**TALKER DETECTION
FOR MULTIMEDIA CONFERENCING**

by

HAN ZHAO, B.ENG., M. SC.

A thesis submitted to the Faculty of Graduate Studies and Research in
partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

October, 2002

Copyright © 2002, Han Zhao

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-79740-6

Canada

The undersigned recommend to the Faculty of Graduate
Studies and Research, acceptance of the thesis

**Talker Detection
for Multimedia Conferencing**

submitted by Han Zhao, B. Eng., M. Sc.

in partial fulfillment of the requirements for the degree of
Master of Applied Science

Thesis Supervisor, Professor John P. Knight

Thesis Supervisor, Professor Rafik A. Goubran

Chair, Professor Rafik A. Goubran
Department of Systems and Computer Engineering

Carleton University

October 2002

Abstract

This thesis deals with multimedia conferencing where a microphone array and a video camera are used. A talker can be coarsely located using the microphone array. The objective of this thesis is to use the speaker's mouth movement to locate the talker within video frames.

In this thesis, an algorithmic model is proposed and developed to detect mouth motion and then pinpoint the talker. It is to be integrated into the system under research by the Carleton Signal Processing Group. A VLSI architecture for motion detection has been developed to meet the real-time speed and storage requirements. This ASIC architecture is computationally highly parallel, and is scalable to process the larger video frame sizes which follow the ITU-R BT. 601-4. Based on this architecture, a gate-level ASIC was designed and verified utilizing a 0.35 μm CMOS standard cell library. The ASIC concentrates large quantities of raw video data into a form manageable by a general purpose processor. Running at a clock rate of 100 MHz, the ASIC can achieve 0.8 ms latency between input frames (320*240 pixels at 30 frames/second) and the finished outputs. This research also investigated the correlation between the audio and video data captured simultaneously from the talker.

Acknowledgements

I am grateful to my supervisors, Professor John P. Knight and Professor Rafik A. Goubran. Thanks to Dr. Knight for his insight, “hall-way talking” and considerable time that was spent reading the many drafts of my thesis, and most of all for his patience and coaching. Thanks to Dr. Goubran for his diligent guidance during the course of research, excellent management and collaboration with industry partners.

I gratefully acknowledge the funding from Mitel Networks Inc. and the Ontario Ministry of Education and Training for an Ontario Graduate Scholarship in Science and Technology. Thanks to the Mitel staff, especially to Graham Thompson, Dieter Schulz, Ramiro Liscano and Michael Tetelbaum for their valuable discussions and suggestions. I also would like to thank Malcolm Stewart, Michel Beshara and Richard Beriault of Lumic Electronics for the information on camera image sensors. I would like to thank all of my friends for their helpful aids. Special thanks to David Lo for close collaboration, discussion and help in the course of research for multimedia conferencing. Thanks to Shing-Fat Fred Ma and Gord Allan for their intelligent suggestions and generous help in the ASIC design.

I would like to extend my appreciation to Ms. Christine Lariviere for her proof-reading of the manuscript, and to Danny Lemay and Jacques Lemieux for their supports on various design tools and computer systems.

I am indebted to my parents and my family. Without their love, support and education, I would not have any work done.

Table of Contents

Title Page	i
Acceptance	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
Glossary & Acronyms	xi
Chapter 1 Introduction	1
1.1 Motivation.....	1
1.2 Problem Definition.....	3
1.3 Thesis Outline	4
Chapter 2 Background	5
2.1 Audio & Video Joint Processing and Analysis	5
2.2 Blueprint of Our System for Multimedia Conferencing.....	7
2.3 Object Localization by Audio Processing.....	9
2.4 Object Localization by Video Processing.....	11
Chapter 3 Exploration of Correlation between Audio and Video	13
3.1 Definition of Motion Area Ratio.....	13
3.2 Exploration of Audio-Visual Correlation of the Speaker	17
3.3 Overview of Experimental Bench.....	23
Chapter 4 ASIC Based Motion Detection & Analysis	25
4.1 Overview of the Motion Detection and Analysis.....	25
4.2 VLSI with Differential Method and Correlation Method	27
4.3 ASIC Based Motion Detection & Analysis Method.....	32

4.4	Differential Motion Detection.....	35
4.5	Motion Analysis with VLSI.....	45
4.6	Region Analysis and Examination.....	52
4.7	Choice of Quantization Levels to Form Binary Discrete Histogram.....	57
Chapter 5 ASIC Implementation and Verification		59
5.1	Subsystem Design Flow and ASIC Design Flow	59
5.2	Hardware Description Language (HDL) and Synthesis	63
5.3	ASIC Architecture	67
5.4	ASIC Verification.....	76
5.5	Discussion of Other Design Issues	82
Chapter 6 Calibration of Experimental Bench		84
6.1	Calibration of Timing.....	84
Chapter 7 Summary and Future Work		89
7.1	Summary and Novel Concepts.....	89
7.2	Future Work	92
References		94
Appendix A:Verilog Behavioral Description of ASIC Modules		98
Appendix B:Sample of ASIC Simulation Data Output of Motion-region Frame		107
Appendix C:Selection of Schematic		109

List of Tables

Table 4.1:	Approximate data rates for two video options	29
Table 5.1:	The delay of the critical path group	81

List of Figures

Figure 1.1:	Illustration of strong sound reflection in multimedia conferencing. This may cause the microphone array to assume that the speaker is positioned at the origin of the reflection, and then cause the camera to capture other persons instead of the speaker [HE00].....	2
Figure 2.1:	Blueprint of multimedia system research. Shaded blocks are the major research interests discussed in this thesis.....	8
Figure 2.2:	Sector division in coarse localization by microphone array	10
Figure 3.1:	Illustration of motion pixels, motion-feature frame, object of interest and area of interest.....	14
Figure 3.2:	Motion-feature frame with the object of interest and the area of interest. The black area contains motion pixels, the white ones contain non-motion pixels; the solid rectangle is the object of interest, the dashed rectangle is the area of interest.	17
Figure 3.3:	Spatial temporal video analysis	18
Figure 3.4:	Amplitude of voice ‘pm’ vs. time, extracted from ‘pm’ audio record	20
Figure 3.5:	Motion area ratio vs. time, extracted from ‘pm’ visual record	20
Figure 3.6:	Amplitude of speech voice ‘hello’ vs. time, extracted from ‘hello’ audio-visual record.....	22
Figure 3.7:	Motion area ratio vs. time, extracted from ‘hello’ audio-visual record	22
Figure 3.8:	Diagram of the experimental bench.....	24
Figure 4.1:	General diagram of ASIC- based digital signal processing flow for motion detection and analysis	33
Figure 4.2:	A more detailed ASIC-based digital signal processing for motion detection and analysis.....	34
Figure 4.3:	The functional steps of motion detection.....	36
Figure 4.4:	The first frame in a pair of two consecutive color RGB frames. It is not in its original resolution and size, and printed here in black and white. It has the approximate effect of a digital image Y frame.	37
Figure 4.5:	The second frame in a pair of two consecutive frames. Compared to the first frame, there are mouth motion and some facial motion. Not in its original resolution and size.	38
Figure 4.6:	The distribution of pixel value in the difference frame. The image size is 320*240 pixels. The small ellipses show the number of points at the integer pixel value. Most of the points are crowded near the X-axis. One point is a circle at 7*104. The envelope was drawn slightly higher than it should be along the X-axis, to make it visible.	39
Figure 4.7:	A thresholded difference frame.....	42
Figure 4.8:	A pair of motion frames during a talking period	43

Figure 4.9:	A pair of motion frames showing eye clipping during a non-talking period.	43
Figure 4.10:	A pair of motion frames with the speaker breathing and in the presence of strong background noise during a non-talking period	44
Figure 4.11:	A pair of motion frames showing the effect of a speaker's body motion ..	44
Figure 4.12:	General diagram for motion analysis of the motion-feature frame.....	47
Figure 4.13:	Two dimensional continuous histogram. There are 16 potential region boxes. Two of them are the strongest.....	48
Figure 4.14:	Improved functional diagram for motion analysis of the motion-feature frame	50
Figure 4.15:	Illustration of the discrete aggregate histogram.....	51
Figure 4.16:	The illustration of two dimensional discrete aggregate histograms. They are converted in size, from 30 to 240 on the Y-axis, from 40 to 320 on the X-axis, and projected back on the motion-feature frame.....	53
Figure 4.17:	Illustration of two dimensional quantized histogram.....	55
Figure 4.18:	Illustration of motion region after motion detection and analysis with only two potential motion region boxes.....	56
Figure 4.19:	Illustration of the choice of quantization level.....	58
Figure 5.1:	System design and development flow [BA00]	60
Figure 5.2:	ASIC design and verification flow for front-end and back-end design. The front-end is in solid boxes while the back-end is in dashed boxes, edited from [CMC99].	62
Figure 5.3:	General diagram of the ASIC for motion detection and processing, related to the motion subsystem of the blueprint of our system under research presented in chapter 2.	68
Figure 5.4:	Motion detection and processing block diagram	71
Figure 5.5:	Diagram of motion processing element unit.....	72
Figure 5.6:	Diagram of the motion analysis unit. Refer to figure 5.9 for the graphic illustration of processing results.	73
Figure 5.7:	Illustration of the timing of critical signals.....	74
Figure 5.8:	ASIC verification model	77
Figure 5.9:	Graphic representation of ASIC output data for verification. Refer to figure 5.6 for ASIC architecture.....	78
Figure 5.10:	High level evaluation of ASIC processing outputs projected on the corresponding digital image Y frame, which is an input to ASIC. This agrees exactly with the results of the algorithmic model.....	79
Figure 6.1:	Calibration of the experimental bench.....	86
Figure 6.2:	Beeping sound wave at the falling edge at around 1.25 seconds.....	87
Figure 6.3:	The corresponding video frame of the falling edge sound wave, captured just before the falling edge at 1.066 second; here the audio had a approximate delay of 0.15 second.	88

Figure 6.4: The corresponding video frame of the falling edge sound wave, captured just after the falling edge at 1.099 second; here the audio had a approximate delay of 0.182 second.88

Glossary & Acronyms

Glossary

area of interest: an observing window set for motion analysis, which is usually a little larger than the *object of interest*. Defined in section 3.1.

control unit: an ASIC circuit module implemented to control the datapath of signal processing for motion detection and analysis. Defined in section 5.3.

digital image Y frame: a frame of YCbCr color space converted from RGB color space. Defined in section 4.3.

difference frame: a frame generated by inter-frame differencing which contains the inter-frame motion information. Defined in section 4.3.

motion-feature frame: a frame generated by motion detection processing from the *difference frame*. Defined in section 3.1.

motion-region frame: a frame generated by motion analysis from the *motion-feature frame*. Defined in section 4.3.

motion processing element unit: an ASIC circuit module implemented for motion detection. Defined in section 5.3.

motion analysis unit: an ASIC circuit module implemented for motion analysis. Defined in section 5.3.

object of interest: the mouth region of the speaker is the *object of interest*. Defined in section 3.1.

RGB: coloured images are often stored as a sequence of RGB triplets or as separate red, green and blue overlays, though this is not the only possible representation. Defined in section 5.4.

TSMC: Taiwan Semiconductor Manufacturing Corporation Ltd, <http://www.tsmc.com.tw/english/default.htm>.

YCbCr color space: a luminance and chrominance color space which can be converted from RGB color space. Defined in section 5.5. It is formally defined in the standard of ITU-R Recommendation BT.601-4.

Acronyms

ASIC: application specific integrated circuit

CMC: Canadian Microelectronics Corporation, <http://www.cmc.ca>.

HDL: hardware description language

MPEG: Moving Picture Experts Group

VLSI: Very Large-Scale Integration

1.1 Motivation

Multimedia conferencing can be used as a substitution for face-to-face interaction to provide a means of multi-collaborator remote conferencing. It also could be used in tele-teaching and learning or in tele-medical consultations.

In multimedia conferencing, the quality of interaction provided by an unsupervised automated system becomes a barrier to wide use. A most important issue is that a basic system equipped with a sound locating microphone array, has difficulty in automatically locating the talker accurately enough for proper pointing of the camera. This is one factor that makes the quality of multimedia conferencing less than that of in-person interaction and communication. There could be several causes leading to the loss of information. One possibility is that the microphone array picks up the acoustic reflection direction and guides the camera to the wrong person in the meeting room, as shown in figure 1.1; another possibility is the microphone array may focus on a noise source. There are also

more subtle issues, such as the video frame severely lagging behind the audio, or some part of the speaker's body other than his head being captured in the video image. Research aimed at alleviating the difficulties of current multimedia conferencing is being carried out by the Carleton Digital Signal Processing Group in collaboration with Mitel Networks Inc.

Figure 1.1: Illustration of strong sound reflection in multimedia conferencing. This may cause the microphone array to assume that the speaker is positioned at the origin of the reflection, and then cause the camera to capture other persons instead of the speaker [HE00].

In a system for multimedia conferencing, one would like to have a computer-controlled camera with the ability to isolate the image of a specific speaker within video frames for real-time adjustment of orientation, zooming rate and tilt. It is also expected

that some visual information also can be extracted from video data to assist audio processing.

1.2 Problem Definition

In multimedia conferences, a microphone array can be used for coarse localization. Using the information from the audio processing of the array, a camera can be directed to the potential sector in which the talker is located. Then the talker will be captured in the view. However, as previously described, the images thus generated may be that of the wrong talker, or it might be poorly centered, or be only a partial image of the talker.

To improve the robustness and adaptability of a multimedia conferencing system, it is desired to extract the visual information of the speaker from video frames to assist the automatic system control. More particularly, locating the speaker by analysis of facial movement is the primary research in this thesis. This motion can then be correlated with the speech envelope to insure that both the microphone array and the camera are capturing the same speaker and the system is not being confused by superfluous noise or motion. This process and analysis must run in real time with an input-output delay in the range of one hundred milliseconds or less.

1.3 Thesis Outline

Chapter 2 provides the reader with background information on the system for multimedia conferencing. It also gives information about audio-visual joint processing and analysis. Chapter 3 describes the research conducted to explore the correlation of video and audio information of the speaker in terms of *motion area ratio* and the amplitude of the speech.¹ Chapter 4 describes the digital signal processing method for motion detection and analysis and its algorithmic model in functional diagrams. Chapter 5 presents the ASIC architecture and circuit, which were designed to concentrate the large amount of visual input data into a manageable form for use by a general purpose processor to meet real-time application requirements. Chapter 6 gives a description of the calibration of the experimental bench, which is used to capture the video and audio data for the correlation exploration in chapter 3. Chapter 7 concludes this thesis and offers recommendations for future work.

1. The terms in italics in this thesis are given a brief definition in glossary.

2.1 Audio & Video Joint Processing and Analysis

In a multimedia conferencing scenario, some special services are desired. For example, one would like to have a computer-controlled camera with the ability to isolate the image of a specific speaker within a video frame to facilitate real-time adjustment of orientation, zooming rate, focus and range, as well as tracking any interesting speaker motion. Additionally, such a system should provide a combined audio-visual processing function to adjust and compensate over a wide range of conferencing environments, examples being lighting conditions and room acoustic reverberations.

Although many object localization methods using either, but not both, audio or video signals already exist, researchers are just beginning to combine the two [AL99, BU95, ST99, ST01, WA00, WA97, WA99, ZO00]. In our research, joint audio-visual methods are explored for some problems in multimedia conferencing: for example, if a face is detected, it is difficult to identify whether it is a speaker, and even should it be

identified as a speaker, determining whether it is the right speaker is still a problem. The joint audio-video signal processing is an attempt to solve these kinds of problems. The combined methodology is just beginning to receive a significant amount of attention in the world.

Some of the special issues are discussed in the latest papers. Recent activities in audio-visual speech processing are reported, with emphasis on lip reading and lip synchronization [CH01]. A presentation of the general methodology on joint audio-video object localization and tracking is presented in [ST01]. Some systems have already been proposed that employ audio processing and video/image processing. Bub et al. [BU95] reported a face localization and tracking system which shows that visually guided audio processing provides a more accurate localization in space. In their system, the face color classification gives a first approximation of areas possibly containing faces. If detected, motion gives additional clues about an object's shape. However, it was mentioned that "To be capable of tracking non-moving speakers, motion is not considered as a necessary feature." Recently, Zotkin et al. [ZO00] developed a prototype for unattended video conferencing which used a combination of acoustical and video processing with a two-camera and multiple microphone setup. The system is implemented on a dual-processor Pentium III 500 Mhz PC with 512 MB of RAM, which performed audio and video data acquisition, acoustic source localization and tracking, video object localization and tracking, camera control and beamforming in real-time. With this advanced configuration, the real-time processing power is 10 frames/second, which is to be compared with the 30 frames/second

required for “smooth” video. A loss of synchronicity between audio and video data is also noticeable; they are working on eliminating it.

2.2 Blueprint of Our System for Multimedia Conferencing

Our research team has members working on many aspects of the system. Figure 2.1 depicts the framework in which this research fits.

In this research, an audio-video correlation analysis was investigated to explore the correlation between the audio and video information originating with the speaker of interest. A digital signal processing approach was proposed for the analysis of a speaker’s mouth motion and its algorithmic model was developed. The component key to meeting the speed and storage requirements of the application was identified. The VLSI architecture of an ASIC for motion detection and analysis was proposed. Based on this ASIC architecture, an ASIC gate-level synthesized design was simulated and verified utilizing the TSMC (Taiwan Semiconductor Manufacturing Company, Ltd.) standard cell library. The ASIC extracts the visual information from a huge amount of input data and “concentrates” it so that general purpose processors can complete the camera control in real time.

Figure 2.1: Blueprint of multimedia system research. Shaded blocks are the major research interests discussed in this thesis

2.3 Object Localization by Audio Processing

In the Strobel's overview [ST01], two classes of acoustic localization methods are generalized as follows:

- Direct source localization methods
- Indirect source localization methods

In our audio-video joint research, a microphone array is utilized for the coarse localization. It is a prototype of a Mitel production model using proprietary acoustic processing algorithms, software and hardware. Besides providing the function of coarse speaker localization, the microphone array is also able to perform other tasks, such as echo cancellation, and reverberation and noise suppression. Here, a brief introduction is given from the joint audio-video view.

Using the microphone array, real-time acoustic processing algorithms, and beam-forming schemes, the audio processing can provide coarse localization. Then the computer controlled camera will be directed to orient itself in approximately the direction.

Figure 2.2 shows that when beamforming is applied in a multimedia conference environment, the horizontal plane is divided into fixed sectors. Each sector is of 30 degree accuracy. But there could exist sufficient error that a speaker near a sector boundary is placed by the audio subsystem in a neighbouring sector.

Figure 2.2: Sector division in coarse localization by microphone array

2.4 Object Localization by Video Processing

According to the survey in [ST01], there are four classes of method for object localization from video sequences in the view of Strobel et al. They are cited as follows:

- class A, knowledge-based face tracking
- class B, feature-invariant face tracking
- class C, template-matching face tracking
- class D, appearance-based face tracking

Class A concerns potential geometrical relationships between facial features. For example, the relative positions of major facial elements, such as eyes, nose and mouth are checked. This method requires preprocessing steps to extract these major facial features.

In Class B, one looks for some specific features that are independent of size, pose, and even lighting conditions. Skin tone is the primary feature that has so far been researched and studied to a great extent.

Class C is made up of one or several manually defined patterns which can characterize the whole face or facial features. The correlation between the input images and these templates is processed and evaluated. The higher the correlation, the closer is the identification between face and template.

Class D processing takes place in two steps. First, feature parameters are learned from a training set. Then these features are searched for in facial images using a pattern-recognition process.

The method used in this thesis does not exactly fall in these classes. It is a bit between class A and class B.

Exploration of Correlation between Audio and Video

In this chapter, we show how potential speakers' mouths are identified using digital signal processing. The motion information of these potential mouths is concentrated to a small data set which can be correlated with the speaker's voice signal.

3.1 Definition of Motion Area Ratio

Through video/image processing, feature images can be generated in which some features are contained or emphasized. The features characterize the information that one expects to extract for the other processing steps. In this application, after motion detection, the *motion feature frames* are generated. The *motion feature frames* contain the motion information of the *object of interest*. The *object of interest* is the mouth region of the speaker in multimedia conferencing. To analyze it, an observing window can be defined. This observing window is defined as the *area of interest* here, which is usually a little larger than the *object of interest*. See Figure 3.1.

Figure 3.1: Illustration of motion pixels, *motion-feature frame*, *object of interest* and *area of interest*

As mentioned in the previous chapter, the camera zooms, focuses and tilts under control of the system. These parameters are known to the image processing modules. After a speaker is captured in the view of the camera, the size of the mouth region of the speaker can be estimated since the variation of human mouth sizes is small and the speaker is captured in the view at a known camera status, e.g. zoom rate, tilt degree, and resolu-

tion. Therefore, a knowledge-based estimation can be made about the size of the speaker's mouth region. The mouth region includes the lips, and their neighbourhood, the chin and the tongue. The mouth region of the speaker is the *object of interest*. Based on the size of the mouth region, an *area of interest* can be explored and sampled for image processing. The *area of interest* can be set the same size or larger than the *object of interest*. Setting it larger allows the analysis on the speaker's mouth region to be augmented with moderate analysis of the speaker's head.

The *motion area ratio* is one of the features that can be extracted from these *motion feature frames* to characterize the motion information of *object of interest*. The *motion area ratio* is defined as follows:

$$\text{motion area ratio} = \frac{\text{The total number of motion pixels in the } \textit{area of interest}}{\text{The total number of pixels in the } \textit{area of interest}}$$

When the total number of motion pixels in the *area of interest* is equal to zero, the *motion area ratio* will be equal to zero. When the total number of motion pixels in the *area of interest* is equal to the total *area of interest*, the *motion area ratio* will be equal to one, theoretically. Therefore, *motion area ratio* is a real number in the range from zero to one, inclusively.

Figure 3.2 shows a motion frame from experimental data with *objects of interest* and *area of interest* generated by the proposed method in this thesis. It can be seen there is a region in which the motion pixels show higher concentration. The *object of interest* is outlined by the solid line in the figure. It includes the objects in the mouth region that are changing when the speaker is in the view of the camera. The dashed line in the figure delimits the *area of interest*. It will be explained in detail in later chapters how the video is processed to generate the motion frame and to extract the image feature of the *motion area ratio*. In this chapter, the discussion is focused on exploring of correlation between the *motion area ratio* and the speech amplitude of the speaker.

Figure 3.2: *Motion-feature frame with the **object of interest** and the **area of interest**. The black area contains motion pixels, the white ones contain non-motion pixels; the solid rectangle is the **object of interest**, the dashed rectangle is the **area of interest**.*

3.2 Exploration of Audio-Visual Correlation of the Speaker

The *motion area ratio* was defined in the previous section; in this section, the correlation between video and audio is explored, in terms of the temporal character of the *motion area ratio* and the temporal character of the speech amplitude of the speaker.

If an *object of interest* is observed over time and the *motion area ratio* is extracted from every frame, the spatial and temporal character of a speaker's mouth motion can be represented by feature data that requires less computation. Figure 3.3 is shown here to illustrate the spatial temporal video processing.

Figure 3.3: Spatial temporal video analysis

Utilizing the experimental bench described in the next section, a speaker was captured in the view of a camera. Then the video and audio was recorded by the camera and microphone simultaneously. Two parts are generated in every audio-visual record: one is a video record of the speaker, the other is a speech record of the speaker. The recorded video was processed by the motion detection and analysis method that is proposed and described in a later chapter. During this image processing, the feature parameter, the *motion area ratio*, is extracted. A database for *motion area ratio* temporal character was built up which is representing the motion information in video. Figure 3.5 graphically depicts one example of *motion area ratio* temporal data. It was extracted from the audio-visual record, which was captured when the speaker was speaking the phrase “pm”. The voice of the corresponding speaker was recorded and processed by audio processing. A database of the amplitude vs. time of the speech was set up for the corresponding speaker audio database. Figure 3.4 depicts the corresponding audio data.

The “pm” visual record was captured at 30 frames/second with a resolution of 320 * 240 pixels. The speech of “pm” was sampled at the rate of 11025 Hz with 8 bits/sample mono quality. These records are captured simultaneously on our experimental bench.

Figure 3.4: Amplitude of voice 'pm' vs. time, extracted from "pm" audio record

Figure 3.5: *Motion area ratio* vs. time, extracted from "pm" visual record

Figure 3.5 shows that the video can be placed into two categories. One is the talking active period, the other is the non-talking period. For instance, there is a non-talking period from time zero to 1 second and from 2.5 seconds to 3 seconds. (Actually, in the experimental data, the non-talking period extended to 4 seconds but it is not shown in the figure.) In the figure, the display windows are of length 3 seconds for both video and audio data. The talking period is only from around 1.25 seconds to 2.25 seconds.

Figure 3.5 and figure 3.4 shows the agreement of the talking active period and non-talking period to the figure of speech voice. The pulse A just before 1 second is due to the speaker's breath. It is normal breath. The pulse B before 2.5 seconds is due to the post-talking mouth movement. Compared to the peak value of talking active pulses, the peak values of these two pulses are around one-fifth to one-tenth and are easy to discard.

Figure 3.6 and figure 3.7 show the experimental data from another audio-visual record which was captured when a speaker was speaking the phrase of "hello". On first glance, the figures look a bit strange. But after the observation of the audio-video data of "hello", it was found that the mouth movement of the speaker (pulse C) leads the phoneme "he". The mouth of the speaker opened at first, and made the pose preparatory for the phoneme "he" as shown (pulse C in figure 3.7).

Figure 3.6: Amplitude of speech voice ‘hello’ vs. time, extracted from “hello” audio-visual record

Figure 3.7: Motion area ratio vs. time, extracted from “hello” audio-visual record

One can see that pulse C is 0.6 - 0.7 seconds ahead of the audio onset of “hello” (pulse D). In the joint audio-visual analysis, an observation period can be set for correlation analysis. This kind of strong leading pulse might be used to put the control system into an “attention” mode. This sort of information is unavailable to the usual audio-only system. If there are strong correlation following these leading pulses within a second, they are closely related with speaker’s talking motion.

A possible quantitative analysis method for correlation analysis of *motion area ratio* and the speech amplitude is covered in the section on future work.

3.3 Overview of Experimental Bench

Figure 3.8 gives the diagram of the major components of the experimental bench. As mentioned in the previous sections, using this experimental bench, audio and visual data from the speakers was recorded simultaneously and processed by the Matlab codes. Once again, video resolution was 320 pixels (h) by 240 pixels (v). The video was captured at 30 frames/second. The speech was sampled at the rate of 11025 Hz with 8 bits/sample. In the chapter titled Calibration of Experimental Bench, the bench is described in detail.

Figure 3.8: Diagram of the experimental bench

ASIC Based Motion Detection & Analysis

4.1 Overview of the Motion Detection and Analysis

“Motion analysis used to be a specialized research area that did not have much to do with general image processing.” (B. Jahne in [JA97]). Fortunately today, the progress in computer hardware and algorithms makes the motion detection and analysis feasible even on a standard personal computer or workstation, although it is still a challenging task to apply it in real-time applications.

When the motion is taken as a feature or character that can be used to identify, characterize, and distinguish objects, and even used to understand dynamic scenes, it is a special and powerful feature. Still images can provide lots of image information. However, only video or image sequence processing and analysis will give a base to explore the dynamic dimension of real life.

Many motion detection and analysis methods have been studied and proposed. There are some good summaries [GO92, JA97] in this area. More theoretical and mathematical background is provided in the latter. There are four main categories of the motion detection and analysis methods:

- differential methods
- correlation methods
- phase methods
- tensor methods

The differential methods are the classical approaches to determining motion from two consecutive images. They compute the difference between two consecutive frames. They are quite sensitive to the changes in background illumination.

The correlation methods are performed by a combination of convolution and point operations. While the differential methods emphasize the difference of two frames, the correlation methods tend to explore the similarity of two frames. This approach has been used in MPEG2 and MPEG4 motion picture standards, where the similarity of two images is examined and utilized at the image encoding side to relieve the computation complexity on the decoding side. Generally, the correlation methods are less insensitive to temporal illumination changes.

The phase method is one of the frequency domain techniques. The earliest phase method was proposed by Fleet and Jepson [FL89], using the phase for computation of optical flow in the frequency domain. Rajala et al. [RA83] presented an application of the one - dimensional Fourier transform that is used for tracking moving projects.

The tensor methods can be further divided into a structure tensor method and an inertia tensor method. The former is a spatial - temporal method. The latter is a Fourier transform based method. The tensor methods can detect whether a local neighbour shows a constant velocity or not [BI87].

4.2 VLSI with Differential Method and Correlation Method

The motion detection method that is utilized in this paper falls into the category of the differential methods. There are several fundamental questions which need to be examined before one presents the motion detection & analysis method and the ASIC implementation which is proposed in this thesis.

It seems that the differential methods are not applied as extensively as the correlation methods in the image industry. The latter are applied in the well-known MPEG2 and MPEG4 standards for motion estimation. There are several reasons why motion estima-

tion is based on the correlation method instead of upon other methods. First of all, in MPEG2 and MPEG4, the motion vector for the next frame needs to be calculated with an accuracy of a pixel or even a sub-pixel. Even though the computation cost of the correlation methods is generally very high, they are still the best candidates for this task because they find similarities. They are better for tracking motion of a moving but slowly changing object. The differential methods, on the other hand, are best at showing where the small changes are.

Secondly, MPEG2 and MPEG4 compress the video data that is captured from various sources, ranging from a football game, or digital movie to a recording studio. The background of these frames and lighting conditions varies from one case to the other. The correlation methods are better able to handle a sudden change of lighting.

Thirdly, the correlation algorithms have been implemented into commercial ASIC chips. This is critical to realize the speeds required for MPEG2 and MPEG4. Otherwise, these standards would still be piles of documents in an advanced research center. Many researchers and engineers have worked on the application issue of the correlation methods to reduce the computation complexity to what can be done on a commercial ASIC. On the algorithm side, much work has been done to reduce the computation for matching and searching [PU87, HA95]. However, the algorithms are still too slow for feasible implementation on general VLSI processors. One needs the parallelism obtainable from dedicated ASICs. This was pointed out by Yu Hen Hu and Surin Kittitornkun in [HU01].

Specific algorithms can be “mapped” onto an ASIC architecture to get computationally seamless high parallelism [BH97, CH98, YE95]. We will call such implementations “functional ASIC algorithms” to differentiate them from the other algorithms which are more suitable for general purpose processors.

One could argue that when MPEG2 or MPEG4 coding is applied in non- real-time applications, the high-speed computation of the ASIC is not required. Only the decoding is done in real time. However, another resource-taxing aspect of the problem must be considered, namely storage. The following approximate data rates are recommended from a user manual of video grabbers from MGI Inc. [MGI]:

Table 4.1: Approximate data rates for two video options

<i>Frame size (pixels)</i>	<i>Frame Rate (per second)</i>	<i>Resulting file size</i>	
		<i>(per second)</i>	<i>(per minute)</i>
320*240	15	2.3MB	138MB
640*480	30	18.4MB	1100MB

In the experiment described in the previous chapter, video sampled at 320×240 pixels, 30 frames/second, leads to a video file size of around 4.9MB/second. From the above examples, it can be seen that video data can take up an enormous amount of storage space instantly before they are processed to a manageable level. High speed computation is needed to reduce the data before the input buffers overflow.

Multimedia conferencing is a real-time application which makes two demands in data processing facilities. One is a time demand; the input data has to be processed before it is sent out. The other is that the processed results which contain the useful information need to be condensed for the next steps. They must either be transferred or stored at a manageable level. In short, high intelligence and high speed in data processing is required to extract the important image features from the mass of input. At the same time, these image features must be represented in a form suitable for real-time post-processing. Generally, when the operations are based on pixel level processing for an image of 360×288 pixels (CIF size), millions of operations per image will be required. General purpose processors are still unable to provide the required processing ability.

In the following part of the thesis, the functional ASIC flow for differential motion detection & analysis and a proposed ASIC architecture will be examined in detail. Following this, the circuit design and simulation, based on an implementation using TSMC standard cells, and its verification, will be presented.

The problem of speaker motion in multimedia conferencing presents several general questions which have to be solved. First, what kind of motion detection methods fit into the multimedia conference scenario? When the theory is applied, can an efficient system architecture and processing flow be implemented to meet the real-time requirement of multimedia conferencing?

In MPEG2 and MPEG4, the motion vectors, for example the directions of motion of objects (like cars), have to be estimated for compression coding. Therefore, the very high computation correlation methods are still the best candidate. Firstly, for multimedia conferencing this is not a requirement. While the mouth motion is the movement of major interest, the differential methods are one possible approach to detect these motion activities since the differential methods are good at emphasizing image difference. Moreover, the mouth shape is changing when a speaker is talking. To follow the mouth movement of a speaker in multimedia conferencing, the differential method is better than correlation methods, since it is not used to track a speaker moving around a conference room. In a conference, people do not usually travel around the room while speaking.

Secondly, the differential methods are sensitive to temporal lighting change compared to the correlation method. (Some researchers have been working on reducing the sensitivity of the differential method to lighting change.) However, the sensitivity to temporal lighting change is not a drawback when applied to multimedia conferencing. It will be assumed that the lighting level is constant. This will be assumed in our attack on the

problem of finding an efficient ASIC implementation of the digital signal processing requirements. Even if it has a sudden change, it should only result in a momentary camera positioning error.

4.3 ASIC Based Motion Detection & Analysis Method

A set of digital signal processing flow, and its algorithmic model for motion detection and analysis in multimedia conferences is proposed and examined, which is based on experiment. Under this, an ASIC process flow and architecture is proposed and discussed, which can process the huge amount of video data in real time. The crucial ASIC core is designed, simulated and verified based on a standard cell library. The verification step, data and results are described in a later chapter.

The digital signal processing flow and its algorithmic model for motion detection and analysis are presented in the following two figures. Figure 4.1 shows the higher level diagram. The other shows more detail. These figures are the guidelines when the discussion is expanded in detail later. It has to be emphasized that these three parts, motion detection, motion analysis and ASIC architecture, are closely affiliated with each other.

Figure 4.1: General diagram of ASIC- based digital signal processing flow for motion detection and analysis

Figure 4.2: A more detailed ASIC-based digital signal processing for motion detection and analysis

4.4 Differential Motion Detection

In differential motion detection methods, the inter-frame differencing approach is commonly used. One group employed it as one step in a hierarchical scheme to extract facial features and model the face [RE95]. The silhouette was found by thresholding of the absolute frame difference with downsampling and temporal filtering. In [EL95] it was used to assist low bit-rate coding for video teleconference by subtracting consecutive low-pass sub-band filtered frames. In [WA99], it was used as the first step in distinguishing potential human bodies from the background environment.

In this thesis, inter-frame differencing is applied as the first step for motion detection. Figure 4.3 shows the functional steps for motion detection to extract the *motion-feature frame* from the *digital image Y frame* (the luminance frame), which is illustrated as only one motion detection box in figure 4.2.

Figure 4.3: The functional steps of motion detection

Coloured images from cameras are often stored as a sequence of RGB triplets or as separate red, green and blue overlays, though this is not the only possible representation. After the *digital image Y frames* are calculated from these RGB color frames, the difference frame is generated by absolute subtraction of two consecutive luminance, or Y, frames. Since the pixel value of a Y frame is in the range of $[0, 255]$, the pixel value of the difference frame has the same range. Thus there is no overflow in this operation. Some of the experimental data is shown in the following three figures. The first two figures, figure

4.4 and figure 4.5 are two consecutive RGB color frames. The difference frame is generated from the corresponding Y frames of these two consecutive RGB color frames. However, the difference frame appears to have little but “zero pixels”.

Figure 4.4: The first frame in a pair of two consecutive color RGB frames. It is not in its original resolution and size, and printed here in black and white. It has the approximate effect of a *digital image Y frame*.

Figure 4.5: The second frame in a pair of two consecutive frames. Compared to the first frame, there are mouth motion and some facial motion. Not in its original resolution and size.

This difference frame does not contain much useful information about mouth motion. But when all the pixel values in it are analyzed, one finds their distribution range is contained in $[0,255]$. Figure 4.6 gives a presentation. It shows that around seventy thousand pixels have a value of zero, or 91% of the total pixels in one frame, since the frame size is $320 * 240$ pixels.

number of pixels

Figure 4.6: The distribution of pixel value in the difference frame. The image size is 320*240 pixels. The small ellipses show the number of points at the integer pixel value. Most of the points are crowded near the X-axis. One point is a circle at 7×10^4 . The envelope was drawn slightly higher than it should be along the X-axis, to make it visible.

After analysis of the experimental data of several thousand difference frames, the distribution of pixel values in the difference frame can be generalized as follows:

- The envelope of the distribution is similar to an exponential distribution in that 80-90% of the pixel values lie in the $[0,10]$ and the remainder are in the range $[10,255]$.
- The peak of the distribution is at the low end.

Assumption of these properties is essential to the pixel-value thresholding operation. This distribution is directly related to whether thresholding of pixel values is necessary, how the threshold should be chosen, and whether any noise filtering is required after thresholding. Thresholding is dividing the data into two categories, above and below a threshold.

First, figure 4.7 shows that thresholding is necessary. After thresholding, the changes can be captured as features to analyze motion while the computation for the next step can be reduced from eight bits per pixel to one bit.

Secondly, Zack et al. developed a model which assumed a unimodal distribution [ZA77]. According to this model, a threshold value can be calculated. After considering several thousand difference frames, we found that any threshold between 15 and 30 all

worked about equally well. This is because the transition band of the envelope of pixel-value distribution becomes very small in the range of $[0,15]$. Figure 4.7 shows the corresponding *motion-feature frame* after thresholding of the previous “black” difference frame with a threshold of 20.

Figure 4.7 shows after the proper thresholding of the difference frame, the motion is captured while the noise in the background is removed or attenuated. Some of the experimental data is presented in the following figures, figure 4.9 to figure 4.12, which shows the exact threshold value is not very critical. These figures are not in their original size and resolution. They are arranged in four pairs for comparison. The two feature frames in each pair are processed from the same input data. The left frame in each group is processed by a threshold of 20; the right one in the corresponding group is by a threshold of 25. All the other parameters are the same.

Figure 4.7: A thresholded difference frame

Figure 4.8: A pair of motion frames during a talking period

Figure 4.9: A pair of motion frames showing eye clipping during a non-talking period

Figure 4.10: A pair of motion frames with the speaker breathing and in the presence of strong background noise during a non-talking period

Figure 4.11: A pair of motion frames showing the effect of a speaker's body motion

From the above experimental data, it can be seen that taking the absolute difference of two frames effectively removes all features that do not change, while highlighting those that do. If the lighting and geometry of view is consistent, the only differences in pixel values where no changes due to motion are caused by statistical variations in the brightness, due to camera or electronic noise. Compounded with a proper thresholding scheme, this differential method can capture the speaker's motion in the talking period and is relatively robust to background noise while the computation is reduced from 8 bit/pixel in the *digital image Y frame* to 1 bit/pixel in the *motion-feature frame*. The *motion-feature frame* is presented by a binary image. Its pixel value is either one or zero. In the figures of this section, to emphasize the detail of noise and motion pixels, the black pixels represent changes due to motion and noise.

Here it has been shown how the *motion-feature frame* can be generated by the digital signal processing. The next section will focus on the motion analysis of the *motion-feature frame*.

4.5 Motion Analysis with VLSI

After the *motion-feature frames* are generated by motion detection, motion analysis must be done. This is the crucial function which has to abstract the motion features and hopefully present them by a few parameters. Therefore it has to be able to handle the com-

putation pressure from a high data rate input to low complexity output. In this section, the general idea is presented to help understanding. Then the proposed ASIC function flow for motion analysis is followed in detail.

To analyze the *motion-feature frame*, pixel based approaches could be used. However, when the frame size is larger, for instance VGA 640 x 480, these approaches are less efficient and have a high demand on memory size.

Here a different method is used for motion analysis. The motion analysis is processed from pixel level to block level. The general idea for motion analysis is depicted in the following figure.

Figure 4.12: General diagram for motion analysis of the *motion-feature frame*

Since *motion-feature frames* are binary frames, a histogram of motion pixels is projected on the X-axis at first. It is called continuous X-histogram. Then the histogram of the motion pixels is projected on the Y-axis to give a continuous Y-histogram. The X-histogram will be converted into a binary X-axis histogram by thresholding. If the histogram value is bigger than the threshold which is set to generate the binary histogram, the corresponding binary histogram will be one, otherwise it is zero. The same thresholding pro-

cess is applied on the Y-histogram to generate a binary Y-histogram. Based on the binary Y-histogram and the binary X-histogram, the potential *object of interest* can be outlined by rectangular boxes. Figure 4.13 presents the processing result.

Figure 4.13: Two dimensional continuous histogram. There are 16 potential region boxes. Two of them are the strongest.

Once the potential *object of interests* are extracted, the region analysis and examination will screen them to select candidates for audio-visual correlation analysis. Figure 4.13 shows these candidates outlined by the light box. They are called potential motion region. The calculation of *motion area* for these potential motion regions is the most important function for region analysis and examination. The definition and calculation were given in the previous chapter.

However, it can be noticed that there are around sixteen potential region boxes. Each has to be screened by region analysis and examination based on box size and *motion area*. If the quantization level for the binary histogram is set to be slightly low, there will be more potential region boxes as input to region analysis and examination. If it is set a bit high, the number of potential boxes will be smaller. But the potential boxes will have smaller sizes. Each of the boxes will only contain part of the potential motion sub-region. Some additional functions have to be added as a postprocess to judge and merge these sub-regions to form a motion region which can efficiently present the motion feature of the speaker's mouth movement. The process can be improved by the method shown in figure 4.15.

Figure 4.14: Improved functional diagram for motion analysis of the *motion-feature frame*

The functional steps of building the histogram on the X-axis & the Y-axis and aggregation of the blocks can be done in parallel on the ASIC. The aggregation is based on 8*8 pixel blocks since the input video size is 320*240 pixels. Instead of extracting the potential region from the continuous histogram, the aggregation will generate two discrete histograms. Since there are 320 columns in the input image, there are 40 discrete data

points for the histogram on the X-axis and 30 discrete histogram data points on the Y-axis.

Figure 4.15 gives an illustration. The aggregation also gives a 40*30 pixels *motion region frame* illustrated in the same figure.

Figure 4.15: Illustration of the discrete aggregate histogram

4.6 Region Analysis and Examination

After the discrete data points for the histogram on the X-axis and Y-axis are generated by the functional flow shown in figure 4.15, they are converted to a binary histogram by setting the quantization level according to the camera status, which is explained in more detail in the next section. From these binary histograms, the potential motion region boxes can be extracted, which contain the mouth region as a whole instead of breaking it into pieces.

Figure 4.16 shows the 2 dimensional histogram projected on the corresponding *motion-feature frame*. This 2 dimensional histogram is taken from the previous 40 discrete X-axis histogram points and the 30 discrete Y-axis histogram points.

Figure 4.16: The illustration of two dimensional discrete aggregate histograms. They are converted in size, from 30 to 240 on the Y-axis, from 40 to 320 on the X-axis, and projected back on the *motion-feature frame*

Figure 4.17 shows, after quantization, the binary X-axis histogram and Y-axis histogram are plotted overlapping the corresponding *motion-feature frame*. Figure 4.18 shows that the potential motion regions are characterized only by two rectangular boxes comparing to the result of sixteen boxes from the continuous histogram based method. More important, the improved processing method has eliminated any extra function step to judge and merge sub-regions into meaningful motion regions. This motion analysis functional flow can be done in parallel by the ASIC as shown in the next chapter.

Figure 4.17 and figure 4.19 show the two motion region boxes which are screened according to their sizes and *motion area* parameters. The biggest one with a reasonable value of *motion area* will be selected as the best candidate for audio-visual correlation analysis. Compared to the previous figure 4.14, the biggest box is shifted a bit to the right, but the potential mouth movement region is captured in it.

Figure 4.17: Illustration of two dimensional quantized histogram

Figure 4.18: Illustration of motion region after motion detection and analysis with only two potential motion region boxes

The input for analysis has been reduced to a manageable level by the method in this chapter. The function of region analysis and examination can either be implemented by embedded microprocessors, for instance ARM7TDMI microprocessor [ARM7], or be implemented by a stand alone software module running on a general purpose processor.

4.7 Choice of Quantization Levels to Form Binary Discrete Histogram

Figure 4.19 shows how to choose the quantization level according to the camera status. For instance, at a certain zoom setting, one can estimate how wide human beings' mouths are in pixels. Then, the quantization level can be set in the range of 1.0 time to 2.0 times this pixel number. Here, the estimated mouth width is around 20 pixels, and the quantization level is chosen at 20. This is a simple, but important concept, which is necessary for the success of the algorithm proposed in this thesis.

Figure 4.19: Illustration of the choice of quantization level

ASIC Implementation and Verification

5.1 Subsystem Design Flow and ASIC Design Flow

Figure 5.1 shows the design flow to build the subsystem for motion processing from concept, through algorithmic modeling, down to the ASIC design. The major steps are numbered on the diagram and described below:

1) The technical specification is captured in high level languages and Matlab code. Various parameters for algorithmic modeling are investigated. Then, the algorithmic model is developed by several interactive rounds of modification and refinements to provide the functions to meet the specification. This methodology was followed to develop the algorithmic model presented in the previous chapter.

2) The algorithm has to settle the bit-true level issues. For example, how many bits for the signals have to be evaluated and determined by algorithmic simulation with large amounts of real-time experimental data.

Figure 5.1: System design and development flow [BA00]

3) After the algorithmic model is frozen at the bit-true level, the ASIC design can be carried out. The behavioral description of ASIC has to be built to capture a circuit design which provides the same processing functionality as the algorithmic model. This circuit behavioral description is the initial point for the ASIC design and verification flow.

Once the algorithm is frozen, the front end implementation of the ASIC can follow the Very Large Scale Integration (VLSI) design flow, as shown in figure 5.2.

Figure 5.2: ASIC design and verification flow for front-end and back-end design. The front-end is in solid boxes while the back-end is in dashed boxes, edited from [CMC99].

5.2 *Hardware Description Language (HDL) and Synthesis*

The advent of logic synthesis in the late 1980s pushed the Hardware Description Languages (HDLs) into the forefront of digital design in improving the productivity of VLSI design. HDLs provide the following advantages: they model the circuit system, facilitate simulation and verification, form a base for synthesis, and build the platform for design entry and integration of the target implementation technology. They can be used to describe hardware at various levels of abstraction, from the behavioral level to the structural level. In behavioral description, designers can describe complex circuits in terms of functionality and data flow. HDLs can also be used for system-level design and simulation. For instance, they are used for simulation of system boards, interconnect buses, Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs).

A common approach to design VLSI is to capture and describe circuit systems using HDLs, synthesize the HDL models into target technologies, and verify the system via simulation and verification. This approach allows designers to capture and verify functions at earlier stages without putting too much attention on details. Even the bit-true algorithm is modeled and frozen; it is still not a push button run to simply convert the algorithm into register transfer level or behavioral description using HDLs. The HDL behavioral description of the circuit system is the initial step of a process which is developed by several interactive rounds of refinements and modification. The number of itera-

tions depends heavily on the complexity of the design. The verification of each step in this process is one of the most time consuming and crucial parts of the design flow.

Every module of ASIC Verilog behavioral design has to be simulated to find any functional errors. Then, the modules are assembled together as a whole ASIC design to check their inter-module connections and functions. The test benches for module verification are developed to assist the design and simulation in this phase. They will also be used to verify the synthesized ASIC design.

After the behavioral description model has been verified, the design is ready for synthesis. Once the functional or behavioral design is frozen, the logic synthesis tools can be used to map the design to any target technology. This makes the design portable and reusable. As well, when designers follow true top-down design methodology, the synthesis allows a quick comparison of various trade-offs in implementing the same design in different technologies, and a comparison of similar technology implementation with different design parameters. High productivity is driven by the design flow and synthesis tools, combined with simulation and verification tools which are integrated in the design flow.

In this design flow, different HDL coding styles have a significant impact on the logic synthesis process. This problem is called *syntactic variance* [GA94]. Some general recommendations are given by D. R. Smith et al. in [SM00]. One is to always clearly separate the datapath from the controller. They should be coded in separate modules, but not

designed separately [KN00]. The design of the datapath and the corresponding controller is somewhat similar to that of key and lock.

The choice of Verilog over VHDL in this thesis is determined by the author's accessibility to the design tools. Verilog was originally designed as a proprietary verification and simulation product language [TH94]. Synthesis tools do not support all the Verilog elements. Sometimes, it is difficult to separate the language for the simulator from that for the synthesis tool because the dynamic aspects of the language are defined by the way the simulator and synthesizer work. To get an efficient and simple hardware implementation, one needs to have a fairly solid knowledge of hardware design techniques and be able to guide the synthesis tools smartly.

Every module of the ASIC is designed, optimized, simulated, synthesized and verified with iterations until an acceptable deliverable is obtained within the constraints of area, timing, and power. Then the built-in test circuits can be added in this front-end ASIC design, for instance, scan test circuits. However, for this thesis, this was left for future work. The back-end ASIC design is initiated by porting the design into a floorplan.

In conventional back-end flows, floorplanning tools perform two major tasks: the placement of circuit blocks and the placement of logic cells in each block. The floorplanning tools take a gate-level netlist and create the placement for route. The floorplan flow can take advantage of the interconnect analysis tool and logic merging features provided

in the Cadence tools. The interconnect analysis tool can select blocks to be merged to minimize the number of external nets which connect these blocks. Then the logic merging can be used to form a new block from these selected blocks. Some of the blocks will be pre-routed and their geometry parameters can be fed into the floorplan tool to perform block merging.

After the iterative, time consuming floorplan and route are finished to meet the constraints of area and time, the post-layout timing information of the VLSI will be obtained through a full-chip resistor and capacitor extraction to calculate the interconnect delay. Back-annotation is then used to read resistance, capacitance, and delay values from the layout file for circuit timing analysis. Then full-timing Verilog simulation with back-annotated delays can be performed. Physical design verification is used to check the final layout result.

5.3 ASIC Architecture

The ASIC architecture and its circuit design followed the recommendation stated by Vasudev Bhaskaran and Konstantinos Konstantinides [BH97], “Both RISC and DSP processors share the same philosophy: simpler is faster.”

Figure 5.3 shows the general ASIC architecture for motion detection and analysis. One *digital image Y frame* of $320 * 240 * 8$ bits needs to be held for motion detection. It can be stored in off-chip memory. The second consecutive frame is scanned pixel by pixel for motion detection processing and stored for the next round of processing when it is required.

The input image data is guided by the control unit through the processing module. The processing results, *motion region frame* and corresponding parameters, are stored in memory and in two register files.

Figure 5.3: General diagram of the ASIC for motion detection and processing, related to the motion subsystem of the blueprint of our system under research presented in chapter 2.

Figure 5.4 shows more details about the ASIC architecture. The processing and analysis module includes two units: a *motion processing element unit* and a *motion analysis unit*. The former is for the motion detection task. The latter is for motion analysis. The *motion processing element unit* has two input ports. One loads the pixel data from memory for the previous *digital image Y frame*. The other is used to scan in the pixel data from the current *digital image Y frame*. An external synchronization signal is used to signal the ASIC to initiate motion detection and analysis. One clock cycle after the *control unit* passes this signal to the other modules, the *motion processing element unit* begins to load

the first pair of pixels of the current and previous *digital image Y frames*. One more clock cycle later, it gives the corresponding thresholded difference pixel value. Seven more clock cycles later, the signal of the number of pixels of the difference frame in the pixel group is generated and sent to the *motion analysis unit*. In the last stage, the *motion-feature frame* and corresponding aggregate histogram data are generated as output of this ASIC.

Figure 5.5 gives details of the *motion processing element unit*. It starts from the absolute difference processing, goes through thresholding, and ends with the comparator and accumulator. These last circuits collect the total number of motion pixels out of a pixel group along a scan line. The pixel group is a unit of 8 consecutive pixels along a scan line/X-axis direction. The result is conveyed through a 4-bit-wide signal called dif8pix into the *motion analysis unit*. By this design, a highly parallel computation is provided, since the pixels of current and previous *digital image Y frames* can be continuously loaded in every clock cycle, once the control signal starting the motion detection is initiated. This gives less latency between the input frame and the output of the processed results.

Figure 5.6 depicts the general architecture of the *motion analysis unit*. The *motion-feature frame* is generated and sent to memory. The corresponding discrete aggregate X-axis histogram is stored in the X register file while the discrete aggregate Y-axis histogram is held in the Y register file. Here, to improve the speed the register files are used instead

of a memory block. When the image size is larger than 320×240 pixels, the design scheme can follow these steps:

- cache data in small size buffers to balance the area and the speed of design
- store the histogram data in memory
- interface this memory with the buffer

The *motion-feature frame* has a size of $40 \times 30 \times 8$ bits while the Y register file is 30×12 bits and X register file is 40×11 bits. The architecture is designed for highly parallel computation. Two *digital image Y frames* are simultaneously loaded into the ASIC and are processed in one round. Then, the *motion-feature frame* and corresponding aggregate histogram data are generated. This architecture allows that the huge input image data are processed with highly parallel computations and the processing results are concentrated to a manageable level that a general purpose processor can handle. The circuit implemented with this architecture can run at a maximum clock rate of 110MHz under worst case conditions without taking interconnection delay into consideration. When running at a clock of 100MHz, the ASIC provides a latency between the frame input and the final output of 0.8 ms, which is adequate to meet the real-time application requirements.

Figure 5.4: Motion detection and processing block diagram

Figure 5.5: Diagram of motion processing element unit

Figure 5.6: Diagram of the *motion analysis unit*. Refer to figure 5.9 for the graphic illustration of processing results.

Figure 5.7 illustrates the timing of critical signals corresponding to the ASIC architecture in figure 5.4, figure 5.5 and figure 5.6.

Figure 5.7: Illustration of the timing of critical signals

The ASIC architecture and design are scalable to a larger video size with higher resolution with some extra storage overhead. For instance, this is applicable to the video size for digital TV, which follows ITU-R Recommendation BT.601-4 [BT601-95]. In this recommendation, a CCIR 601 image is defined as 720 pixels *480 lines for National Television Standards Committee (NTSC) and 720 pixels *576 lines for Phase Alternating Line (PAL). Only the designs of the *control unit* and *motion analysis unit* need to be scaled to increase the bus-width, count size and storage size.

5.4 ASIC Verification

The verification of the ASIC in this thesis followed the model in figure 5.8. The same recorded experimental data drove the algorithmic model and test bench for the ASIC as input. The output of the algorithmic model is the “gold standard” that is used to check the output of the ASIC behavioral description design and the gate-level ASIC design synthesized from the Synopsys synthesis tool. Static timing analysis was also used to verify the synthesized ASIC circuit.

The ASIC behavioral simulation was performed to verify the functional correctness of the ASIC behavioral model. After the behavioral model was synthesized into a gate-level netlist, the gate-level simulation was performed to verify the functional equivalence of this gate-level ASIC design. Figure 5.9 shows the graphic illustration of the output of the ASIC. Then, these outputs were imported into the algorithmic model to project the box of *object of interest* onto the input *digital image Y frame* for high level evaluation of the ASIC processing results, as shown in figure 5.10.

In simulation and verification, the test vectors are generated in two categories: pre-determined data and experimental data captured by camera.

Figure 5.8: ASIC verification model

Figure 5.9: Graphic representation of ASIC output data for verification. Refer to figure 5.6 for ASIC architecture.

Figure 5.10: High level evaluation of ASIC processing outputs projected on the corresponding *digital image Y frame*, which is an input to ASIC. This agrees exactly with the results of the algorithmic model.

The functionality verification of the ASIC requires an external data memory. However, the gate-level design of these components is not part of this thesis. The RAM blocks which are used to hold the input frame data and the output result are modeled by the behavioral Verilog entities. Their behavior was described as a Verilog behavioral description and simulated together with the ASIC behavioral design and its synthesized gate-level circuit. Use of these RAM block behavioral models allows running the actual experimental frame data to test the ASIC's functions. In the layout phase, a pre-compiled memory block would be inserted on the chips [BR93, JO91].

The technology used for this ASIC is 0.35 μm CMOS with three metal layers, from the technology foundry Taiwan Semiconductor Manufacturing Corporation. The library cells were accessed through the Canadian Microelectronics Corporation [CMC].

The final synthesis results show that the critical path has 9.34 ns delay, enabling a maximum clock rate of 110 MHz under worst case conditions without taking interconnect delay into consideration. This critical path begins from the control signal and ends at the output to the X register file. Table 5.1 summarizes the delay of the critical path group and the components that account for the major delay in it.

Table 5.1: The delay of the critical path group

<i>Critical path group</i>	<i>Maximum delay in the group (ns, Worst case)</i>	<i>Major components and their delay</i>	
		<i>Component</i>	<i>Maximum Delay (ns, Worst case)</i>
computation and output of X register data	9.34	control signal for computing X register data	5.5
		computing and data flow of X register data	3.8

The above table shows that the computing of the X register data does not take much computation time. This ASIC design avoids any multiplication operation in the processing flow to meet real-time application requirements.

The ASIC design has a circuit complexity which is equivalent in area to 18500 2-input AND gates. Without taking pads and inter-module routing into account, Synopsys Design Compiler gives the following estimation on area and power consumption. The total area of the synthesized ASIC is $820000 \mu\text{m}^2$. The *control unit* takes $27000 \mu\text{m}^2$. The *motion processing element unit* takes $11700 \mu\text{m}^2$, while the *motion analysis unit* takes the most, $780000 \mu\text{m}^2$, partially because of the register files. The total dynamic power dissipation without pads is 45.6 mw while leakage power is 4.5 nw with a 3.0 v power supply.

5.5 Discussion of Other Design Issues

Most camera CMOS sensors [AG01,IC00,OM76] give the YCbCr signal outputs which follow the ITU-R Recommendation BT.601-4 to provide an interface for digital color signals. The ASIC designed in this thesis accepts the Y signal of these CMOS sensor outputs as the *digital image Y frame* data for motion detection and analysis. This allows a seamless integration to any possible System-on-Chip commercialization.

In any case, if there is no *digital image Y frame* data ready to be input, the color space conversion from RGB signals of camera sensors to YCbCr signals can be performed by the design suggested in [Bh97].

The color conversion between YCbCr and RGB follows these calculations:

$$Y = 0.299 R + 0.587G + 0.114 B$$

$$Cb = -0.169 R - 0.331G + 0.500 B$$

$$Cr = 0.500 R - 0.419G - 0.081 B$$

Using a parallel multiply-accumulate unit, these calculations can be completed in nine cycles per pixel. However, an alternative implementation can conduct this task using barrel shifter and adders. This shares the same design motivation as the ASIC in this the-

sis, and avoids using multipliers. This barrel shifter and adder design can be easily integrated into the ASIC in this thesis to perform the color conversion option.

The barrel shifter and adder design for color conversion uses approximate calculations. For example, 0.299 can be approximated by the sum $2^{-2} + 2^{-5} + 2^{-6} + 2^{-8}$. Therefore, $0.299R$ can be broken into $R(2) + R(5) + R(6) + R(8)$, where $R(n)$ denotes a right shift of R by n bits. Finally, the calculation of Y can be in this form:

$$Y = R(2) + R(5) + R(6) + R(8) + G(1) + G(4) + G(6) + G(7) + B(4) + B(5) + B(6) + B(8).$$

The data of R can be loaded from the barrel shifter four bits at a time, for instance, $R(2)$, $R(5)$, $R(6)$, $R(8)$ in the first cycle, then $G(1)$, $G(4)$, $G(6)$, $G(7)$ in the second, etc. A Y value is given after three cycles. Using pipelining, a color conversion calculation can be done in nine cycles per pixel. Therefore, when this is integrated into the ASIC design in this thesis to provide the option of a color conversion function, the ASIC still can meet the real-time application with less effort than using a multiplier.

However, we propose a hard-wired carry save adder to be implemented instead of a barrel shifter and adder design, which is used for color space conversion. Using this design, the silicon area can be reduced while the computation speed remains high.

Calibration of Experimental Bench

6.1 Calibration of Timing

In chapter 3, the correlation of video data and audio data captured simultaneously from a speaker is explored in the time domain. In the experiment, the camera was aimed at the speaker to capture video at 30 frames/second, 320*240 pixels, while the microphone recorded the speech at 11025 Hz with 8 bits/sample mono quality. Then the video and audio data were processed and encoded by a control computer. The experimental bench was calibrated to judge the time delay between the capturing and processing of audio and video. The experimental bench includes the various hardware components and software tools. The major elements are listed as follows:

- Creative Sound Blaster AudioPCI
- Canon communication camera VC-C4
- Belkin USB VideoBus II
- MGI VideoWaveIII

- X86 Family 6 Model 8 Stepping 3 CPU
- 384 MB RAM Memory
- Microsoft Windows 2000

Figure 6.1 shows the calibration of the experimental bench. The signal generator is used to generate a square wave. This square wave output drives an analog oscilloscope and a beeper simultaneously. The beeper is triggered on the rising and falling edges of the square wave. The oscilloscope captures the square wave and shows it on the screen.

Figure 6.1: Calibration of the experimental bench

A camera is used to aim at the screen of the oscilloscope to record the display in video at 30 frames/second while simultaneously the sound of the beeper is captured and recorded by a microphone at 11025 Hz with 8bits/sample. Then the video and audio data are sent to the computer to be encoded together. When the data is recorded, the video data is given priority for processing.

The analysis shows that in this experimental bench the time delay between the video data and audio data is in the range of 0.18 - 0.21 seconds in the worst case, and around 0.10 - 0.15 seconds typically. When audio and video parameters are in correlation processing, these time delays can be take into consideration for high accurate computation. Figure 6.2 shows one sample of the beep sound wave while the corresponding video frames are shown in figure 6.3 and figure 6.4. The time delay between the video data and audio data could be due to the priority given to the encoding of video data.

Figure 6.2: Beeping sound wave at the falling edge at around 1.25 seconds

Figure 6.3: The corresponding video frame of the falling edge sound wave, captured just before the falling edge at 1.066 second; here the audio had a approximate delay of 0.15 second.

Figure 6.4: The corresponding video frame of the falling edge sound wave, captured just after the falling edge at 1.099 second; here the audio had a approximate delay of 0.182 second.

7.1 Summary and Novel Concepts

An ASIC-based digital signal processing methodology was proposed for motion detection and analysis to be integrated into our system under research for smart multimedia conferencing. The processing accepts the video data from most CMOS camera sensors. Then the motion detection and motion analysis are performed to extract the useful motion information of the speaker from the huge amount of video data generated in real time. At the same time, the processing results are concentrated to a manageable level for audio-visual correlation analysis and data fusion of the system by a general purpose processor.

General Features

The digital signal processing methodology was modeled by Matlab and C language programs. The method succeeded in extracting useful motion information about the

speakers' mouths while efficiently reducing the computation complexity for following processing steps. This method reduced two 320*240 pixels (2*76800 pixels) frames to one 40*30 pixel (1200 pixels) *motion-region frame*, plus typically 8 integer numbers for the X-axis and Y-axis coordinates of the motion region.

The difference frame was converted to a binary frame using a threshold. The choices of the threshold for the histogram were found by analyzing several thousand *difference frames*. The result gave a very simple heuristic rule. An alternate was proposed by G. W. Zack [ZA77]. However, that method would require real time computation of the statistics of the data.

For motion analysis, the setting of quantization level for the two-dimension binary discrete histogram was examined. An approach for choosing the quantization level was proposed, based on the mouth width in pixels. As a result, false detection due to hand, eye, and background-object motion was greatly reduced.

ASIC Architecture

An ASIC architecture was proposed and its circuit was designed to perform the task of motion detection and analysis in real time. This ASIC concentrates the motion

information of the speaker to a level that general purpose processors can manage, in order to meet the requirements on time and storage.

The ASIC architecture was designed for highly-parallel computations while placing more relaxed requirements on the memory interface. Especially, a novel processing method of discrete aggregate histogramming was proposed and mapped onto this architecture to concentrate the mouth motion information of speakers with less effort than pixel-based methods. After one round of processing, the motion information of the inputs was extracted and concentrated to provide motion parameters in two dimensions.

This architecture is also a parameterized design, which is scalable for larger video frame sizes that follow the ITU-R Recommendation BT. 601-4. It also accepts the YCbCr video frame format that most commercial camera sensors provide.

Design Performance

The front-end ASIC implementation was synthesized utilizing TSMC's 0.35 CMOS technology and cell library. The design was verified against the algorithmic model for signal processing functionality. Running at a clock rate of 100 MHz, the ASIC can achieve 0.769 ms latency between the input frame and the output of the ASIC processed results, when the input frames are 320*240 pixels, at a rate of 30 frames/second. The

ASIC can run at a maximum clock rate of 110 MHz under worst case conditions without taking interconnect delay into consideration. The ASIC design has a circuit complexity which is equivalent in area to 18500 2-input AND gates.

Exploration of Correlation between Video and Audio Data of the Speaker

The correlation of the video data and the audio data of the speaker in multimedia conferencing is investigated by a new approach in terms of *motion area* and the speech amplitude. This correlation approach shows the relationship in the time domain between the audio and video signal of the speaker captured simultaneously. It was found that some words which contained mouth motion did not result in sound. For example, when the speaker is speaking the phonon beginning with “h”, the mouth motion precedes the speech.

7.2 Future Work

A prototype system composed of different software and hardware modules needs to be built up. The methodology of software and hardware co-design and co-verification is the critical element on this prototyping path.

Further research and investigation in video-audio joint processing and analysis can be carried out to continue improvements to the intelligence of the system for multimedia conferencing. One possible method to analyze the correlation between *motion area* and the speech amplitude includes the following steps:

- Extract the envelope of the energy of speech
- Downsample this envelope at 30 Hz/second
- Correlate this downsampled audio data with *motion area ratio* data
- Try to remove the impact of nonvocal mouth motion on correlation computation.

References

- [AG01] Agilent Technologies Inc., Palo Alto, CA. U.S.A., *HDCS-1020, HDCS-2020 CMOS image sensors data sheet*, Feb. 20, 2001.
- [AL99] A. Al-Qayedi and A. F. Clark, "An algorithm for face and facial-feature location based on grey-scale information and facial geometry," *Image Processing and its application, IEE Conference Publications*, issue N. 465, PT.2, 1999, pp 625-629.
- [ARM7] *ARM7TDMI (Rev4) Technical Reference Manual*, <http://www.arm.com/Documentation>, visited Sept. 2001.
- [BA00] H. Bacher and H.U.Roeck, "Enhancing Hearing Aid Performance with SOC's", *EETimes Network*, May 2000, <http://www.eedesign.com/story/OEG20000509S0034> visited Jan. 2001.
- [BH97] V. Bhaskaran, *Image and video compression standards : algorithms and architectures*, Kluwer Academic Publishers, 1997, 2nd edition, pp 183-184, 254-255, 284-291.
- [BI87] J. Bigun et al., "Optimal orientation detection of linear symmetry," *Proceedings ICCV' 87, IEEE Computer Society Press*, 1987, pp 433-438.
- [BR93] T. Brodnax, "The PowerPC 601 design methodology," *IEEE International Conference on Computer Design*, October, 1993, pp 248-252.
- [BT601-95] ITU Radiocommunication (ITU-R) standard number BT.601-5 (10/95), *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, October, 1995.
- [BU95] U. Bub et al, "Knowing who to listen to in speech recognition: visually guided beamforming," *International Conference on Acoustics, Speech, and Signal Processing 1995, ICASSP-95*, vol. 1, 1995, pp 848 -851.

- [CA00] *Design Framework II user guides*, version 4.4.x, Cadence Design Systems Inc., San Jose, U.S.A., 2000.
- [CH01] Tsuhan Chen, "Audiovisual speech processing: lip reading and lip synchronization," *IEEE Signal Processing Magazine*, vol.18, no. 1, Jan 2001, pp 9-21.
- [CH98] Y.-K. Chen and S. Y. Kung, "A systolic methodology with applications to full-search block matching architectures," *Journal of VLSI Signal Processing*, 1998. vol 19, no.1, pp 51-77.
- [CMC] Canadian Microelectronics Corporation, <http://www.cmc.ca>.
- [CMC99] Canadian Microelectronics Corporation, *0.35-Micron CMOS Design Kit for Synopsys and Cadence V4.2*, August 09, 1999.
- [EL95] A. Eleftheriadis and Jacquin, "Automatic face location, detection and tracking for model-assisted coding of video teleconferencing sequences at low bit-rates," *Signal Processing: Image Communication*, vol. 7, no. 3, pp 231-248, 1995.
- [FL89] D. J. Fleet and A.D. Jepson, "Hierarchical construction of orientation and velocity selective filters," *IEEE Trans. PAMI*, vol. 11, no.3, pp 315-324, 1989.
- [GA94] D. Gajski, "Introduction to high-level synthesis," *IEEE Design and Test of Computers*, Winter, 1994, pp 45-54.
- [GO92] R. C. Gonzalez et al., *Digital Image Processing*, Addison-Wesley Publishing Company, 1992, pp 465-478.
- [HA95] H.- M. Hang and Y.-M. Chou, "Motion estimation for image sequence compression," *Handbook of Visual Communications*, Academic Press, pp 147-164.
- [HU01] Yu Hen Hu and Surin Kittitornkun, "Application-specific multimedia processor architecture," *Multimedia image video processing*, CRC press LLC, 2001, pp 35-66.
- [IC00] IC MEDIA CORPORATION, San Jose, U.S.A., *ICM-205D VGA Resolution color CMOS Image sensor data sheet*, July 18, 2000.
- [JA81] R. Jain, "Dynamic scene analysis using pixel-based processes," *Computer*, 1981, vol. 14, no. 8, pp 12-18.

- [JA97] B. Jahne, *Digital Image Processing : Concepts, Algorithms, and Scientific Applications*, Springer, 1997, pp 395-450.
- [JO91] D. Jones , et al., "Verification techniques for a MIPS compatible embedded control processor," *IEEE International Conference on Computer Design*, October, 1991, pp 329-332.
- [EL00] H. Gilad, "Speech enhancement using microphone array," *M.Eng. Thesis*, 2000, pp 5-40.
- [KN00] J.P. Knight, Bridge Camp VLSI design, http://www.doe.carleton.ca/~jknight/Bridge_Camp2000/index2.htm, visited Jan. 2001.
- [KO81] T. Koga et al., "Motion-compensated interframe coding for video conferencing," *National Telecommunication Conference*, 1981, pp G5.5.3.1-5.
- [MGI] *MGI VideoWave III Reference Manual*, MGI Software Corp., 2000, P 38.
- [OM66] Omnivision Technology Inc., Sunnyvale, U.S.A., *OV6620 single-chip CMOS CIF color digital camera and OV6120 single chip CMOS CIF B&W digital camera*, Rev. 1.3 (5/13/00).
- [OM76] Omnivision Technology Inc., Sunnyvale, U.S.A., *OV7620 single-chip CMOS CIF color digital camera data sheet*, Rev. 1.3 (5/13/00).
- [PI97] G.S. Pingali, "Integrated audio-visual processing for object localization and tracking," *Proc. SPIE*, 1997, vol. 3310, pp. 206-213.
- [PU87] A. Puri, H.-M. Hang, and D. L. Schilling, "An efficient block-matching algorithm for motion compensated coding," *Proc. IEEE International Conference on Acoustic, Speech, Signal Processing*, April 1987, pp. 25.4.1-25.4.4.
- [RA83] S.A. Rajala, A. N. Riddle et al., "Application of the one - dimensional Fourier transform for tracking moving objects in noisy environments," *Computer Vision Graphic Image Processing*, vol. 21, pp. 280-293.
- [RE95] M.J.T. Reinders , et al., "Ficial feature localization and adaptation of a generic face model for model-based coding," *Signal Processing: Image Communication*, vol 7, no. 3, 1995, pp 57-74.
- [RE98] S. J. Reeves, " A survey of image processing software and image data-bases," *The Digital Signal Processing Handbook*, CRC Press , 1998.

- [SM00] D. R. Smith et al., *Verilog style of synthesis of digital systems*, Prentice Hall Inc., 2000, pp 137-139.
- [ST99] N. Strobel et al., "Classification of time delay estimates for robust speaker localization," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, vol. 6 , 1999, pp 3081 -3084.
- [ST01] N. Strobel , et al., "Joint audio-video object localization and tracking," *IEEE Signal Processing Magazine*, vol. 18, no. 1 , Jan 2001 , pp 22 -31.
- [SY01] *Synopsys online documentation*, version 2001.08, Synopsys Inc., Dec. 2001.
- [SY01P] *PrimeTime reference manual*, version 2001.08-sp1, Synopsys Inc., Dec. 2001.
- [TH94] D.E. Thomas et al., *The Verilog Hardware Description Language*, Kluwer Academic Publishers, 1994, 2nd edition, p2.
- [TSMC] Taiwan Semiconductor Manufacturing Co., Ltd, <http://www.tsmc.com.tw/english/default.htm>, visited Oct. 2000 .
- [WA00] C. Wang et al., "Robust automatic video-conferencing with multiple cameras and microphones ," *IEEE International Conference on Multimedia and Expo, 2000, ICME 2000*, vol. 3 , 2000, pp 1585 -1588.
- [WA99] C. Wang et al., "Multi-source face tracking with audio and visual data ," *Multimedia Signal Processing, 1999 IEEE 3rd Workshop*, 1999, pp 169 - 174.
- [WA97] H. Wang, and P.C. Chu, "Voice source localization for automatic camera pointing system in videoconferencing ," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97.*, vol.1, pp 187-190.
- [YE95] H. Yeo, and Y. H. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Trans. on Circuits System Video Technology*, 1995. vol.5, no.5, pp.407-416.
- [ZA77] G. W. Zack et al., "Automatic measurement of sister chromatid exchange frequency," *the Journal of Histochemistry and Cytochemistry*, vol. 25, 1977, pp 741-753.
- [ZO00] D. Zotkin et al, "Smart videoconferencing ," *IEEE International Conference on Multimedia and Expo, 2000, ICME 2000*, vol. 3, pp 1597 -1600.

Appendix A: Verilog Behavioral Description of ASIC Modules

```
//-----  
// Copy right © 2002, Han Zhao, hzhao@doe.carleton.ca  
//-----  
  
// paralist.v  
  
'define pix8          3'd1    //3'b001  
'define everyseventh  3'b111  // indicate linecount is at every seven.  
'define DIF8PIXWIDTH  3'd4    //The max number of dif8pix will be 'd8, therefore  
                                //need 4 bit for register size  
  
'define BYTESIZE      4'd8  
'define SEGADDRESSIZE 'd5  
'define BLKADDRESSIZE 'd6  
'define DIFTHRESSIZE  'd8  
  
'define BLKCOLUMNREGSIZE 'd8  
'define BLKCOLUMNREGACCSIZE 'd11  
  
'define MDPROWACCSIZE 'd12  
  
// parameter for full size frame c320 by r240  
// +-----+  
'define columnlength  11'd320 //h140  
'define rowlength     8'd240  //hF0  
  
'define lastline      8'd239 //hEF  
'define SEGCOLUMNLENGTH 6'd40 //h28  
'define SEGROWLENGTH   5'd30  //h1E  
  
'define FRAMESIZE  
'define IMADDRESSIZE  'd17  
'define DIFBLKSIZE    'd1200  
'define DIFADDRESSIZE 'd11  
//-----
```

```

'include "paralist.v"

module CONTROL (
clock, difstart,
prepixaddress ,curpixaddress, done8pix, doneseg, doneframe,
segblkaddress, done8pix4lastsegline,
blkaddress,segaddress

);

parameter
pixsign = 'pix8, linesign = 'columnlength ,
framesign = 'rowlength ,
lastseglinesign = 'everyseventh, lastframelinesign = 'lastline ,
segcolumnlength = 'SEGCOLUMNLENGTH,
imAddressSize = 'IMADDRESSSIZE ,
segaddresssize = 'SEGADDRESSSIZE ,
blkaddresssize = 'BLKADDRESSSIZE ,
difAddressSize = 'DIFADDRESSSIZE ;

input clock, difstart;

output done8pix, doneseg, doneframe, done8pix4lastsegline;
reg done8pix, doneseg, doneframe;

reg doneline;

reg donelinepre;

output [blkaddresssize-1:0] blkaddress;
reg [blkaddresssize-1:0] blkaddress;

wire [blkaddresssize-1:0] blkaddresspre;
reg [blkaddresssize-1:0] blkaddressprenxt;

output[segaddresssize-1:0] segaddress ;
reg [segaddresssize-1:0] segaddress ;

wire [segaddresssize-1:0] segaddresspre ;

output [imAddressSize-1:0] curpixaddress , prepixaddress ;
wire [imAddressSize-1:0] curpixaddress , prepixaddress ;

```



```

output [difAddressSize -1:0] segblkaddress;
wire [difAddressSize -1:0] segblkaddress;

wire done8pix4lastsegline;
reg lastsegline , lastframeline ;
reg lastseglinepre,lastframelinepre;
reg donesegpre, doneframepre;

reg [8:0] count , countnext;

reg [7:0] linecount , linecountnext;
wire [7:0] memrowaddress;

reg [imAddressSize-1:0] memaddress;

always @(posedge clock)

begin
    if (~difstart) count <= 0;
    else count <= countnext;
end

always @(count or doneline)
begin
    countnext = count +1;
    done8pix = 0;
    donelinepre = 0;

    if ( ( (count[2:0] == pixsign) && ( count != 1) ) || doneline == 1 )
        done8pix =1; // mod june30

    if (count == linesign) begin donelinepre =1; countnext=1; end

end

always @(posedge clock)
doneline <= donelinepre;

always @(posedge clock)
begin
    if (~difstart) linecount <=0;
    else if (donelinepre) linecount <= linecountnext;
end

```

```

always @( linecount or count )
begin

linecountnext = linecount + 1;
donesegpre = 0 ;
doneframepre = 0;

lastseglinepre = 0;
lastframelinepre = 0;


if (( linecount[2:0] == 3'b000) && ( linecount!= 0) && count == 1)

    donesegpre = 1;

if ( (linecount == framesign) && count == 1)
    begin
        doneframepre = 1;

    end


if ( linecount == lastframelinesign )
    lastframelinepre = 1;

if (( linecount != 1 ) && ( linecount[2:0] == lastseglinesign))
    lastseglinepre = 1;


end


always @(posedge clock)
begin
lastsegline <= lastseglinepre;
lastframeline <= lastframelinepre;

doneseg <= donesegpre;
doneframe <= doneframepre;
end

assign blkaddresspre = count[8:3];

assign segaddresspre = linecount[7:3] ;

assign done8pix4lastsegline = lastsegline && done8pix;

```

```

always @(posedge clock)
begin
if (donelinepre)
    blkaddressprenxt <= 0;
else
    blkaddressprenxt <= blkaddresspre;
end

always @(posedge clock)
begin
segaddress <= segaddresspre;
blkaddress <= blkaddressprenxt;
end

assign segblkaddress = segaddress * segcolumnlength + blkaddress;

assign          memrowaddress = linecount;

always @ ( memrowaddress or count )
begin

    memaddress = memrowaddress * linesign + count;
end

assign          curpixaddress = memaddress;
assign          prepixaddress = memaddress;

endmodule // end module control

module MDP(
clock, difstart, difthres, pre, cur, done8pix,

dif8pix //,

);

parameter dif8pixsize = 'DIF8PIXWIDTH, bytesize = 'BYTESIZE,
            difthressize = 'DIFTHRESSIZE;

input clock, difstart, done8pix;

input [ difthressize -1:0] difthres;

input [bytesize-1:0] pre,cur;

output [dif8pixsize -1:0] dif8pix;

```

```

reg [dif8pixsize -1:0] dif8pix ;

reg difstartnext;

reg dif;

reg [difthressize -1:0] difference ;

reg [dif8pixsize -1:0] dif8pixnew;

always @( cur or pre or difthres)
begin

    if (pre > cur)
        difference = pre - cur;
    else
        difference = cur - pre;

    if (difference >= difthres) dif = 1'b 1;
    else dif = 1'b 0;
end

always @( posedge clock)

difstartnext <= difstart;

always @( posedge clock)
begin

dif8pix <= dif8pixnew;
end

always @( difstartnext or done8pix or dif or dif8pix)
begin
casez( { difstartnext,done8pix,dif } )
3'b0?? : dif8pixnew = 3'b000;
3'b110 : dif8pixnew = 3'b000;
3'b111 : dif8pixnew = 3'b001;
3'b100 : dif8pixnew = dif8pix;
3'b101 : dif8pixnew = dif8pix + 1;
3'b10? : dif8pixnew = dif8pix;
default: dif8pixnew = 3'b000;
endcase
end

endmodule

```

```

module STOREMDPREG(
clock, difstart,
done8pix, doneseg, doneframe , done8pix4lastsepline,
dif8pix ,
blkaddress, segaddress,

difblkdata,
blkcolumnregACCdata , blkcolumnregACCdatanext , rowsegregdata,

difblkdatapre
);

parameter columnreglength = 'SEGCOLUMNLENGTH ;
parameter rowreglength = 'SEGROWLENGTH;
parameter      bytesize = 'BYTESIZE ,
                dif8pixsize = 'DIF8PIXWIDTH ,
                blkaddresssize = 'BLKADDRESSSIZE ,
                segaddresssize = 'SEGADDRESSSIZE ,
                blkcolumnregsize = 'BLKCOLUMNREGSIZE,
                blkcolumnregACCsize = 'BLKCOLUMNREGACCSIZE,
                mdprowACCsize = 'MDPROWACCSIZE;

input clock, difstart, done8pix, doneseg, doneframe , done8pix4lastsepline;

input [dif8pixsize -1:0] dif8pix ;

input [blkaddresssize -1:0] blkaddress;

input[ segaddresssize -1:0] segaddress ;

output [blkcolumnregsize-1:0] difblkdata;
reg [ blkcolumnregsize-1:0] difblkdata;

output [ blkcolumnregsize-1:0] difblkdatapre;
wire [ blkcolumnregsize-1:0] difblkdatapre;

output [ blkcolumnregACCsize-1:0] blkcolumnregACCdata;
wire [ blkcolumnregACCsize-1:0] blkcolumnregACCdata;

output [ blkcolumnregACCsize-1:0] blkcolumnregACCdatanext;
reg [ blkcolumnregACCsize-1:0] blkcolumnregACCdatanext;

output [ mdprowACCsize-1:0] rowsegregdata;
wire [ mdprowACCsize-1:0] rowsegregdata;

```

```

reg [blkcolumnregsize-1:0] blkcolumnreg [ 0 : columnreglength -1 ];
reg [blkcolumnregsize-1:0] blkcolumnregnext [ 0 : columnreglength -1 ];

reg [blkcolumnregACCsize-1:0] blkcolumnregACC [ 0 : columnreglength -1 ];
reg [blkcolumnregACCsize-1:0] blkcolumnregACCnext [ 0 : columnreglength -1 ];

reg [mdprowACCsize-1:0] mdprowACC ;

reg [mdprowACCsize-1:0] mdprowACCnext ;
reg [mdprowACCsize-1:0] rowsegreg [ 0 : rowreglength -1 ];

integer j, k, l, m;
always @ (posedge clock)
begin
  if ( (~difstart) || doneseg )
    begin
      for ( j= 0; j <= columnreglength-1 ; j=j+1 )
        blkcolumnreg [j] <= 0 ;
    end

  if ( done8pix) blkcolumnreg [ blkaddress ] <= difblkdata;

end

assign difblkdatapre = blkcolumnreg [ blkaddress ];
assign blkcolumnregACCdata = blkcolumnregACC [ blkaddress ];

assign rowsegregdata = rowsegreg [ segaddress ];

always @ (difstart or doneseg or dif8pix or difblkdatapre )
begin
  if ( (~difstart) || doneseg )
    begin
      for ( k=0; k <= columnreglength-1 ; k=k+1 )
        blkcolumnregnext [k] = 0 ;
    end

  else
    difblkdata = difblkdatapre + dif8pix;

end

```

```

always @ (posedge clock)
begin
if ( (~difstart) || doneframe )

    begin
        for ( l= 0;l <= columnreglength-1 ; l=l+1 )
            blkcolumnregACC [l] <= 0 ;

        end
    if ( done8pix4lastsegline)
        blkcolumnregACC [ blkaddress ] <= blkcolumnregACCdatanext;

    end

always @ ( difstart or doneframe or difblkdata or blkcolumnregACCdata)
begin

if ( (~difstart) || doneframe )
    begin
        for ( m= 0; m <= columnreglength-1 ; m=m+1 )
            blkcolumnregACCnext [ m ] = 0 ;

        end
    else

        blkcolumnregACCdatanext = blkcolumnregACCdata + difblkdata;

    end

always @ (posedge clock)
begin

mdprowACC <= mdprowACCnext;
if ( doneseg)
rowsegreg [ segaddress ] <= mdprowACC;

end

always @ (difstart or doneseg or done8pix or dif8pix or mdprowACC )

begin
case( { difstart, doneseg, done8pix } )
3'b0xx : mdprowACCnext = 3'b000;
3'b100 : mdprowACCnext = mdprowACC;
3'b101 : mdprowACCnext = mdprowACC + dif8pix;
3'b11x : mdprowACCnext = 3'b000;
default: mdprowACCnext = 3'bxxx;
endcase
end

endmodule // end module

```

Appendix B: Sample of ASIC Simulation Data Output of Motion-region Frame

A sample of the simulation result of the ASIC is attached here. It is divided into two parts for illustration.

The first part of the data of *motion-region frame*, in hexadecimal form.

[illegible]

Continuation showing the lower part of *motion-region frame*, in hexadecimal form.

	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
	0	0	3	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	8	9	3	7	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	0	1	0	4	7	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	2	2	4	0	7	0	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	5	8	1A	9	1	2	3	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	10	26	29	21	22	1F	1A	3	1	4	2	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	0	19	3B	3D	30	37	38	35	12	12	A	C	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	1	E	2	F	38	36	32	2E	8	8	5	3	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
	0	C	25	23	35	34	23	29	D	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2
	0	1	6	17	27	17	2C	3	0	0	4	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
	0	0	5	35	26	23	2B	12	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
	0	2	14	1C	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A
	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

They show the data of one motion-region frame. Every two consecutive rows of data corresponds to one scanline of the frame. The data is graphical represented in binary form in figure 5.9. In that figure, any digital pixel that has a non-zero value is presented as white, otherwise it is black.

Appendix C: Selection of Schematic

The selected schematics of circuit block are presented. They are the modules: control, motion processing element unit (MDP), and motion analysis (storemdpreg). The RAM blocks are separate intellectual property blocks and are not synthesized. One example of a lower level subblock is shown. It is used in MDP as comparator. This was automatically synthesized by Synopsis Designware (Registered trade mark of Synopsys).

Overview of complete circuit

Motoin analysis
(storendpreg)

Subblock:
Comparator used in MDP

810-012
TC

DESIGN: CONTROL CASE 4011.3.3.3	DATE: 8/22/88
Author: J. J. J.	Sheet: 1 of 1