Multiprocessor Implementation of a Face Detection System

Sankalita Saha[1], Neal K. Bambha[2] and Shuvra S. Bhattacharyya[1],
[1]Department of Electrical and Computer Engineering, University of Maryland, College Park and
[2]US Army Research Laboratory, Adelphi, Maryland
ssaha@eng.umd.edu, nbambha@arl.army.mil, and ssb@eng.umd.edu

Introduction

Face detection and recognition research has attracted great attention in recent years. Automatic face detection has great potential in a large array of application areas; including banking and security system access control, video surveillance, and multimedia information retrieval, etc. Face detection is a complex problem characterized by computation- and memory- intensive operations. However, many face detection algorithms have inherent parallelism in them − both data as well as instruction-level − which when properly exploited can yield significant performance gains. Proper exploitation of this parallelism is not always easy, however, since there are significant memory operations that can overshadow the performance gain obtained by parallelization. In this work we explore various target multiprocessor architectures for this important application and study the resulting performance-area trade-offs.

Face Detection

Face detection research has been an active area of research for the past few decades. There are several approaches that make use of shape and/or intensity distribution on the face. In this work, we use a shape-based approach as proposed by Moon et al [1] where a face is assumed to be an ellipse. This method models the cross-section of the shape (ellipse) boundary as a step function. Moon [1] proves that the derivative of a double exponential (DODE) function is the optimal one-dimensional step edge operator, which minimizes both noise power and mean squared error between the input and the filter output. The operator for detecting faces is derived by extending the DODE filter along the boundary of the ellipse. The probability of the presence of a face at a given position is estimated by accumulating the filter responses at the centre of the ellipse. At the implementation level, this reduces to finding out correlations between a set of ellipse shaped masks with the image in which a face is to be detected. Figure 1 shows the complete flow of the employed face detection algorithm. In this work, we assumed that the number of masks and the mask sizes are fixed, where, as mentioned before. For operational value, the implementation should be able to handle variability in size of the faces as that information is not usually available a-priori. We handled this by creating several elliptical masks of varying sizes. This entails building of a large mask set and consequently a large number of correlation computations.

Target Architectures

We examine three different architectures to explore different area and performance results. The first target is a general-purpose distributed-memory multiprocessor system. The processors are a combination of IBM Netfinity and Dell Poweredge hardware nodes. Specifically, each node is a dual processor PIII-550 (2xPIII-550Mhz) with 1GB memory and 2x18GB of disk capacity. Each node has a 1Gigabit link to the other nodes.

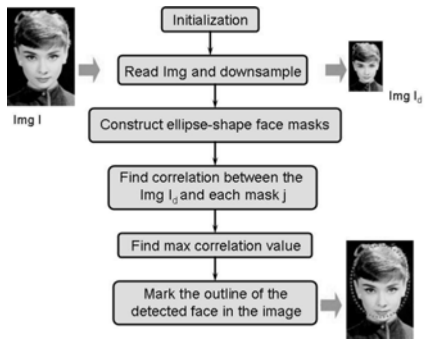
Initialization
Read Img and downsample
Img I
Img I_d
Construct ellipse-shape face masks
Find correlation between the Img I_d and each mask j
Find max correlation value
Mark the outline of the detected face in the image

Figure 1: The flow of the face detection algorithm

The second architecture is a full hardware implementation, utilizing multiple processing elements in hardware that each perform a single task (a correlation function). The target consists of a reconfigurable system on chip − Xilinx's ML310 board, which contains a Virtex II Pro field programmable gate array (FPGA) device. It also includes on-chip and off-chip memory resources. Access to the off-chip memory is assumed to be through a shared bus. On-chip memory access can be performed through a shared bus or via DMA.

The third architecture is an embedded multiprocessor design consisting of multiple Xilinx Microblaze soft cores, interconnect infrastructure, and hardware co-processing elements. In this architecture, there is a choice of executing functions in either hardware or software.

Distributed-Memory Multiprocessor Implementation

Profiling results show that computing the correlation between the image and the mask set (mask correlation) is computationally the most expensive operation. However, this operation is inherently parallelizable. The mask set can be divided into subsets and each subset can be handled by a single processor independent from the rest. Thus the main tasks are building of masks (BM*i*) and finding correlation between masks and image to find the best match (PE*i*). The processor task assignment is shown in Figure 2 for the case of 3 processors for the correlation operation. Besides using multiple processors for the correlation operation, we use a separate processor to handle the required I/O operations (reading of image RI and finalizing result RI).

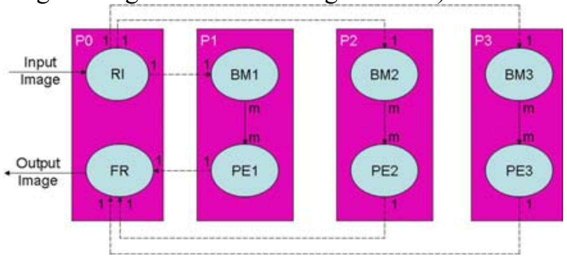

Figure 2: Assignment of tasks to processors for the case of 4 processors.

The experiments were run on configurations of 4, 5, and 6 processors and 2 test benches were used. These test benches, respectively, involved 126 and 114 masks; each of size 121x191 and 127x183 pixels and image of size 128x192 pixels. Sample results for the case of 5 processors is shown in table 1.

Table 1: Execution times for 1 frame for 5 processors and 2 testbenches for MPI implementation

Processor id	Testbench 1 (secs)	Testbench 2 (secs)
0	1.544	1.356
1	1.427	1.31
2	1.434	1.302
3	1.433	1.315
4	1.438	1.327

FPGA Hardware Implementation

Figure 3 shows the high-level architecture model for the face detection system. Our architecture consists of multiple processing elements (PEs) that can concurrently execute multiple instances of the correlation function and thereby process masks simultaneously. From the face detection algorithm, shown in Figure 1, a large set of masks is created with which the image frame is compared. This mask set can be created *offline* and moved to external memory. Each PE reads masks with which it performs the correlation operation from the external memory one at a time. Also no two PE uses the same mask. Thus, each PE requires at least one mask available on chip. To facilitate faster processing, we stored 2 masks per PE at a time; when a PE operates on the first mask the second one is read. Thus, this model requires us to have multiple masks and copies of the frame (concurrent access by the PEs) available on the chip, e.g., 3 PEs require 3 frame copies present on chip.
To minimize the rate of memory accesses and hence power consumption, we partitioned the image into stripes and processed the image one stripe at a time where a stripe is defined to be the minimum size of the image that can be processed on one pass. We run our mask set on a given stripe of the image and find the maximum correlation value, repeat the process for the next stripe, and continue in this manner until we have exhausted all the stripes, and hence the image. For a set of *N* masks and *n* PEs (i.e., masks can be processed simultaneously), it will take *ceil(N/n)* processing passes to cover all masks for a single stripe. Here, *ceil* denotes the greatest integer that is greater than or equal to the rational number.

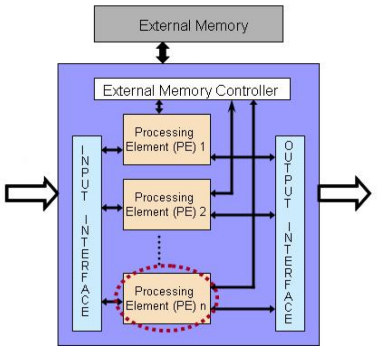
External Memory
External Memory Controller
Processing Element (PE) 1
Processing Element (PE) 2
Processing Element (PE) n
INPUT INTERFACE
OUTPUT INTERFACE

Figure 3: Face detection system architecture

To keep the design space manageable, we fixed the frame size (frame size = 240×320, stripe size = 65×160), and varied other the number of PEs (i.e., *n*), the steps — the granularity at which the image is correlated with a mask, and the fine-grain parallelism (the number of additional multiplications done simultaneously in each PE) up to the permissible HW limits. The performance results for the various design cases are shown in table 2.

Table 2: Execution times for 1 frame for different design parameters for FPGA implementation

n	Degree of parallelism	Execution time (in ms)	
		Steps = 2	Steps = 4
6	0	697	205
1	20	227	79
2	10	227	79
3	6	249	85
4	5	227	79
5	4	227	79
6	3	249	85

References

[1] H. Moon, R. Chellappa, and A. Rosenfeld, "Optimal edge-based shape detection", IEEE Transactions on Image Processing, 11: 1209–1227, 2002.

[2] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "A message passing standard for MPP and workstations", Communications of the ACM, 39(7):84–90, 1996.