primary image, and both integer and sub-pixel disparity will be computed from its data set. The right image is used only for the LRLOS check, and therefore only integer disparity is required. In addition to the LRLOS check, there are additional stereo consistency checks such as a min/max check and curve check. The overall architecture is illustrated in Figure 5.

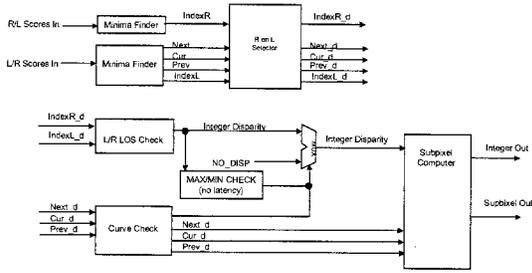


Figure 5: Integer/Subpixel Disparity Computer

The first step in computing disparity is to find the minimum value in the SAD score array. The index of the minimum value will be the integer portion of the disparity for that pixel. The value of the minimum score of the left image vector, plus the value of the next and previous indices will be used to compute the fractional portion of the disparity. The LRLOS check will compare the integer disparity of the left and right images, and if they differ by more than 1 index, it will fail the LRLOS check.

Figure 6 illustrates one step of the minima finder. This step is repeated 32 times, with each step discarding the previously checked value. Each step compares the current value with the previously found minimum value, and if the current value is smaller, the current value's index, and neighbor's values are saved. The same is done for the right image, however only the minimum value and index are saved, and only the minimum index value is output.

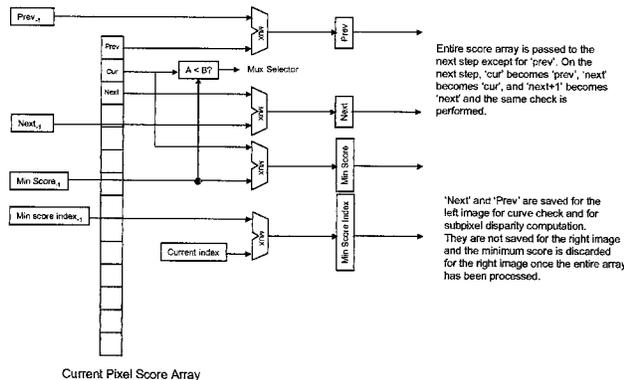


Figure 6: Minima Finder  
Repeated N disparity times

After the minimum is found, it is run through the stereo consistency checks. If any check fails, a no-disparity flag is inserted in place of the current pixel. After the checks have been performed, the data then goes to the sub-pixel computation element. Since the sub-pixel element requires a signed divide, a pipelined signed integer divider is used and the integer portion is added to the result after being delayed by the latency of the divider. The architecture is illustrated in Figure 7.

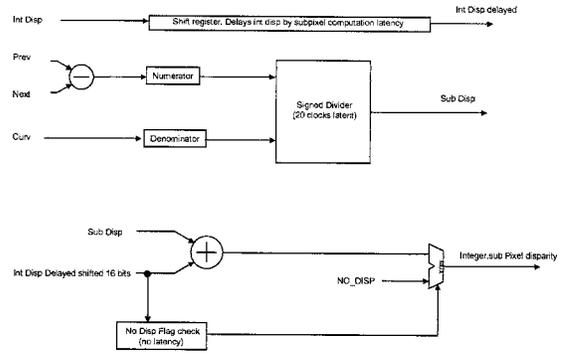


Figure 7: Subpixel Computer  
Integer + (Next-Prev/Curv)

#### 4. Results

The stereo accelerator has been implemented in the MAM and results match the software only implementation. Implementing the current subset of stereo computation in the FPGA for a 320x240 image and searching over 32 disparities required a usage of roughly 40% of a VirtexII Pro 2VP40 using 7850 slices, 9148 slice flip flops, 11087 4 input LUTS, and 68 BRAMS. The accelerator was designed for a clock speed of 100MHz, giving a theoretical max frame rate of roughly 1100fps. In the Mobility system, the module is integrated not using direct memory access, (DMA) instead it is integrated using programmed I/O, which limits throughput.

On the VirtexII Pro, with its embedded PPC405 running at 300MHz, processing stereo operating on a 320x240 image and searching across 32 disparities, processing time takes roughly 1000ms without FPGA based acceleration. According to code profiling,

approximately 200ms are taken in pre and post processing, and 800ms are taken in the correlation and disparity computation. If we move the correlation and disparity computations to hardware, the 200ms for pre and post-processing remain the same, however the correlation and disparity computation take only 50ms. That is a 16 times speedup on disparity computation over the software-only solution. We expect even faster cycle times when we implement DMA and integrating other parts of stereo in hardware.

## 5. Summary and Discussion

The project goals initially were to determine what resources and architectures were needed to do stereo processing on FPGAs. At the beginning, it was decided to do a fully parallel, highly pipelined architecture and focus on the fastest possible frame rate. However, further analyzing what is needed to scale this architecture to larger image sizes and/or disparities resulted in the determination that the resources required scaled roughly on a factor of  $O(N^3)$  with image size. While computational elements scaled linearly based on the number of disparities, the storage of intermediate products and the shift registers needed to pass data from one stage to the next quickly exhausted the available resources on a 2VP40. The parallel minima finder is also a resource hungry item. Each stage is not a simple shift register, but a series of MUXes as well as computational elements. Single bit MUXes are a limited resource on a VirtexII Pro. For example, a 640x480 by 64 disparity module requires 110% of a 2VP40 using this architecture. However, now that the requirements are understood, we can now look at other alternatives such as a running serially across disparities instead of parallel. This will reduce the resources needed for intermediate products while increasing the number of clocks needed per pixel. Other alternatives included letting the processor take care of intermediate products and instead use the FPGA as a programmable vector unit.

The disparity module is successful and was able to increase the computation speed tremendously

due to the ability to put a large amount of computation elements in hardware. Other stereo tasks can also be speed up tremendously. Work is ongoing to look at porting other stereo and vision algorithms to FPGA fabric.

## 6. Acknowledgements

This work was sponsored by the JPL Research and Technology Development program under the Mobility Avionics Module program with Kevin Watson as lead. Thanks to the JPL Machine Vision group for consultation and instruction in vision algorithms, including, but not limited to, Larry Matthies and Mark Maimone of JPL, and Steve Goldberg of Indelible Systems.

## 7. References

- [1] L. H. Matthies, Obstacle Detection for Unmanned Ground Vehicles: A Progress Report. *Robotics Research, the 7<sup>th</sup> International Symposium*. 475-486.
- [2] G. Bolotin, *et al.*, Advanced Mobility Avionics: A Reconfigurable Micro-Avionics Platform for the Future Needs of Small Planetary Rover and MicroSpacecraft. AIAA Space 2004. AIAA-2004-5997.