

Real-Time Stereo Vision on the PARTS Reconfigurable Computer

John Woodfill
Interval Research Corporation
Palo Alto, CA 94304
woodfill@interval.com

Brian Von Herzen
Rapid Prototypes, Inc.
Interval Research Corporation
brianvon@fpga.com

ABSTRACT

This paper describes a powerful, scalable, reconfigurable computer called the PARTS engine. The PARTS engine consists of 16 Xilinx 4025 FPGAs, and 16 one-megabyte SRAMs. The FPGAs are connected in a partial torus—each associated with two adjacent SRAMs. The SRAMs are tightly coupled to the FPGAs so that all the SRAMs can be accessed concurrently. The PARTS engine fits on a standard PCI card in a personal computer or workstation.

The first application implemented on the PARTS engine is a depth from stereo vision algorithm that computes 24 stereo disparities on 320 by 240 pixel images at 42 frames per second. Running at this speed, the engine is performing approximately 2.3 billion RISC-equivalent operations per second, accessing memory at a rate of 500 million bytes per second and attaining throughput of over 70 million point \times disparity measurements per second.

1. INTRODUCTION

This paper describes a powerful, scalable, reconfigurable computer called the PARTS engine. The acronym PARTS stands for “Programmable And Reconfigurable Tool Set.” Like most reconfigurable computers, the PARTS engine can be optimized to perform specialized computations efficiently, for example real-time video or audio processing.

The PARTS engine brings together several critical requirements for high-performance computations: high computational density, high memory bandwidth, and high I/O bandwidth. Arranged in a regular partial torus, computations can be easily relocated to any site in the array. This flexibility makes it easy to mix and match varied computations across the array. Using the locally distributed memory and relatively large FPGAs arranged in a uniform torus configuration, the PARTS engine has attained over 2.3 billion RISC-equivalent operations per

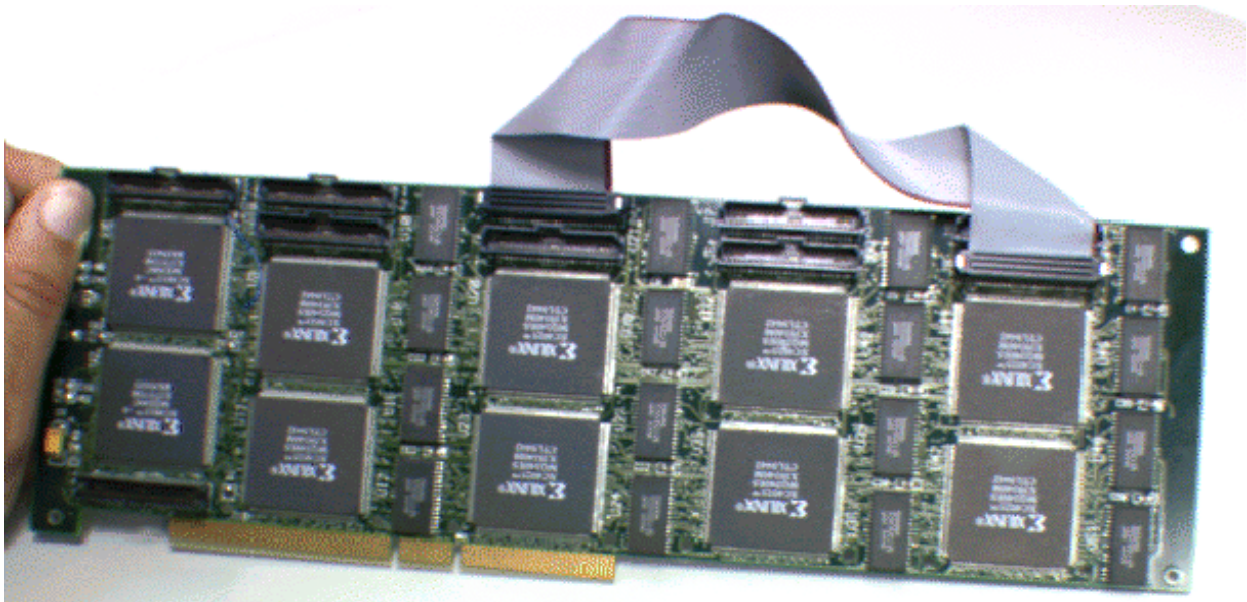


Figure 1: The PARTS engine.

second, and memory bandwidth of 500 MB/sec. With improved bus performance, the PARTS architecture is expected to achieve over 10 billion RISC-equivalent instructions and 2 GB/sec of memory access. The PARTS engine can attain I/O bandwidth of 2 GB/sec over external connectors.

The first large-scale application of the PARTS engine is an implementation of the Woodfill and Zabih census stereo-correspondence algorithm [ZabWoo94]. The stereo algorithm takes in pairs of stereo images as input, and produces images of scene depth as output. At present, the implementation performs census transforms and computes 24 stereo disparities (depth values) on 320 by 240 images at 42 frames per second.

This paper describes the architecture of the PARTS engine and introduces the recent census stereo-correspondence algorithm. Later sections show how this algorithm has been mapped onto the PARTS architecture and describe its operation.

1.1 Related work

A variety of reconfigurable engines have been developed in recent years. The PAM computer utilizes a fixed-size mesh of 16 FPGAs with global memory on a TURBOchannel adapter [Vuille95]. The Splash-2 processor uses a one-dimensional array of FPGA elements with a crossbar on a VME bus [Arnold93]. The CHAMP-1 and CHAMP-2 engines are also configured in a one-dimensional array with crossbar global interconnect on a VME Bus [Box94]. The Virtual Computer from VCC consists of a two-dimensional array of FPGAs and interconnect chips, with memory located at the edges of the array [Cass93]. Giga-Ops makes modules containing pairs of FPGAs and RAM that mount on a VL-bus card, utilizing several buses to communicate globally [Giga96].

The real-time stereo vision community has primarily relied on special purpose hardware. Nishihara has developed FPGA-based stereo vision systems on custom boards using his Laplacian-of-Gaussian Sign-Correlation algorithm [Nishih93]. These systems have tended to compute selectable, sparse depth measurements. A stereo vision algorithm based on normalized correlation was implemented on the PAM board [Fauger93]. Using the metric of points \times disparity-measures per second (PDS), this system achieved theoretical performance of 7.4 million PDS. Kanade *et al.* [Kanade96] at CMU describe a system believed to be the world's fastest in 1995. This system is composed of five VME boards, including three

custom boards built up from discrete components as well as a C40 DSP-array board, and a real-time OS board. It attains 30 hertz performance on 200 by 200 images doing sum-of-absolute-differences correlation. It uses a multi-camera technique and computes 32 disparities, performing 30 million PDS. By comparison, reconfigurable PARTS engine computing stereo depth as described below currently performs 77 million PDS.

2. ARCHITECTURE OF THE PARTS RECONFIGURABLE COMPUTER

The PARTS engine combines a homogeneous array of Xilinx FPGAs with tightly-coupled local SRAM to maximize memory bandwidth. One of the major objectives in building PARTS was to create a homogeneous computer with a minimum number of edge-conditions and heterogeneous resources. Proper handling of edge conditions usually entails special case mechanisms. Heterogeneous resources create contention and bottlenecks for those resources. By distributing resources evenly throughout the PARTS engine, for example SRAM resources, overall throughput can be improved for general-purpose computation. A homogeneous architecture also has the benefit of translation invariance, whereby an FPGA configuration can be translated to any of the FPGAs in the array.

The PARTS engines have utilized a variety of Xilinx chips ranging from the 4005H to the 4028EX. The 4028EX engines approach half a million Xilinx gates in capacity on a single PCI board. A PCI host could contain two or three PARTS boards, resulting in a million configurable Xilinx gates in a single standard personal computer.

2.1 Toroidal topology for minimal edge effects

The PARTS engine consists of a 4x4 extensible, two-dimensional array of computing elements, shown in Figure 2. In addition there is a PCI-bus interface chip, a clock control chip, and a datapath chip. All of these chips are Xilinx FPGAs. The latter three chips coordinate the flow of signals to the array. The array itself has horizontal connections that wrap around the edge of the board.

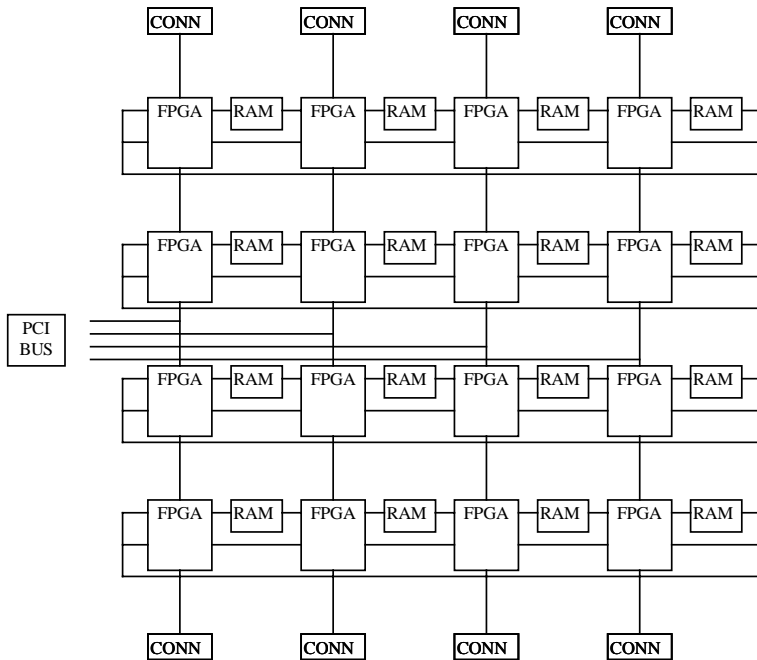


Figure 2: Architecture of the PARTS engine array.

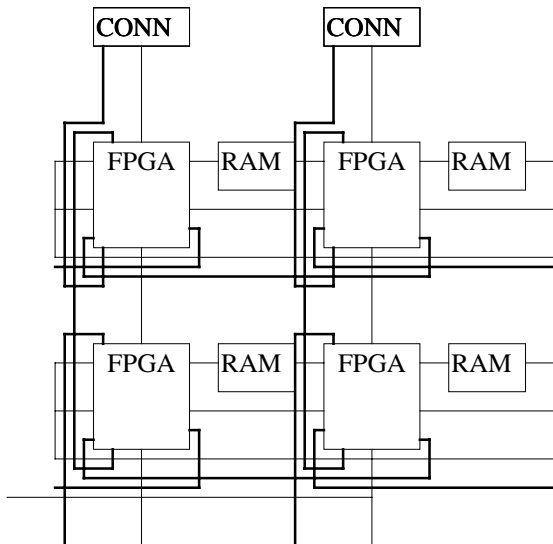


Figure 3: Superpin buses are shown between each pair of adjacent chips in lines.

The top and bottom connectors have 50-pins that are suitable for extending the array, closing the torus, or adding peripheral devices. Closing the torus involves attaching short 1-cm long jumper cables between adjacent connectors. Multiple PARTS boards can be

daisy-chained together to form larger tori, extending the PARTS engine to 4 x 8, 4 x 12, etc. Special applications can attach other ribbon cables and peripheral devices that need special controls or signal conditioning.

Current designs have pipelined data paths between array chips, including paths through jumpers, that run at 33 MHz. Limited testing has been done at speeds of up to 50 MHz with success, confirming the ability of the PARTS engine to provide high-bandwidth communication for real-time data intensive algorithms.

2.1.1 Superpin Buses

The array not only has nearest-neighbor mesh connections, but also a set of 8 “superpin” connections on each side of each computing element [HauBor94]. The superpin connections

make it possible to go from one chip to the next using a single connection between adjacent pins. Thus it is possible to construct soft “pipeline buses” or token rings or other distribution networks without using many of the routing resources on the computing elements. So far the superpins have been used mainly for local interconnections between neighboring chips. When used for local communication, superpins are slower than regular pins and can consume additional routing resources in the array chips. New applications might make use of pipelined busing using the superpin interconnections. Figure 3 shows how the superpins are configured in the toroidal topology of the PARTS engine.

2.2 Standard PCI bus

The host computer talks directly over the PCI bus to a Xilinx 4013E-2 on the PARTS machine. This chip, called PCI-32, controls all 32-bit accesses and is the master chip for the PARTS engine. The design for this chip has been adapted from the Xilinx PCI design. While the datapath chip controls data communications between the array and the PCI bus, it also connects directly to the 64-bit extension of the PCI bus. The datapath chip is programmed by the PCI-32 chip and can be reconfigured dynamically as applications require. The current design uses a non-burst interface for the PCI bus.

A burst interface is being designed to run at 33 MHz which should be capable of 130 MB/sec.

2.3 Distributed local SRAM

Graphics and video applications are large consumers of memory bandwidth. A single channel of video can consist of 30 MB/sec of data or more. If an application needs to save intermediate results, memory bandwidth requirements can be many times this number. In many applications, such as stereo vision, two or more source channels and a destination channel are needed, and the memory bandwidth requirements can grow correspondingly.

To satisfy this demand for memory bandwidth, the PARTS engine is designed so that each FPGA can control its own SRAM memory locally. Each memory is 8 bits wide and can operate at 33 MHz, providing a peak external memory bandwidth of over 500 MB/sec.

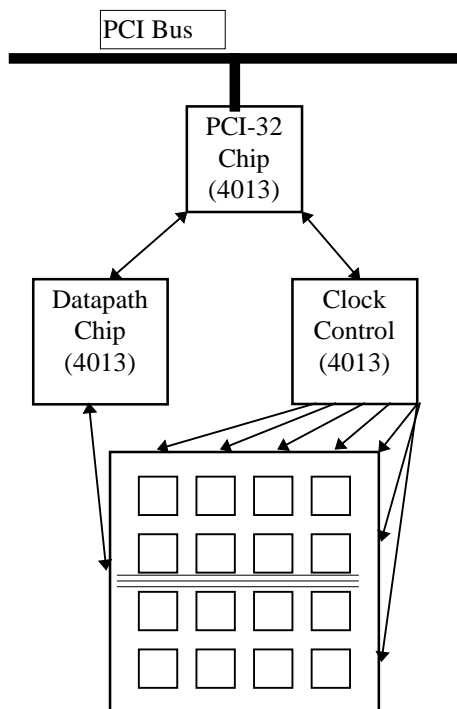


Figure 4: Block diagram of the control chips for the PARTS engine.

2.4 Bootstrapping for flexibility on PARTS engine

The PARTS engine requires a 3-level bootstrapping process to completely configure the board. This

multistage process is useful because it provides run-time flexibility in programming and accessing the array.

Referring to Figure 4, the first bootstrapping step is to program the PCI-32 chip, which controls the entire PARTS engine. This chip can be programmed either with a Xilinx Xchecker cable connected to the PARTS engine, or with a serial PROM. The Xchecker method makes it easy to modify a PCI design, download it, and test the design from a host PC. Once the design of the PCI-32 chip is complete, a serial PROM is configured to program the PCI-32 chip automatically on power-up.

The second bootstrapping step is to configure the clock control and datapath chips. The clock control chip distributes all clock and control signals to each chip in the array, while the datapath chip handles the data I/O to the array and also the 64-bit PCI bus extension. Both of these chips are programmed by the PCI-32 chip.

Once the clock control and datapath chips are configured, the third step of the bootstrapping process is to use the clock control chip to configure the rest of the array. It passes configuration data to the array directly, sending 16 bits at a time, one bit to each of the 16 array chips. Once the array chips are programmed, the clock control chip manages the clock distribution to the array, along with any strobe or control signals. This three-step bootstrapping approach maintains maximum flexibility for the PARTS engine while preserving a fast reconfiguration time of <50 msec for the array chips and <150 msec for the whole engine.

2.5 Commercial tools for configuring the PARTS engine

The development of new applications on the PARTS engine has been predominantly limited by the speed of designing configurations using the current tool set. Designs have been developed so far using commercial schematic capture tools from Viewlogic to design each FPGA, followed by place and route using the Xilinx toolset. Place-and-route times of four hours for each chip have not been uncommon.

The Xilinx floorplanner tool has been very useful for improving the performance of the PARTS control chips. The floorplanning process permits the grouping of locally-related elements together to provide for shorter communications paths and reduced routing bottlenecks. By preserving the hierarchy of the design it is possible to maintain the structure down to the layout level, and improve the resulting time delays relative to a fully

automatic placement. It is not uncommon for a floorplanned design to produce twice the speed and density of an automatically placed design.

3. THE CENSUS STEREO ALGORITHM

The initial application developed on the PARTS engine is a census stereo disparity algorithm. The census stereo disparity algorithm takes a pair of images as input and produces a dense depth map as output. Sample input images and results are shown in Figure 5. Darker objects are farther away, while bright objects are closer.

The census stereo algorithm is a novel stereo correspondence algorithm developed at Interval, and is described in some detail in [ZabWoo94] and in [Zabih94]. The problem of stereo correspondence involves taking a stereo pair of images, and determining for each pixel in one image, the corresponding pixel in the other. If stereo images are taken from two cameras oriented with parallel principal axes, perpendicular to the line that connects them, then points that are infinitely far away will appear at the same relative position in both images, while points that are nearby will have a considerable horizontal translation from one image to the other. Thus by determining correspondence between points in the two images, this shift (or horizontal disparity) can be measured, and in turn the depth of scene elements can be estimated. The images captured by the cameras are said to be in *standard* form if the following three conditions are met: the imaging surfaces of the two cameras lie on the same plane; the principal axes are perpendicular to the image planes; and the scanlines of both cameras are parallel to the line between the two cameras. Ignoring lens distortion effects, for two images in standard form, for each pixel on one image, there is a

standard form. Without loss of generality, standard form is assumed for the remainder of the analysis.

3.1 The census transform for robust correspondence

One problem that makes determining stereo correspondence difficult is that the images come from distinct cameras with distinct viewpoints, and hence corresponding regions in the two images may have differing absolute intensities resulting from distinct internal gains and biases, as well as distinct viewpoints. A second difficulty in stereo correspondence is that wherever there is a discontinuity in depth in a scene, image regions corresponding to either side of the depth discontinuity will have distinct disparities. An image window overlapping this depth discontinuity will match two half windows in the other image at different places. Assuming that the majority of pixels in such a region fall on one side of the depth discontinuity, the depth estimate should agree with the majority, and not have the result skewed by the minority opinion. The census algorithm attempts to address both of these difficulties by taking an approach to determining correspondence based on non-parametric statistics. Rather than parametrically comparing intensity values across images, the census algorithm performs a transform on the input images based on intra-image comparisons and then uses the transformed images, the results of the intra-image comparisons, to determine correspondence. This non-parametric approach ameliorates the above difficulties in that it compares the intra-image intensity structures from both images. In the case of noise pixels, or pixels coming from a minority sub-window, the discrepancies are weighted by how they change the relative orderings, and



Figure 5: A stereo pair, and a resulting depth image. The red regions denote ambiguous depth without stereo matching not at all by the magnitude of the discrepancies.

3.2 The census transform

The census transform lies at the heart of the census algorithm. The transform maps each pixel and its surrounding neighborhood into a vector of Boolean variables, each denoting the ordering relation between the center pixel and a neighboring pixel, shown in Figure 6. The dissimilarity between two such Boolean vectors can be measured using the number of elements that differ between the two vectors, i.e., the Hamming distance. Thus two pixel regions with nearly the same intensity structure will have nearly the same census transform, and the Hamming distance between their two representative census transformed values will be small. The number of pixels in the comparison set can vary from perhaps 8 to 64. As the window gets larger, more information can be taken into account, but the effects at discontinuities are worsened. The first step to computing stereo disparity using the census algorithm is to apply the census transform to the left and right input images.

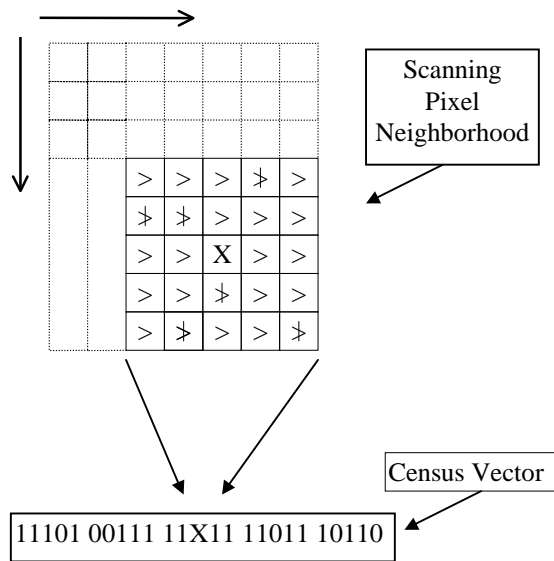


Figure 6: The census transform. For a neighborhood around each pixel (5x5 in this example), determine if the neighbor intensity is greater than the central pixel. If so, assign a 1 to the census vector, else assign a zero.

3.3 Stereo disparity computation

The goal of stereo disparity computation is to determine which pixels correspond between two images in a stereo pair. Assuming standard form, for each pixel in one image, there is a corresponding scanline in the other image on which the corresponding pixel will lie (unless it is occluded). Using this fact, the disparity for a given pixel can be estimated by determining the best matching pixel lying within a fixed search window on its associated

scanline. In order to effectively compare matching pixels using census transforms, Hamming distances are summed over a small local area. Given the transformed census images, the best matching pixel P' for a given pixel P is determined to be the one with the minimal summed Hamming distance between P and P'. The result of this computation is, for each pixel, the index, or disparity, of the best matching pixel.

4. IMPLEMENTATION OF THE CENSUS ALGORITHM

The census algorithm is well suited to the PARTS engine in that the algorithm is very uniform and extremely computationally intensive. In order to perform this large amount of computation, the algorithm is manually mapped onto the PARTS engine by evaluating the memory, the computation, and the inter-chip communication needed to perform the stereo disparity algorithm on real-time video. Debugging has been performed with canned image sequences sent over the PCI bus from disk. At present, live video sources are sent over the PCI bus with images coming from video frame-grabbers. In the future, digital cameras will send video data directly through the connectors on the top of the board. The disparity algorithm on the PARTS engine produces real-time video in which brightness corresponds to proximity of scene elements to the video cameras, as shown in Figure 5.

Summing Hamming distances requires considerable memory bandwidth, while routing census vectors requires considerable communication bandwidth. Camera pixels can be conveniently assumed to arrive at about 12.5 MHz, while the PARTS engine is capable of interfacing with its bus and external SRAM at 33 MHz. The current design uses a strobe signal for pixel data on every other clock cycle. This two-phase convention allows two data transfers and two external SRAM accesses per pixel.

4.1 Data flow

The census stereo algorithm can be implemented in a fully pipelined fashion, given the assumption of standard form. This data flow is shown in Figure 7. As pixels from the left and right cameras come into the system, they are immediately fed into two parallel census transform units. Coming out of the two census units are two streams of census vectors. Immediately the Hamming distance of these two vectors, the zero-disparity Hamming distance can be computed. By delaying one census vector an

additional cycle, and computing the Hamming distance for disparity two can be computed, and so on up to 24 disparities in the current implementation. This structure is similar to pipelined VLSI correlators [VonHer91]. Thus 24 summed Hamming distances are computed concurrently for each pixel. The final output of the computation for each pixel is the disparity with the minimum Hamming distance.

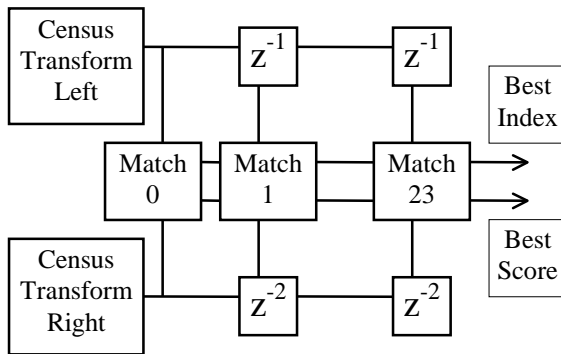


Figure 7: Dataflow for the census algorithm.

4.2 Multiplexing communication

The communication between each stage of the correlation pipeline includes two census vectors, an index, a summed Hamming distance and a couple of bits of control information. This adds up to more than the 43 pins available to provide such communication on the North-South axis. Given the regimen of two clock cycles per incoming pixel, communication can be duplexed. Hence, 86 bits can be communicated by multiplexing the outputs and inputs on these North-South 43 pin connections. This approach is similar to the method used in the Virtual Wires engines [Babb93]. This multiplexing will be simplified by the Xilinx EX chips with their strobed I/O registers and multiplexed I/O pins.

4.3 Memory use

A census transform requires several scanlines of data in order to form the census vector for a pixel in one cycle. The computation needs access to several pixels from each of these scanlines on each cycle. This translates to several bytes of memory read, and one write per transform per pixel. Assuming that an array chip uses

both adjacent SRAMS, the most bandwidth the external SRAM can provide is four bytes per pixel. For reasonably sized census transforms, this bandwidth is not enough, dictating that the census transform must be implemented using on-chip SRAM.

Summing Hamming distances requires reading and writing data every cycle. However, since switching from reading to writing external SRAM costs a clock cycle, it is not possible to switch during the active pixels in a scanline. The current design uses only eight of the FPGAs for correlation. Each correlation FPGA uses two SRAMs, one for reading, and one for writing. Every few scanlines, the roles of these memories swap.

4.4 Mapping to the PARTS architecture

Given the constraints regarding which block needs to use which memory and which chip needs to communicate what to which other chip, the choices of how to lay out the computation become quite limited. In the current configuration left and right pixel information comes in from the PCI bus on column A. The census transform is applied to one pixel of this data in the top two chips of column A. The other byte of data is shipped sideways to column C where it has the census transform performed on it in the top two chips of column C. The top chips of columns A and C spit out census data at double speed over the 16 wires available on the left and right of the top chip in column B. The top chip in column B performs 3 stages of the correlation algorithm, using SRAM on both sides of it. From here on data flows down through the rest of column B and is cabled over to the top of column D. Each chip in this path computes 3 stages of the correlation path. Finally, the bottom chip in column D passes the result to the bottom of column C. The result is passed up to the central bus, where it is read from the host processor. Figure 8 shows this datapath. A total of four array chips are used for census transforms, eight are used for correlation, two for communication, and two remain unused.

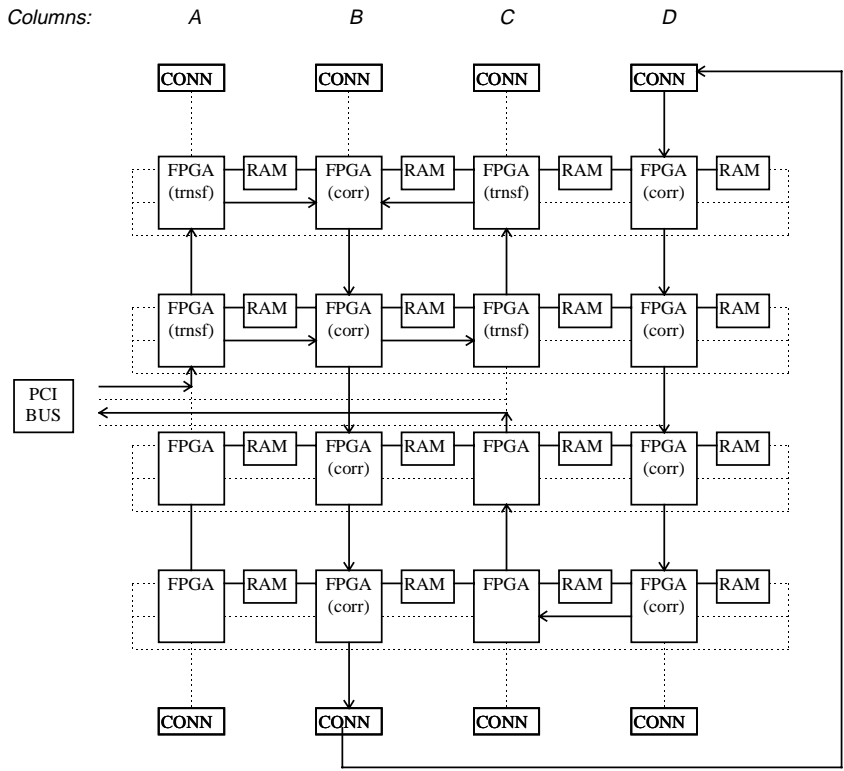


Figure 8: Dataflow for the census algorithm on the PARTS engine.

4.5 Performance

In order to characterize the amount of computation performed by the system, we use the notion of RISC-equivalent instructions, i.e., the number of arithmetic operations, loads and stores that a RISC machine would perform in order to produce the same result. Loop overhead is ignored in this analysis. For a pair of images, the stereo computation involves performing a census transform at each pixel in each image, followed by a search over a fixed search window at each pixel. The census transform involves comparing the center pixel with a given number, *K*, of other pixels surrounding it.

The search for the best disparity is restricted to *N* possible disparities for each pixel. The computation for each pixel involves pairing the transformed census pixel for one image, and each of *N* transformed pixels for the other image and computing the minimum summed Hamming distance over the *N* pairs. Our current implementation runs at 42 frames per second, which is approximately 2,320,000,000 RISC equivalent

operations per second, and about 500 MB/sec of data loaded or stored. A design incorporating a burst-mode PCI interface should achieve 225 frames per second using approximately 12,400,000,000 RISC equivalent operations per second and 2,690 MB/sec of memory loaded or stored. Higher resolution images can also be used at correspondingly lower frame rates.

4.6 Power consumption

Since the current configuration requires 43 communication pins, and up to 56 memory pins on each of the correlation chips to drive output pins at 33 MHz, the whole PARTS engine can consume considerable power. The PCI specification allows for up to 5 amps power consumption on the bus. The current design consumes 4.5 amps at 5 volts in steady-state with 24 disparities at resolution of 320 x 240 pixels.

5. CONCLUSIONS

This paper has shown that general-purpose, reconfigurable machines with toroidal topology, distributed memory and wide bandwidth I/O are capable of solving challenging applications at real-time speeds. Although actually designing and debugging software to run on contemporary FPGAs is currently tedious and difficult, such machines can provide exceptional computational power.

ACKNOWLEDGEMENTS

Bob Alkire was instrumental in developing the schematics for the PARTS engine, and in working through the architectural and design issues for the engine. Dick Shoup participated in the design and development of the board, has provided essential long-term support and encouragement for the project, and was very helpful in reviewing this paper. Andrew Singer developed the digital camera interface. Harlyn Baker gave assistance with camera and digitizing issues. Numerous others at Interval participated in discussions to improve the

architecture and suggest applications for the PARTS engine.

6. REFERENCES

[ZabWoo94] R. Zabih, and J. Woodfill, Non-parametric Local Transforms for Computing Visual Correspondence, *Proceedings of 3rd European Conference on Computer Vision*, pp. 150-158, May 1994.

[Vuille95] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, P. Boucard. Programmable Active Memories: Reconfigurable Systems Come of Age, *IEEE Transactions on VLSI Systems*, Vol. 4, No 1, pp. 56-69, March 1996.

[Arnold93] J. Arnold, D. Buell, D. Hoang, D. Pryor, N. Shirazi, and M. Thistle, The Splash-2 Processor and Applications, *Proceedings of the 1993 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, pp. 482-485, 1993.

[Box94] B. Box, Field Programmable Gate Array Based Reconfigurable Preprocessor, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Engines*, pp. 40-48, April 1994.

[Cass93] S. Casselman, Virtual Computing and the Virtual Computer, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Engines*, pp. 43-48, April 1993.

[Giga96] <http://www.gigaops.com>.

[Nishih93] H. K. Nishihara, Real-Time Stereo- and Motion-Based Figure-Ground Discrimination and Tracking using LOG Sign-Correlation, *Proceedings of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers*, pp. 95-100, November 1-3, 1993.

[Fauger93] O. Faugeras, B. Hotz, H. Mathieu, T. Vieville, Z. Zhang, P. Fua, E. Theron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy, *Real-time correlation-based stereo: algorithm, implementations and applications*, INRIA Technical Report #2013, August 1993.

[Kanade96] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, A Stereo Engine for Video-rate Dense Depth Mapping and Its New Applications, *Proceedings of Conference on Computer Vision and Pattern Recognition*, pp. 196-202, June 1996.

[HauBor94] Scott Hauck, Gaetano Borriello, Pin assignment for multi-FPGA systems, *IEEE FCCM*, p.11, April 10-13, 1994.

[ZabWoo94] R. Zabih, and J. Woodfill, Non-parametric Local Transforms for Computing Visual Correspondence, *Proceedings of 3rd European Conference on Computer Vision*, pp. 150-158, May 1994.

[Zabih94] R. Zabih, *Individuating Unknown Objects by Combining Motion and Stereo*, Ph.D. Thesis, Stanford, CA, August 1994.

[VonHer91] B. Von Herzen, VLSI Partitioning of a 2 GHz Digital Spectrometer, *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 5, pp. 768-772, May 1991.

[Babb93] J. Babb, R. Tessier, and A. Agarwal, Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators, *IEEE FPGAs for Custom Computing Engines*, pp. 142-151, April 1993.