tool allows algorithms to be quickly *tweaked* without re–programming, re–compiling, or re–loading.

MIRTIS Performance

MIRTIS is capable of performing a pipeline of user defined computations on 512x480, 8bit per pixel frames at the target rate of 30 frames per second, given that enough C40s are available in the hardware architecture. The digitization resolution (width and height of the digitized frames), the data depth (bits per pixel), and the frame rate are adjustable. However, 512x480x30=7.0 Mbytes per second is the highest data rate yet achieved due to hardware limitations (the ribbon cables connecting the C40 communication ports limit the communications performance). A large system containing 51 nodes was benchmarked at $520\frac{Mops}{sec}$ sustained (counting only useful computations) while performing a complex edge detection on live video. However, systems as small as 4–6 processors can be used for simpler applications. This is one of the most powerful features of the MIRTIS system: the flexible, open architecture provided by the model–based approach allows the system to be scaled up or down or completely re–configured for an application by simply manipulating the models.

Conclusions

MIPS has proven to be useful for hiding the complexities involved in implementing parallel image processing applications on a network of DSPs. The MIRTIS architecture allows the user to program, configure, and control a large, complex DSP network with simple, high–level interfaces. By automatically data parallelizing image processing algorithms and scaling the parallelism, real–time performance can be achieved for user–defined computational dataflows.

In the future, we plan to increase the data rates supported by the hardware by using a doubly–linked pipeline topology. This should increase the maximum data rate to $14\frac{Mbytes}{sec}$.

References

[1] M. S. Moore: "A Model Based Real Time Image Processing System", Final report for the 1994 USAF-RDL summer research program, AFOSR Contract F49620-93-C-0063, September 1994.

[2] P. A. Laplante, "Issues in Real-Time Image Processing," Proceedings of the 1993 IEEE Systems, Man, and Cybernetics Conference, Le Touquet, France, October, 1993, pp. 323-326.

[3] B. Abbott, T. Bapty, C. Biegl, G. Karsai, and J. Sztipanovits: "Model–Based Software Synthesis", IEEE Software, May 1993.

[4] G. Karsai: "A Configurable Visual Programming Environment", IEEE Computer, March 1995.

[5] B. Abbott and A. Ledeczi: "TICK: TI TMS320C40 Utility Program", Proceedings of the International Conference on Signal Processing Applications and Technology, Oct. 1994.

tems [4]. By using domain specific models and interpreters, MGA allows the domain experts to specify a system in familiar terms and provides an insulation from the underlying implementation details.

MIRTIS Architecture

We have applied the MGA to the real–time image processing domain, creating a real–time image processing system with high–level programming and user interfaces. MIRTIS generates split–and–merge PCT applications for a parallel C40 network by building the PCT schedules and DMA programs automatically, The complexities of communications (DMA programming), sub–task decomposition, sub–task allocation, and scheduling are hidden from the user. The components of the MIRTIS system can be seen in figure 2.

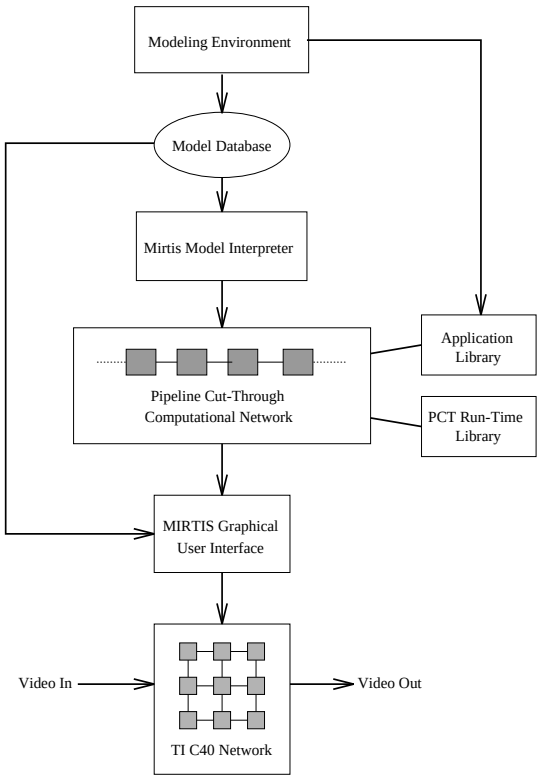

Figure 3: The MIRTIS Architecture

MIRTIS Modeling

The MIRTIS modeling paradigm includes three aspects, or views, of the system. (1) computation models, (2) hardware models, and (3) performance goal models. *Computation models* describe which algorithms are to be performed, and in what order. The computation model is represented graphically in a graph dataflow style: blocks represent computations and inter–connecting lines represent communication. *Hardware models* describe the C40 network, including information about each node (memory configuration, etc.), and about the interconnection topology. The hardware model is acquired automatically using the *TICK* [5] network detection tool. *Performance goal models* allow the specification of the target throughput and latency which is needed by the application. The three aspect models are viewed and manipulated in the *modeling environment* (see figure 3). The modeling environment is based on *XVPE*, a configurable visual programming environment developed at Vanderbilt University [4] which runs under **X** and uses the **obst** object oriented database package for model storage. XVPE has been ported to **linux**, which makes a PC a convenient platform for modeling.

Model Interpreter

The MIRTIS model interpreter transforms the aspect models into a PCT computational network which will implement the application with the given performance, assuming that enough processors are available in the C40 network.

Application and Run–Time Libraries

The *image processing application library* contains the code which actually implements the algorithms. Because of the simplicity of the split–and–merge model, it is possible to utilize existing C40 image processing libraries.

The *PCT run–time library* provides a scheduler, a connection/memory manager, and other run–time facilities. The C40 network is loaded with the *TICK* network loader, which also provides a network debugger.

Mirtis GUI

The *MIRTIS Graphical User Interface* is used for loading and interacting with the C40 network. The GUI has a dynamic command interface to the PCT network which allows the user to adjust parameters of the running computations using slider bars, and other interactive graphical widgets. For example, a user can interactively adjust a convolution kernel while viewing the output of the running system. The changes in the computations are apparent in the output almost immediately. This

technique which is applicable to image processing is the *split–and–merge* programming model. In this technique, the input data is *split* into N pieces, which are distributed to several *worker* processors and processed concurrently. The results are then *merged* to form the output. See figure 1. Since each worker computes $\frac{1}{N}th$ of the result, there is a computational speedup of at most N.

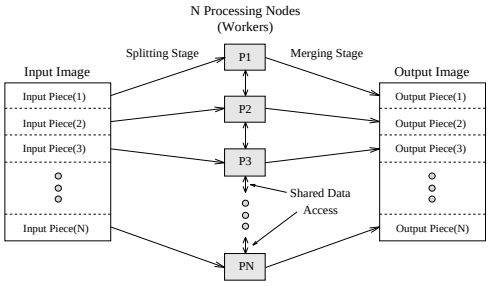

Figure 1: The Split–and–Merge Paradigm

Pipeline Cut–Through

PCT is a communications technique which implements the communications necessary for split–and–merge processing on a pipeline connected network of C40s. PCT programs the autonomous Direct Memory Access (DMA) co–processors of the C40 to automatically split and merge the data stream without CPU intervention, leaving the 40Mflop CPUs free to perform the image processing calculations. The communication and computation processes are synchronized once per image through DMA interrupts, but otherwise run independently.

The reason that the split–and–merge communication can be done with very little overhead on a pipeline hardware topology is that the C40 DMAs can be programmed to transfer from a communication port to memory, from memory to a port, or directly from a port to another port. The port to port communication makes the C40 a virtual wire connecting its two neighbors. Keeping in mind that the DMAs operate independently of the CPU, it is possible then for the data stream to *cut through* one or more C40s in the pipeline, potentially allowing any two processors in the network to communicate. Since the DMAs can re–program themselves upon completion of a transfer, communication state table can be implemented which cause pipeline connected C40s to split and merge an image sequence datastream.

In figure 2, a group of 3 C40s is shown which are programmed to split images into 3 groups of rows and merge the sub–image outputs into the output datastream. Note that in each state one of the C40s is effectively reading from the input stream and writing to the output stream by cutting through the other 2 processors. The input and output streams flow un–interrupted as long as each worker processor is finished computing its piece of the image before its turn to communicate again. In figure 2 the horizontal arrows between memory segments indicate computation. Note that in the given example, the processor communicating is not computing, which causes the speedup to be at most $N - 1 = 2$ instead of $N = 3$. This is easily alleviated by double buffering, but for simplicity, the double buffered case was not shown.

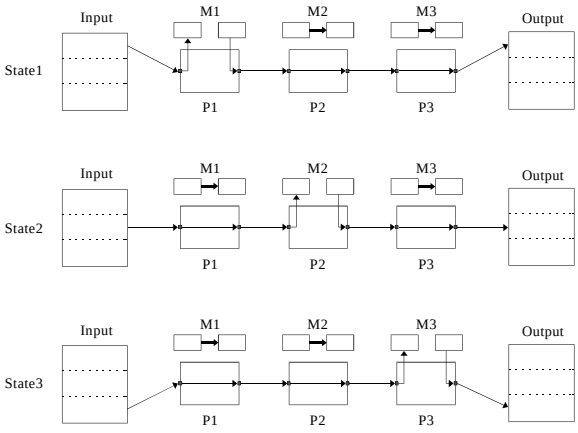

Figure 2: The PCT Communications Scheme

The way that the PCT DMAs are programmed depends upon the size of the data, the way that the data is to be split, the number of processors to be used, and the way that the algorithm accesses the data. Building these programs by hand for each application is not feasible, so a way of automatically generating the DMA programs is needed.

Model Integrated Synthesis

Model Integrated Program Synthesis (MIPS) is a method of managing complexity in large scale engineering systems [3]. Applications are generated from *models*, which specify the system in terms of a *paradigm*, or language. The Multigraph Architecture (MGA) is a MIPS architecture developed at Vanderbilt which provides a framework and tools for building domain specific models and transforming the models into executable sys-

this system.

Real–Time Image Processing

Image processing is the mathematical manipulation of digital images by computers. It may be used to enhance the quality of or extract information from the visual data. Examples of image processing algorithms are look–up table operations, 2–D convolution, and morphological filtering. Image processing algorithms are particularly compute intensive due to the fact that images are very large datasets. Image sequences (digitized video) are even larger data sets, so processing video requires a huge number of computations. For example, a common video camera produces $30\frac{frames}{sec}$, and each frame may be digitized into $512x480x8bit$ pixels. The data–size resulting from digitizing 1 second of video at this resolution is $7.0Mbytes$. Since a $5x5$ convolution operation requires 25 multiplies and 25 adds for each pixel, filtering this 1 second of video using a $5x5$ convolution will take 370 $Million$ mathematical operations.

A *real–time* system is one which must produce outputs which are not only numerically correct, but that also meet meet timing constraints. A real–time image processing system obtains digitized video data from sensors and computes the correct response in compliance with specified temporal constraints. The relevant constraints for image processing are *throughput* and *latency*. Throughput is the rate at which outputs are produced, and latency is the time required to compute one entire frame. Either or both types of temporal constraints may be needed for a particular real–time imaging application, so a general system must have methods of controlling throughput *and* latency. For example, in real–time video enhancement the throughput is most important, since it is critical that all of the data be processed with no down–sampling. However, in robotics latency is most important. The determination of the robot's action must be made as soon as possible after the visual sensing, and the next sensing cannot occur before the action has been made.

Example Applications

In addition to the video enhancement application which motivated this effort, there are several contemporary, specialized applications which require real–time image processing, as discussed by Laplante in [2]. These include remote command and control, broadcast and multi–media communications, high speed modeling, rapid image identification, medical imaging, and robotics.

Hardware Approaches

Traditionally, real–time image processing has been done using hardware specially designed to implement the operations. For instance, a common configuration is to put one or several boards into a PC or workstation host, each which has certain real–time capabilities built in. By changing the inter–connections between the boards, and setting hardware registers, a sequence of operations can be performed on live video. These types of systems are a cost effective solution for some applications, and many different such systems are on the market. For example, AEDC has used a Quantex QX–7 system for video enhancement for several years.

Limitations of Specialized Hardware

There are several limitations to using specialized hardware for real–time image processing.

- The largest limitation is that the systems are not *end–user programmable*, in the sense that user–defined functionality cannot be added to the system. (Some systems are programmable, but the performance may suffer dramatically when user defined operations are used.)
- Specialized hardware solutions are *expensive*, both in hardware and training costs.
- The systems are not *extensible*, or *scalable*, in that the amount of computations which can be performed cannot always be increased by adding more hardware.

At AEDC, a system without the limitations of specialized hardware is needed. We determined that by taking advantage of the natural data parallelism of images and high performance parallel DSP hardware, such as the C40, a real–time image processing machine could be built which is programmable, scalable, and cost–effective.

Split–and–Merge Parallelism

Because image processing algorithms usually perform the same, relatively simple computations for each pixel in the image, they are easily data parallelizable. A simple data parallel programming

A DSP–BASED REAL–TIME IMAGE PROCESSING SYSTEM *

Michael S. Moore
Measurement and Computing Systems Laboratory
Department of Electrical & Computer Engineering
Vanderbilt University
msm@vuse.vanderbilt.edu

Abstract

A real–time image processing system has been developed which is based on the Texas Instruments TMS320C40 DSP (C40). The system employs model–integrated program synthesis to automatically data parallelize image processing algorithms, scale the parallelism to meet the performance specifications of the application, and realize the parallel implementation on a network of C40s. The underlying parallelism is transparent, so the user is insulated from the complexities normally involved in programming parallel machines. This makes parallel DSP hardware a feasible platform for real time–image processing.

Introduction

Because of the huge data rates and the computational power required to process digital images, real–time image processing systems have traditionally been based on specialized hardware. These systems, though useful, have had the drawback of being limited in programmability and scalability. The types of computations they can do are fixed. Due to the recent gains in VLSI and parallel processing technology, however, the means are now available to build much more programmable and scalable image processing machines. By exploiting the natural parallelism of image processing algorithms, and using networks of high performance DSPs, very powerful real–time image processing solutions can now be built.

A real–time parallel image processing system is being developed through a joint effort between Vanderbilt University and Arnold Engineering Development Center (AEDC). The goal of the project, which began during the 1994 AFOSR Summer Research Program [1], is to create a machine and development environment to be used for processing video sequences on–line (in real–time) during turbine engine tests, reducing video data off–line, and for interactively experimenting with image processing algorithms. Three major requirements of the application domain are

- **Affordable programmability.** To reduce programming expense, the system must be programmable by the end users, which may have no knowledge of parallel programming or real–time systems. A high–level parallel programming interface is necessary.
- **Real–time performance.** User defined computations must be performed in live video sequences at rates up to 30 frames per second (512x480x8bit frames).
- **Scalability/Flexibility.** The system must be easily scaled up, scaled down, or completely reconfigured to meet the requirements of a particular application.

This paper presents the Model–Integrated Real–Time Imaging System (MIRTIS). MIRTIS employs model–based techniques [2][3] to transparently and automatically parallelize image processing computations, which it executes on a parallel DSP hardware platform. System models are used to manage the complexity of the underlying implementation, mask the underlying parallelism, and simplify the system's use. The MIRTIS hardware platform is a network of Texas Instruments TMS320C40 DSPs (C40s). Real–time execution is achieved by scaling the data parallelism in the computational dataflow to the appropriate level to reach the target throughput. The scaling is done with virtually no overhead via a communications scheme called Pipeline Cut–Through (PCT). Following is a description of our on–going work on

*This work was supported by the AFOSR/AFMC, United States Air Force, contract number F49620-94-C-0076.