Dissertation

学位請求論文

A Study on High Speed and Low Power Image Sensors with Variable Resolution Scan using Tree Structure of Images

画像信号のツリー構造を用いた 可変解像度走査機能を持つ 高速低消費電力画像センサに関する研究

Supervisor: Professor Kunihiro Asada
指導教官: 浅田邦博教授

Junichi Akita
秋田 純一

Department of Electronic Engineering,
University of Tokyo
東京大学工学部電子工学科

December 19th, 1997
1997 年 12 月 19 日

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A Brief History of Image Sensors' Development

The history of image processing technology in electronics dates from the invention of television system in 1927. From this time, the image processing technology has been studied and developed mainly in terms of television broadcasting technology, such as NTSC or PAL.

In recent years, the development of image processing technology seems to have come at a dramatic turning point, for the development of high definition television (HDTV) or the popularization of "multi-media." For example, in the applications for HDTV, the drastical increase of the quantity of image information makes various difficulties for the real-time image processing technology. In case of the applications for "multi-media," the importance of image signals has coming up, and the forms of the usage of image signals are not only single direction broadcasting such as televisions, but also the bi-directional communications such as television conference systems. The recent image broadcasting systems, such as VOD (Video on Demand) also need the one-to-many broadcasting and the information quality adaptivity for the needs of users. The needs of digital image processing systems have also increased since these development of image processing systems has been based on the development and the popularization of computers and their digital signal processing technologies.

On an image capturing devices, the solid-state imagers have been invented in 1970s, which employ the CCD (charge coupled device) as a signal transformer. The recent development of VLSI technologies enables us to fabricate not only photo-detectors and signal transfer circuits, but also a simple image processing circuits in one image sensor chip, so called "computational sensors." An integration of image processing systems on image sensor has a ability to decrease the quantity of image information coming outside the image sensor chip, which can also overcome the bottleneck of the channel capacity between image sensors and image processing systems.

1.2 Previous Works on "Computational Sensors"

The conventional image processing systems have separated image sensor and image processor connected by the signal channel. Seen in the previous section, the channel capacity is one of the most important problems in image processing systems by the increase of the quantity of image information. One solution for this problem is the integration of image sensor and some of signal processing systems in one chip, which was enabled by the highly developed VLSI technologies. Such image sensors which have the functions of more than simply capturing images are called "smart sensor," or "computational sensor." Most of the recent studies on computational sensors are made in an approach of integrating the functions of pre-processing for images in sensor chips.

In this section, some of previous studies on computational sensors are summerized.

1.2.1 Early vision problem

The pre-processings for images, such as noise reduction, edge detection and so on, are called "early vision problem," which have the following characteristics:

- useful as pre-processing for images.
- can be processed parallelly.

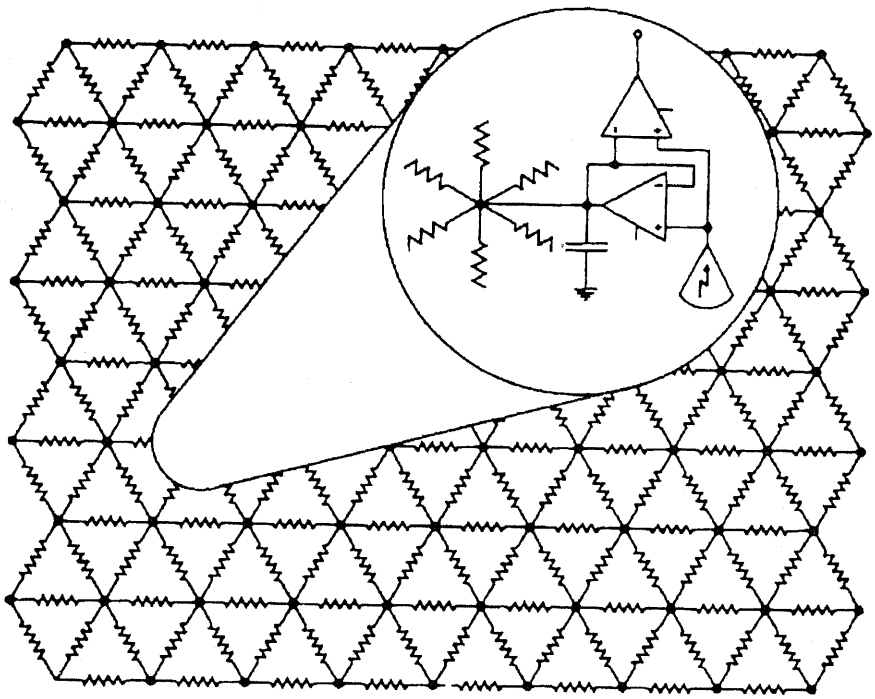


Figure 1.1: Model of "silicon retina."(From [3])

- can be easily mathematically modeled.

These are the reasons why the early vision problems are often studied as the computational sensors. One of the most previous studies in computational sensors are the approach of modeling the structure of retina of creatures[1, 2, 3, 4], The function of early vision processing in these image sensors are implemented by the network of resistors as shown in Figure 1.1, which are called "silicon retina."

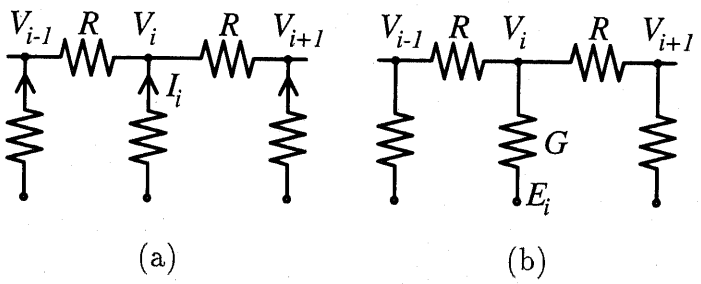


Figure 1.2: Models of one-dimensional resistor network.

Figure 1.2 shows a simple model of the resistor network. Assuming that the potential of the node i is V_i, the current injected to node i is I_i and the resistance between two nodes is R in Figure 1.2(a), the following equation is derived from the Kirchhoff's law.

$$2V_i - V_{i+1} - V_{i-1} = 2RI_i \tag{1.1}$$

Since this equation is equivalent to the discreted Laplacian ∇^2, the following differential equation is derived for the whole area.

$$\nabla^2 V = RI \tag{1.2}$$

Assuming that V_i gives the intensity of a pixel, the distribution of I_i gives the edge-detected images.

On the other hand, the following equation is derived for the resistor network in Figure 1.2(b).

$$2V_i - V_{i+1} - V_{i-1} = RG(E_i - V_i) \tag{1.3}$$

This equation is equivalent to the discrete equation of the following Helmholtz's equation, the V gives the smoothed images of the image given by the distribution of E_i.

$$\nabla^2 V + GRV = GRE \tag{1.4}$$

The network of non-linear resistors is expected to implement the more useful functions[5, 6]. For example, the network of "resistive fuse," which is the resistor that opens for the preset voltages shown in Figure 1.3, can implement the image smoothing with maintaining the contrast at the edge of images[7] as shown in Figure 1.4.

1.2.2 Range finding

One of the most simple methodologies for range finding is "Light-Stripe" method, which is a kind of triangle method using the angle of rotated mirror and the output of image sensor as shown in Figure 1.5. Some computational sensor for range finding

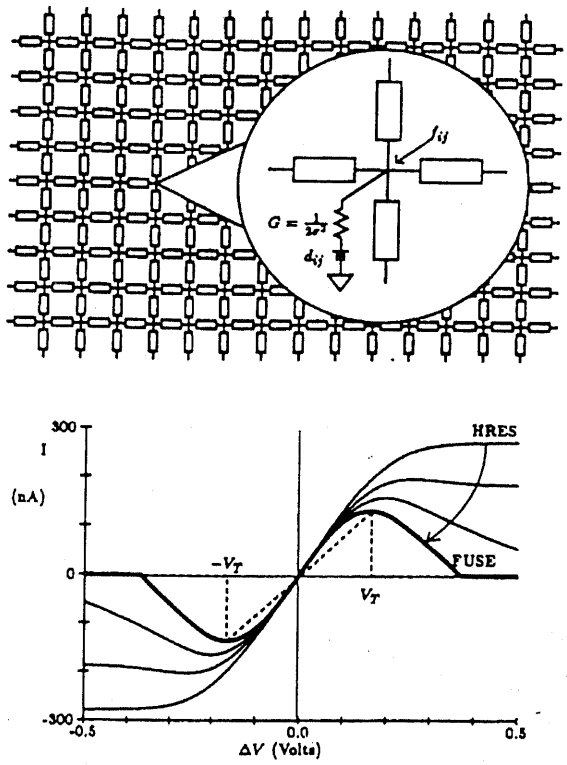
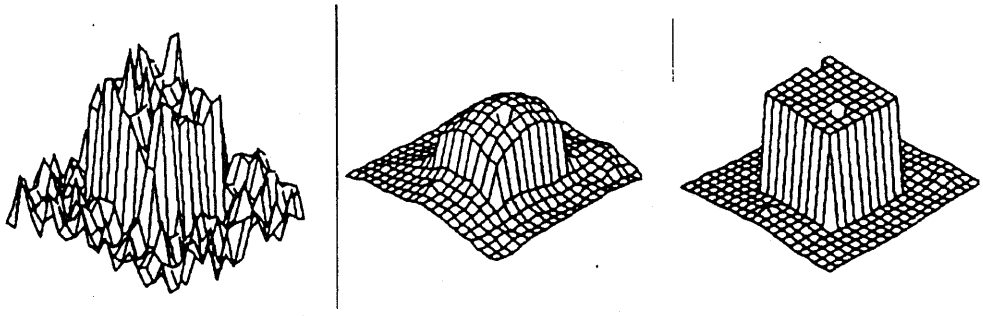


Fig. 6. (a) Schematic diagram of a 20 by 20 pixel surface interpolation and smoothing chip. A rectangular mesh of resisitive fuse elements (shown as rectangles) provides the smoothing and segmentation ability of the network. The data are given as battery values d_{ij} (eq. 6) and the conductance G depends on the variance σ^2 of the additive Gaussian noise assumed to corrupt the data. If no data are available, $G = 0$. The output is the voltage f_{ij} at each node (compare Fig. 4a). Parasitic capacitances (not shown) provide the dynamics. The steady-state of the circuit corresponds to one of the local minima of the non-convex variational functional of eq. (6). (b) Measured $I - V$ relation for different settings of the resistive fuse. The $I - V$ curve can be continuously varied from the hyperbolic tangent of Mead's saturating resistor (HRES) to that of an analog fuse. The $I - V$ curve of a binary fuse is also indicated (dashed line). For a voltage of less than $V_T = \sqrt{\alpha}$ across this two-terminal device, the circuit acts as a resistor with fixed conductance. Above V_T, the current is either abruptly set to zero (binary fuse) or smoothly goes to zero (analog fuse). Independent voltage control lines allow real-time changes of both the slope (over four orders of magnitude) as well as V_T (over one order of magnitude). From Harris.[37]

Figure 1.3: The characteristics of "resistive fuse."(From [7])



(a) (b) (c)

Figure 1.4: Examples of edge enhancement and smoothing using the networks of various resistive elements.(From [7]) (a)original image with noise, (b)output of the network of HRES, (c)output of the network of "resistive fuse."

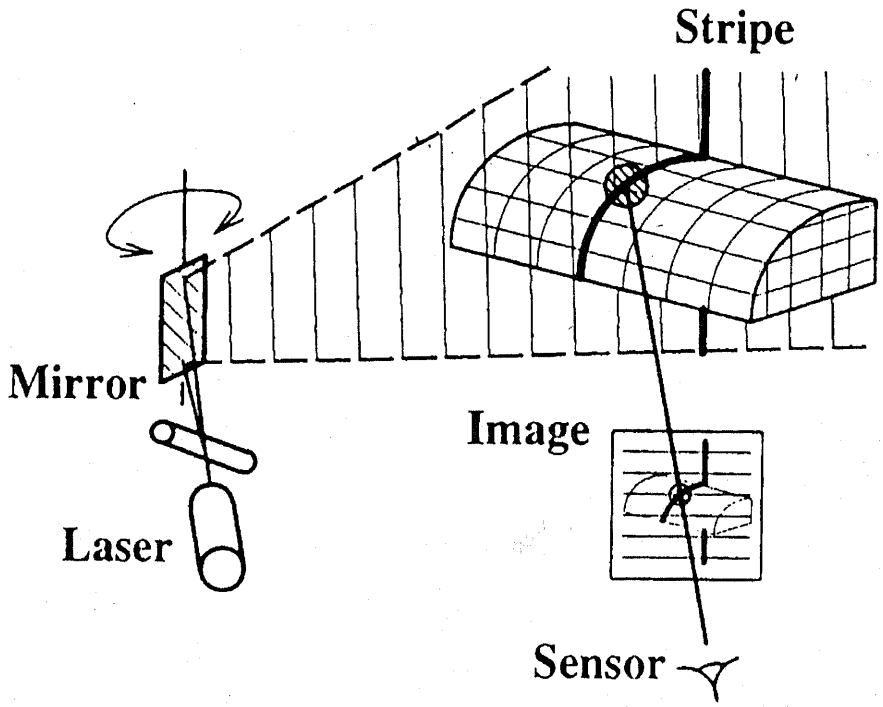


Fig. 1. Traditional light-stripe range imaging.

Figure 1.5: "Light-Stripe" method for range finding. (From [8])

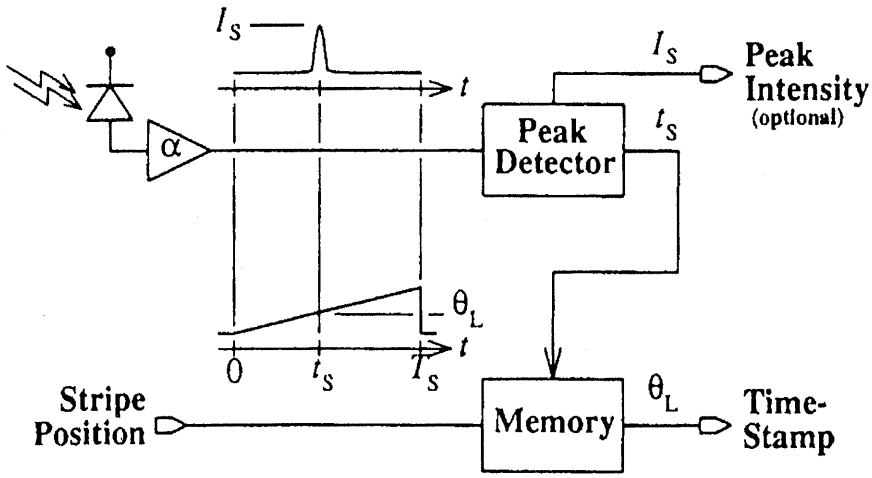


Fig. 4. Basic sensing element block diagram.

Figure 1.6: Pixel circuit for fast range finding computational sensor.(From [8])

are reported. For example, a range finding computational sensor whose pixels are shown in Figure 1.6 is reported, and its range finding system can archive the error less than 0.1% with very fast finding speed[8].

1.2.3 Movie compression

On-chip image compression is adequate for the reduction of the traffic in the channel between the image sensor and the image processing systems, especially in case of the increased quantity of image signals, such as HDTV. Since the algorithms for movie compression needs very high ability of computing, few studies on on-chip movie compression are previously reported. One of the reported studies on on-chip movie compression is the movie-compression sensor by taking the differences between one pixel and its neighbor pixels for the block of 3×3[9]. An computational sensor for movie compression by taking inter-frame difference is also reported[10].

1.2.4 Circuit technologies

Many of the computational sensors employ analog signal processing circuits, not digital signal processing circuits. The advantages of analog signal processing circuits

are follows.

- can implement operation by the physical law using simple circuit elements, for example, add by the Kirchhoff's current law using resistor network.
- can solve equations as a equilibrium of the system about in the time of RC time constant($\sim \mu$s).

Analog signal processing circuits also have disadvantages as follows.

- its accuracy depends on the distribution of fabrication process.
- very sensitive for noise.
- has a difficulty on design system.
- has a difficulty on employing memory elements.

By the reasons above, some studies on computational sensors employing digital signal processing circuits has been reported[11, 12, 26].

1.2.5 Fabrication technologies

Since the solid-state imager has the photo detectors on a focal plain, the fill factor, which is the ratio of the photo detectors' area to the pixels' area, is expected to be kept as high as possible to obtain a high light sensitivity. Computational sensors has an essential problem of low fill factor, because each pixel has both photo detectors and signal processing circuits.

In recent years, the stacked device structure, which is called "three dimensional IC," can have been fabricated by the development of SOI (Silicon On Insulator) technologies. For example, a computational sensor which has 25 photo detectors on top layer, A/D converters for each pixel on middle layer, and signal processing circuits on bottom layer, as shown in Figure 1.7 has been reported.

It is also studied to fabricate three-dimensional structure by pasting bulk chips, and using copper terminal for inter connections between layers[13].

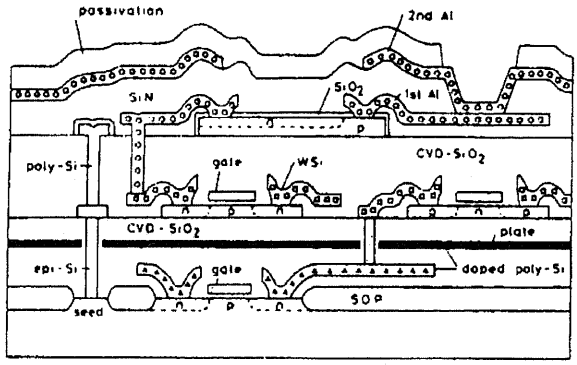


Fig.4 Schematic cross section of the 3-D structure

Figure 1.7: Cross-sectional view of computational sensor with three dimensional structure.(From [11])

1.3 Outline of This Thesis

Seen in the previous sections, the channel capacity between image sensors and image processing systems will become one of the most important problems in near future, while few studies on on-chip image compression are reported. The increase of power consumption and the bottleneck of operation speed of signal outport will also be the important problems in the mega-pixels image sensors.

In this thesis, the studies on a image sensor employing a tree structure for image signals, which has a kind of data compression, that is efficient both power reduction and high speed operation.

Chapter 2 describes the power reduction methodologies for VLSI circuits including image sensors with the probabilistic models of power consumption.

Chapter 3 describes a kind of image encoding method using tree structure to treat image signals. The analytical model of this method also has been described.

In Chapter 4, the applications of the method using tree structure for image signals for some adaptivities of image sensors, such as spatial adaptivity and intensional adaptivity. The applications for movie compression sensor has been described in this chapter.

Chapter 5 describes the two difference implementations of tree-structured image sensor using CMOS technologies, which can archive high speed and low power operation. The implementation of decoder of 1:4 tree code are also discussed. The functions and the circuits of pixel for on-sensor image processing are also discussed in this chapter. The idea of ultra-low power image sensor which uses the energy of light is also described.

Chapter 6 describes the evaluations for tree-structured image sensors. The evaluation of the prototype system using FPGAs and full-custom design of tree-structured image sensors are described. The evaluation of designed full-custom tree-structured image sensors are also discussed in this chapter.

The summary and the conclusion are discussed in Chapter 7.

Chapter 2

Power Reduction Methodologies for Image Sensors and CMOS Circuits

In this chapter, the problems of power consumption in image sensors and CMOS circuits will be discussed. In section 2.1, the models of power consumption in conventional image sensors will be described. In the following section 2.2 and section 2.3, the probabilistic models of power consumption in CMOS combinational logics and sequential circuits will be described. It will be also discussed the power reduction methodologies for CMOS combinational logics and sequential circuits with maintaining the performance in these sections.

2.1 Power Consumption Modeling of Image Sensors

Figure 2.1 describes the typical architecture of CCD transfer image sensors. Image signals are converted to charge in each photo detector, shown as squares in Figure 2.1, and the charge is carried to the vertical transfer CCDs and horizontal transfer CCDs outside to the image sensor.

The clock signals provided to each CCD are usually multi-phase clock signals, and their swings are often very large compared with the CMOS logic circuits. For

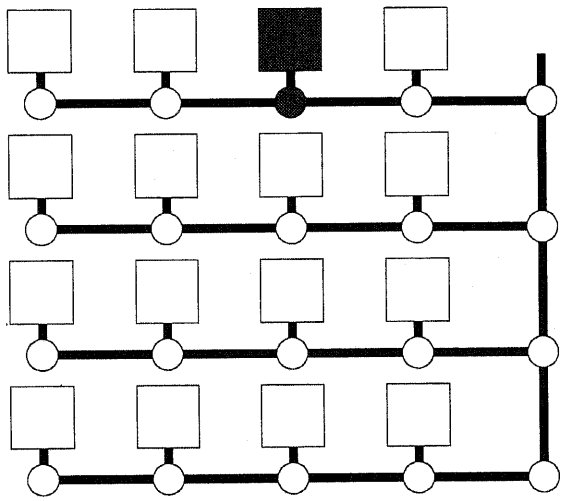


Figure 2.1: The simplified model of CCD transfer image sensors. (The square presents the photo detector, and the circle presents the transfer CCD.)

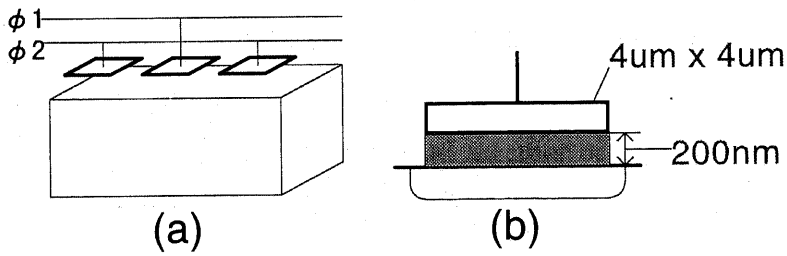


Figure 2.2: Model structure of CCD transfer gate. (a)a part of transfer line, (b)example size of unit CCD transfer gate.

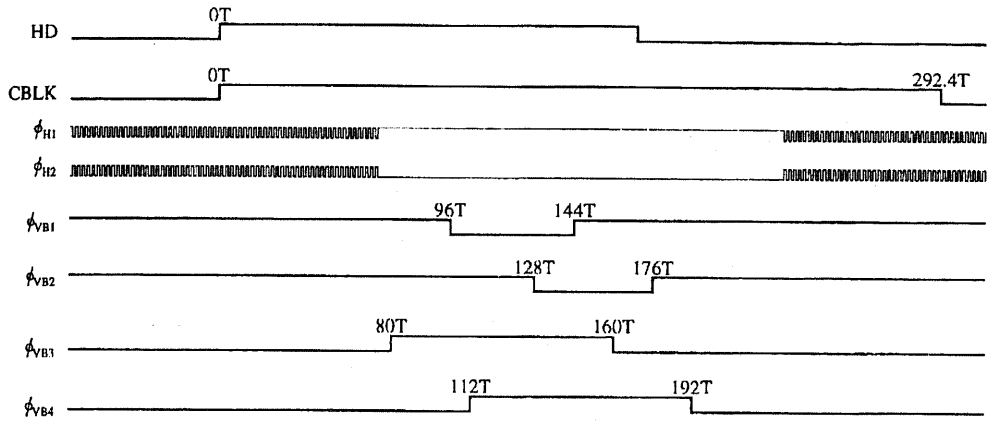


Figure 2.3: Example of multi-phase clocks for CCD devices (From [15])

example, the voltage swing of -7V to 0V is provided for vertical transfer CCDs, and 0V to $+5$V for horizontal transfer CCDs. Such a high clock voltage is needed because of the large size of devices, which is not easy to be scaled down to keep the charge transfer efficiency. (On the other hand, the size of MOSFETs in conventional CMOS logics can be reduced by the advanced fabrication technology, and it results in both power and delay reduction[14].)

The multi-voltage and multi-phase clock signals, for example as shown in Figure 2.3, should make the control circuits more complex.

It is notable that the clock signals are always provided to all transfer CCDs, and the number of transfer CCDs is quite large, it is almost same to the number of pixels, for example about 350,000 for VGA resolution image sensors.

Figure 2.4 shows the model structure of CCD transfer gates in image sensors. Assuming that the vertical and the horizontal number of pixels are both equal to n, the number of vertical and horizontal transfer CCD, n_v and n_h, respectively, are equal to $n_v = n^2$ and $n_h = n$, respectively. In case of scanning pixels, the transfer clocks for vertical CCDs are provided n times, while the transfer clocks for horizontal CCD are provided n^2 times. The total number of clocks provided for vertical and horizontal CCDs are equal to $2n^3$, and the power consumption in transfer CCDs is expected to be proportional to $O(n^3)$, which will be one of the most important problems in mega-pixel CCD image sensors.

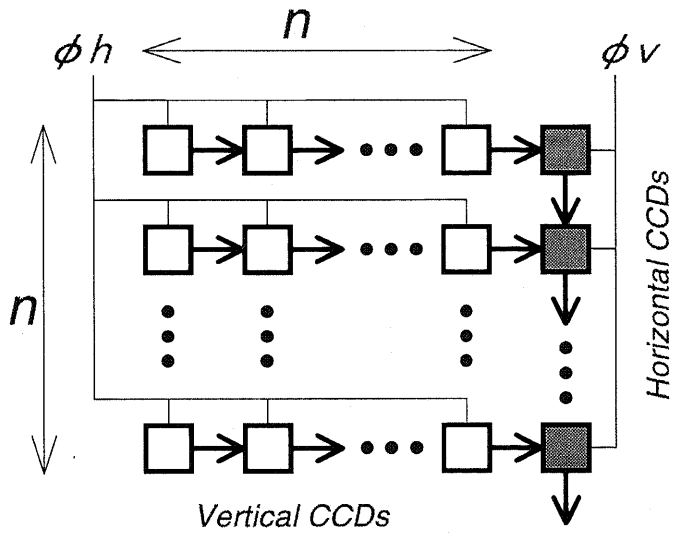


Figure 2.4: Model of CCD image sensor

Discussed in above, the reasons of increasing power consumption of CCD image sensors are follows.

- Clock signals are always provided to the all transfer CCDs. The number of transfer CCDs is almost same to the number of pixels.
- Voltage of clock signals can not be reduced because of the large size of devices. It is difficult to reduce the size of devices to keep the transfer efficiency.

The another type image sensor is based on memory-like architecture as shown in Figure 2.5. The non-CCD image sensor of this architecture has the advantages as follows.

- Just the address lines to the pixels being scanned are activated, and the most of address lines are not activated.
- The row and column decoders are combinational logics, and it can be "scaled down" in order to reduce power consumption. The driving voltage for each pixel can be kept to be equal to the supply voltage of decoder logics, which is lower than that of CCD image sensors.

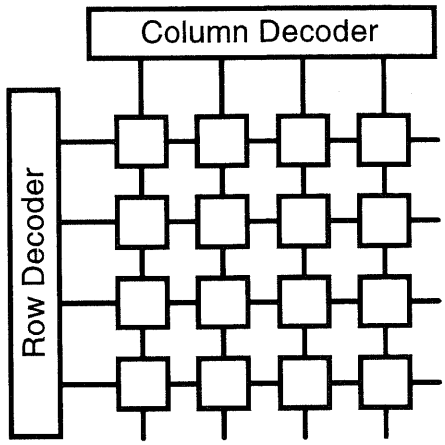


Figure 2.5: Model of non-CCD image sensor. (The squares represents the photo detectors, and the horizontal and vertical lines acrossing the photo detectors are address lines.)

In spite of the switching noise problem, the non-CCD image sensors (they are often called as "CMOS sensors") are expect to be a low power image sensor compared with CCD image sensors.

2.2 Models of Power Consumption in CMOS Combinational Logics and Its Reduction

In non-CCD type image sensors seen in previous section, the conventional logic elements, such as combinational logic gates and sequential circuits including flip-flops are used in non-CCD image sensors for signal scanning circuits. In this section, the power consumption modeling of CMOS combinational logics are described. It is also discussed the power reduction methodologies for combinational logics with keeping circuit function and circuit performance at the design step after logic synthesis.

2.2.1 Probabilistic power model of CMOS combinational logics

Here we assume that the input signals for combinational logic have the following characteristics.

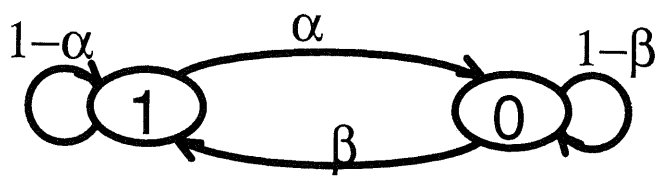


Figure 2.6: Definition of probabilistic parameters, (α, β).

- Input signals change synchronously with the system clock.
- Input signals change probabilistically concerning with the value in the previous cycle. (Markov chain)
- Input signals change independently.

Letting α and β to be conditional probabilities of "0" to "1" and "1" to "0", respectively as shown in Figure 2.6, the probabilities from "0" to "0" and "1" to "1" are equal to $(1-\alpha)$ and $(1-\beta)$, respectively. Here we call the pair of (α, β) as the probabilistic parameters of input signal. Assuming the stable state of input signals, the probability of input signal being "0", P_0 is derived from the following equation.

$$P_0 = P_0(1-\alpha) + (1-P_0)\beta \tag{2.1}$$

From the Equation (2.1), the P_0 and the probability of input being "1", P_1 are derived as follows.

$$P_0 = \frac{\beta}{\alpha+\beta} \tag{2.2}$$

$$P_1 = \frac{\alpha}{\alpha+\beta} \tag{2.3}$$

The load capacitances of logic circuits are charged or discharged along to the input signals, which are the most part of the power consumption of CMOS logics. For example, 2 input NAND gate can be simplified as shown in Figure 2.7. It has two load capacitances, the load capacitance C_L and the internal node capacitance C_i. Assuming that the probabilistic parameters of input x and y are (α_x, β_x) and (α_y, β_y), respectively, the stable probabilities for all combinations of input signals

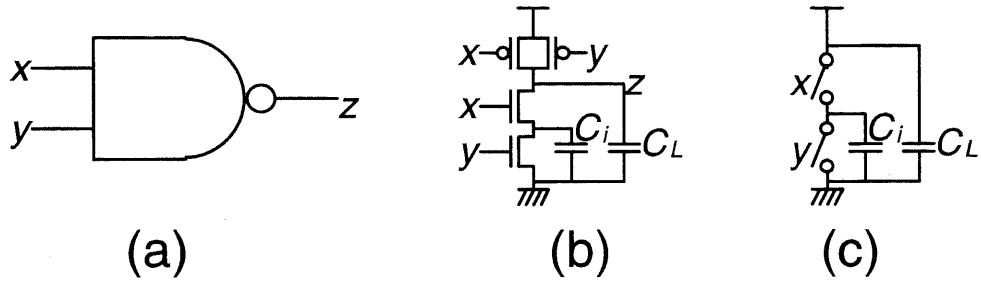


Figure 2.7: Models of 2 input NAND gate. (a)gate model, (b)transistor model, (c)switch model.

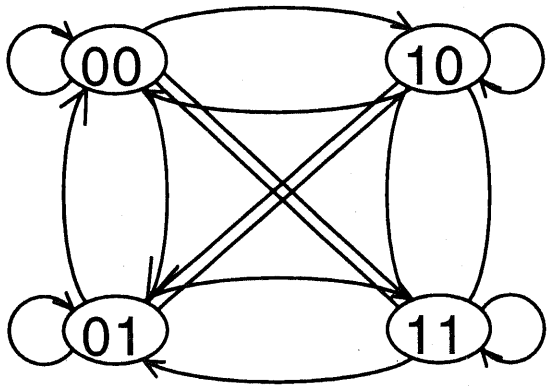


Figure 2.8: Transition diagram of two inputs.

are derived as follows.

$$P_{00} = \frac{\beta_x}{\alpha_x + \beta_x} \cdot \frac{\beta_y}{\alpha_y + \beta_y} \tag{2.4}$$

$$P_{10} = \frac{\alpha_x}{\alpha_x + \beta_x} \cdot \frac{\beta_y}{\alpha_y + \beta_y} \tag{2.5}$$

$$P_{01} = \frac{\beta_x}{\alpha_x + \beta_x} \cdot \frac{\alpha_y}{\alpha_y + \beta_y} \tag{2.6}$$

$$P_{11} = \frac{\alpha_x}{\alpha_x + \beta_x} \cdot \frac{\alpha_y}{\alpha_y + \beta_y} \tag{2.7}$$

Here P_{XY} means the probability of $x = X$ and $y = Y$.

Using these probabilities, the all transition probabilities between two combinations of inputs, as shown in Figure 2.8 can be derived. For example, the probability of transition from $(x, y) = (0, 1)$ to $(x, y) = (1, 1)$ is equal to the product of P_{01} and

the conditional probability of $x = 0$ to $x = 1$, α_x and $y = 1$ to $y = 1$, $1 - \beta_y$, as follows.

$$P_{01\to 11} = P_{01}\alpha_x(1-\beta_y) = \frac{\beta_x}{\alpha_x+\beta_x}\frac{\alpha_y}{\alpha_y+\beta_y}\alpha_x(1-\beta_y)$$

It also can be derived the probabilistic parameters of output z, (α_z, β_z) along to the transition diagram of output node z, as follows.

$$\alpha_z = 1-(1-\beta_x)(1-\beta_y) \tag{2.8}$$

$$\beta_z = \{P_{00}(1-\alpha_x\alpha_y) + P_{10}(1-\alpha_y+\beta_x\alpha_y) + P_{01}(1-\alpha_x+\alpha_x\beta_y)\} / (P_{00}+P_{01}+P_{10}) \tag{2.9}$$

In the switch level model shown in Figure 2.7, both load capacitances have two states; charged state ("1") and discharged state ("0"). The states of load capacitances C_L and C_i change between "1" and "0", and it is easy to be describe the probabilistic parameters of charging/discharging each load capacitance, $(\alpha_{C_L}, \beta_{C_L})$ and $(\alpha_{C_i}, \beta_{C_i})$ as follows.

$$\alpha_{C_L} = \alpha_z \tag{2.10}$$

$$\beta_{C_L} = \beta_z \tag{2.11}$$

$$\alpha_{C_i} = \frac{P_{00}^{(0)}(\alpha_x-\alpha_x\alpha_y) + P_{01}\alpha_x\beta_y + P_{11}(\beta_y-\beta_x\beta_y)}{P_{00}^{(0)}+P_{01}+P_{11}} \tag{2.12}$$

$$\beta_{C_i} = 1-\alpha_y \tag{2.13}$$

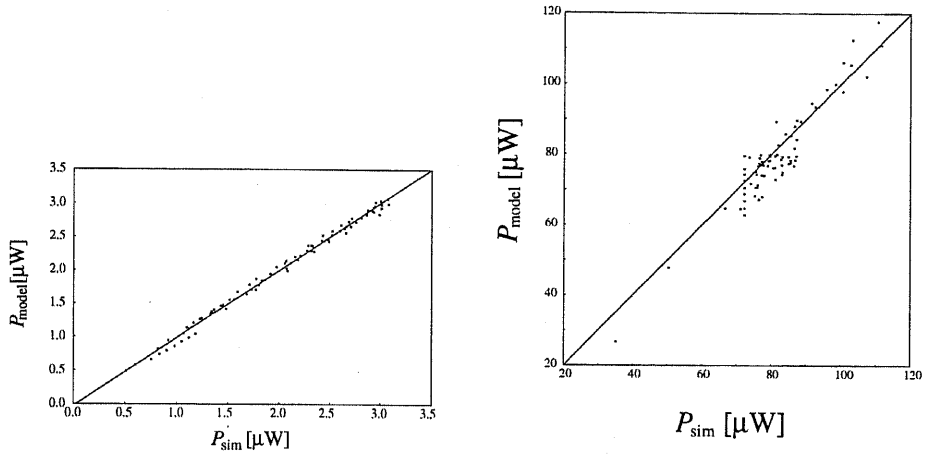
Here $P_{00}^{(0)}$ is the probability of inputs are both "0", with the discharged state of C_i, which is floating in this case, as follows.

$$P_{00}^{(0)} = \frac{P_{01}(1-\alpha_x)\beta_y + P_{11}\beta_x\beta_y}{\alpha_x+\alpha_y-\alpha_x\alpha_y} \tag{2.14}$$

The probabilistic parameters of load capacitances give the probabilities of charging load capacitances, $\overline{N_{C_L}}$ and $\overline{N_{C_i}}$, respectively, as the product of the probability of state "0" and the transition from "0" to "1", as follows.

$$\overline{N_{C_L}} = \frac{\alpha_{C_L}\beta_{C_L}}{\alpha_{C_L}+\beta_{C_L}} \tag{2.15}$$

$$\overline{N_{C_i}} = \frac{\alpha_{C_i}\beta_{C_i}}{\alpha_{C_i}+\beta_{C_i}} \tag{2.16}$$



(a) (b)

Figure 2.9: Calculated power by probabilistic model, Pmodel and `spice` simulation, P_{sim}. (a)2-input NAND gate and (b)1-bit full adder

Using these probabilities, the expectation of power consumption per one system clock cycle, $\overline{P}$ is described as follows.

$$\overline{P} = f\{\overline{N_{C_L}} C_L V_{dd}^2 + \overline{N_{C_i}} C_i V_{dd}(V_{dd} - V_T)\} \tag{2.17}$$

Here f is the system clock frequency, V_{dd} is the supply voltage, and V_T is the threshold voltage of n-channel MOSFETs.

This probabilistic model of power consumption can be extended for a large logic circuits containing many logic gates by assuming that the probabilistic parameters of a input signal are equal to those of the output signals connected to it.

The accuracy of this probabilistic model is checked by `spice` simulation, as shown in Figure 2.9. P_{sim} and P_{model} represent the power consumption calculated from `spice` simulation results[16], and from the probabilistic model by Equation (2.17), respectively. Simulations for both 2-input NAND gate and 1-bit full adder, containing 9 NAND gates shown in Figure 2.10, are shown in Figure 2.9(a) and 2.9(b), respectively. Figure 2.9(b) indicate a small difference between the power calculated by `spice` simulation and that of the probabilistic model in case of 1-bit full adder. The reason is that signals of each gate in the circuits are not always independent each other.

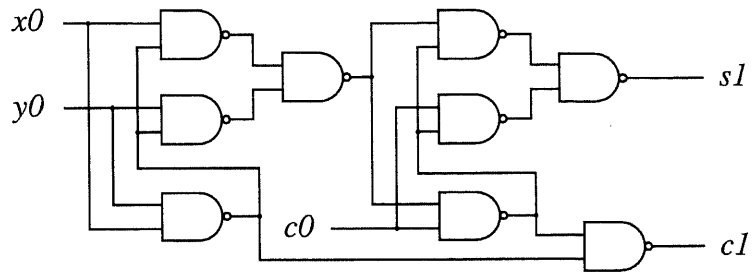


Figure 2.10: Circuit of 1-bit full adder

2.2.2 Power reduction methodologies for CMOS combinational logics

Figure 2.7 gives the hints of reducing power consumption without changing logic function of 2-input NAND gate. The inputs x and y are symmetric logically, though they are asymmetric topologically. The charge and discharge for load capacitance C_L will happen depending on the state of output signal, but charge and discharge for internal capacitance will depend on the state of C_i, since C_i will be opened when $x = y = 0$ and keeps the previous state. This situation will depend on whether input signals S_x and S_y are connected to the input terminals x and y, respectively, or connected to the input terminals y and x, respectively, though these two cases give the identical logic function of 2-input NAND gate.

We can choice the optimal input assignment by comparing the expectation of power calculated from the probabilistic model.

For example, we carried out the simulation for the 1-bit full adder shown in Figure 2.10, which has 9 2-input NAND gates and tree inputs; x_0, y_0 and c_0. There are $2^9 = 512$ possible input signal assignments for all NAND gates, and the maximum power consumption, $U_{\max}$ and the minimum power consumption, $U_{\min}$ in all possible assignments are shown in Figure 2.11. Here we assume that C_L=40fF, C_i=20fF, $\alpha_{x_0} = \beta_{x_0} = 0.1$, $\alpha_{y_0} = \beta_{y_0} = p_{y_0}$ and $\alpha_{c_0} = \beta_{c_0} = p_{c_0}$. The result shows that there are the ratio of $U_{\max}/U_{\min}$ of 1.74 in maximum and 1.10 in minimum, and this implies that the power consumption of 1-bit full adder can be reduced about 40% in maximum by the optimum input assignment, without the modification of logic function.

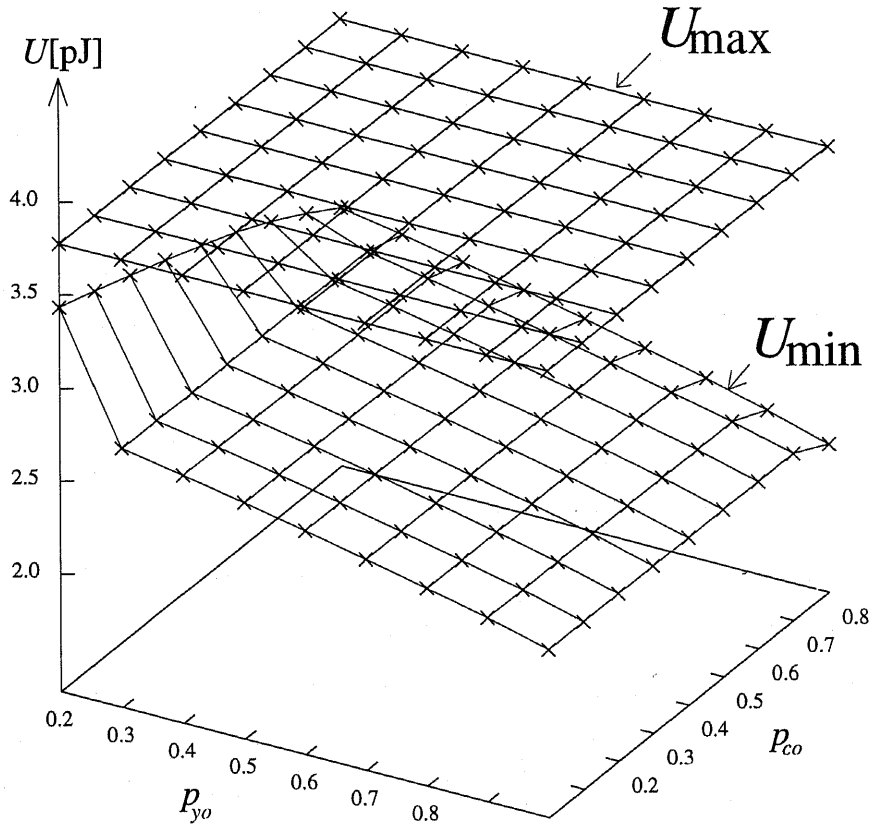


Figure 2.11: Maximum and minimum of power consumption of 1-bit full adder for various probabilistic parameters of input y_0 and c_0.

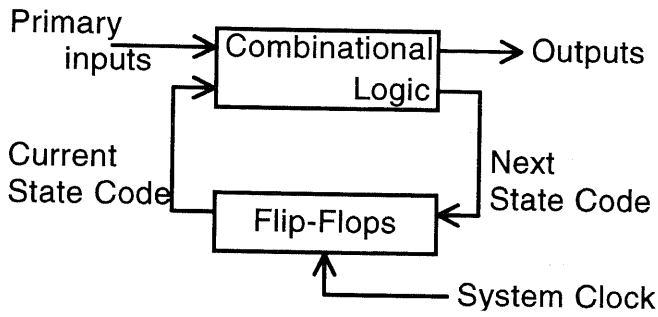


Figure 2.12: Architecture of finite state machine (FSM).

2.3 Models of Power Consumption in CMOS Sequential Circuits and Its Reduction

In this section, the power consumption modeling of CMOS sequential circuits are described. It is also discussed the power reduction methodologies for sequential circuits by the state code assignments for each state.

2.3.1 Probabilistic power model of CMOS sequential circuits

Here we consider the sequential circuit as the finite state machine (FSM), as shown in Figure 2.12. The most part of CMOS circuits' power consumption is the dynamic power consumption, caused by charging and discharging load capacitance, as discussed in the previous section. The dynamic power consumption of sequential circuits consists of two parts; the power consumption of flip-flops, P_{FF} and that of combinational logics, P_{logic}. The power consumption of flip-flops is composed of two parts; the power consumption by charging and discharging the load capacitance of flip-flops based on the state transition, $P_{\mathrm{FF(load)}}$ and that by charging and discharging system clock lines, $P_{\mathrm{FF(clock)}}$.

$P_{\mathrm{FF(load)}}$ depends on the number of flip-flops whose outputs have changed, while $P_{\mathrm{FF(clock)}}$ is almost constant for every clock cycle, whether the all or no flip-flops have made transition.

This fact implies that there is a possibility of reducing power consumption by load capacitance, $P_{\mathrm{FF(load)}}$, since the way to assign of the binary state code for each

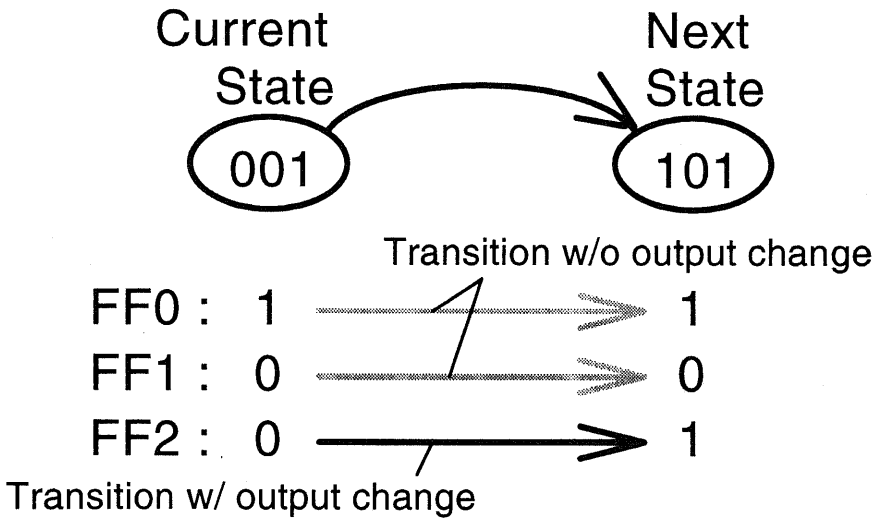


Figure 2.13: Charging and discharging load capacitance at every state transition.

state in the transition diagram is not unique; it is often determined to minimize the complexity of combinational logics.

Assuming that the conditional probabilities of the transition from state i to state i, $\overline{p}_{i,j}$ are known, the probabilities of occupying state i at each cycle, q_i should satisfy the following equation.

$$q_i = \sum_{j=1}^{N} \overline{p}_{i,j} \cdot q_j \tag{2.18}$$

Here N is the number of states in the transition diagram. Using the matrix $\overline{\boldsymbol{P}}$, whose (i,j) element is $\overline{p}_{i,j}$ and the vector $\boldsymbol{q}$, whose i element is q_i, the Equation (2.18) can be described as follows.

$$\boldsymbol{q} = \overline{\boldsymbol{P}}\boldsymbol{q} \tag{2.19}$$

The solution of Equation (2.19) under the condition of $\sum q_i = 1$ gives the state probability of each state, q_i. The product of q_j and $\overline{p}_{i,j}$, $p_{i,j} = \overline{p}_{i,j} \cdot q_i$ gives the transition probability at each clock cycle.

It is notable that the number of flip-flops whose output have changed at clock cycle is equal to the Hamming distance, $d_{i,j}$ between two state. The expectation of the number of flip-flops whose output have changed at clock cycle should be equal to the mean Hamming distance, $\overline{d}$, as is described as follows.

$$\overline{d} = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{i,j} \cdot d_{i,j} \tag{2.20}$$

This $\overline{d}$ is expected to be as an index of the power consumption in flip-flops.

2.3.2 Power reduction methodologies for CMOS sequential circuits

The simple methodology of reducing P_{FF} in the sequential circuits is to find the binary state code assignment for each state which minimizes the mean Hamming distance, $\overline{d}$. There are $N!$ possible assignments of state codes for all N states, and it is undesirable to try the all possible assignments to find the assignment minimizing $\overline{d}$ for N states in practical design.

The alternative way to find the optimal state code assignment is the following heuristic algorithm.

1. calculate transition probabilities, $p_{i,j}$ and add $p_{i,j}$ and $p_{j,i}$, as $\tilde{p}_{i,j}$.

2. sort $\tilde{p}_{i,j}$ in order.

3. assign "0"(decimal) and "1"(decimal) for state i and j whose $\tilde{p}_{i,j}$ is the largest.

4. assign to the state i whose state code has not assigned yet, as one of the available state codes whose Hamming distance from state j is minimum, in order of $\tilde{p}_{i,j}$.

The simulation results shows that this heuristic algorithm can give the optimal assignment, with the error of about 4% against the optimum assignment by the exhaustive method, with the calculation time proportional to about $O(N^3)$ against about $O(N!)$ of the exhaustive method.

The simulation results of power consumption and the number of gates in combinational logics by this heuristic assignment compared with both the random assignment and the assignment aiming to minimize the complexity of the combinational logic[17] are shown in Table 2.1. The results shows about -10% power reduction against the conventional assignment for minimize logic complexity, and the number

	compared with	
	random assignment	logic minimize assignment
Power in flip-flops	−40%	−30%
Power in combinational logic	—	−11%
Total Power	—	−10%
# of gates	—	+11%

Table 2.1: The comparison of power in each part and the number of gates by the assignment for minimum $\overline{d}$.

of gates in combinational logic increase about 10%, but the total increase of hardware costs will be smaller than this case, since the number of flip-flops is equal in each assignments.

2.4 Summary and Conclusion

In this chapter we discussed the power consumption in image sensors, both CCD sensors and "CMOS" sensors. This chapter also has described the probabilistic model of power consumption of both the combinational logics and the sequential circuits. It is shown that these models are adequate to analytically express the expectation of power consumption. It is also proposed the methodologies to reduce power consumption of both the combinational logics and the sequential circuits without modifying the designed functions. The efficiency of power reduction in the combinational logics is up to about −30% against the conventional random input signal assignment. The efficiency of power reduction in the sequential circuits is up to about −10% against the conventional assignment aiming to minimize the complexity of combinational logics.

Chapter 3

Tree Structure of Image Signals for Image Sensors

This chapter will describe the idea of using tree structure of image data for a kind of data compression. The analytical model of tree structure and its evaluation will be discussed in section 3.2 and section 3.3. It will be also discussed the decoding algorithms and its implementations of encoded codes by tree structure in section 3.4.

3.1 Selective Scanning of Image Signals and Tree Structure

3.1.1 Image scanning for selected areas

The procedure of CCD image sensors to read out the image data is to read out the all images of photo detectors regardless of its value, which is called "raster scan." In the raster scan, all the pixels have be scanned, even in case of the distributed data with two-dimensional correlation, as shown in Figure 3.1(a), and it will result in redundant scan steps for the large white areas. It will be very remarkable in the extreme case of very few pixels are active, which is often caused by taking the effective pixels of inter-frame difference for movie compression.

On the other hand, when we see something by our eyes, we will make attentions

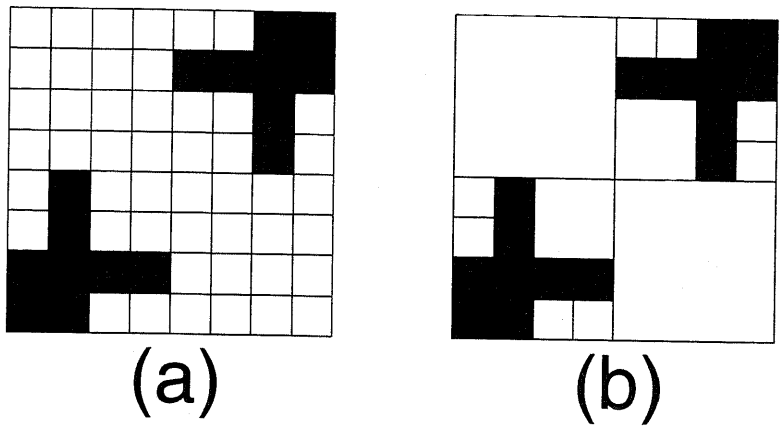


Figure 3.1: Sample of distributed image signals with two-dimensional correlation(a), and the area division for its distribution.

for the restricted area in the view and obtain the detail information from there, not always get the sharp images of whole area. The application of raster scan for such the scan for the restricted area will also result in redundant cycles for the non-interested areas.

3.1.2 Tree structure of image signals for selective scan

In such case of the image in Figure 3.1(a), one of the idea to reduce the redundant cycle is to divide the areas according to whether the divided areas contain the interesting data or not, for example, as shown in Figure 3.1(b).

Here we propose to apply the tree structure for image signals. The tree contains nodes, which have b nodes in the lower level and connected to the node in the higher level, and the nodes in the lowest level have b pixels, whose value is binary, in its lower level. The all connections between two nodes have three signal channels; the *start* signal to order the scan for its lower node, *completion* signal to inform the finish of its scan for its higher node, and *value* signal to inform its value for its higher node. (They are summerized in Figure 3.2.) We define the functions of each nodes as follows.

- It has the value of logical-OR of its lower b nodes. The node in the lowest level has the value of logical-OR of the b pixels.

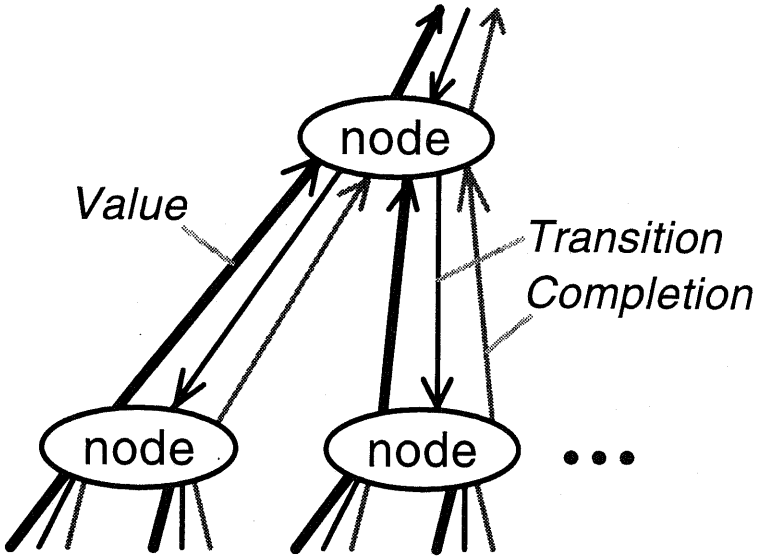
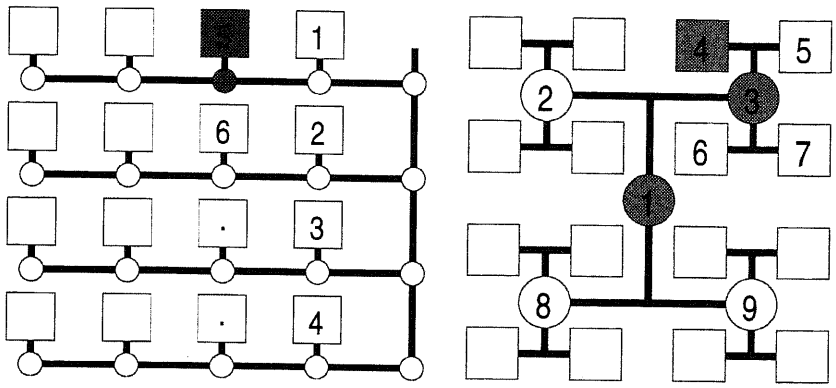


Figure 3.2: Signals of the nodes in the tree structure.

- It returns its value when it receives *start* signal from the node in the higher level.
- When its value is "1", it implies that there are at least one "1" in its lower level, and executes the scan by activating *start* signals of its lower b nodes in order. If the scanned lower node node has returned "1", it will wait for the finish of the lower node's scan until it returns *completion* signal. When the all scans of the lower b nodes have finished, it returns *completion* signal to its higher nodes.
- When its value is "0", it implies that there are no "1"s in its lower level and all the descending nodes to the pixels, and finishes the scan by returning *completion* signal.
- It has $(b+1)$ internal states; `W` for waiting state, and `Si`$(i = 1, 2, ...b)$ for the state of scanning the i-th nodes in its lower level.

For example, the raster scan of 4×4 pixels as shown in Figure 3.3(a) will proceed in order as shown in figure, and it will make a 16 bits code, which is equal to the number of pixels. (Note that white pixels or nodes represent "0" and gray pixels or



(a) (b)

Figure 3.3: The two procedures of image scan; (a)conventional raster scan, (b)scan using tree structure.(The square and the circle represents the pixels and scanning circuits, respectively. The numbers in the figure represents the steps of scan.)

nodes represent "1".) On the other hand, the scan using the tree structure of the nodes defined above, as shown in Figure 3.3(b) will proceeds as the following steps. (Note that $b = 4$ is assumed in this case.)

1. First the highest nodes 1 is scanned, and it returns "1".

2. Next, the scan proceeds to the second level, the node 2 at first. The node of 2 returns "0", and no more scan for its lower level are needed.

3. The scan proceeds to the node 3, and it returns "1". According to this value, the following four pixels are scanned in order, 4, 5, 6, and 7, and then the scan for node 3 has finished.

4. Next, the node 8 and 9 are scanned in order, and both of them return "0", and then the all scans have finished.

In the above procedure of scan using tree structure, we obtain 9 bits code, which is shorter than that of the raster scan. This scan procedure using tree structure (we call "tree scan" afterward), will have the following advantages.

- The scan step for the area containing all 0s are drastically skipped, which is a kind of data compression.

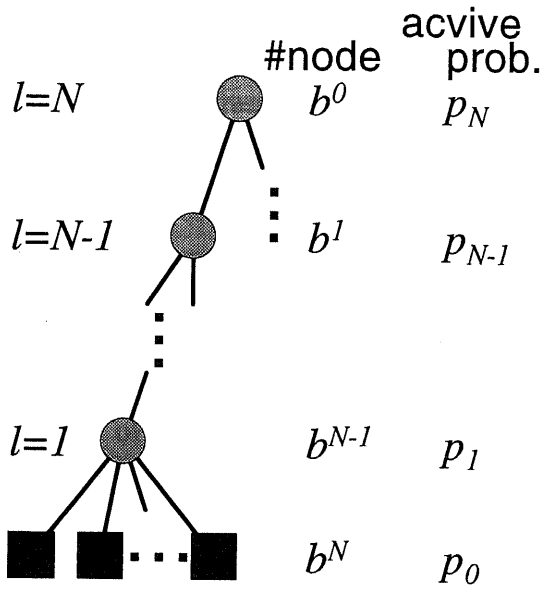


Figure 3.4: Analytical model of tree structure

- Only the signal path from the nodes or pixels scanned to the outside of image sensor is activated, and most of signal pathes rest are not activated, which will be effective for power reduction.

It is also notable that the more the tree scan step proceeds to the lower level, the smaller divided areas, or precise details, are scanned. This characteristic of tree scan gives the idea of executing the scan of the needed resolution for the needed sub-areas, or "adaptive resolution scan", which will be discussed in section 4.2.

3.2 Model and Analysis of the Scan Procedure of Tree-Structured Image Signals

3.2.1 Model of tree structure of image signals

We consider the model of tree structure discussed in the previous section as shown in Figure 3.4. Here b is the number of nodes connected under one node, and N is the number of levels, and p_l is the probability that the node in l-th level has the value of "1", and the number of pixels is expressed as b^N. Note that p_0 is the probability

that the pixel has the value of "1", or it is active. The node in l-th level has the value of "1" when at least one of b nodes in its lower level has the value of "1", and then p_l should satisfy the following equation.

$$p_l = 1 - (1 - p_{l-1})^b \tag{3.1}$$

Solving the Equation (3.1), p_l is expressed as follows.

$$p_l = 1 - (1 - p_0)^{b^l} \tag{3.2}$$

3.2.2 Mean code length of tree-scan

Here we can calculate the expectation of the number of steps in the procedure of the tree scan, or the expectation of code length $\overline{L}$ as the following procedures.

1. First, the node at the highest level, $l = N$ is scanned, and $\overline{L} = 1$.
2. If the node at the highest level has the value of "1", the b nodes in $l = N - 1$ are scanned, and $\overline{L} = \overline{L} + b$.
3. The following scan are executed just for the nodes whose value is "1" in b nodes at $l = N - 1$, which are $b \cdot p_{N-1}$ nodes, and $\overline{L} = \overline{L} + b \cdot b \cdot p_{N-1}$.
4. The following scan are executed just for the nodes whose value is "1" in b^2 nodes at $l = N - 2$, which are $b^2 \cdot p_{N-2}$ nodes, and $\overline{L} = \overline{L} + b \cdot b^2 \cdot p_{N-2}$.

Iterating the procedures above, the expectation of code length, $\overline{L}$ is derived as follows.

$$\begin{aligned} \overline{L} &= 1 + \sum_{l=1}^{N} b^{N-l+1} p_l \\ &= 1 + \sum_{l=1}^{N} b^{N-l+1} \{1 - (1 - p_0)^{b^l}\} \end{aligned} \tag{3.3}$$

The relations of $\overline{L}$ and the active probability of pixel, p_0 are shown in Figure 3.5 for various b, in case of the number of pixels, b^N is 1,024. The results in Figure 3.5 indicates that the tree structure with the number of branches, b is 4 gives the shortest code length for all p_0, and thus we treat the tree structure with $b = 4$, we

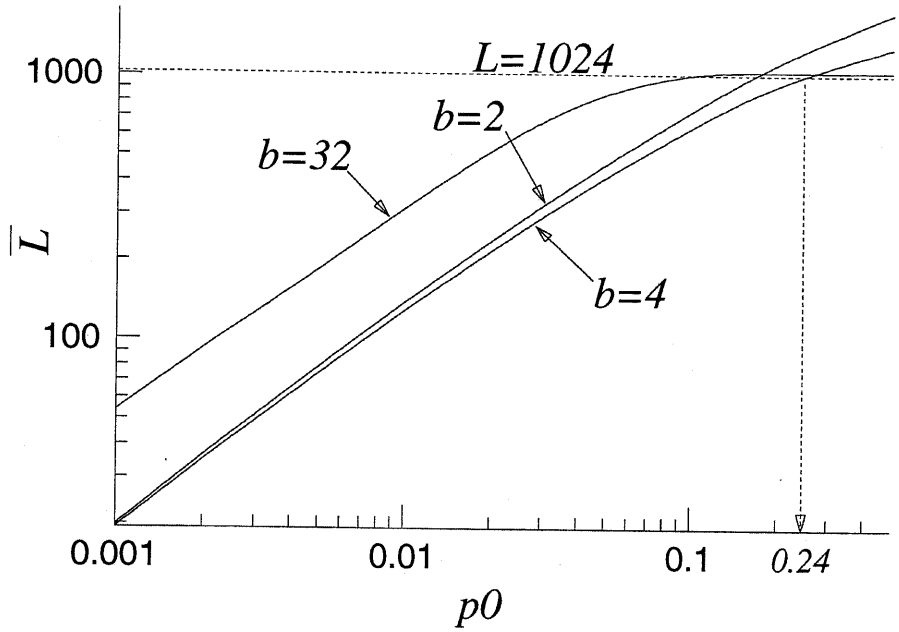


Figure 3.5: The relations of mean code length,$\overline{L}$ and the active probability of pixel, p_0 for various b.

call "1:4 tree structure" afterward. It is also indicated that $\overline{L}$ of 1:4 tree scan is shorter than the length of raster scan, 1024, in case of smaller p_0 than 0.24. (These results were derived not only for 1,024 pixels, but also for the more pixels.)

The number of branches, b and the number of levels, N are correlated through the number of pixels, n, as $n = b^N$, or $N = \log_b n$. Using this equation, Equation (3.3) can be easily extended to the case of the various b as follows.

$$\overline{L} = 1 + \sum_{l=1}^{\lfloor \log_b n \rfloor} b^{\log_b n - l + 1}\{1 - (1 - p_0)^{b^l}\} \tag{3.4}$$

Figure 3.6 shows the optimum b which gives the shortest $\overline{L}$ for given p_0. For large p_0, the scan steps will be executed for almost all nodes, and the case which the total number of nodes is small, or b is large, gives the shortest $\overline{L}$. For small p_0, the optimum b for various p_0 is between 3 and 4, and it indicates the properness to employ 1:4 tree structure in terms of minimizing $\overline{L}$.

In Equation (3.3), we assume no spatial correlations between pixels, but the pixels in usual images are expected to have distributions with the spatial correlations. The

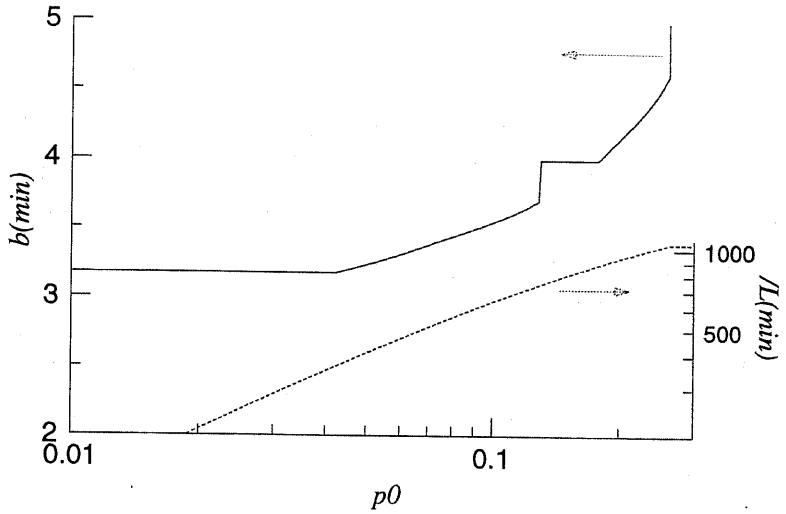


Figure 3.6: The optimum b which gives the shortest $\overline{L}$ for various p_0, and $\overline{L}$.

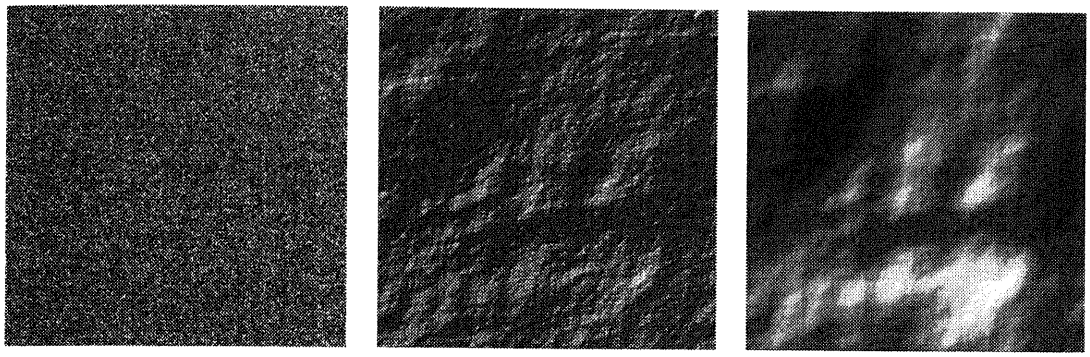
tree scan for two dimensional images is executed for the smaller divided areas as the step of scan proceeds to the lower level, and we can say that this procedure has a characteristic of gathering the pixels in the target sub-areas at each step. It is expected that $\overline{L}$ will be smaller for real images with spatially correlated pixels, than the case without spatial correlations as derived in Equation (3.3), or the white noise images containing very high spatial frequency.

We have generated the random images whose spatial power spectrums are proportional to $1/f^n$, where f is the spatial frequency, and carried out the procedure of 1:4 tree scan for them. Note that the images with larger n contains the less component of higher spatial frequency, as shown in Figure 3.7. Power spectrum $P(f)$ of two dimensional images $p(x, y)$ is defined by the following equation.

$$P(f) = P(\sqrt{k^2 + l^2}) \propto \left| \iint p(x,y) e^{j(kx+ly)} dxdy \right| \tag{3.5}$$

The random images whose power spectrum $P(f) = 1/f^n$ are given by invert Fourier transform of $e^{j\theta_f}/f$, where θ_f is the phase of each frequency components, and they should be determined randomly.

The results of calculating $\overline{L}$ for various images are shown in Figure 3.8. (The number of pixels is $2^{16} = 65,536$.) It is known that the spatial power spectrum of usual real images has a tendency to having between $1/f^{-1}$ and $1/f^{-2}$, and the



(a) (b) (c)

Figure 3.7: Sample of random images with various spatial power spectrum. (a)$1/f^0$, (b)$1/f^{-1}$, and (c)$1/f^{-2}$.

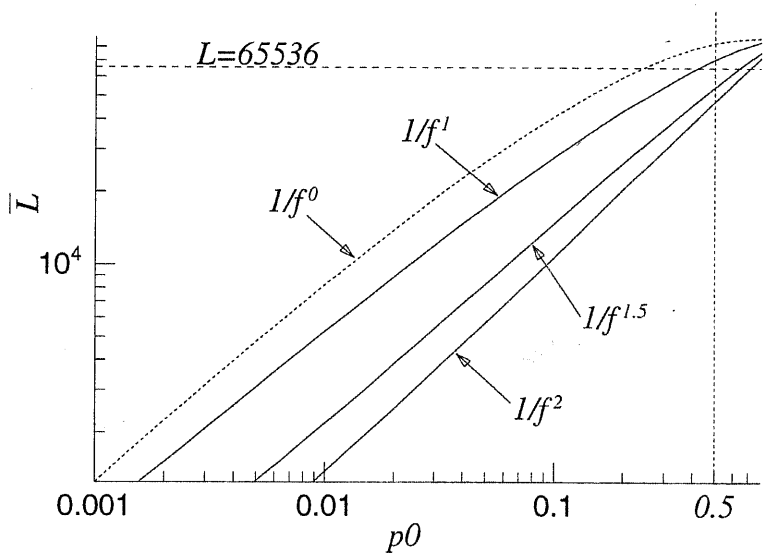


Figure 3.8: Calculated $\overline{L}$ for the images with various spatial power spectrum.

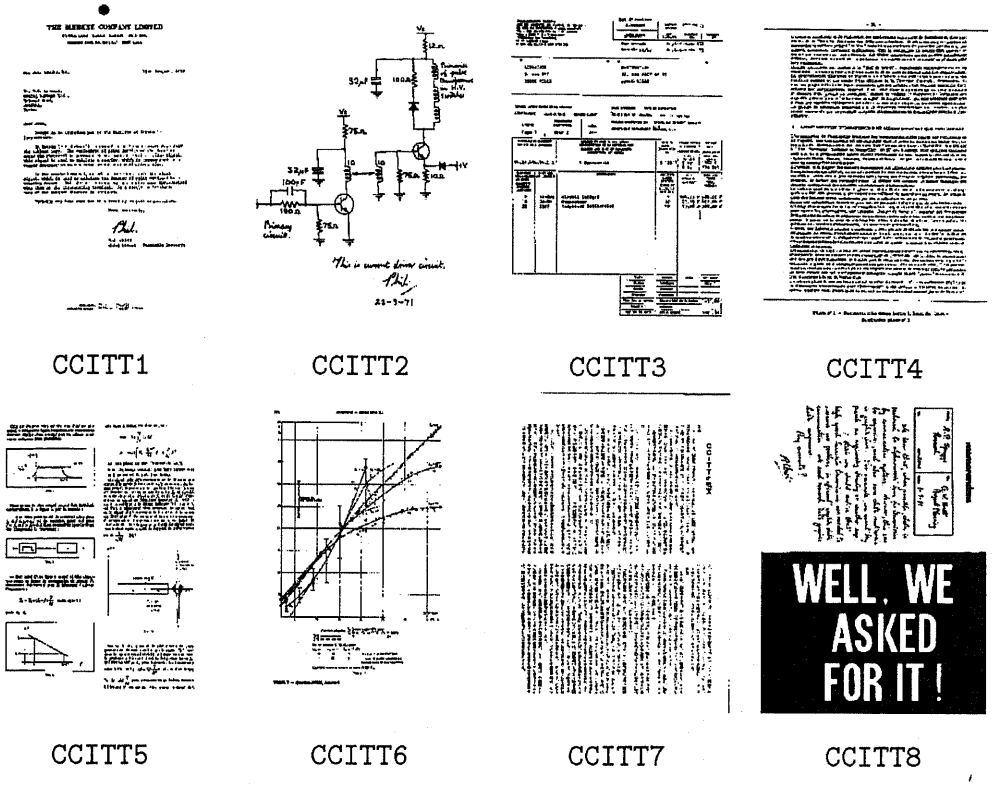


Figure 3.9: ITU-T facsimile standard images

possibility was indicated that $\overline{L}$ using 1:4 tree structure will be shorter than that of the raster scan, even in case of p_0 is larger than 0.5, or the more than half of pixels are black in usual real images.

The history of the idea applying quad tree structure for image signals is very old[18], but the efficiency applying quad tree for image signals was derived in terms of the image encoding efficiency discussed in above.

3.3 Experiments of Tree-Scanning for Images

In this section, the simulation results of 1:4 tree scanning for various real images are shown, with comparing the other scanning methods; the raster scan or run-length compression.

3.3.1 Tree scanning results for binary still images

Table 3.1: Scanning results for ITU-T facsimile standard binary images. (L_{t0} is the 1:4 tree code length per pixel, H_{r0} is the run-length entropy per pixel, and p_0 is the ratio of black pixels.)

Name	p_0	L_{t0} [bit/pix]	H_{r0} [bit/pix]
CCITT-1	0.037	0.175	0.178
CCITT-2	0.045	0.159	0.240
CCITT-3	0.080	0.310	0.304
CCITT-4	0.141	0.573	0.452
CCITT-5	0.075	0.296	0.327
CCITT-6	0.049	0.188	0.269
CCITT-7	0.105	0.460	0.534
CCITT-8	0.450	0.753	0.313

The 1:4 tree scan was carried out for the standard binary images for the evaluation of facsimile given by ITU-T (formerly CCITT) as shown in Figure 3.9.

The run-length compression is often used to scan binary images by one line, and to encode for the obtained binary bits. Some encoding code for run-length compression are known, for example MH (modified Huffman) code which is employed in G3 facsimile standard, and the lower bound of compression efficiency by one dimensional run-length compression is given by the following run-length entropy per pixel, H_{r0}.

$$H_{r0} = H_r/T = -\sum_{i=1}^{N} P_i \log_2 P_i \;/\; \sum_{i=1}^{N} iP_i \tag{3.6}$$

Here P_i is the relative frequency of run whose length is i, H_r is the run-length entropy, and T is the mean run length.

The scanning results are shown in Table 3.1 with the comparison with run-length compression. Table 3.1 shows that 1:4 tree scan is more effective especially in case of smaller p_0, which is just $1/2$ to $1/7$ against that of the raster scan. It is also notable that 1:4 tree scan is often more effective than run-length compression in most cases, especially in case of p_0 is small.

Table 3.2: Scanning results for inter-frame difference of movies. (L_{t0} is the 1:4 tree code length per pixel, H_{r0} is the run-length entropy per pixel, and $\overline{p_0^m}$ is the ratio of black pixels.)

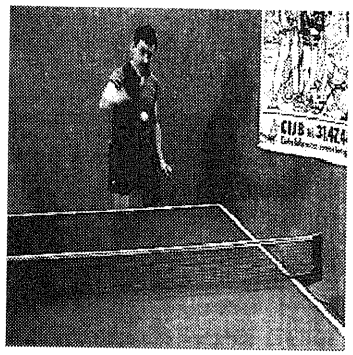
name	$\overline{p_0^m}$	L_{t0} [bit/pix]	H_{r0} [bit/pix]
MissAmerica	0.011	0.055	0.106
TableTennis	0.114	0.454	0.584
FlowerGarden	0.172	0.601	0.724
Neck	0.017	0.067	0.125
Rail	0.036	0.122	0.149

3.3.2 Tree scanning results for moving pictures

We carried out the 1:4 tree scan for inter-frame difference of some example movies. It is one of the most simple algorithms for movie compression to take inter-frame differences and scan just the pixels which have differences. Some example movie sequences are digitized to binary images, and the scan was carried out for the inter-frame differences, which is just exclusive OR of pixels in two succeeding frames. The scanning results are shown in Table 3.2 with the comparison of run-length compression. Here "MissAmerica", "TableTennis", and "FlowerGarden" are the standard of movies, and "Neck" and "Rail" are the sample movies captured by Hatori-Aizawa Laboratory, University of Tokyo, and the cuts of movies are shown in Figure 3.10. In "MissAmerica," the upper half of a lady is shown, and she talks with the motion of the body slowly. In "TableTennis," the view of table tennis game is shown, and it has two drastical scene change, with zoom out. In "FlowerGarden," the view of the garden is shown, and the whole scene moves slowly by the move of camera. In "Neck," the toy of moving rabbit is shown, and its head and hands are moving quickly. In "Rail", the toy of train is shown, and it moves slowly.

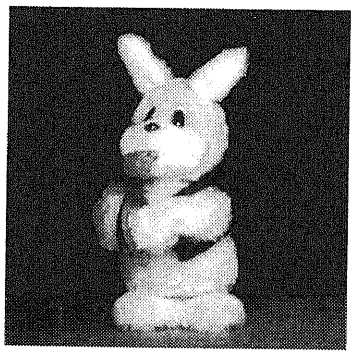
Table 3.2 indicates that only a few percents of pixels should be scanned, and 1:4 tree code length is about 1/14 to 1/8 over that of the raster scan. It is also indicated that the 1:4 tree code length is about 1/2 at maximum over that of the run-length

"MissAmerica"

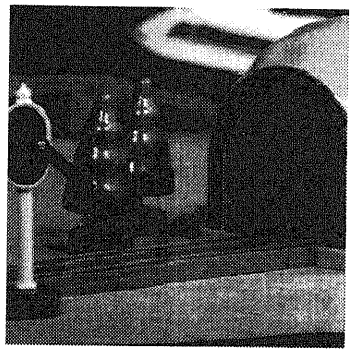


"TableTennis"

"FlowerGarden"



"Neck"



"Rail"

Figure 3.10: Scene cuts of used sample movies

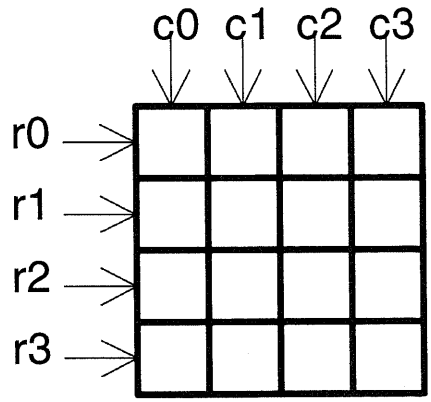


Figure 3.11: Example of 4×4 memory cells and the selection signal lines.

compression.

3.4 Decoding procedure of tree-scanned image signals

The encoded code by 1:4 tree scan can be decoded to binary two dimensional images, of course, since 1:4 tree scan is reversible procedure. Assuming that the decoded binary images are stored in two dimensional array of memory cells, and they are selected by the selection signal lines controlled by the external controller, as shown in Figure 3.11. The memory cells at (i, j) are selected when both column select line c_i and row select line r_j are activated, and the binary data is stored for all of the selected memory cells. This architecture of memory cells is similar to that of the conventional data memory, such as DRAM, but the difference is that more than two select signal lines can be activated so as to select the areas.

For example, the 1:4 tree code of `101101000` can be decoded by the following procedures, as shown in Figure 3.12.

1. First bit of 1:4 tree code, "1" represents the logical-OR of whole area, and all the memory cells are selected to be stored "1", as shown in Figure 3.12(a).

2. The following bit of code, "0" represents that the logical-OR of the upper-left 2×2 area is "0", and it is stored to this area with activating the corresponding select lines, as shown in Figure 3.12(b).

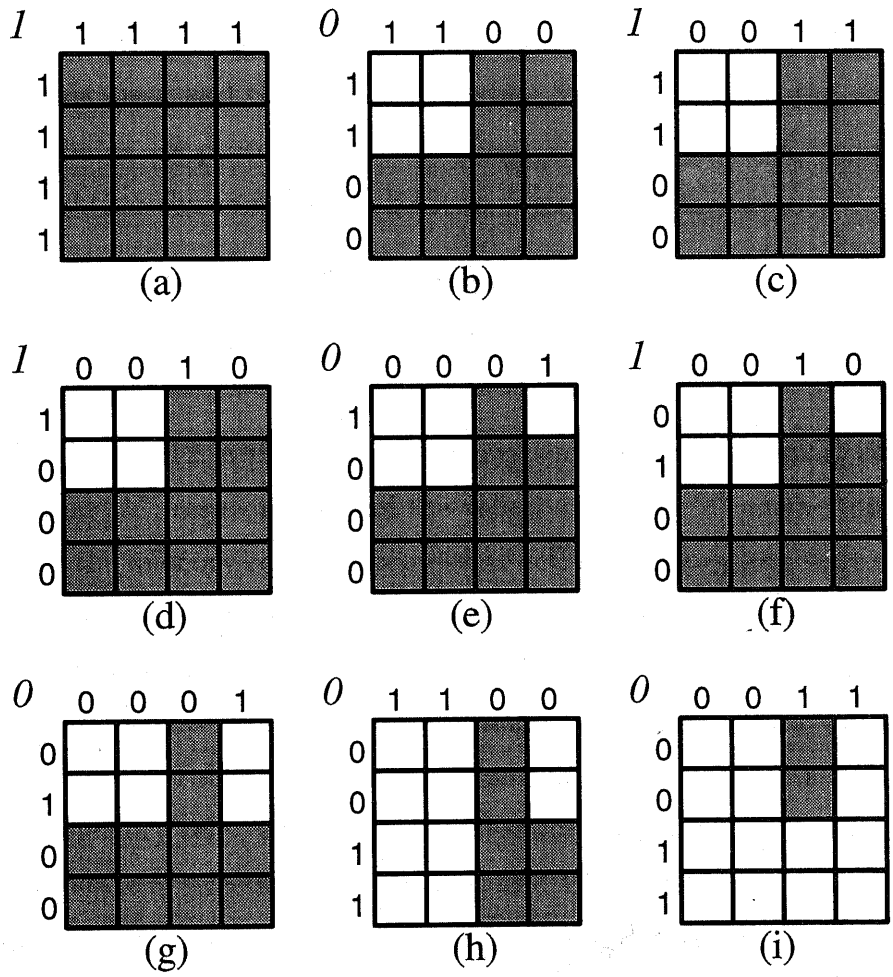


Figure 3.12: Example of decoding procedure of 1:4 tree code.

Table 3.3: States of automata in the decoding process shown in Figure 3.12.

1:4 tree code	1	0	1	1	0	1	0	0	0
upper(s2)	W	A	B	B	B	B	B	C	D
lower(s1)	W	W	W	A	B	C	D	W	W

3. Next bit of 1:4 tree code of "1" are stored to the upper-right 2×2 area, as shown in Figure 3.12(c). This bit means that there are at least one "1" bit in this area, and the decode procedure for 1×1 area in the upper-right 2×2 subarea are processed in order of upper-left, upper-right, lower-left, and lower-right, as shown in Figure 3.12(d), (e), (f), and (g).

4. The following two bits of "00" represent the lower-left and lower-right 2×2 area as all 0, and they are stored, respectively, as shown in Figure 3.12(h) and (i).

Note that in Figure 3.12, white squares and gray squares are the memory cells whose value is "0" and "1", respectively, and the values of "1" or "0" on the top side and the left side of the memory cell plain represents activated or negated state of column and row select lines, respectively. The value at the upper-left corner in each step is the 1:4 tree code according to each step.

The decoding process discussed above is implemented by making automata for each level, as shown Table 3.3. The upper automaton can select the area by the unit of 2×2 memory cells, and the lower automaton can by 1×1 memory cells. `W` represents the state of selecting all of its area, and `A`, `B`, `C` and `D` represents the states of selecting upper-left, upper-right, lower-left, and lower-right subarea, respectively. For example, the pair of two automata's states, (`s2`,`s1`)=(`B`,`A`) represents that the selected area is in upper-right 2×2 area by `s2`=`B`, and the precise position in this 2×2 area as the upper-left 1×1 memory area by `s1`=`A`, which is just corresponds to the state in Figure 3.12(f).

The selection lines of memory cells are implemented by using the `ra`, `rb`, `ca`, and `cb` signals defined in Table 3.4 as follows.

`r0` = `ra2` && `ra1`

Table 3.4: Truth table for select lines.

state	ra	rb	ca	cb
W	1	1	1	1
A	1	0	1	0
B	1	0	0	1
C	0	1	1	0
D	0	1	0	1

r1 = ra2 && rb1

r2 = rb2 && ra1

r3 = rb2 && rb1

c0 = ca2 && ca1

c1 = ca2 && cb1

c2 = cb2 && ca1

c3 = cb2 && cb1

Here the number following ca, and so on, represents the order of automata, for example ca2 represents the ca signal according to the state of automata s2, and && represents the logical-AND operator.

3.5 Summary and Conclusion

In this chapter, a concept of applying the tree structure for image signals is proposed. The image scan for tree structure including the nodes, which have logical-OR of lower nodes, can implement a kind of data compression by skipping the redundant scan step for white sub-areas. The size of sub-areas scanned can be changed dynamically according to the scanning step, the lower level of tree structure represents the higher spatial resolution. It is also derived that the quad tree structure is the most adequate to treat the image signals, which we call "1:4 tree structure." The 1:4 tree scan is more effective for some sample images than the raster scan, and than the run-length compression in most cases.

The decoding algorithm of 1:4 tree code is also discussed, ant it can be implemented by using the state automata and selection lines, with the array of memory cells, which can be selected simultaneously.

Chapter 4

Applications of 1:4 Tree Sensor in Image Processing

Our eyesight has the functions of not only just "looking at" the objects, but also "watching" them, for example, making attentions to the restricted objects, or having the intensional adaptivity in dark field. Some approaches for implementing these functions in the solid-state image sensors are studied, and most of them take approaches of implementing the structure of the retina using electronic circuits[3].

In this chapter, the previous studies for implementing the functions of the eyesight in electronic circuits are summerized at first. The efficiency and the way to apply the 1:4 tree scan for implementing some functions of the eyesight, mainly in terms of various adaptivities are discussed in section 4.2 and section 4.3.

It is also discussed the applications of 1:4 tree scan for the movie compression implemented on image sensors, mainly in terms of inter-frame differences and motion compensation, which are often used in conventional movie compression, such as MPEG, section 4.4.

4.1 Previous Approaches for the Functions of Eyesight

Figure 4.1 shows the structure of the typical retina in the creatures' eyes. The retina mainly consists of the following elements.

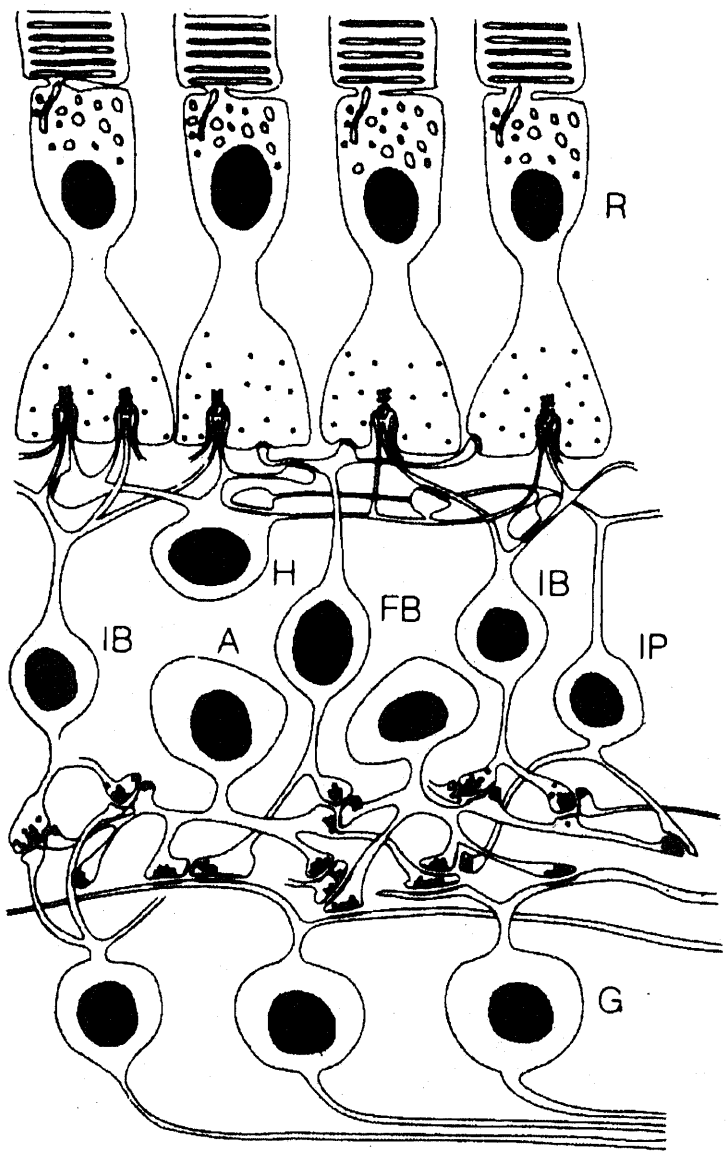


Figure 4.1: Cross-sectional view of the retina's structure.(From [3])

- Photo receptors(R), which convert the light signal to the electrical signal, and are often weakly connected each other.

- Horizontal cells(H), which receive the signal from the photo receptors, and connect them each other. This function implements the smoothing of the images.

- Bi-polar cells(FB and IB), which take the differences of the signal from photo receptors and that from horizontal cells. This function implements a kind of edge detection.

The functions implemented in the retina, the smoothing and the edge detection, are called "early vision problem," and they have been studied for a long time as the pre-processing of image signals.

One of the most previous works in computational sensors is "silicon retina", as introduced in section 1.2. It implements the smoothing of received images by the resistor networks, as the horizontal cells in the retina.

One of the other studies for "silicon retina" is "foveated" image sensors[19], as shown in Figure 4.2. In this sensor, the photo receptors and transfer CCDs are placed like the concentric circles, which is very similar to the placement in the retina, which has the characteristics of higher spatial resolution at the center part. It is also notable that the obtained data from this image sensor will be invariant against the rotation of the objects.

As seen above, most of approaches for implementing the functions of eyesight are made by implementing the structure of the retina in electronic circuits[20, 21].

4.2 Adaptivity for Spatial Resolution

4.2.1 Spatial adaptivity of eyesight

We will usually see the objects, with making attentions just to the area we have interests. In this case, we will obtain the detail information for the areas of interests,

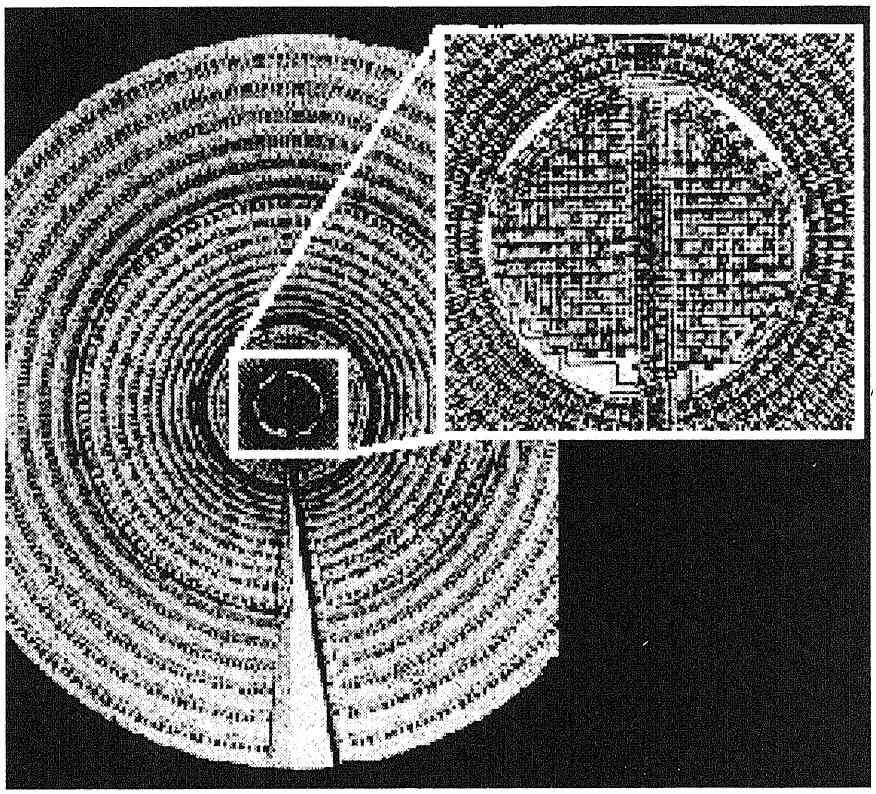
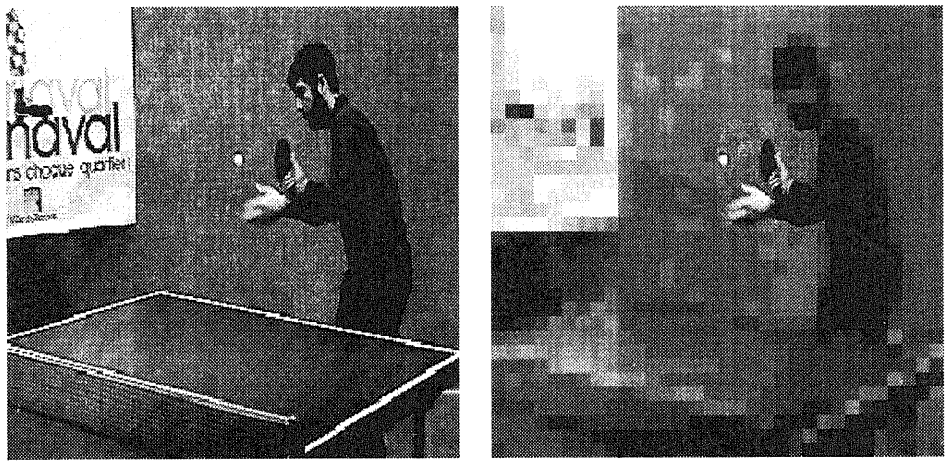


Figure 4.2: Photograph of the foveated CCD retina[19].



(a) (b)

(a') (b')

Figure 4.3: Samples of the original images(a)(a'), and the images with making attentions to the restricted area(b)(b').

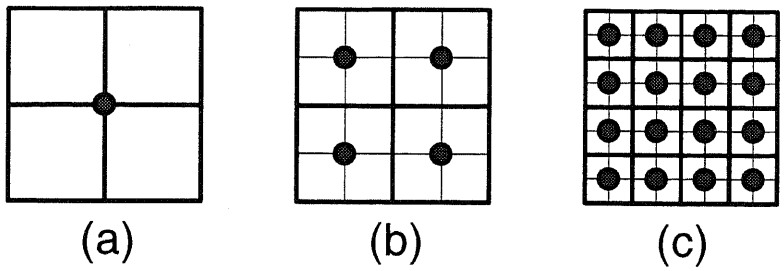


Figure 4.4: Area division by the step of 1:4 tree scan. (a)unit of 4×4 pixels , (b)unit of 2×2 pixels, and (c)unit of 1×1 pixel. (The gray circles in this figure represents the higher nodes for each sub-area.)

and less information from the other areas, such as just the intensity or the information of objects' existence, which will be mosaicked images as shown in Figure 4.3(b)(b').

This fact is expressed as follows; the higher spatial resolution for the areas of interests, and the lower for the other areas, for example by the unit of 8×8 pixels.

4.2.2 Implementing spatial adaptivity using tree sensor

It is notable that the 1:4 tree scan proceeds from the nodes in the higher level to the nodes in the lower level, and to the lower the scan proceeds, the higher spatial resolutions will be obtained, since the nodes have the logical-OR, or *summerized information* of their lower nodes, as shown in Figure 4.4.

This procedure can be implemented by adding the functions for each node, to select whether proceeding the scan normally, or finishing the scan regardless of its value. The nodes for the areas of interests should continue its step of scan normally, and the nodes for the other areas should finish its scan to return the summerized information of its sub-area.

The procedure of "spatial adaptive scan" by 1:4 tree structure is expected to be the efficiency of a kind of data compression, by skipping the redundant scan for the areas of no interests. For example, the conventional 1:4 tree scan of the digitized image in Figure 4.3(a) and (a') give the 1:4 tree code of 59,061 and 42,953 bits, respectively, where the images have $256 \times 256 = 65,536$ pixels. (Note that the

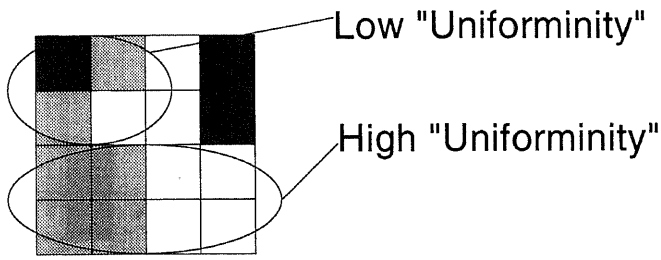


Figure 4.5: Sample of gray scale image with partial uniformity.

pictures in Figure 4.3 are illustrated as gray scale images for readers' convenience. These images are processed after digitizing to binary images in this study.) On the other hand, the extended 1:4 tree scan to obtain the mosaicked images in Figure 4.3(b) and (b') give the 1:4 tree code of 1,581 and 8,177 bits, respectively, which is about 1/27 and 1/8 of the conventional 1:4 tree code, respectively. Note that the scan for the mosaicked areas is proceeded to the resolution up to 8×8 and 4×4 pixels for Figure 4.3(b) and (b'), respectively. These results indicate that 1:4 tree scan can implement a kind of data compression according to the attentions for the images of interests.

4.2.3 A concept of using spatial adaptivity for still image scan

In the still images, whether they are gray scale or binary, there are two parts of areas; the areas whose pixels have almost the same intensity, and the areas whose pixels have much difference intensities. Here we express the "uniform" areas for the former, and the "ununiform" areas for the later. Since the pixels in the "uniform" areas will have almost the same intensity, the scan for the all pixels are not needed; just the scan of the summerized (mean) intensity of the areas is needed, while the complete scan will be needed for the "ununiform" areas to obtain the informations for the details.

This procedure can be implemented by the 1:4 tree scan, by the partially stopped scan for the "uniform" areas, and the complete scan for the "ununiform" areas. This methodology can be expressed as that it uses "uniformity" as the criteria of interest.

Assuming that the intensity of each pixel should has the analog value, the "uni-

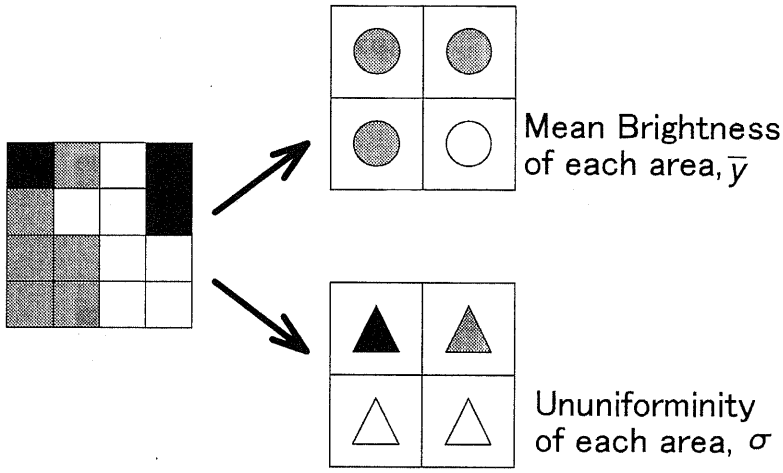


Figure 4.6: Models of the nodes which have two values; mean intensity, $\overline{y}$ and the standard deviation of the intensity, σ.

formity" of the area can be defined by the standard deviation of intensity in the areas, σ as follows.

$$\sigma^2 = \frac{1}{4}\sum_{i}^{4}(y_i - \overline{y})^2 \tag{4.1}$$

$$\overline{y} = \frac{1}{4}\sum_{i}^{4} y_i \tag{4.2}$$

Here the y_i is the mean intensity of each four sub-areas, and $\overline{y}$ is the mean intensity of all four sub-areas, and the models of this equation are shown in Figure 4.6.

The procedure considering the uniformity of the images using the hierarchical structure of this y_i and σ for each node, can be implemented by the following procedure.

1. Scan the uniformity value, σ of the node, as shown in Figure 4.7(a).

2. If σ is smaller than the threshold value, σ_θ, it implies that the area of the node will be uniform enough, and it finishes the scan just by returning the mean intensity, $\overline{y}$, as shown in Figure 4.7(b).

3. If σ is larger than σ_θ, it implies that the area of the node will contain variations of pixels' intensity, and it continues the following scan for the lower nodes, as shown in Figure 4.7(c).

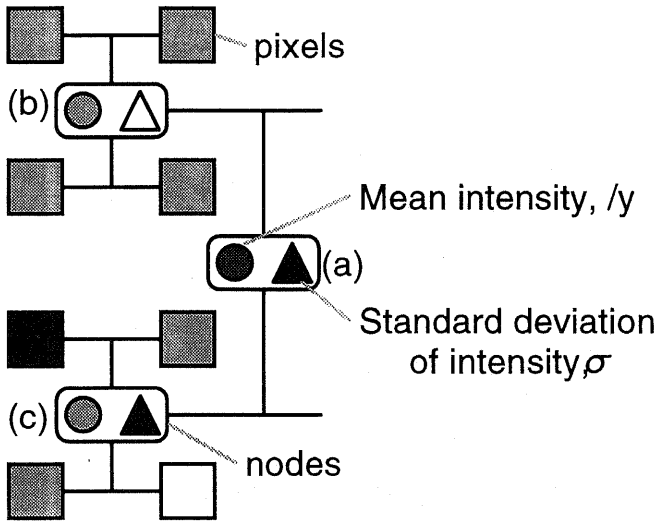


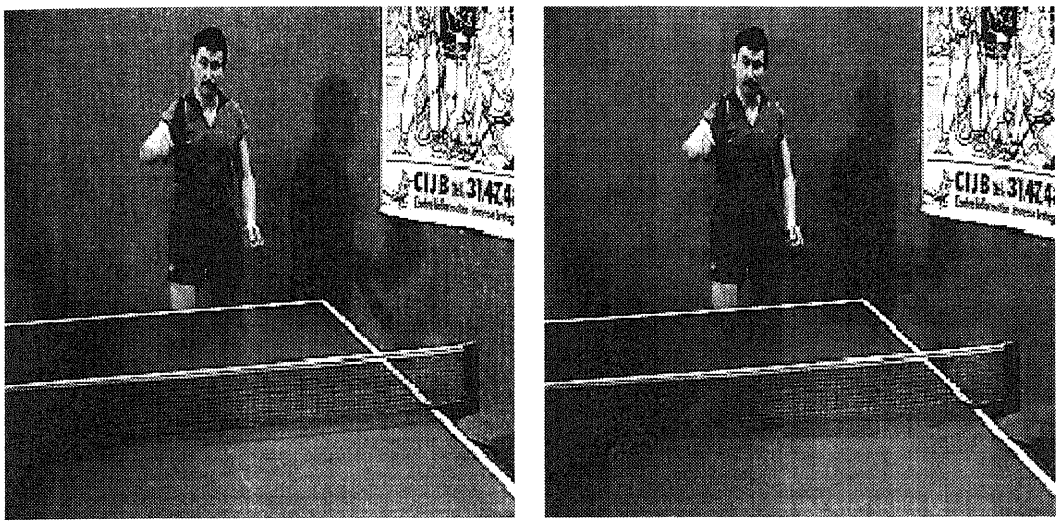
Figure 4.7: The procedure of scan considering the uniformity of each area.

Figure 4.8 shows the example of the above procedure. The original gray scale image with 256×256 pixels is shown in Figure 4.8(a), and the reconstructed image using the scan considering the uniformity of each area discussed above is shown in Figure 4.8(b). In Figure 4.8, the just nodes whose σ is larger than $\sigma_\theta = 10$ are scanned normally, for the smaller sub-area of 16×16 pixels. (Note that the intensity depth of the image is 256) Seen in Figure 4.8(b), the objects with low uniformity, such as the man or the poster on the wall, are scanned to the detail, while the areas with high uniformity, such as the gray background or the table tennis court, are scanned with lower spatial resolution.

The number of scan steps is 19,660, which is about 1/3 of the conventional raster scan, and the signal-to-noise ratio (SNR) of the reconstructed image, defined by the following equations, is 31.4dB,

$$\mathrm{SNR[dB]} = 10 \log_{10} \frac{255^2}{\frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} \{p(x,y) - p'(x,y)\}^2}, \tag{4.3}$$

where M and N are the number of pixels in horizontal and vertical direction, respectively, and $p(x,y)$ and $p'(x,y)$ are intensity at (x,y) in the original and reference image, respectively.



(a) (b)

Figure 4.8: Sample of gray scale image(a), and the reconstructed image using the scan considering the uniformity of each area(b).

This result indicates one possibility of the spatial adaptive scan with a kind of data compression maintaining the quality of the images, using 1:4 tree structure.

4.3 Adaptivity for Intensional Resolution

4.3.1 Intensional adaptivity of eyesight

Our eyesight has the function to see the objects both in the bright condition and the dark condition, or have the adaptivity for the intensity of the field. This function is implemented by having two photo receptors in the retina; *the rod* for the dark-field without the sense of color, and *the cone* for the bright-field with the sense of color[3].

One difficulty of receiving the photo signals by the electronic device is that the range of intensity of the field is broad up to 5 decades, and it is difficult to keep the high dynamic range for such a broad range of intensity by using the simple photo-electronic converter. It is known that the sensitivity of the photo receptor in the retina is proportional to the logarithm of the input intensity, which can keep the

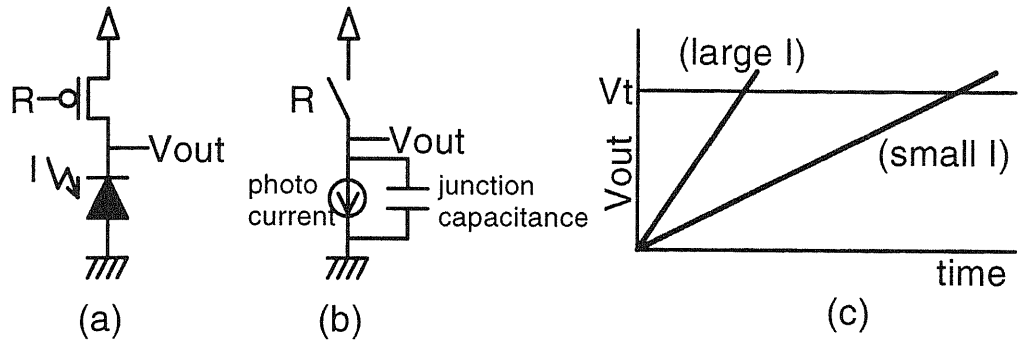


Figure 4.9: A simple circuit of photo-electronic converter(a), its equivalence circuit(b), and its output with the light of small and large intensity(c).

high dynamic ranges[3]. Some studies for the photo-electronic converting circuits with the logarithmic sensitivity[22, 23, 24], and some of them intend to obtain the more wide dynamic range by selecting the gain of photo-electronic converting circuits corresponding to the light intensity[24].

4.3.2 A concept of gray-scale scan using 1:4 tree with intentional adaptivity

Figure 4.9 shows a simple circuit of photo-electronic converter. The charge in the junction capacitance of the photo diode is cleared by the reset signal at first, and then the charge is integrated in the capacitor by the photo current, I_{photo}. Since the photo current is proportional to the intensity of light, the output voltage V_{out} will expressed as follows;

$$V_{\text{out}} = C \cdot I_{\text{photo}} \cdot t \propto I \cdot t, \tag{4.4}$$

where t is the time of integration, and I is the intensity of light. Using the Equation (4.4), the time until V_{out} reaches the threshold voltage, V_t is estimated to be proportional to I^{-1}, and we can obtain the information of the light intensity by this time.

The flag indicating that V_{out} has reached V_t will be activated at the local areas in the focal plain, since the distribution of the intensity is expected to be local in two dimensional focal plain, as shown in Figure 4.10. It is the case which can

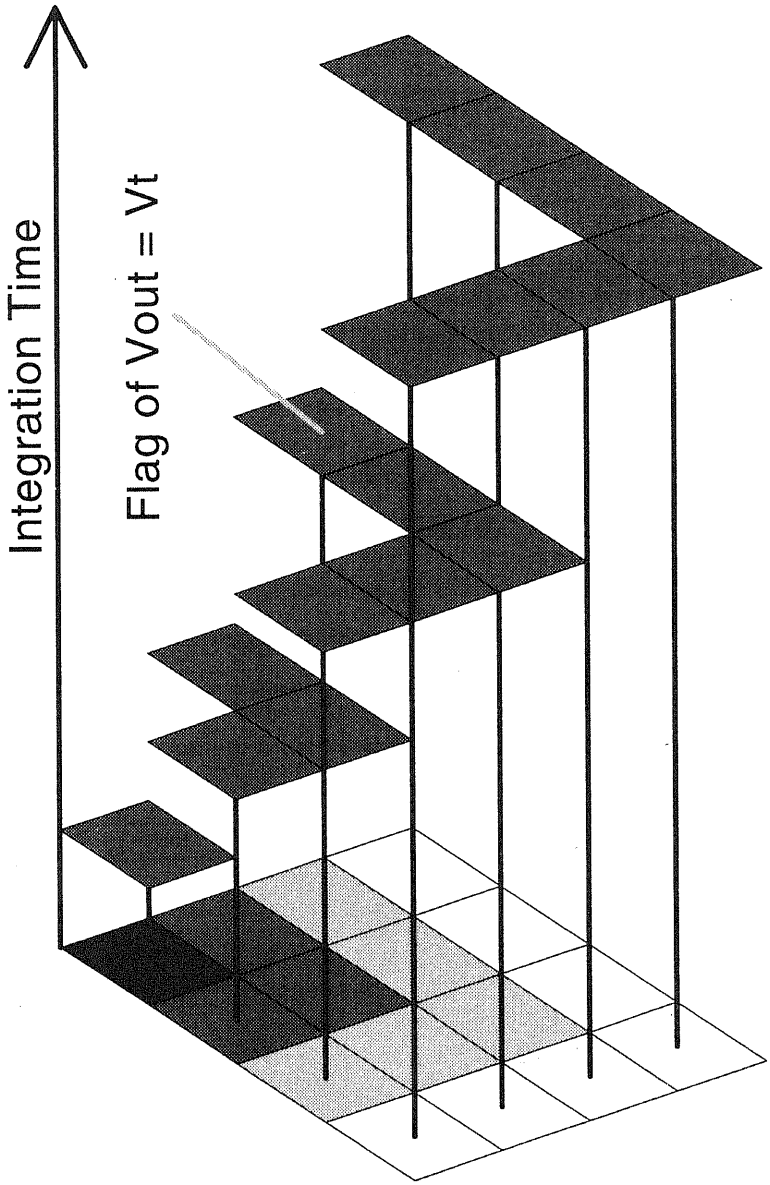


Figure 4.10: Intensity distribution in focal plain and integration time to $V_{\text{out}} = V_t$

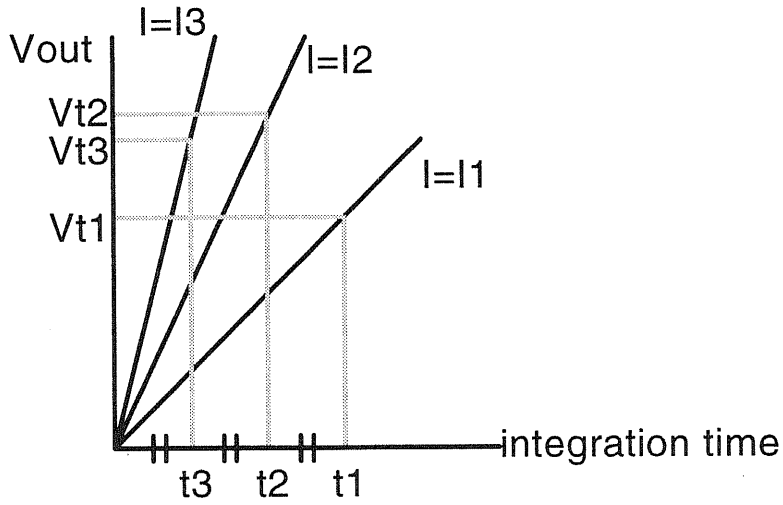


Figure 4.11: Controlled V_t to keep the interval time of V_{out} to reach V_t.

be efficiently scanned by the 1:4 tree scan, where the pixels of "1" are distributed locally, as shown in section 3.2.2. The flag once scanned should be cleared, since the pixel whose V_{out} has reached V_t is not need to be scanned any more.

One problem of the above technique is that V_{out} for the higher intensity will reach V_t rapidly, while that for the lower intensity will rise very slow, and this fact will result in the rush of flags at the early terms in integration time.

One concept to solve this problem is to control V_t according to the terms in integration time, as shown in Figure 4.11. The V_t will be set small at the early terms in integration time to keep the interval time long, while V_t will be set larger at the middle term in the integration time, and V_t will be set small again at the later term for the lower intensity.

Assuming that the slope of V_{out} is proportional to the light intensity I, as $V_{\text{out}} = kIt$. The integration time until V_{out} reaches to V_t is expected to have the equal interval, for the intensity of the equal interval, and the intensity and the integration time of i-th brightness, I_i and t_i, respectively, will expressed as follows.

$$I_i = I_0 + i\Delta I \tag{4.5}$$

$$t_i = t_0 - i\Delta t \tag{4.6}$$

V_t for the intensity of I_i, V_t^i is expressed as the V_{out} at the time of t_i, as follows.

$$V_t^i = kI_i t_i = k(I_0 + i\Delta I)(t_0 - i\Delta t) \tag{4.7}$$

Using this equation, V_t^i is expressed as the function of integration time, by eliminating i from the all equations, as follow.

$$V_t(t) = -k\frac{\Delta I}{\Delta t}\left(t - \frac{\Delta I - I_0\Delta t}{2\Delta I}\right)^2 + \frac{(\Delta I + I_0\Delta t)^2}{\Delta I \Delta t} \tag{4.8}$$

The interval of the integration of time to reach a threshold voltage will be equal by changing $V_t(t)$ according to the integration time, as derived in the Equation (4.8). This control will make the almost constant number of pixels at each scanning term.

4.4 Applications for Movie Compression

4.4.1 Inter-frame difference of movies and movie compression

The differences of pixels between successive two frames in moving pictures, which is called *inter-frame difference*, have the values when there were some motions of objects in the scene, and most of the pixels will have no difference except the restricted area where motion occurred. One of the simplest, and most important technique to compress the movie data is to take the inter-frame difference, and to read out just the value of pixel whose value has changed between two frames, which is called "effective pixel."

4.4.2 Implementing scan of inter-frame difference using 1:4 tree

The ratio of the effective pixels in inter-frame difference is expected to be small enough, except the special case, such as when the scene has changed. The scan for the effective pixels using 1:4 tree structure will make a high efficiency, since 1:4 tree scan is suitable for the images containing a few effective pixels, whose value is "1", as shown in section 3.3.2.

This function can be implemented by modifying the pixels in 1:4 tree structure to have the value as the inter-frame difference, by adding the memory to keep the value in the previous frame, with modifying the nodes for binary images.

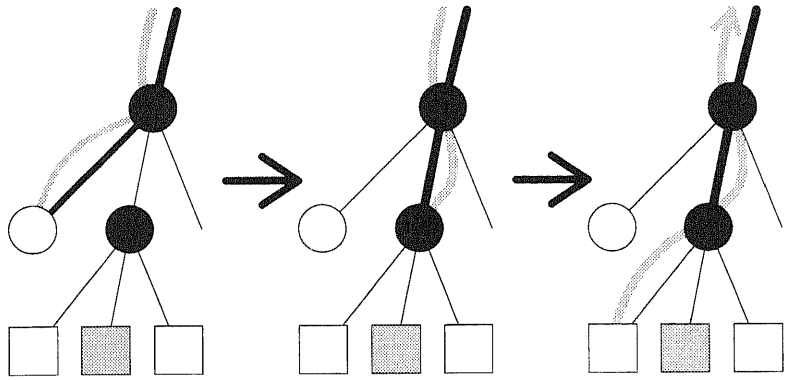


Figure 4.12: 1:4 tree structure for scanning inter-frame difference.

Here we consider the method to extend the above compression methodology for gray-scale movies. One of the simplest algorithms for gray-scale movie compression using inter-frame difference is "Conditional Replenishment Method" as the following procedure.

1. take the inter-frame difference; the arithmetical subtraction of the values of pixels in the successive two frames. (The value of pixel in the previous frame is stored in the memory.)

2. if the difference is smaller than the threshold, the scan for this pixel will be skipped.

3. if the difference is larger than the threshold, the value of this difference is read out outside the sensor.

4. the stored value in the memory is replaced by the value of the pixel in the current frame when scanned.

We can apply the 1:4 tree scan and conditional replenishment method by adding the analog signal pathes as follows.

- each pixel has the flag which indicates that it has the larger inter-frame difference than threshold; "1" for effective pixels.

- the flags are scanned by the conventional 1:4 tree structure.

Table 4.1: 1:4 tree scan code length per pixel, $\overline{L}$ and mean SNR of reproduced images, $\overline{\text{SNR}}$ using conditional replenishment method for various threshold of intensity, θ.

name	$\theta = 5$		$\theta = 20$	
	$\overline{L}$	$\overline{\text{SNR}}$	$\overline{L}$	$\overline{\text{SNR}}$
MissAmerica	0.474	41.1	0.115	32.5
TableTennis	0.774	41.7	0.345	30.7
FlowerGarden	1.137	44.8	0.984	30.8
Rail	0.113	44.4	0.030	38.3
Neck	0.313	42.5	0.126	35.5

- when the scan step has proceeded to each pixel, the analog difference is read out, through the analog signal path along to the scan path from the top to the pixel.

An example of the above procedure is shown in Figure 4.12, where the squares and the circles represent the pixels and nodes, respectively, and black circle is the node whose value is "1". The gray curve represents the analog signal path for the analog inter-frame difference, from the pixel to the outside of the sensor, and the thick line between nodes represents the current 1:4 tree scan path. The analog signal path will be easily implemented by the transfer gates by the pair of n-channel and p-channel MOSFETs.

Table 4.1 shows the results of the above procedure for some sample movie sequences, where $\overline{L}$ is the mean 1:4 tree code length per pixel for the scan of effective pixels whose difference between two frames is larger than θ, and mean signal to noise ratio (SNR) over all sequences of the reproduced image by conditional replenishment method over the original images.

In most cases, the larger θ gives the shorter 1:4 code length and worse SNR, and we can select θ for the purposes. In the sequence of "FlowerGarden," the whole area of images moves slowly, and the ratio of effective pixels is constantly very large, which results in the larger $\overline{L}$ than the raster scan.

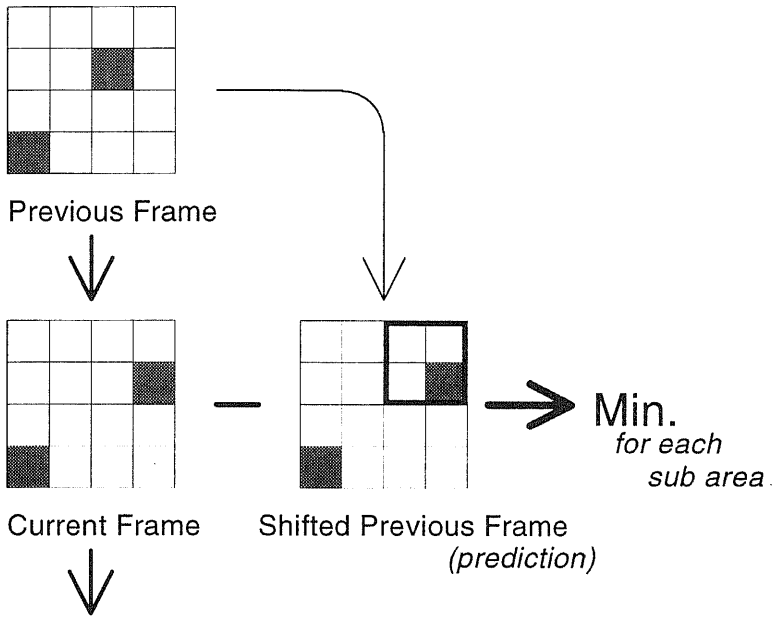


Figure 4.13: The simplest methodology for motion compensation.

4.4.3 Motion compensation and movie compression

Another of the most important techniques for movie compression is "motion compensation," which is the prediction of the motion. The error between the predicted image by the motion compensation and the practical frame will be read out, using the conditional replenishment method, if needed. The standard movie compressions, such as MPEG, employ both the inter-frame difference and motion compensation[25].

The simplest methodology to calculate the motion compensation, which is often done in the conventional movie compression, is as follows, as shown in Figure 4.13.

- scan the images and store the buffer memory.
- create the moved image to various directions with the various distances for divided sub-areas, from the stored previous frame, and calculate the difference between this moved image and the practical image.
- choice the direction and the distance to minimize the difference for each sub-area, and they are compensated motion.

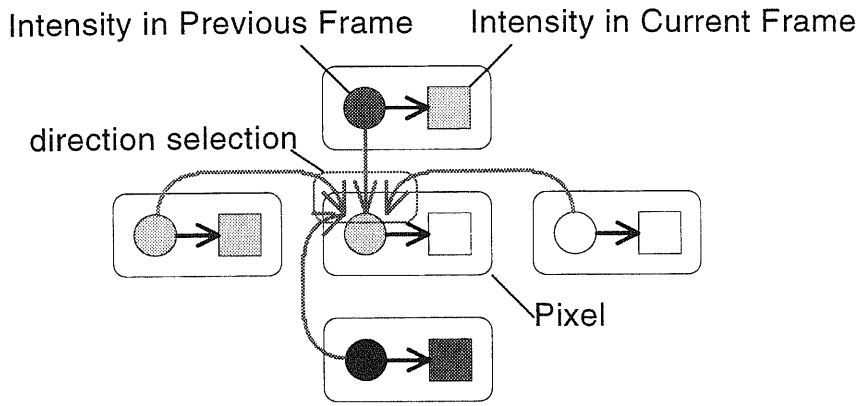


Figure 4.14: Pixels connection for motion compensation in 1:4 tree

This procedure is very complex, and the motion compensation is one of the most complex operations for movie compression.

4.4.4 Implementing motion compensation using tree sensor

The motion compensation is implemented using 1:4 tree structure by adding the following functions as shown in Figure 4.14.

- The pixel has the value of difference between the value in the current frame and that in the previous frame, but it is used the value in the previous frame of the neighbor pixel from the selected distance and direction, instead of the current pixel.

- The node in the 1:4 tree structure has the value as the mean of the connected four nodes' value. The value of one node represents the mean of values in the all sub-area under it.

- The nodes according to the size of area needed for motion compensation are scanned from the outside, and this scan is iterated by changing the distance and direction for the motion to be compensated.

- Select the direction and the distance to minize the difference, for each sub-areas, and they are compensated motion for each sub-area.

Assuming that the pixel should has the mean value of the lower four nodes, the scanned value at any step gives the mean difference at the sub-area lower of the node, and we can select the block size of motion compensation by selecting the scan step.

Figure 4.15 shows the samples of simulation of motion compensation, using the standard movie, "MissAmerica".

The SNR between the practical image and the predicted image by compensated motion using the above algorithm is shown.

In case of Figure 4.15(a), SNR for the two size of sub-areas for independent motion compensation is shown, whole area of "1 of [256x256]" and 16×16 sub-areas of 16×16 size as "256 of [16x16]".

In Figure 4.15(b), the experience results by changing the searching distance are shown, searching just the neighbor pixel, $d = 1$, and the pixels within 15 pixels, $d < 15$.

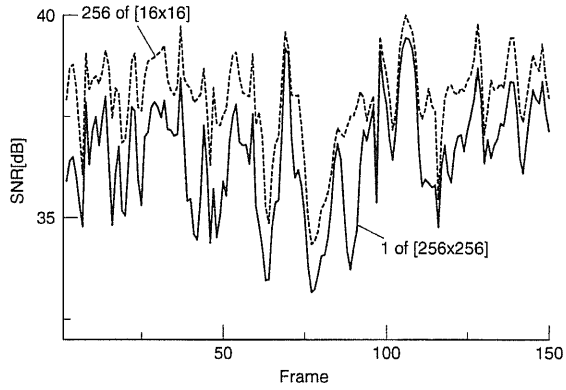
The comparison of the changing the searching directions is shown in Figure 4.15(c), with 8 directions of "8Dir" and 32 directions of "32Dir".

The simulation results shows that the more the range of searching directions and distances or the smaller the size of sub-areas, the prediction by compensated motion will be more precise.

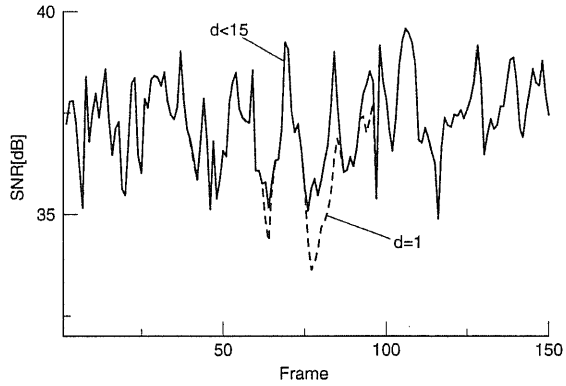
It is indicated that in case of very high frame rate, for example 1ms per frame, the motion will occurred within at least one pixel, since the object may not move with the faster velocity[26]. In such case with scanning with very high frame rate, the motion will be restricted to within one pixel, and then it will be enough for precise motion compensation the search range of just one neighbor pixels and 8 directions.

It will be can easily implemented by adding the signal paths between one pixel and the 8 neighbor pixels and the memory to store the pixel value in the previous frame, and the global signal to select the direction to be moved in Figure 4.14.

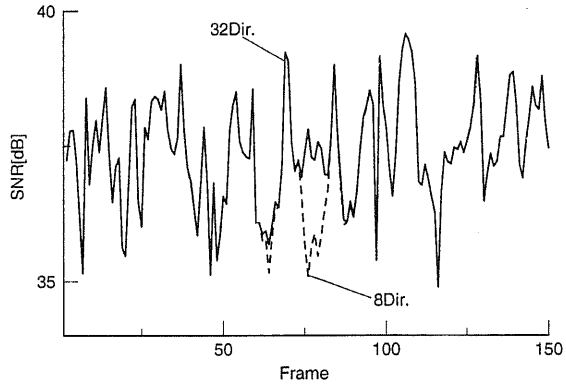
Table 4.2 shows the mean 1:4 tree code length per pixel for the effective pixels, $\overline{L}$ and the mean SNR of reproduced images, $\overline{\mathrm{SNR}}$ using conditional replenishment method. Here the conditional replenishment method is carried out for the difference



(a)



(b)



(c)

Figure 4.15: Simulation results of SNR for the predicted image by compensated motion and the practical image.

Table 4.2: 1:4 tree scan code length, $\overline{L}$ per pixel and mean SNR of reproduced images, $\overline{\text{SNR}}$ using conditional replenishment method after motion compensation.

	$\theta = 5$		$\theta = 20$	
name	$\overline{L}$	$\overline{\text{SNR}}$	$\overline{L}$	$\overline{\text{SNR}}$
MissAmerica	0.364	41.3	0.046	34.8
TableTennis	0.753	41.4	0.296	30.7
FlowerGarden	1.097	42.9	0.809	29.9
Rail	0.116	44.3	0.030	37.3
Neck	0.321	42.0	0.093	35.5

between the original image and the predicted image, which is generated from the previous image using the compensated motion within 5 pixels and 8 directions.

In most cases, the 1:4 tree code length and SNR are better than those without motion compensation, but SNR is worse than without motion compensation in some cases, since the precision of motion compensation is not enough for these movies captured in slow frame rate of about 30ms/frame. Assuming high frame rate, the precision of motion compensation will be enough to keep enough SNR.

4.5 Application for Visual Tracking

In this section, the simple algorithm of visual tracking using 1:4 tree structure is described. The methodology of masking target image used for selecting targets in visual tracking is described at first, and the visual tracking algorithm using image masking is described next.

4.5.1 Image masking using 1:4 tree structure

Here we consider to scan just for the restricted area in images, and the other area should be white, as shown in Figure 4.16. The original image is Figure 4.16(a), and here the target area drawn gray in Figure 4.16(b) in the original image is to be scanned as shown Figure 4.16(c). We assume the bitmap image, we call "window"

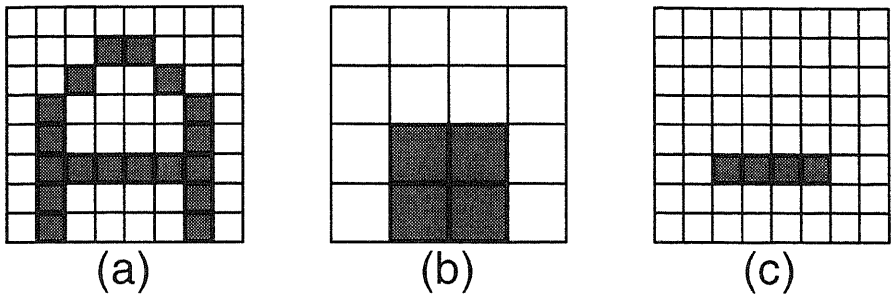


Figure 4.16: Samples of original image(a), target area to be masked(b), and the masked image(c).

here, as the raster image of "1" for the target area and "0" for for the other area, and the resolution of the "window" can be assumed independent on the original image, for example 4×4 in Figure 4.16(b). The 1:4 tree code for this "window" in Figure 4.16(b), here we call "window code" is obtained as follows, with the order of level in 1:4 tree structure l.

Level(l)	1	2	2	2	3	2	3
Window code	1	0	0	1	0101	1	1010

Here we extend the functions of the nodes in 1:4 tree structure as follows.

- Refer the bit of window code corresponding to the level l of the conventional 1:4 tree scan for the original image.
- If the corresponding bit of window code is "0", it implies that the lower level than the current node is the area out of the target, and it returns 1:4 tree code of "0" since no more scan steps are needed.
- If the corresponding bit of window code is "1", it implies that the area is in the target area. The following 1:4 tree scan steps for the lower level are carried out, and the corresponding level of window code is also descended.
- If the 1:4 tree scan step has reached the lowest level, the scan is executed for each pixel in target area, and scan steps are carried out regardless of the bit of window code.

For example, in case of Figure 4.16(a), the extended 1:4 tree scan steps proceeds as follows. (C is the current bit of 1:4 tree code)

1. First, since the bit of window code for $l = 1$ is "1", the usual 1:4 tree scan is executed, and we obtain $C = 1$.

2. The scan proceeds to $l = 2$, and the corresponding bit of window code is "0". No more scan steps for the lower level is needed, since this area is out of target, and we obtain $C = 0$.

3. The next step for $l = 2$ is also out of target, since the corresponding bit of window code is "0", and $C = 0$ is obtained.

4. The next bit of window code is "1", and it implies that the following scan steps are in the target area, and we obtain $C = 1$ at first. Because the following four bits of window code are "0101", the second and the fourth 2×2 areas (upper-right and lower-right, respectively) are the target area, while the first and the third (upper-left and lower-left, respectively) are out of target.

5. The first 2×2 area is out of target, since the corresponding bit of window code is "0", and we obtain $C = 0$.

6. The second 2×2 area is in the target area, since the corresponding bit of window code is "1", the usual 1:4 tree scan steps are executed in order, and we obtain C=1-0011.

7. The third 2×2 area is also out of target, and we obtain $C = 0$. The last 2×2 area is in the target area, and we obtain C=0, since the all pixels in this area is "0", and here the scan steps for $l = 3$ have finished.

8. The scan step should back to $l = 2$ again, and since the corresponding bit of window code is "1", we obtain $C = 1$, and the following procedures for the lower level are executed similarly, and we obtain C=1-0011-0-0-0.

Here the following 1:4 tree code is obtained.

Level(l)	1 2 2 2 3 3 4	3 3 2 3 4	3 3 3
1:4 Tree Code	1 0 0 1 0 1 0011	0 0 1 1 0011	0 0 0

The image masking procedure described above can be executed by referring only the corresponding bit of window code in order to decide whether usual 1:4 tree scan steps should be carried out or not.

4.5.2 Visual tracking algorithm using 1:4 tree structure

The motion in the movies is detected simply by taking inter-frame difference, if no motion predictions or motion compensations are executed. The simple visual tracking algorithm is to trace only the moving objects. The moving objects include moving pixels, which are "effective pixels" in the successive two frames. Here we consider the visual tracking algorithm as follows, by determining the target area based on the motion.

- If there are effective pixels in the area corresponding to the window code of "0", it implies that the new motion has occurred in the non-target area, and this area should be added to the target area. The window code of current "0" should be replaced as "1-0000", if the corresponding level is not the lowest, while the current 0 in window code should be changed to "1" if the corresponding level is the lowest.

- If there are no effective pixels in the area corresponding to the window code of "0", no motions have occurred in the non-target area, and the window code should no be modified.

- If there are no effective pixels in the area corresponding to the window code of "1", the motion in the target area has stopped, and this area should be set non-target area. The corresponding bit of window code should be replaced as "0", and the bits for the lower level also should be replaced as "0", if the corresponding level is not the lowest.

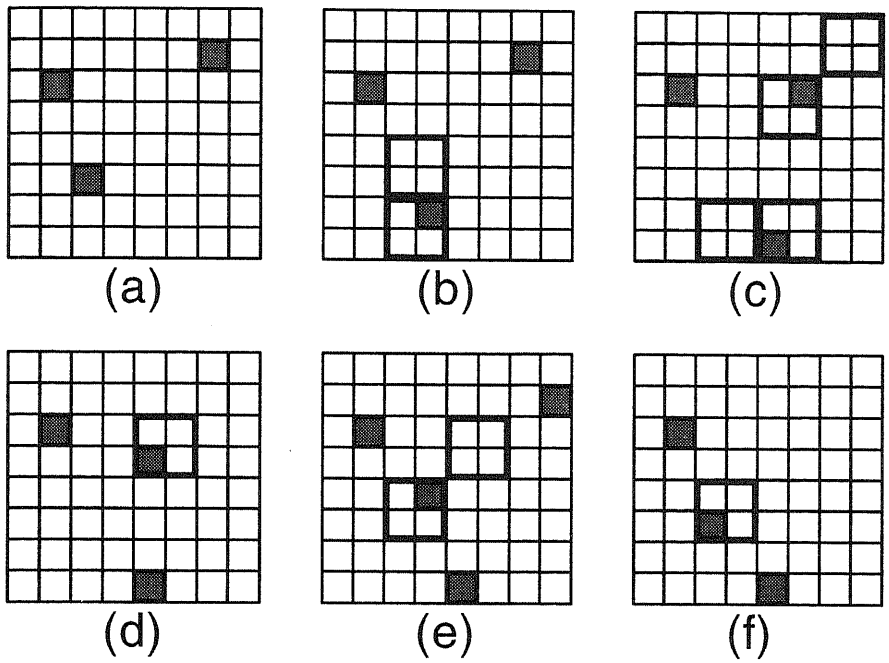


Figure 4.17: Steps of the visual tracking using 1:4 tree scan and window code for masking. (The area surrounded by bold line represents the target area.)

- If there are effective pixels in the area corresponding to the window code of "1", some motion have occurred in the target area, and window code should not be modified.

Figure 4.17 shows the steps of visual tracking using the algorithm discussed above. The visual tracking steps are follows.

1. Figure 4.17(a) indicates the start frame. There are no target areas, since no motions have occurred.

2. Figure 4.17(b) indicates the following frame. The point at the lower-left area has begun to move, and the areas of the motion are set as the target area, which are represented by surrounded area by bold lines.

3. In the following frame in Figure 4.17(c), the point at the upper-right area has also begun to move, and both of the motions are traced by the target areas.

4. In Figure 4.17(d), the point at the lower-left area has stopped moving, and the corresponding area is set out of target.

5. One-shot noise has appeared at the upper-right area in Figure 4.17(e).

6. In Figure 4.17(f), temporary appeared noise has not be traced, since it has disappeared before the window code was modified.

The visual tracking algorithm discussed above has the following characteristics.

- The number of target area is free from restriction.
- Only the successive two frames are used, but motion compensation has not been implemented.
- The targets are disappeared if they have stopped.
- The targets can be traced regardless to their direction or velocity.
- One-shot noise which will disappear within one frame is neglected.
- There are only two modifications of window code; "0" to "1-0000", and "1-xxxx" to "0". These modification will be implemented without the huge global pixel memory.

4.6 Summary and Conclusion

In this chapter, some applications of 1:4 tree structure for image processing are discussed.

The spatial adaptivity is easily implemented by the 1:4 tree scan, since the lower the 1:4 tree scan proceeds, the image with the higher spatial resolution is obtained. It is also discussed the methodology to scan the gray-scale still image considering the local uniformity, and the possibility is shown that this methodology can compress the image about 1/3, with maintaining the quality of the image.

It is also discussed the applications of 1:4 tree structure for movie compression. It is indicated that the two most important techniques in movie compression, inter-frame difference and motion compensation will be implemented effectively by modifying the conventional 1:4 tree structure.

Chapter 5

Implementation of Image Sensors with 1:4 Tree Structure

In this chapter, the implementation of the 1:4 tree structure discussed in chapter 3 will be discussed.

It will be discussed the implementation of the 1:4 tree structure in CMOS circuits, mainly aiming the low power operation by activating the target area selectively in section 5.1. In the following section 5.2 will describe the alternative implementation of 1:4 tree sensor by placing the pixel select circuits external of the pixel plain. It will be discussed the architecture and its circuits for the decoder of 1:4 tree code in section 5.3.

In the section 5.4, the functions and the circuits for each pixel will be discussed, aiming the intelligent image sensor. It will be discussed the concept of the image sensor driven by the received photo energy for the ultra low power operation in th following section 5.5.

5.1 Image Sensor with Tree Structure of Node Automata

5.1.1 Circuit of node automaton

In order to implement the function of nodes discussed in section 3.1, the node is described as the finite state machine whose state transition diagram is shown as

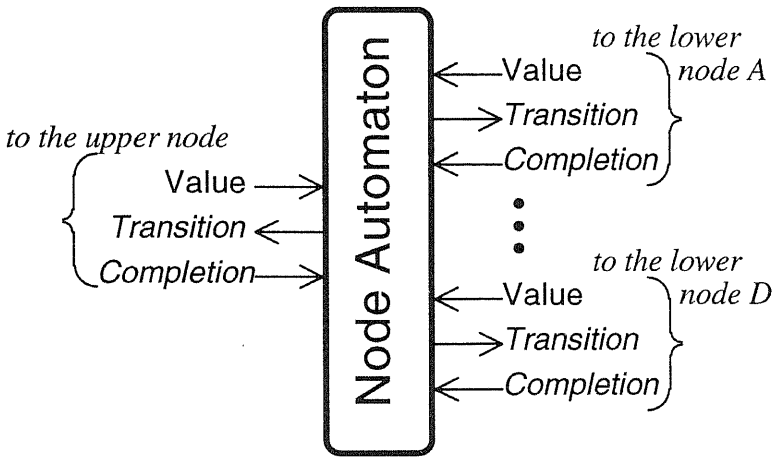


Figure 5.1: Signals of the nodes in the tree structure.

Table 5.1: State transition diagram for the node automata of 1:4 tree structure.

Inputs								Outputs					
T	V	CA	CB	CC	CD	S	S'	TA	TB	TC	TD	V	C
0	–	–	–	–	–	W	W	0	0	0	0	VO	0
↑	0	–	–	–	–	W	W	0	0	0	0	VO	1
↑	1	–	–	–	–	W	A	↑	0	0	0	VA	0
↑	–	0	–	–	–	A	A	↑	0	0	0	VA	0
↑	–	1	–	–	–	A	B	0	↑	0	0	VB	0
↑	–	–	0	–	–	B	B	0	↑	0	0	VB	0
↑	–	–	1	–	–	B	C	0	0	↑	0	VC	0
↑	–	–	–	0	–	C	C	0	0	↑	0	VC	0
↑	–	–	–	1	–	C	D	0	0	0	↑	VD	0
↑	–	–	–	–	0	D	D	0	0	0	↑	VD	0
↑	–	–	–	–	1	D	W	0	0	0	0	VO	1

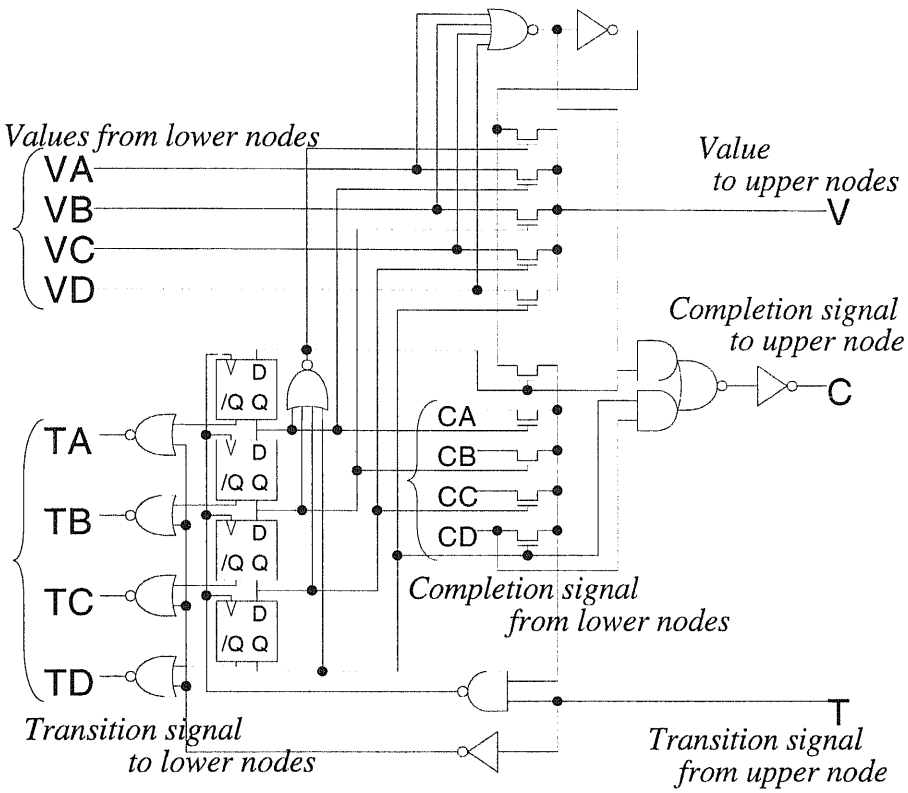


Figure 5.2: Circuit of node automaton.

Table 5.1.

Here the state W represents *waiting* states, and the state SA, ..., SD are the states of *scanning* lower nodes A, ..., D, respectively.

T is the *transition* signal to the automaton, and TA, ..., TD are the *transition* signals to its lower nodes A, ..., D, respectively. CA, ..., CD are the *completion* signals of the scan from the lower nodes A, ..., D, respectively. VO is the logical-OR of VA, ..., VD, and V and E is the output value and *completion* signal of automaton, respectively. These signals are summerized in Fig.5.1.

The designed circuit of the node automaton is shown in Figure 5.2. The tree structure containing a number of the conventional sequential circuits will consume much redundant power especially in clock line, since the most of the node automata have no transition except the nodes along the scan path. In the designed circuit in Figure 5.2, the clock signals for flip-flops are provided only in case of the flip-flops

have made state transition, in order to reduce the redundant power consumption.

The integration of photo detectors and signal processing circuits in computational sensors, will result in the reduction of the ratio of the photo detectors' area to the pixels' area, which is called "fill factor." This is a main reason why the functions on computational sensor are restricted to the simple signal processing, such as early vision problems. The number of transistors in many studies on computational sensors, will keep less than about 50, to maintain a large fill factor[5, 7, 8].

The number of transistors of the circuit in Figure 5.2 is 142. In case of the 1:4 tree structure, the node automata will be shared by the four sub-areas, and the effective number of transistors per pixel will be smaller than this number. Assuming the number of branches as b and the number of levels as N, the total number of pixels, n_{pixel} and the nodes, n_{node} are derived as follows.

$$n_{\text{pixel}} = b^N \tag{5.1}$$

$$n_{\text{node}} = \sum_{l=1}^{N} b^{l-1} = \frac{b^N - 1}{b-1} = \frac{n_{\text{pixel}} - 1}{b-1} \approx \frac{n_{\text{pixel}}}{b-1} \tag{5.2}$$

Thus the effective number of the transistors of node automata in 1:4 tree per pixel is derived as $142 \div (4-1) \approx 47$, which is expected to be reasonable to keep the large fill factor.

It is also notable that the signal pathes for values in the circuits of Figure 5.2 are implemented by the transfer gates, which will be easily extended for the analog signal pathes.

5.1.2 Layout of node automata

The node automata and pixels should be placed in the two dimensional focal plain, since the 1:4 tree image sensor are implemented using the conventional CMOS technology. A possible layout is shown in Figure 5.3. The pixels are placed with the equal intervals, and the nodes at the lowest level is placed at the center of the four neighbor pixels, and the nodes for the upper nodes are also placed at the center of the four sub-areas. This layout can be implemented for any number of levels, by creating the magnified self-similar cross-shaped node automata.

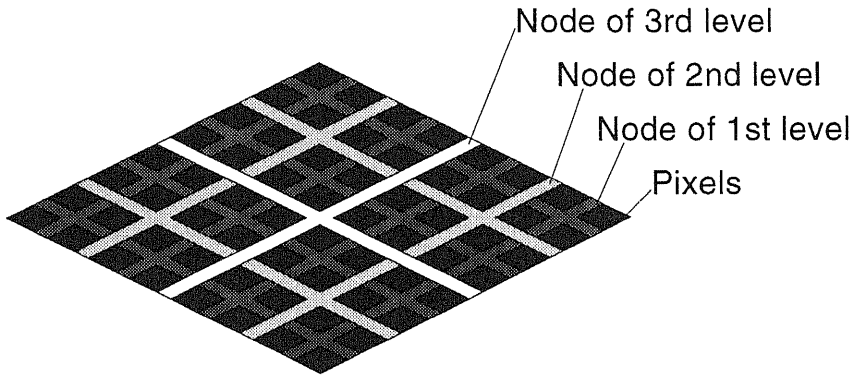


Figure 5.3: Possible two dimensional layout of the 1:4 tree structure.

As shown in Figure 5.3, since the upper nodes can occupy the larger areas, the upper nodes can have the larger signal drivers to drive the longer signal path, which is reasonable in order to minize the delays[27].

5.2 Alternative Implementation of Tree-Structured Image Sensor

5.2.1 Alternative architecture of tree sensor

The direct implementation of 1:4 tree sensor using the tree structure of node automata cannot avoid the problem of low fill factor intrinsically.

Here we propose the alternative architecture to implement the 1:4 tree scan, using the decoder architecture discussed in section 3.4. In each step of decode, the pixels are selected according to the selected sub-areas, for the values to be stored, while the scan will be executed for such selected sub-areas by reading the logical-OR of in the sub-areas alternatively. For example, the 4×4 binary image shown in Figure 5.4(a), the 1:4 tree scan proceeds by the following steps. (Here the number at the top side and the left side of pixel plain in Figure 5.4 is the column and row select lines, respectively, and the pixel whose column and row select lines are both 1, is selected. The logical-OR of selected pixels is shown at the upper-left corner of the pixel plain.)

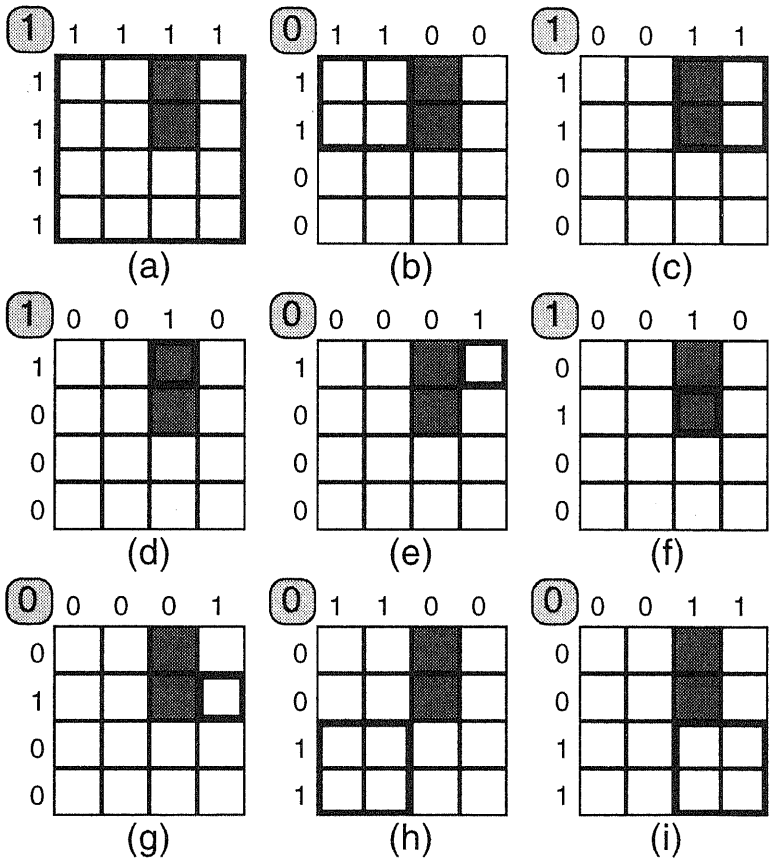


Figure 5.4: Example of 1:4 tree scan step using external address selectors

1. First, the logical-OR of whole pixels should be read out. The whole pixels are selected by the row and column address lines, and the logical-OR of the all selected pixels is read out, as shown in Figure 5.4(a).

2. Next, the scan proceeds to the upper-left 2×2 sub-area, and the pixels in this sub-area are selected, as shown in Figure 5.4(b). The logical-OR of these pixels is 0, and no more detail scan is needed for this sub-area.

3. The scan proceeds to the upper-right 2×2 sub-area as shown in Figure 5.4(c). Since the logical-OR of these pixels is 1, the more detail scan for each 1×1 sub-area (or pixel) is proceeded in order, as shown in Figure 5.4(d), (e), (f), and (g).

4. The following two steps of scan return 0 as the logical-OR of lower-left and lower-right 2×2 sub-areas, respectively, and now the all scan step has finished.

The logical-OR of the selected pixels can be made by the circuit as shown in Figure 5.5, where `PRB` and `PRW` are the signal of precharging each bit line of one column and word line, respectively, and ϕ is the clock of output latch for `v`. `r0` and `r1` are row select lines which indicate the selected rows in pixel plain, and `c0` and `c1` are the column select lines.

The scanning procedures at each step are follows, as shown in Figure 5.6.

1. precharge the each bit line and word line by keeping `PRB` and `PRW` as low.

2. set `r0`,`r1`, `c0` and `c1` according to the state of selection.

3. make `PRB` as high, and the each bit line will be discharged if there are at least values of pixels, `v` of "1" at the selected row.

4. make `PRW` as high, and the word line will be discharged if there are at least one discharged bit line in the selected column.

5. latch the value of word line by the latch clock ϕ.

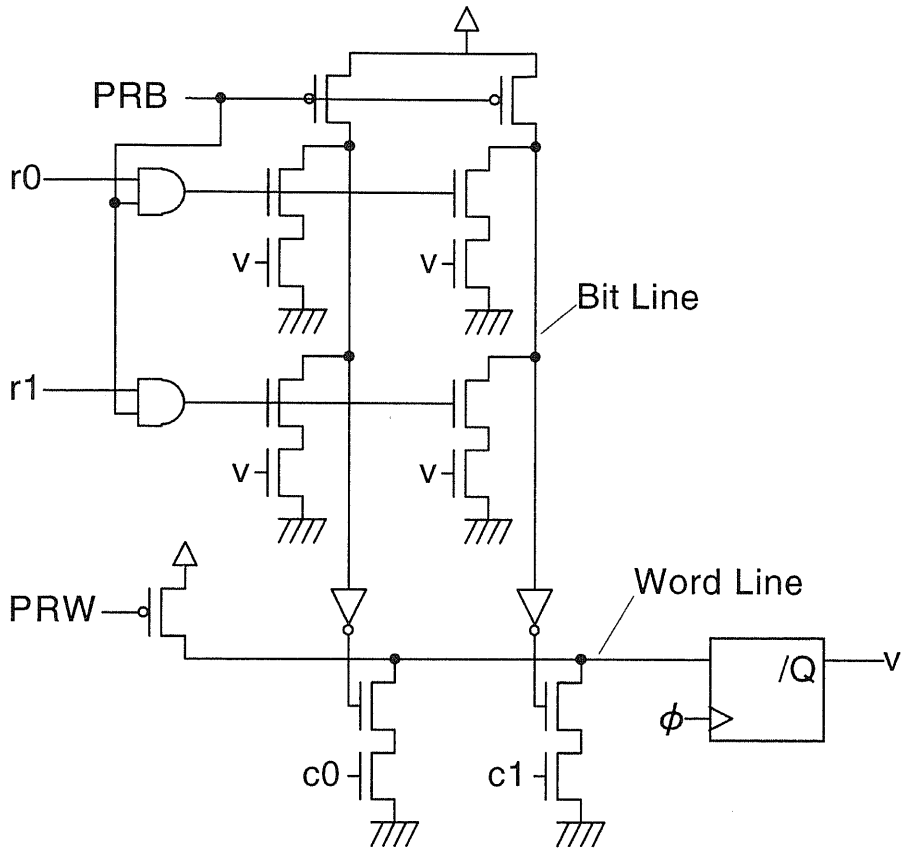


Figure 5.5: Architecture of pixel plains to make the logical-OR of the selected pixels.

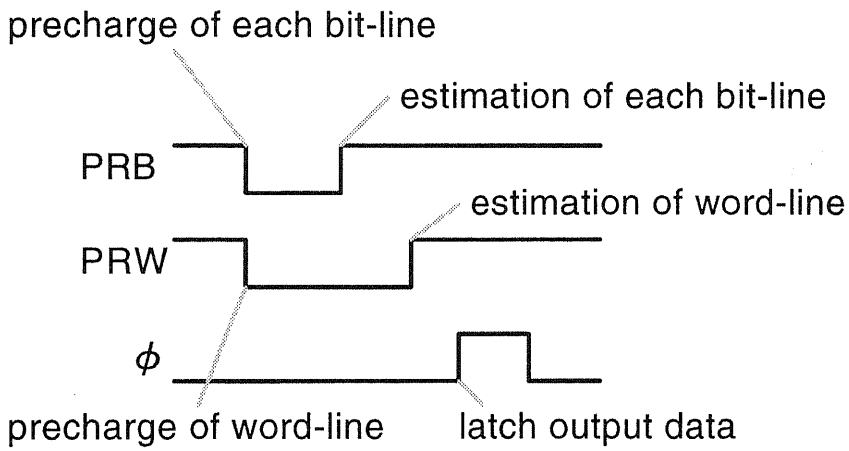


Figure 5.6: Timing chart of scan procedure of each step

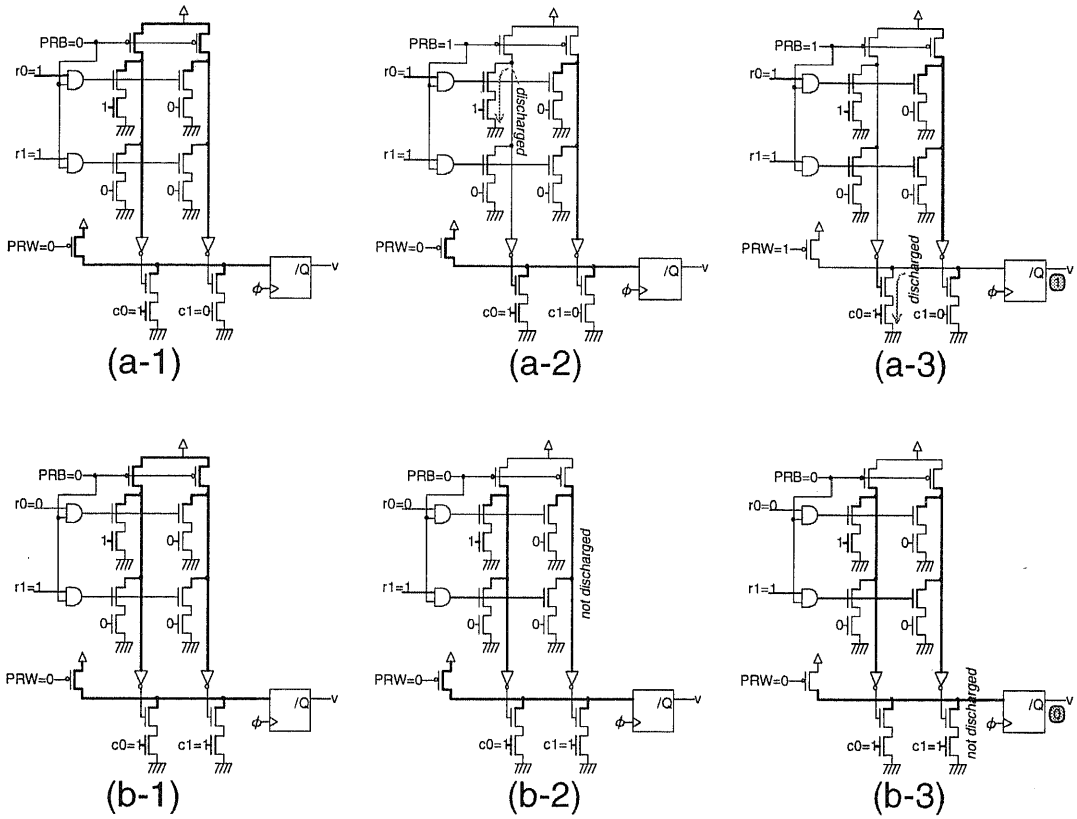


Figure 5.7: An example of scan steps using the pixel plain circuit in Figure 5.5. (The bold lines are charged lines)

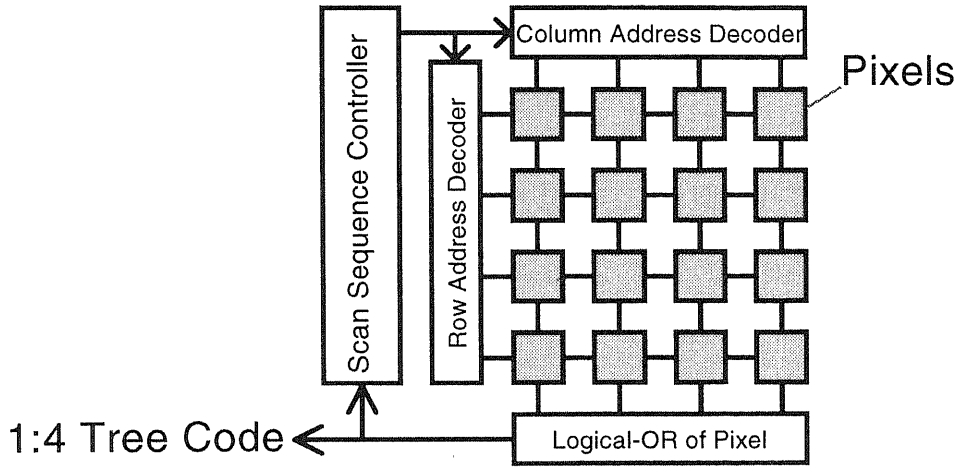


Figure 5.8: Architecture of 1:4 tree sensor using external address decoders

An example of scan steps with the state of each signal line are shown in Figure 5.7. In case of Figure 5.7(a-1), ...(a-3), the left half of four pixels are selected by (r0,r1)=(1,1) and (c0,c1)=(1,0). The bit lines are precharged in Figure 5.7(a-1), but the left bit line is discharged since upper-left pixel whose value is "1" is selected in Figure 5.7(a-2). Since the left bit line is selected by c0, the word line is discharged in Figure 5.7(a-3), and logical-OR of selected pixels is read out as "1."

In the other case in Figure 5.7(b-1), ...(b-3), the lower half of four pixels are selected by (r0,r1)=(0,1) and (c0,c1)=(1,1). The bit lines are precharged in Figure 5.7(b-1), and they are not discharged, since there are no discharge pathes from each bit line to ground, as shown in Figure 5.7(b-2). The word line is not also discharged since there are no discharged bit lines in selection, and the logical-OR of selected pixels is read out as "0."

The whole architecture of implementing these procedures is shown in Figure 5.8. The "scan sequence controller" makes selection signals of pixels based on the logical-OR of all the selected pixels, and "row and column address decoders" make address selection signals for pixels along to the currently selected sub-areas.

The problem of low fill factor because of the placed node automata in pixel plain will be solved by this architecture, which we call "external address decoder" architecture afterward, with implementing completely the same functions of 1:4 tree scan

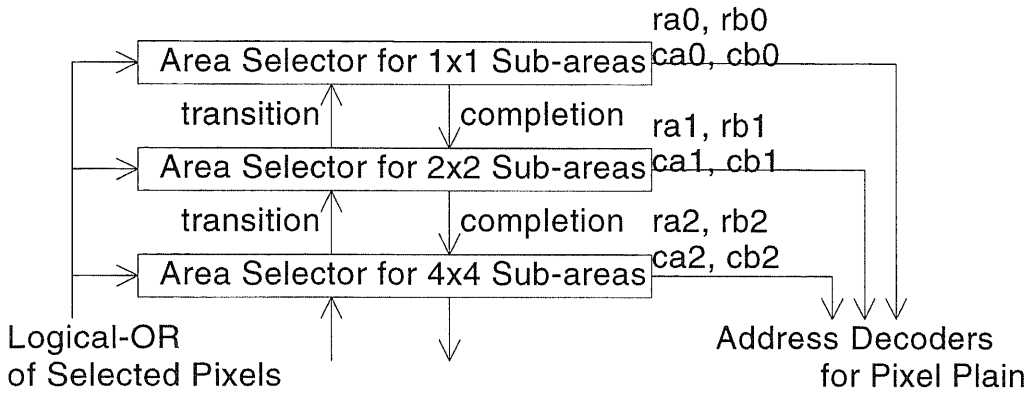


Figure 5.9: Architecture of selection controllers for pixel plain.

Table 5.2: States of automata in the decoding process shown in Figure 5.4.

upper(s2)	W	A	B	B	B	B	B	C	D
lower(s1)	W	W	W	A	B	C	D	W	W
1:4 tree code	1	0	1	1	0	1	0	0	0

discussed in the previous section.

It is also notable that the external controller of scan procedure and address decoders are completely identical with those for the decoders of 1:4 tree code discussed in section 3.4.

5.2.2 Circuit of controller

The selection procedure of pixels discussed above is implemented by the hierarchical structure of controllers which correspond to each size of sub-area, as shown in Figure 5.9. Each controller has output of `ra` and `rb` for the selection of the upper half and the lower half of its sub-areas, respectively, and `ca` and `cb` for the selection of the left half and the right half of its sub-areas, respectively.

For example, the scan procedure shown in Figure 5.4 can be implemented by two level automata, and their states should be transient as shown in Table 5.2. Here `s1` and `s2` are the automata indicating 1×1 and 2×2 sub-areas, respectively, and state `W` represents selecting all of its sub-area. The state `A`, ...`D` represent the selecting upper-left, upper-right, lower-left, and lower-right of its sub-area, respectively. For

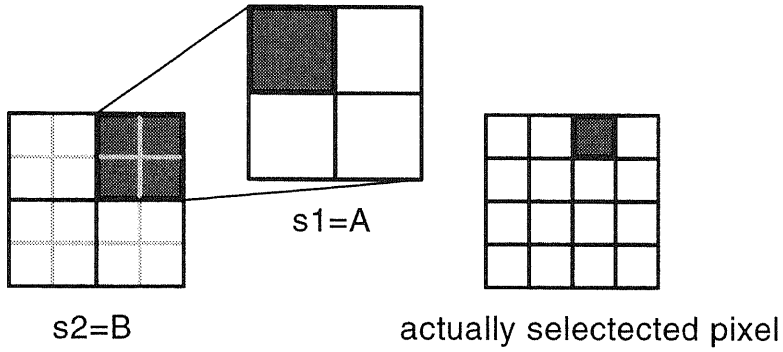


Figure 5.10: Selected pixel by the states of (s2,s1)=(B,A)

example, the states of (s2,s1)=(B,A) in the fourth step in Table 5.2, the upper-left 1×1 area (or pixel) which is indicated by s1=A, in the upper-right 2×2 sub-area which is indicated by s2=B, as shown in Figure 5.10.

The logical-OR of the selected pixels' value is given back to each controller, and the controllers make transition according to its current state and the value, as shown in Table 5.3. Here the state of W represents the state of selecting the whole of its sub-area, and A, B, C, and D represents the state of selecting the upper-left, upper-right, lower-left, and lower-right of its sub-area, respectively. Ti and Co is the transition and the completion signals as inputs, respectively, and To and Co is the transition and the completion signals as outputs, respectively, and V is the logical-OR of the selected pixels' value in the pixel plain.

The designed circuit of the controller is shown in Figure 5.11. It is notable that the new signal SC is added, to control the scan process in order to implement the spatial adaptive resolution scan, discussed in section 4.2. If SC is low, the scan is proceeded as usual 1:4 tree scan, while the scan step for this controller no longer proceeds to its lower level if SC is high; it returns the *completion* signal immediately.

5.2.3 Circuit of address decoder

The select lines for each pixel are decoded according to Ra, Rb, Ca, and Cb of each controller. For example, the row select line for the top row, r0, is activated when Ra0, ...Ran are all 1, since Ra represents the selection of upper-half in each subarea.

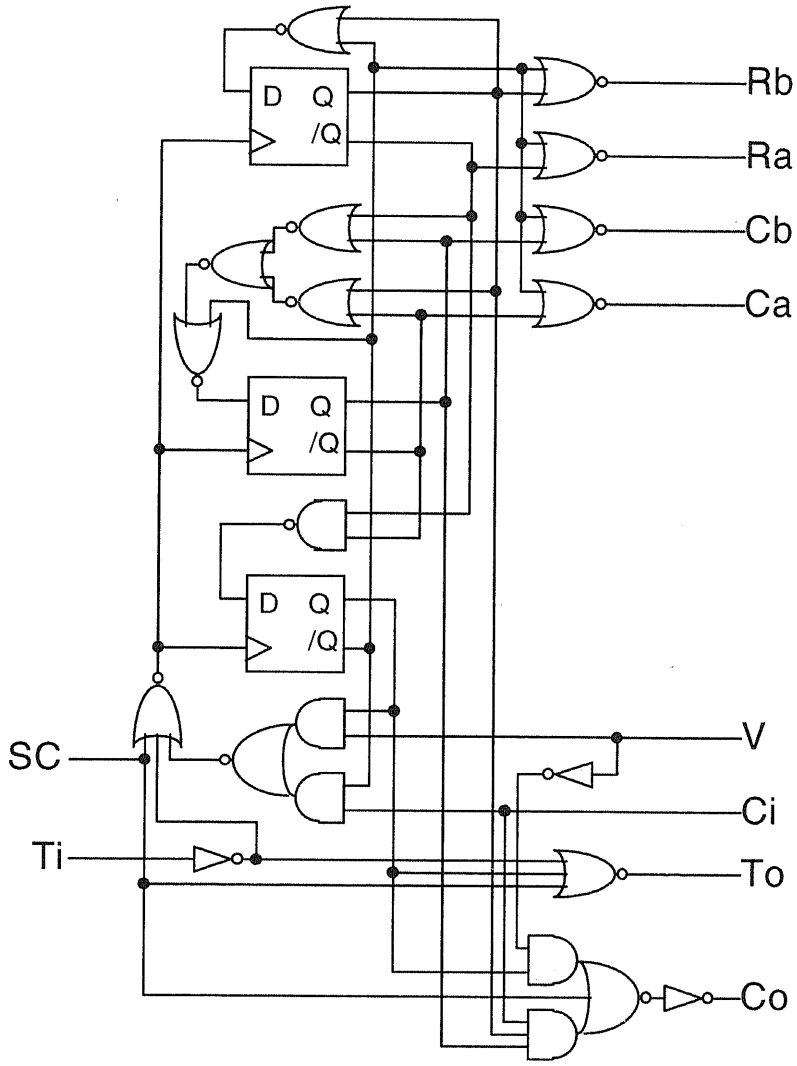


Figure 5.11: Circuit of the controller for the external address decoders.

Table 5.3: State transition diagram of the controller in each level for the external address decoder of 1:4 tree sensor.

Inputs					Outputs					
V	Ci	Ti	S	S'	Ra	Rb	Ca	Cb	Co	To
0	-	↑	W	W	1	1	1	1	1	0
1	-	↑	W	A	1	1	1	1	0	↑
-	0	↑	A	A	1	0	1	0	0	↑
-	1	↑	A	B	1	0	1	0	0	↑
-	0	↑	B	B	1	0	0	1	0	↑
-	1	↑	B	C	1	0	0	1	0	↑
-	0	↑	C	C	0	1	1	0	0	↑
-	1	↑	C	D	0	1	1	0	0	↑
-	0	↑	D	D	0	1	0	1	0	↑
-	1	↑	D	W	0	1	0	1	1	↑

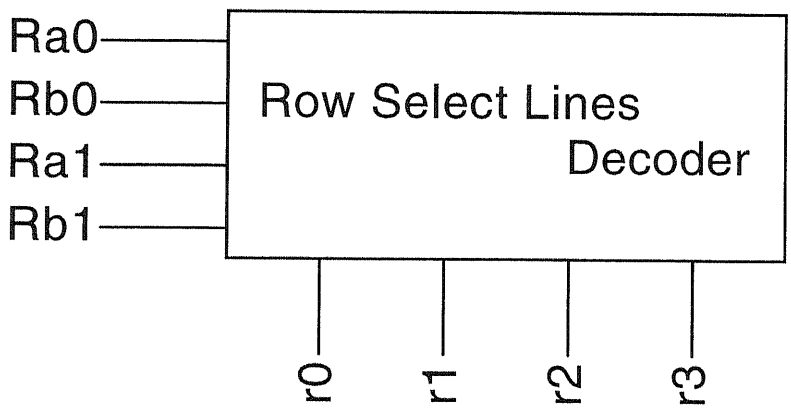


Figure 5.12: Functional block of row decoder for four rows of pixel plain.

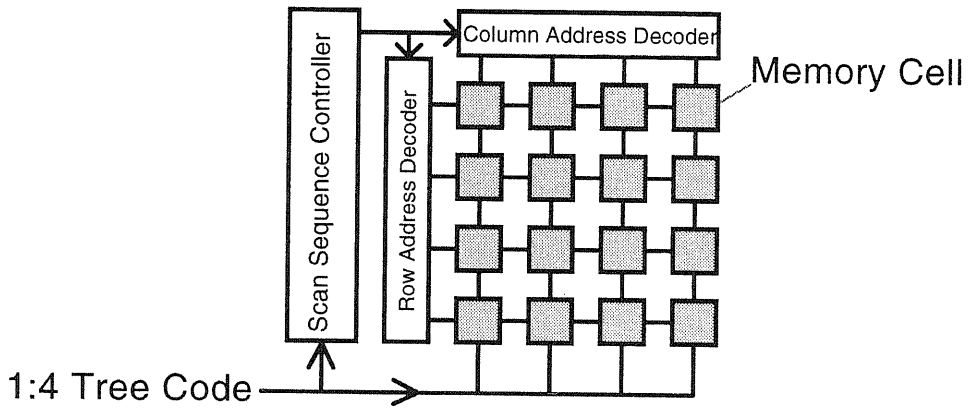


Figure 5.13: Architecture of 1:4 tree code decoder

The four row select lines of pixels, r0, ...
r3, as shown in Figure 5.12 are decoded as the following equations, which is similar to those discussed in section 3.4.

$$r0 = Ra1 \;\&\&\; Ra0$$

$$r1 = Ra1 \;\&\&\; Rb0$$

$$r2 = Rb1 \;\&\&\; Ra0$$

$$r3 = Rb1 \;\&\&\; Rb0$$

Note that && represents the logical-AND. The equations in case of N levels, which can select 2^N rows of pixels, are easily derived by extending the above equations for using N-input AND gates, and the column select lines are also implemented by the same way.

5.3 Circuits of Memory Cell Array for Decoding 1:4 Tree Code

The controllers and the address line decoders designed in section 5.2 are also used to implement the decoder of 1:4 tree code. The architecture of 1:4 tree code decoder is shown in Figure 5.13, and the "scan sequence controller" and "row and column address decoders" are completely identical to those in 1:4 tree sensor using external

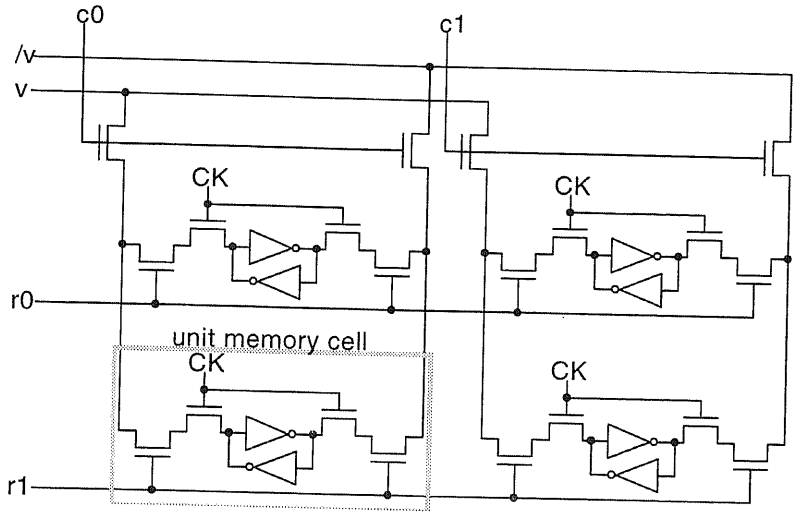


Figure 5.14: Architecture of memory cell plains to implement the decoders of 1:4 tree code

address decoders described in section 3.4.

The architecture of memory cell plain for the decoder, alternatively the pixel plain in Figure 5.5 is shown in Figure 5.14. Here `c0`, `c1` and `r0`, `r1` are column and row select lines generated by the controllers and the address decoders designed in the section 5.2, respectively. `v` and `/v` is the value to be stored for the selected areas, and its inverse, respectively, and `CK` is the clock signal to store the values for the memory cells.

This architecture is similar to that of the conventional SRAM, while more than two memory cells can be selected in this decoder architecture.

5.4 Functions and Circuits of Pixels for On-Sensor Image Processing

5.4.1 Pixel circuit for the inter-frame difference

Figure 5.15 shows the designed pixel circuit with the functions of making inter-frame difference for movie compression, as discussed in section 4.4.

The photo signal is received by the photo diode, and the photo current is converted

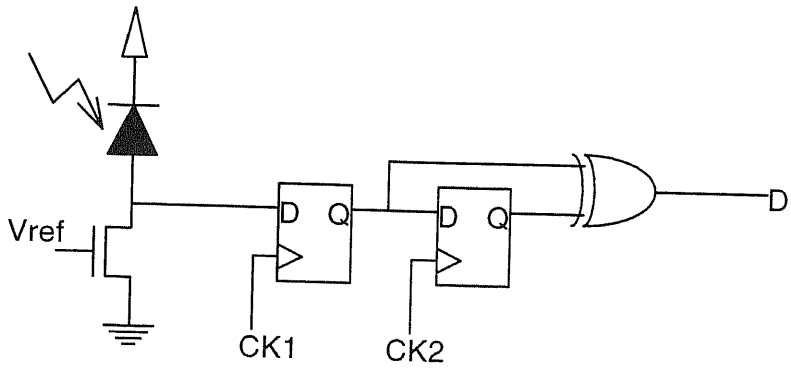


Figure 5.15: Circuit of pixels with the functions of making inter-frame difference.

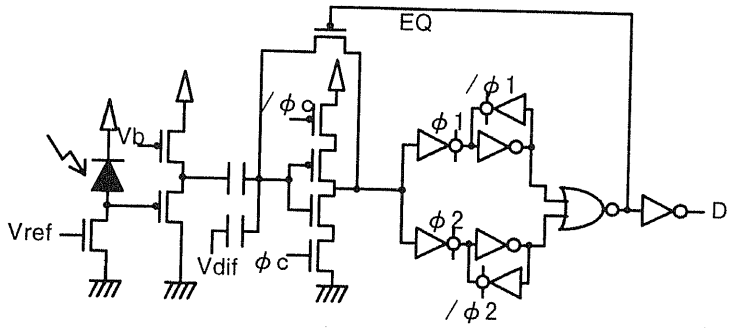


Figure 5.16: Pixel circuit for taking analog inter-frame differences

by the load transistor controlled by `Vref`, as the dark and bright light generates the logical low and high, respectively. The signal is latched by the former D-latch, controlled by `CK1`. The output of the former D-latch is transferred to the later D-latch after reading the output of `D`, which represents the difference of the two D-latchs' outputs. Keeping `CK2` as low after reset sequence, the output `D` is equal to the output of the former D-latch, which is just equal to the output of photo signal.

Figure 5.16 shows the pixel circuit for taking analog inter-frame differences for movie compression. Here `Vref` and `Vb` are bias voltage for load transistor and source follower, respectively. The clocked inverter at the center of figure is equalized by `EQ`, and then ϕ_c and $/\phi_c$ are provided for the clocked inverter to compare the input with the inverter threshold voltage. If the input is higher than inverter threshold voltage V_{inv}, the output goes to low, while it goes to high if the input is lower than V_{inv}. The output of source follower at the previous cycle, V_o^0 is stored at the upper

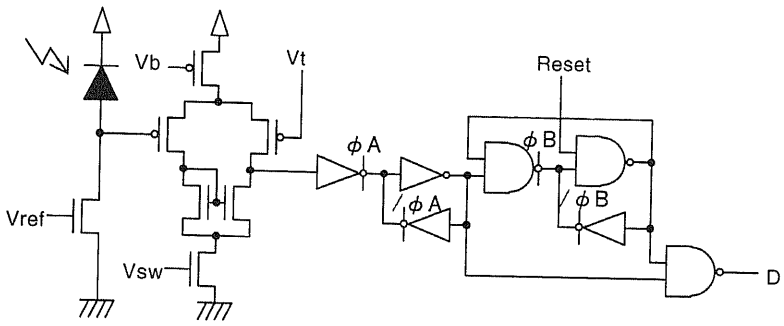


Figure 5.17: Pixel circuit for the scan of intensity by referring the charge-up time

capacitor, and `Vdif` $= V_{\text{inv}} - V_\theta$ is given, where V_θ is the effective difference to be detected between the current output of source follower, V_o and V_o^0. If V_o is larger than $V_o^0 + V_\theta$, the output of clocked inverter goes to 1, and this output is latched to the upper D-latch using ϕ_1 and $/\phi_1$. Next the `Vdif` of $V_{\text{inv}} + V_{\text{theta}}$ is given, and the output of clocked inverter goed to 1 when V_o is smaller than $V_o^0 - V_\theta$. This output is stored the lower D-latch using ϕ_2 and $/\phi_2$. Thus the output `D` is 1 if the absolute value of difference, $|V_o - V_o^0|$ is larger than V_θ, which indicates the effective difference in the successive two frames.

5.4.2 Pixel circuit for the gray-scale converting

In this section, we consider the pixel circuit for the scan of intensity by referring the charge-up time using 1:4 tree structure, discussed in section 4.3.2. Figure 5.17 shows this pixel circuit. The photo current is charged to the junction capacitance of photo diode, and the output voltage of the junction capacitance is compared by the following differential amplifier with the reference voltage `Vt`. `Vb` is the bias voltage for the differential amplifier, and `Vsw` is used to shutdown the amplifier when it is not used. At the first step of scan, the later D-latch is cleared by `Reset` signal, and the output of the differential amplifier is transferred to the former D-latch using ϕ_A and $/\phi_A$. The output of pixel `D` goes high if the output of photo-electronic converter is higher than reference voltage `Vt`, and the time from the reset sequence up to `D` goes high represents the intensity of light. The output `D` is scanned by 1:4 tree structure, and it is cleared after the scan, by transferring "1" of the former D-latch

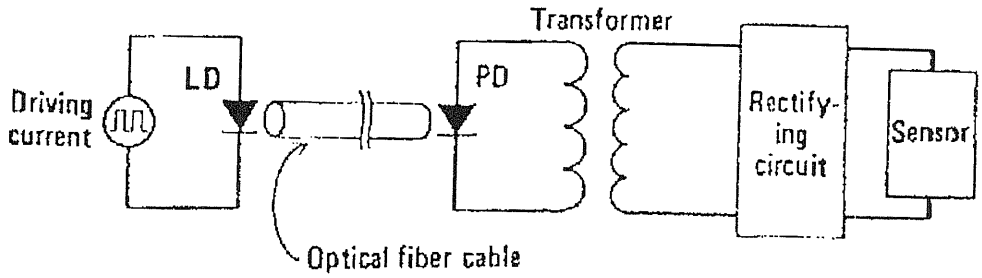


Fig. 1 Basic configuration of power supply of the pulsed-light-power-supply-type sensor

Figure 5.18: Block diagram of the power supply system using the energy of laser diode and optical fiber. (From [28])

to the later D-latch using ϕ_B and $/\phi_B$.

5.5 Light-Powered Image Sensor for Ultra-Low Power Operation

The conventional image sensors are operated by the external power supply. This section describes a concept of the image sensor operated by the power of received light.

Some previous studies of the methodologies to supply the power for image sensor using the laser diode and optical fiber are discussed, as shown in Figure 5.18[28].

The current of photo diode is expressed as follows,

$$I = I_s(e^{qV/nk_BT-1} - 1) - I_p, \tag{5.3}$$

where I_s and I_p is the saturation current and the photo current, respectively, and V is the given voltage of the photo diode. q, n, k_B, and T is the elementary charge, n-value, the Boltzmann's constant, and the operation temperature, respectively.

Assuming $I_p = 1\mu$A, $I_s = 20$pA, $n = 1$, and $T = 300$K, the supplied power by the photo diode and the operation voltage, and the results shows that one photo diode has a ability to supply about 0.1μW. Assuming that the consumed energy per each step of 1:4 tree scan as 100pJ, which will be estimated in section 6.1, the

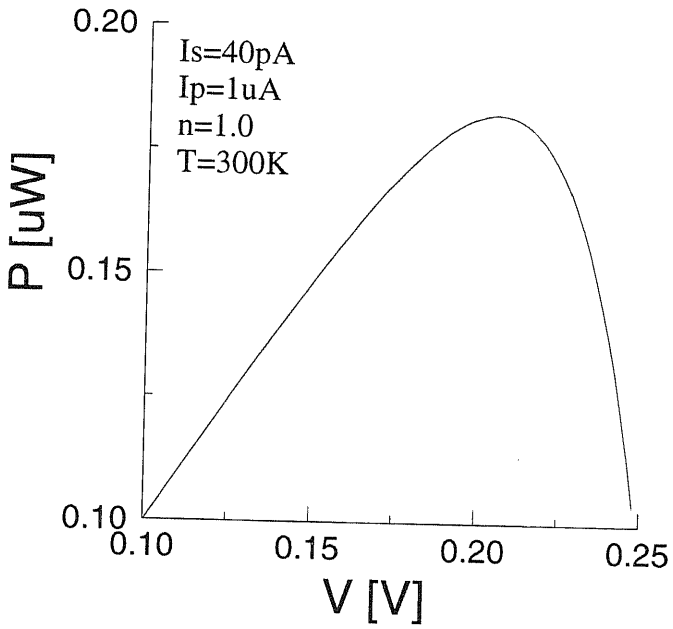


Figure 5.19: Calculated supplied power by the photo diode and the operation voltage.

operation frequency is estimated as 0.1μW÷100pJ=1kHz, which is expected to be enough operation frequency for the ultra-low power applications.

Figure 5.20 shows a concept of the architecture of 1:4 tree structure using the current mode operation. Each node selects the pixels, by closing the current paths to the upper, and the sum of the photo currents for the selected areas is generated.

One of the problems to implement such a circuit in Figure 5.20 is the power supply for the selection controllers. A possible solution is to place two photo diodes for each pixel; one for signals, and the other for power supply for the selection controllers, as shown in Figure 5.21.

5.6 Summary and Conclusion

In this chapter, the circuit implementations of 1:4 tree structure are discussed.

The circuit of node automata for the 1:4 tree structure are described, which has the advantage of lower operation by the gated clock. The reasonable way of two dimensional layout of nodes is also described, which is also reasonable to minize the

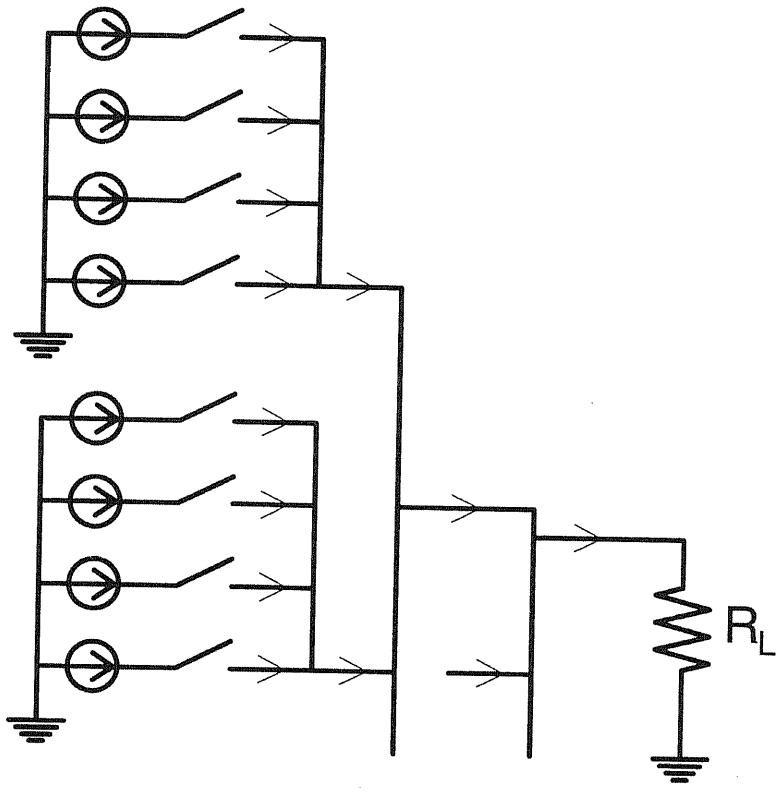


Figure 5.20: Architecture of 1:4 tree structure using current mode operation.

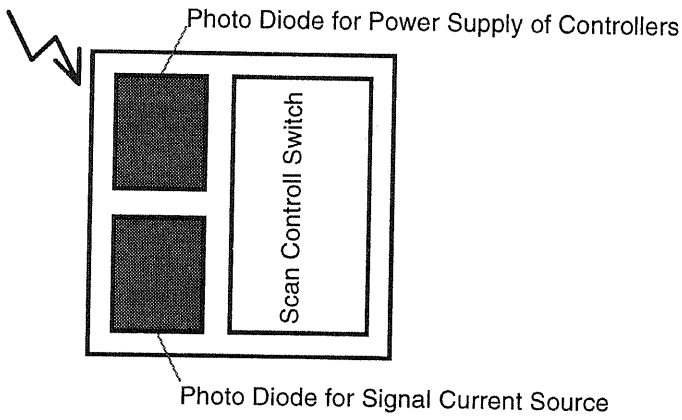


Figure 5.21: Pixel layout for light-powered 1:4 tree sensor

signal delay, since the upper node automaton can occupy the larger area for the larger signal driver.

It is also described the alternative architecture of the 1:4 tree sensor, the architecture of placing the pixel selection controllers outside the pixel plain, which has the advantage of keeping the high fill factor. The architecture of decoders of the 1:4 tree codes to two dimensional raster image is also described, which can easily implemented by using the address select controllers identical with the external-addressed 1:4 tree sensor.

It is also discussed the circuits and the functions of pixels, in order to have the functions more than simple photo-electric conversion. The idea and implemenatation of the pixels for inter-frame difference and gray-scale conversion are discussed.

The concept of the 1:4 tree sensor driven by the power of received light is also discussed.

Chapter 6

Design and Evaluation of 1:4 Tree Sensors

In this chapter, the designs and the evaluations for the manufactured 1:4 tree sensors are described. The characteristics of photo diodes fabricated the used CMOS 1.5μm process is discussed at first.

The prototype systems of the 1:4 tree image sensor using FPGAs will be described in section 6.2. It will be described the design and the evaluation of the 1:4 tree image sensor having the tree structure of node automata in the following section 6.3.

The design and the evaluation of the 1:4 tree sensor with the architecture of placing address selector externally will be described in section 6.4.

The design and the evaluation of the decoder of 1:4 tree code will be described in the section 6.5.

6.1 Design and Evaluation of Photo Diodes

We have designed the photo diode TEGs (test element groups) to estimate the photo current. The designed TEG chip in 2.3mm×2.3mm using CMOS 1.5μm technology is shown in Figure 6.1[1]. The designed photo diodes are shown in Figure 6.2.

[1]The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo with the collaboration by Nippon

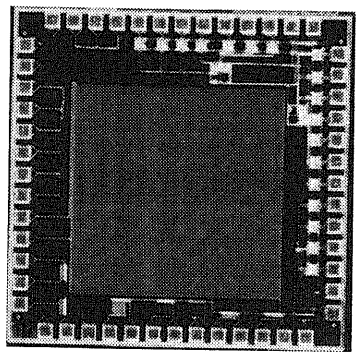


Figure 6.1: Chip photograph of the designed photo diode TEGs

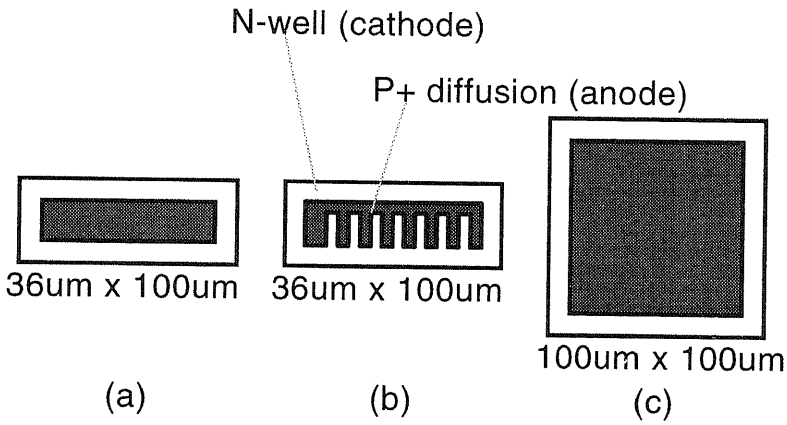


Figure 6.2: Designed photo diodes of tree types. (a)small diode, (b)small comb-shaped diffusion diode, and (c)large diode.

Three types of photo diodes are designed; the small one whose size is $36\mu\text{m} \times 100\mu\text{m}$ shown in Figure 6.2(a), the one of same size with the comb-shaped p+ diffusion in order to obtain the larger edge length of diffusion area shown in Figure 6.2(b), and the large one whose size is $100\mu\text{m} \times 100\mu\text{m}$ shown in Figure 6.2(c).

The characteristics of photo diodes are shown in Figure 6.3. Here the intensity of light is measured by exposure meter in exposure value (EV), defined as the follows,

$$E = 2^{\text{EV}} C/S. \tag{6.1}$$

where E is the illuminance in lux[2], C is the correction coefficient determined ex-

Motorola and Dai Nippon Printing Corporation.

[2]1[lux] is defined as the illuminance at the distance of 1m from the light source of 1[cd]. 1[cd]

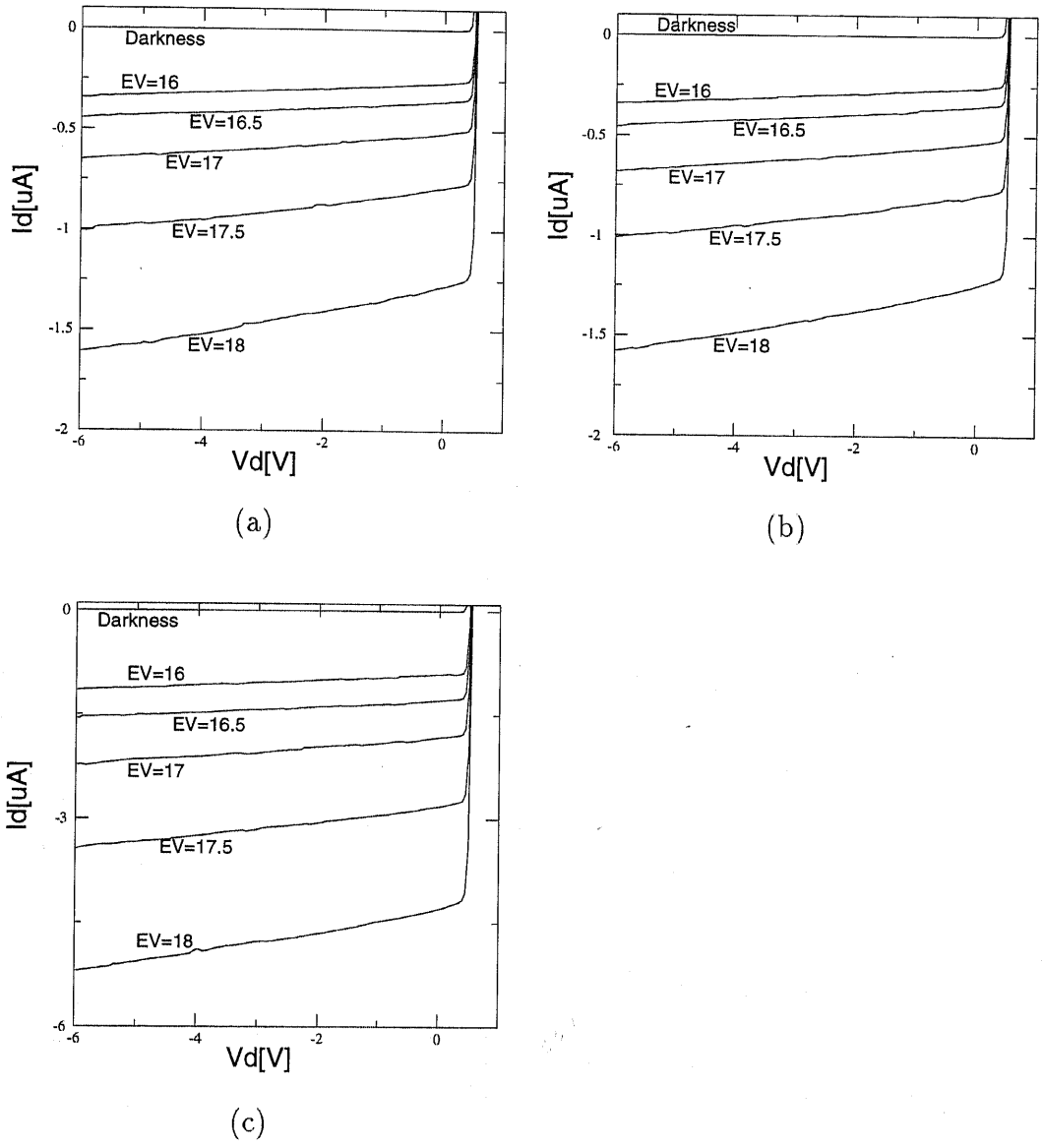


Figure 6.3: Measured characteristics of fabricated photo diodes. (a)small diode, (b)small comb-shaped diffusion diode, and (c)large diode.

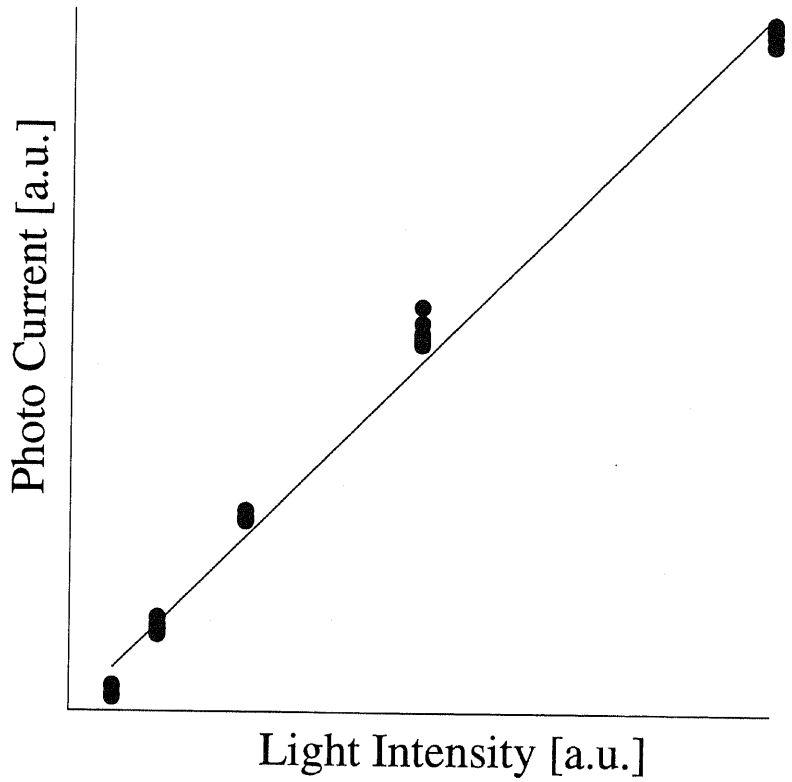


Figure 6.4: Relation of the normalized photo current per area and the normalized light intensity

perimentally which is 330, and S is the film sensitivity used in the exposure meter which is ISO400 in this case. The distance between exposure meter and light source is 3cm.

Seen in Figure 6.3, the photo currents for the photo diode(a) and (b) in Figure 6.2(a) and (b), respectively, are almost the same, and the reason is that the depletion layer between teeth in comb-shaped diffusion area in photo diode (b) is overlapped, and the distribution of depletion layer, which is the area photon generates the pair of electron and hole, is identical with that of photo diode (a).

It is also indicated that the intensity of light increases twice, the photo current is also increase about twice, and the photo current is almost proportional to the size

is defined as 1/60 of the light intensity of the black body whose temperature is the melting point of platinum, 2045K.

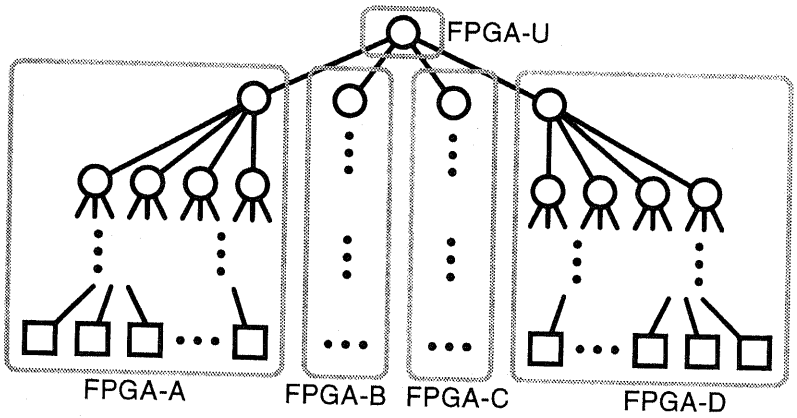


Figure 6.5: Configure of the 1:4 tree structure using five FPGAs.

of the diffusion area. The relation of the normalized photo current per area and the normalized intensity is shown in Figure 6.4, and it indicates that the photo current is proportional to both light intensity and size of diffusion area.

6.2 Prototype System for 1:4 Tree Sensor using Node Automata

6.2.1 Prototype implementation 1:4 tree sensor using FPGAs

We have developed the prototype system of the 1:4 tree sensor using FPGAs(Field Programmable Gate Array). We have employed the FPGA of XC4025[29] , which is the production of Xilinx Inc., which has logic blocks whose logical functions can be defined by the configure ROMs outside. The developed prototype systems consists of five FPGAs, four for the lower sub-trees, and one for the top node, as shown in Figure 6.5, where the number of pixels is $1,024 = 2^{10}$ with the 6 levels of 1:4 tree structure. Figure 6.6 shows the photograph of the developed prototype system.

The function of the pixel is emulated by having the flip-flops as the pixel value, which can be set serially from outside using shift registers, as shown in Figure 6.8. The image data captured by the video camera are transferred to the PC, then the PC transfers them to the prototype system serially.

The 1:4 tree scan for the prototype systems is executed, with the clock signal from

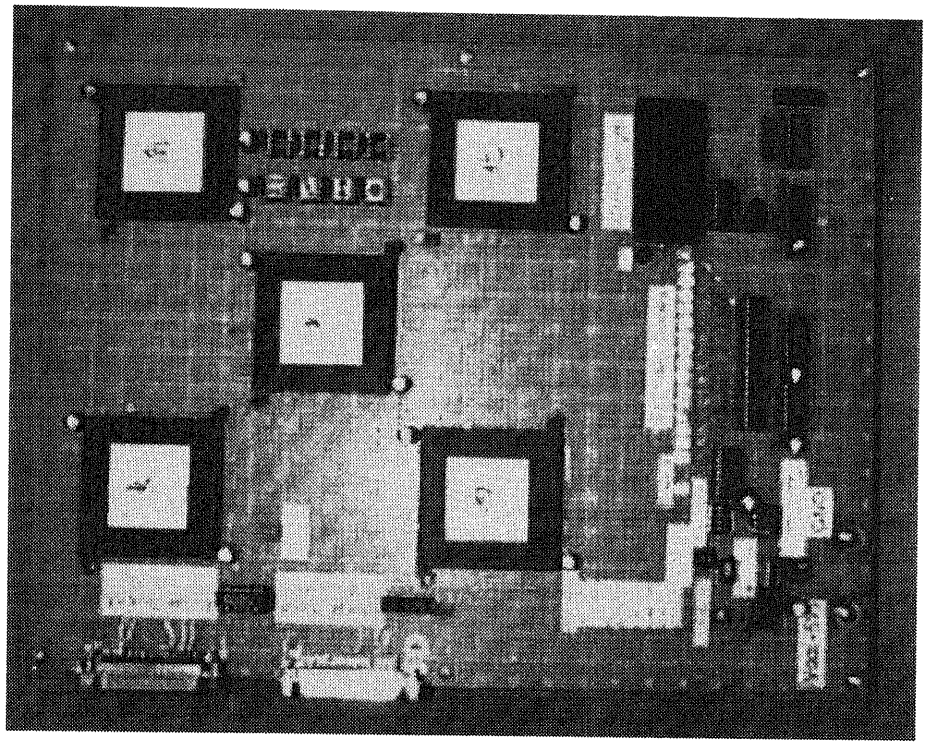


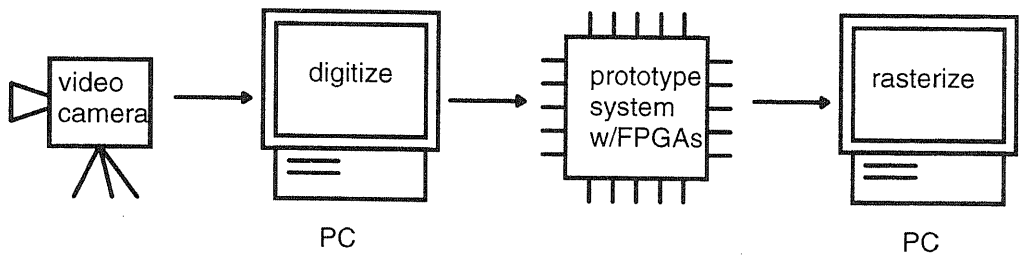
Figure 6.6: Photograph of the developed prototype system for 1:4 tree sensor.

the another PC; this PC is prepared to receive the 1:4 tree code and decode it to two dimensional images, too.

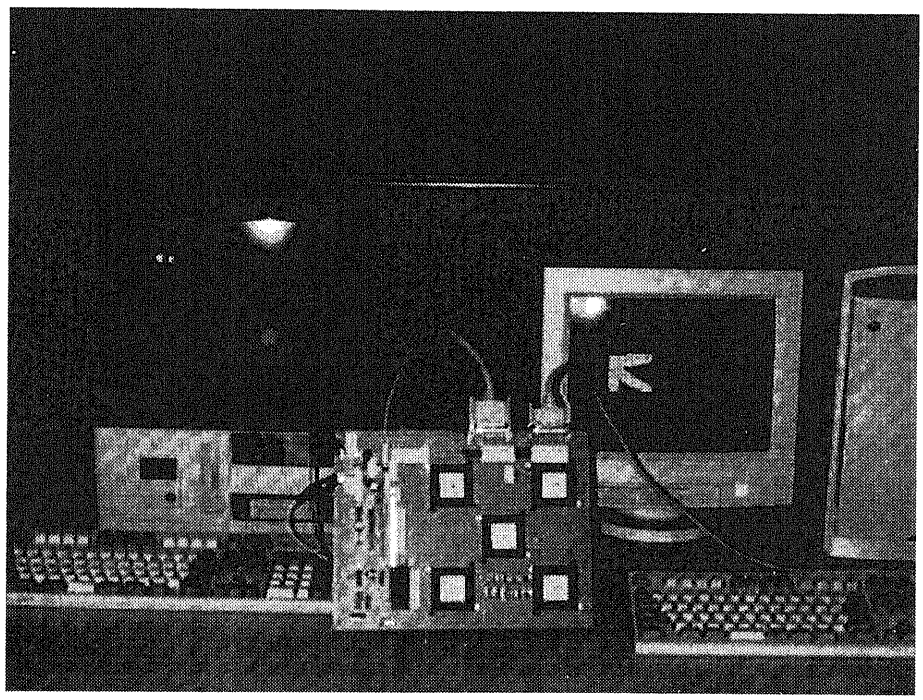
The configure of this system is described in Figure 6.7(a), with the photograph in Figure 6.7(b).

6.2.2 Evaluation of prototype system

Figure 6.9 shows the decoded images from the 1:4 tree code generated by the developed prototype system, from the image captured by video camera. The proper operations of the developed prototype system are observed, and it is also shown that 1:4 tree code length can be smaller than that of the raster scan, 1024. The operation of the developed prototype system with the clock frequency up to 1MHz is observed, which is restricted by the ability of the clock generator. The operation with the higher clock frequency is expected, since the number of inter-connection between FPGAs is very small.



(a)



(b)

Figure 6.7: Model of the developed prototype system of 1:4 tree sensor(a), and its photograph(b).

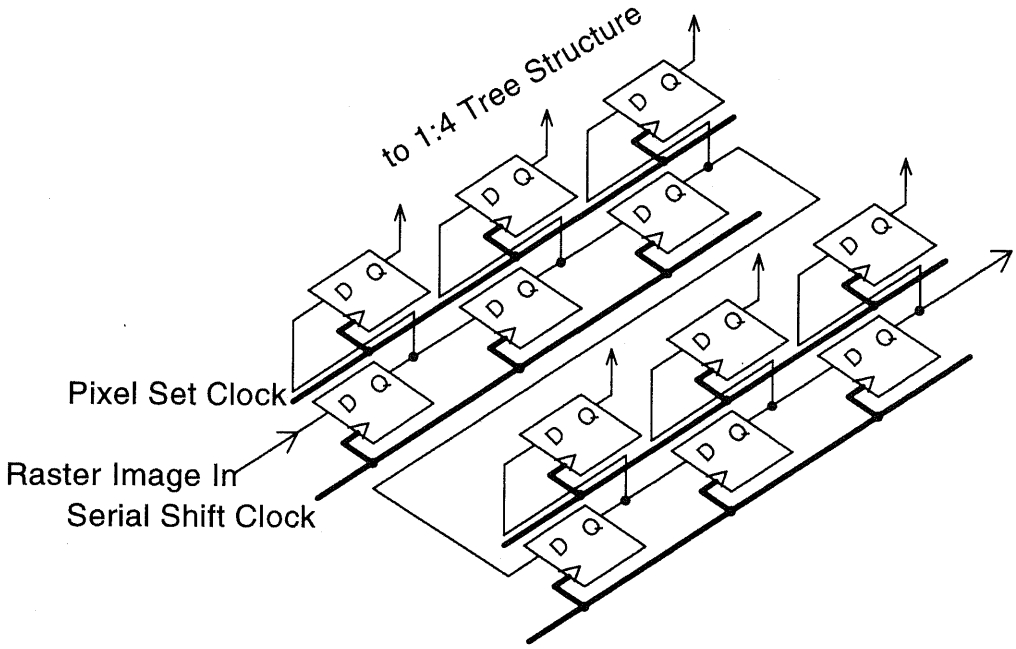
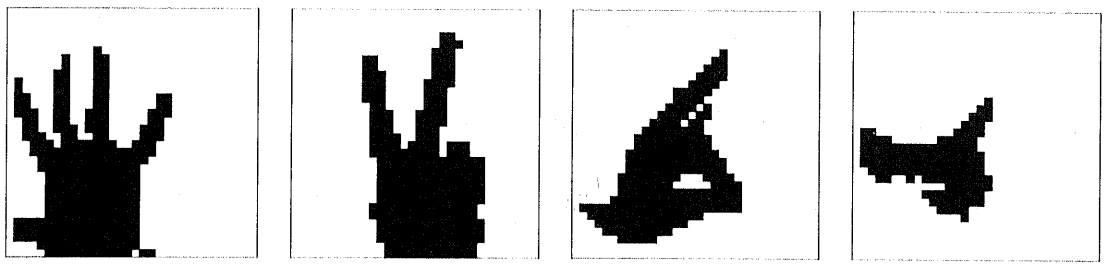
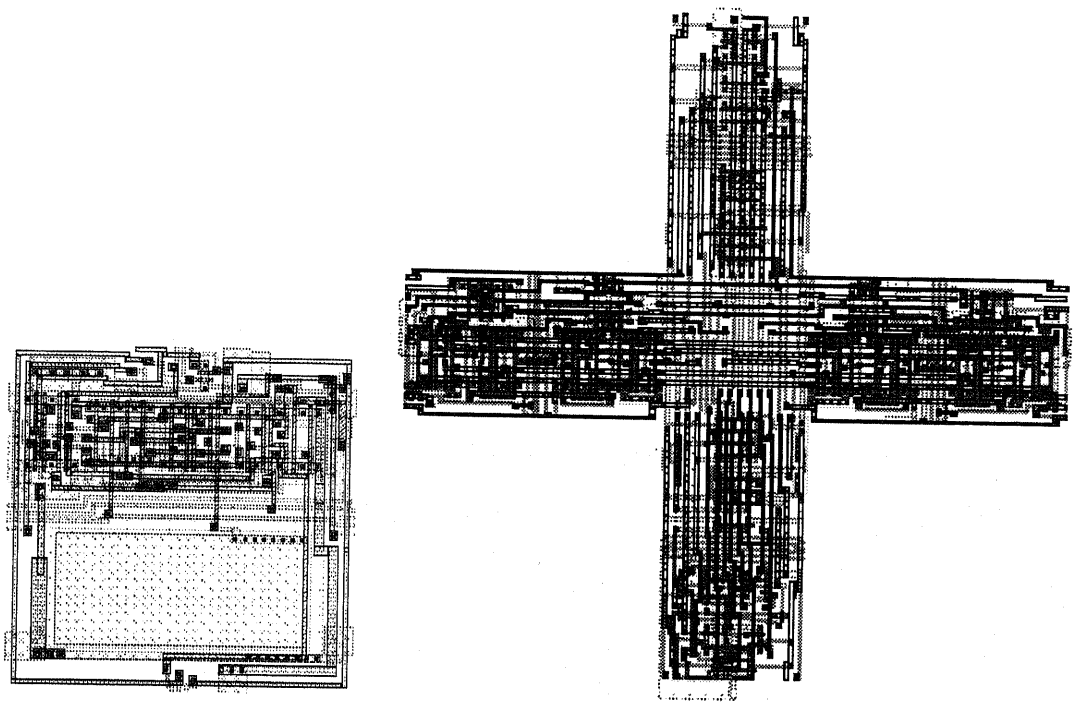


Figure 6.8: Architecture of serial shift registers to emulate the pixels



"Par"	"Peace"	"Fox"	"Good"
(573bit)	(529it)	(433it)	(293bit)
(a)	(b)	(c)	(d)

Figure 6.9: Four sample images decoded from 1:4 tree code generated by the developed prototype system. The code lengthes are also shown below.



(a) (b)

Figure 6.10: Layout of the pixel(a) and the node automata(b) in the designed 1:4 tree sensor.

6.3 Design and Evaluation of 1:4 Tree Sensor using Node Automata

6.3.1 Design of 1:4 tree sensor using node automata

The 1:4 tree image sensor with the tree structure of node automata has been designed, by using the circuits of node automata designed in Figure 5.2, with the pixel which can take inter-frame difference, as designed in Figure 5.15. The layouts of the each pixel and the each node are shown in Figure 6.10, using CMOS 1.5μm technology, and the whole layout of the chip is shown in Figure 6.11. The chip photo graph of the fabricated chip is also shown in Figure 6.12[3]. The chip size is

[3]The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo with the collaboration by Nippon Motorola and Dai Nippon Printing Corporation.

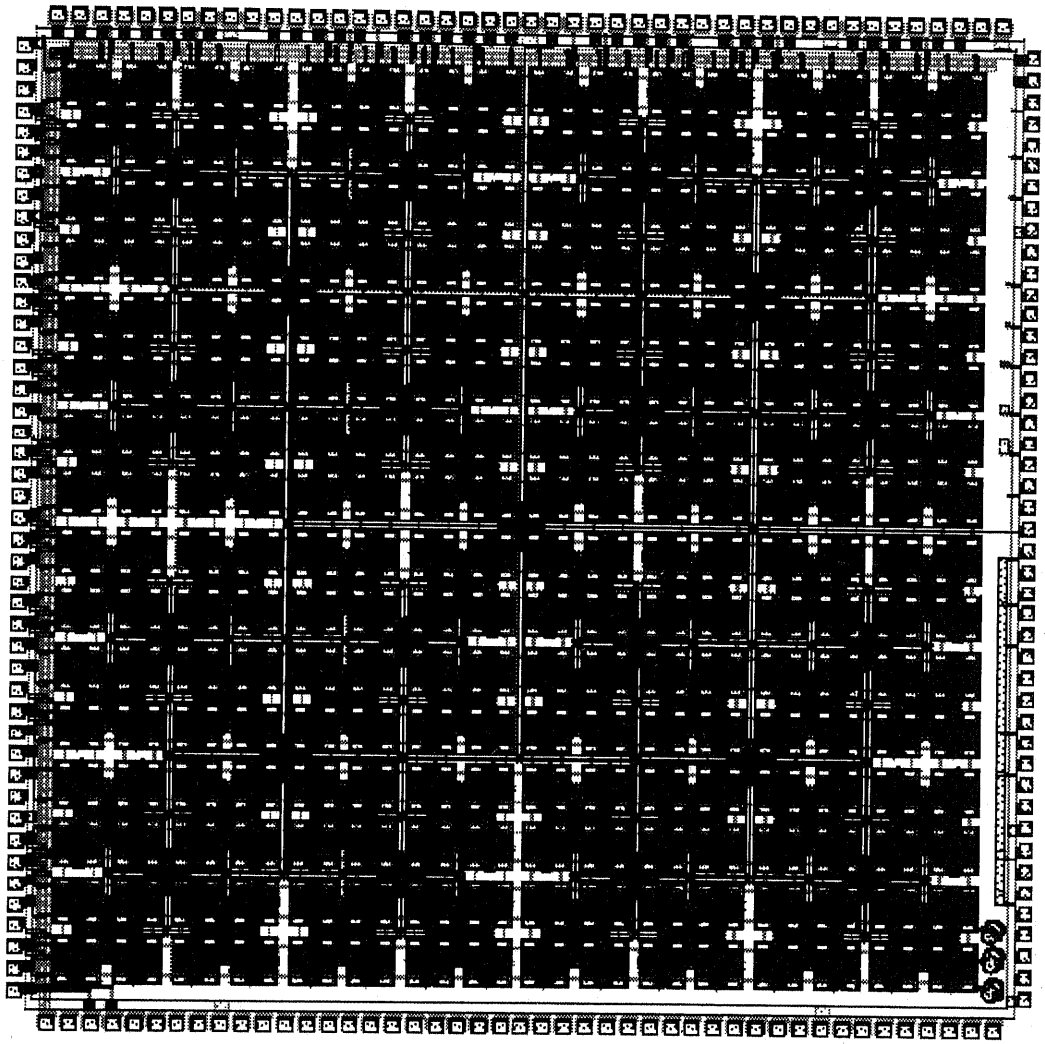


Figure 6.11: Whole layout of the designed 32 × 32 pixels 1:4 tree sensor.

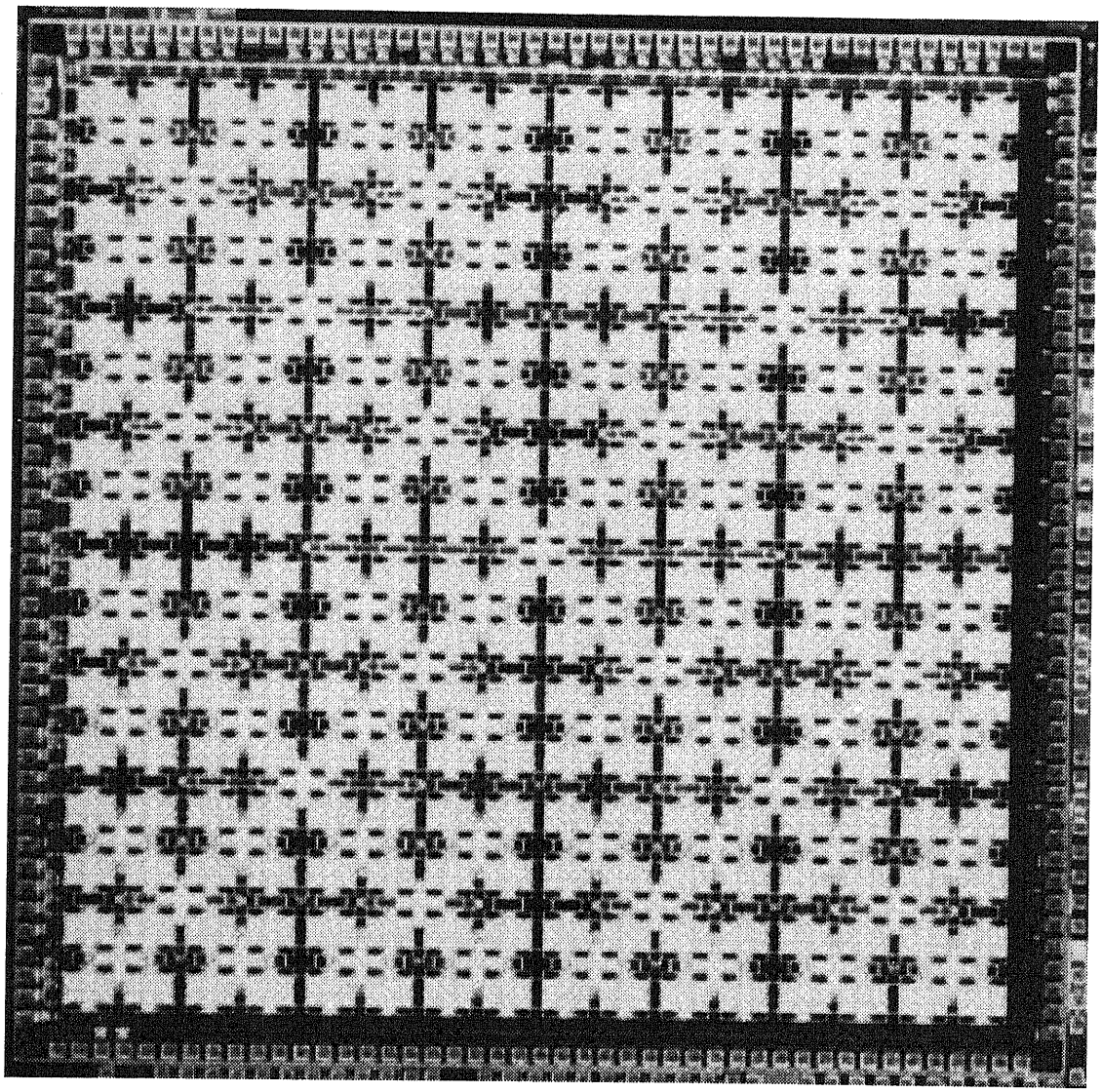


Figure 6.12: Chip photograph of the designed 32×32 pixels 1:4 tree sensor.

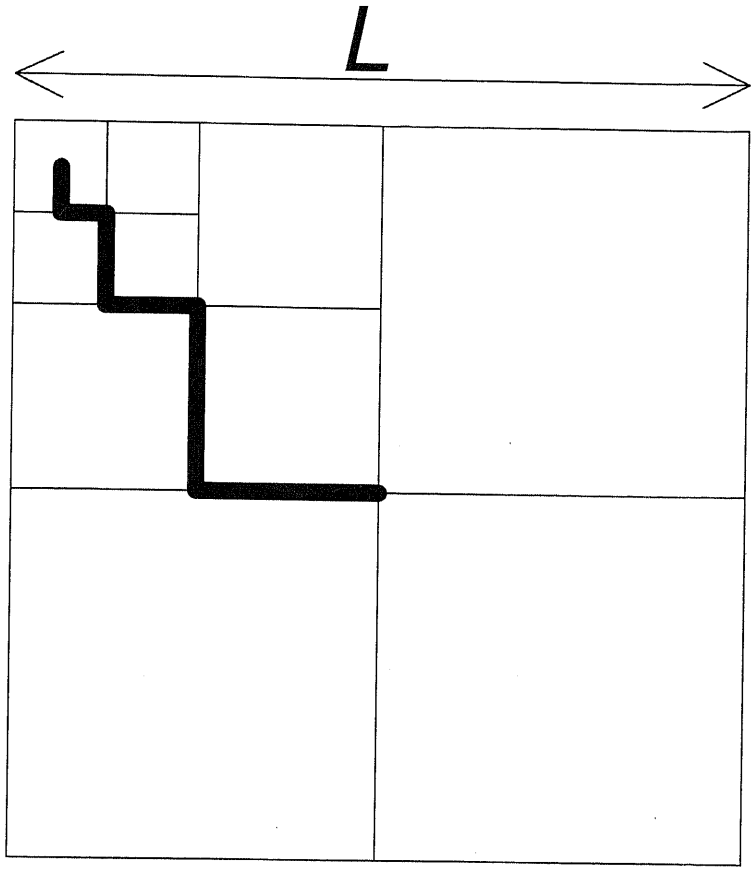


Figure 6.13: Model of scan path to from the top node to the pixel. L is the length of the edge in whole chip.

7.2mm×7.2mm, and the number of transistors 104,784, with the fill factor of about 20%.

6.3.2 Evaluation of 1:4 tree sensor using node automata

The power consumption of designed 1:4 tree sensor is estimated by the model and the simulations. In the designed node circuit, the clock signal is provided just along the scanned signal path, which is expected to be suitable to reduce power consumption.

Assuming the number of pixels is $n \times n$ and the length of edge in whole chip is L, the activated signal scan path from the top node to the pixel is shown in Figure

6.13. The length of this scan path, L_t, is derived as follows.

$$L_t = \sum_{l=1}^{N} \frac{L}{2} \times 2^{-(l-1)} \approx L \propto n \tag{6.2}$$

This scan path is activated at every step of 1:4 tree scan, and the power consumption per step by charging and discharging this scan path is expected to be proportional to n.

Assuming the number of branches as b, the number of transitions of each node through all the scan steps until finish, is equal to $(b+1)$ for the nodes whose values are "1", while no transitions occur in the nodes whose values are "0". Total number of transition of nodes in whole tree structure, N_{node} is expressed as follows,

$$N_{\text{node}} = (b+1)\sum_{l=1}^{N} b^{N-l} p_l, \tag{6.3}$$

where p_l is the probability that the node at the level l has the value of "1".

The expectation of power consumption per scan step, $\overline{N_{\text{node}}}$, is derived by dividing N_{node} by the mean code length $\overline{L}$, as follows.

$$\begin{aligned} \overline{N_{\text{node}}} &= N_{\text{node}}/\overline{L} \\ &= \frac{(b+1)\sum_{l=1}^{N} b^{N-l} p_l}{1+\sum_{l=1}^{N} b^{N-l+1} p_l} \approx \frac{b+1}{b} \end{aligned} \tag{6.4}$$

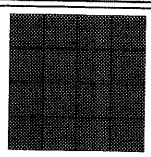
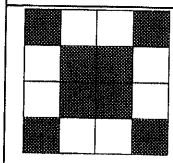
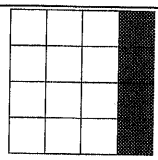
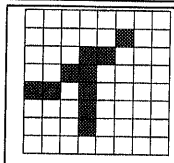
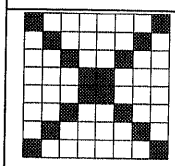
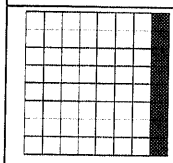
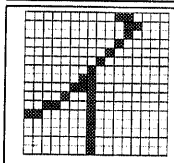
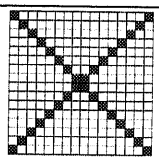
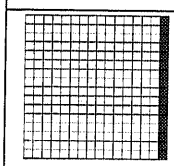
In case of 1:4 tree, assuming $b = 4$, $\overline{N_{\text{node}}}$ is derived as $5/4$. In the designed circuit of node automaton shown in Figure 5.2, the expectation of the number of flip-flops per each transition is derived as follows.

$$\frac{1+1+2+2+2}{5} = \frac{8}{5} \tag{6.5}$$

Thus the total expectation of the number of flip-flops which make transitions in the whole 1:4 tree structure is derived as $5/4 \times 8/5 = 2$, which is constant through all the scan steps.

The total power consumption per scan step, U_{tree} is expressed the sum of the constant part, U_0^t and the part proportional to n, $U_1^t \cdot n$, as follows.

Table 6.1: Power consumption of 1:4 tree structure per scan step

Image	# of pixels	Code length	U[pJ/bit]
	4×4	21bit	19.5
	4×4	21bit	19.8
	4×4	13bit	19.1
	8×8	41bit	24.4
	8×8	53bit	24.6
	8×8	29bit	24.2
	16×16	93bit	32.5
	16×16	117bit	34.7
	16×16	61bit	32.5

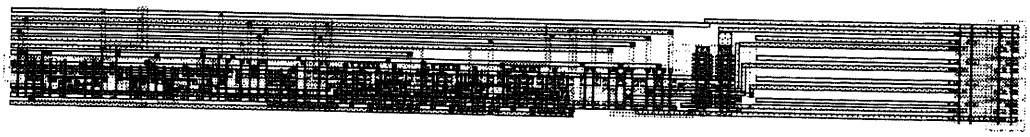


Figure 6.14: Layout of the designed address select controller.

$$U_{\text{tree}} = U_0^t + U_1^t n \tag{6.6}$$

Table 6.1 shows the calculated power consumption of 1:4 tree structure per scan step, using the simulation results of spice for the circuits of 1:4 tree structure consists of node automata and voltage sources as the value of pixels. The load capacitance of each node is determined to be proportional to 2^l, where l is the level with assuming the standard 1.5μm CMOS process, and the total number of levels, N is $N = 2$ for 4×4 pixels, $N = 3$ for 8×8 pixels, and $N = 4$ for 16×16 pixels, respectively.

The two part of power consumption in Equation (6.6) is derived as $U_0^t = 14.4$pJ and $U_1^t = 1.26$pJ, respectively. Seen above, it is shown that the power consumption in 1:4 tree structure is proportional to $O(n)$, while $O(n^2)$ for CCD raster scan image sensor, which is expected to be small enough for larger n.

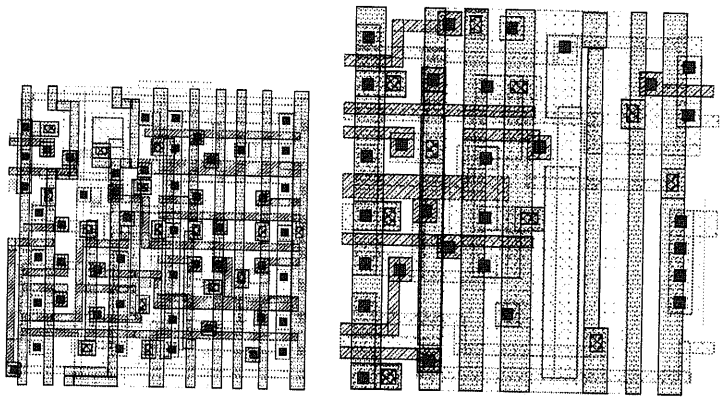
Because of the design mistake in pad assignment suitable for LSI tester, the measurement of the fabricated chip has not been carried out yet.

6.4 Design and Evaluation of 1:4 Tree Sensor using External Address Decoders

6.4.1 Design of the 1:4 tree sensor using external address decoders

The 1:4 tree image sensors using the external address decoders have been designed, by using the circuits of the controller and the address decoders designed in Figure 5.11, which is shown in Figure 6.14.

The two types of pixels are designed; the pixels with the function of taking inter-frame difference, as shown in Figure 6.15(a), and the compact pixels without taking inter-frame difference, which has only one latch to hold data, as shown in Figure



(a) (b)

Figure 6.15: Layouts of the designed pixels for the 1:4 tree sensor using external address decoders. (a)pixel with taking inter-frame difference, (b)simple, compact pixel without taking inter-frame difference.

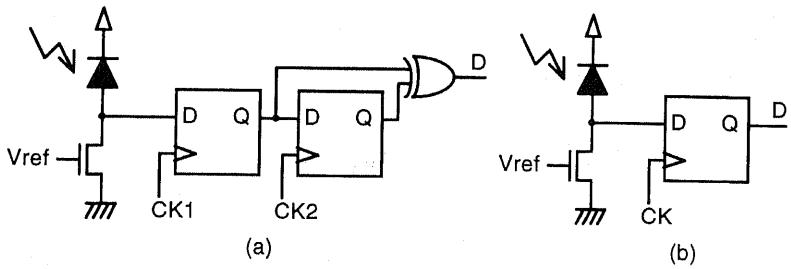


Figure 6.16: Circuits of pixels used for the 1:4 tree sensor using external address decoders. (a)pixel with taking inter-frame difference, (b)simple, compact pixel without taking inter-frame difference.

6.15(b). The circuit of each pixel is shown in Figure 6.16.

Two chips are designed using CMOS 1.5μm technology; 64×64 pixels chip with the function of taking inter-frame difference using the pixels in Figure 6.15(a), and 128×128 pixels chip without the function of taking inter-frame difference using the pixels in Figure 6.15(b). Whole layouts of the designed chip are shown in Figure 6.17 and Figure 6.18, respectively.

The chip photographs of both chips are shown in Figure 6.19 and Figure 6.20, respectively[4].

The chip size of the 64×64 chip is 4.8mm×4.8mm, and the number of transistors is 120,045, with $64 \times 64 = 4,096$ pixels which have the functions of taking inter-frame differences.

The chip size of the 128×128 chip is 7.2mm×7.2mm, and the number of transistors is 207,661, with $128 \times 128 = 16,384$ pixels which have the simple photo-electron conversion function without taking inter-frame differences.

6.4.2 Evaluation of the 1:4 tree sensor using external address decoders

The power consumption in the 1:4 tree architecture using external address decoders should be larger than that of the 1:4 tree sensor designed in section 6.3, since the long pixel select lines across the pixel plain is activated at scan step, while the only one signal scan path is activated in the 1:4 tree sensor as discussed in section 6.3.2.

The power consumption in the 1:4 tree sensor using external address decoders is composed of the following parts.

- power consumption of charging and discharging each bit line, B_i.
- power consumption of charging and discharging each row select line, R_j.
- power consumption of charging and discharging the word line, W.

[4]The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo with the collaboration by Nippon Motorola and Dai Nippon Printing Corporation.

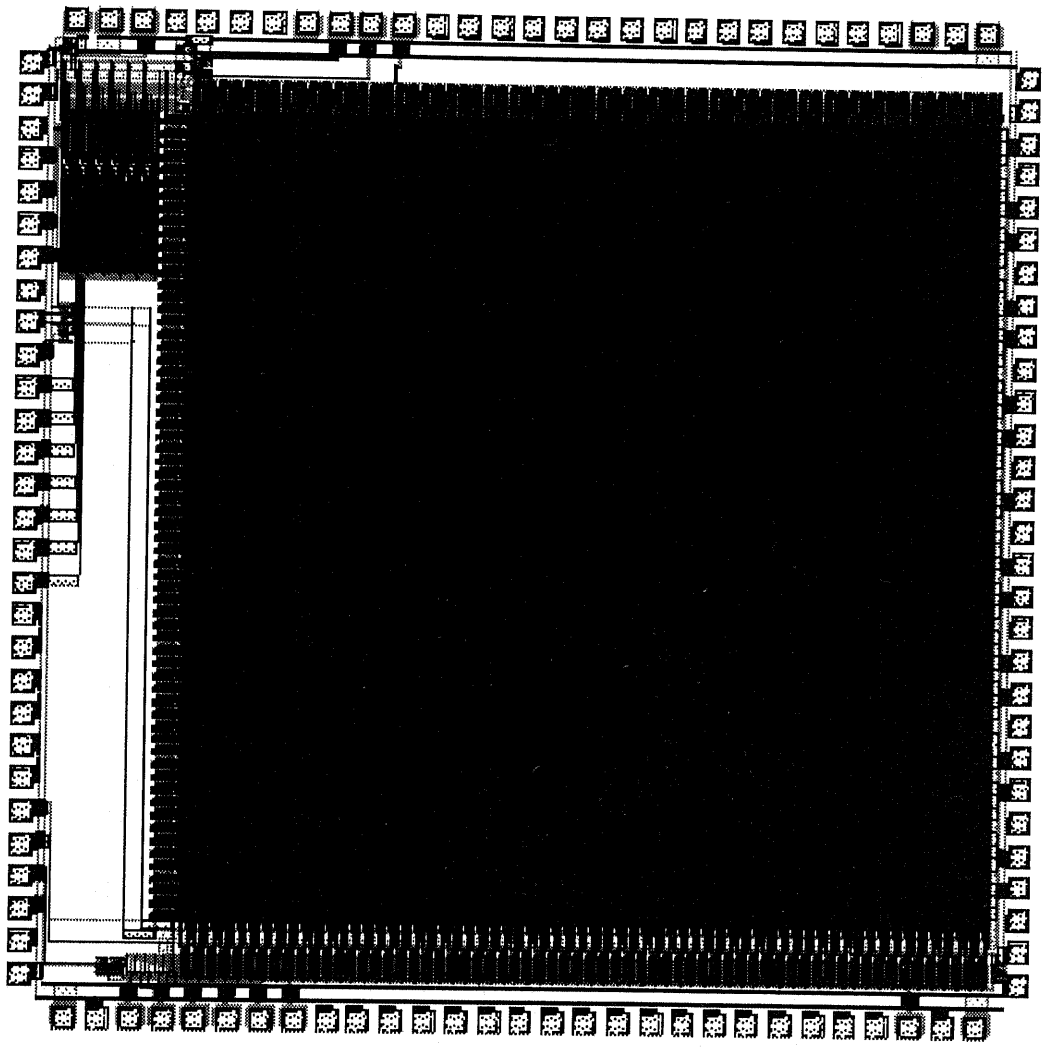


Figure 6.17: Whole layout of the designed 64 × 64 1:4 tree sensor using external address decoders, with taking inter-frame difference.

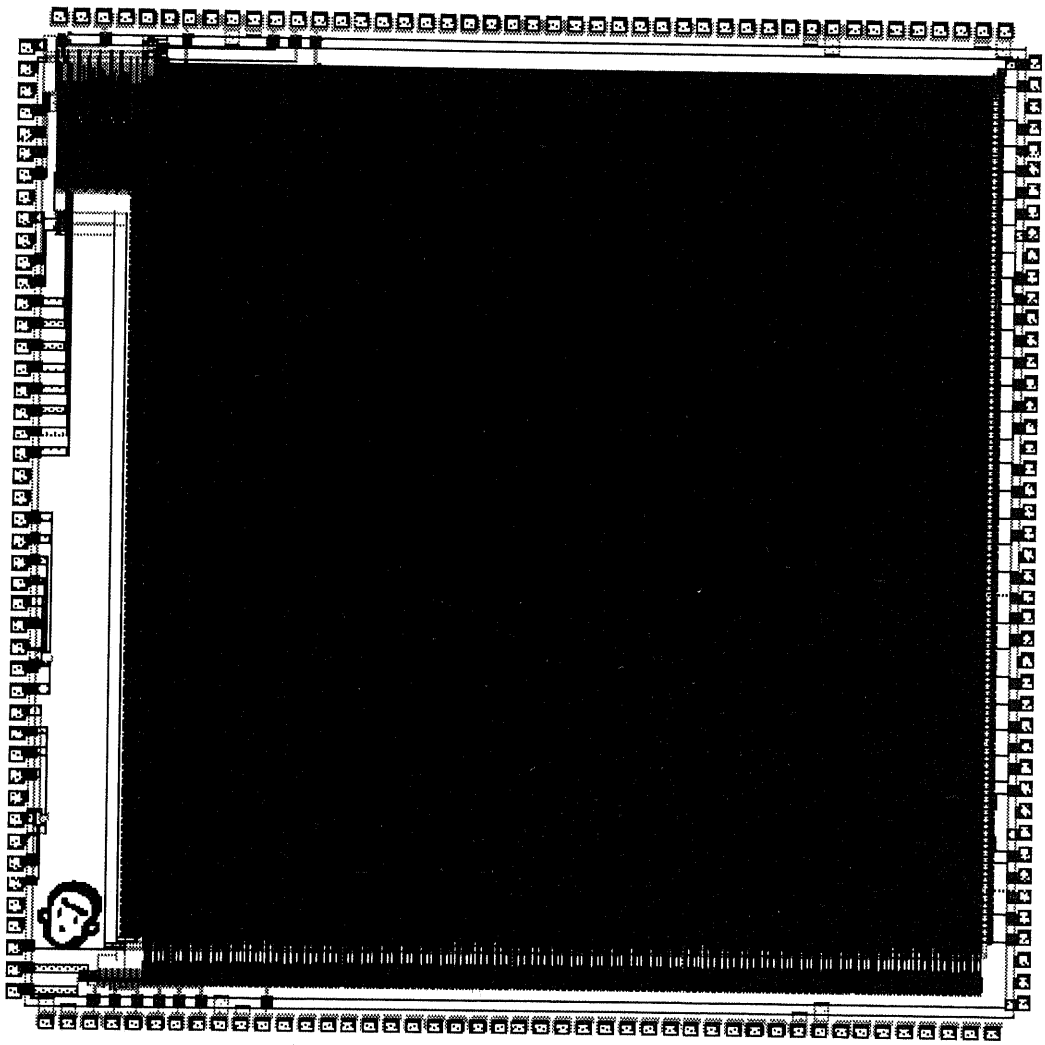


Figure 6.18: Whole layout of the designed 128 × 128 1:4 tree sensor using external address decoders.

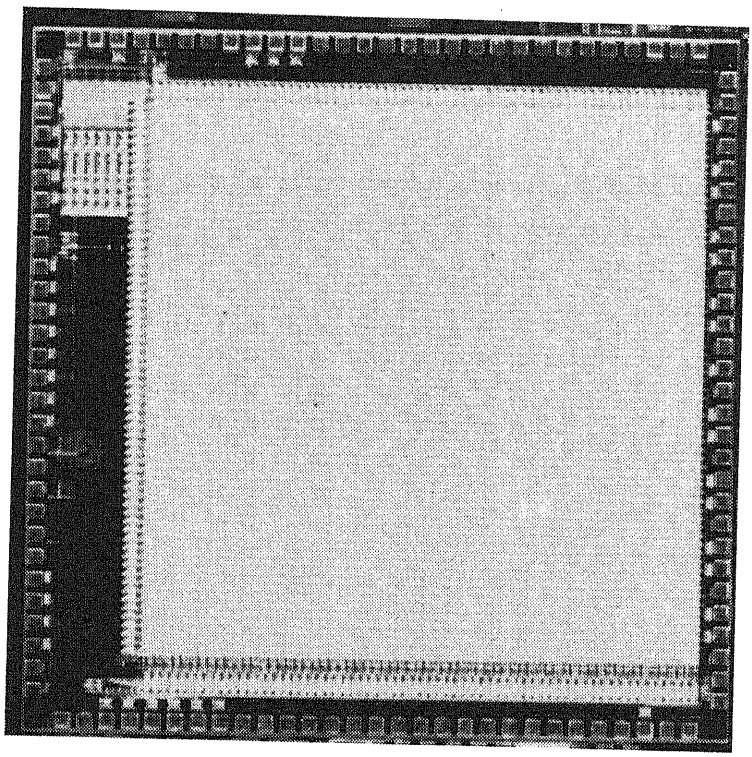


Figure 6.19: Chip photograph of the designed 64×64 1:4 tree sensor using external address decoders, with taking inter-frame difference.

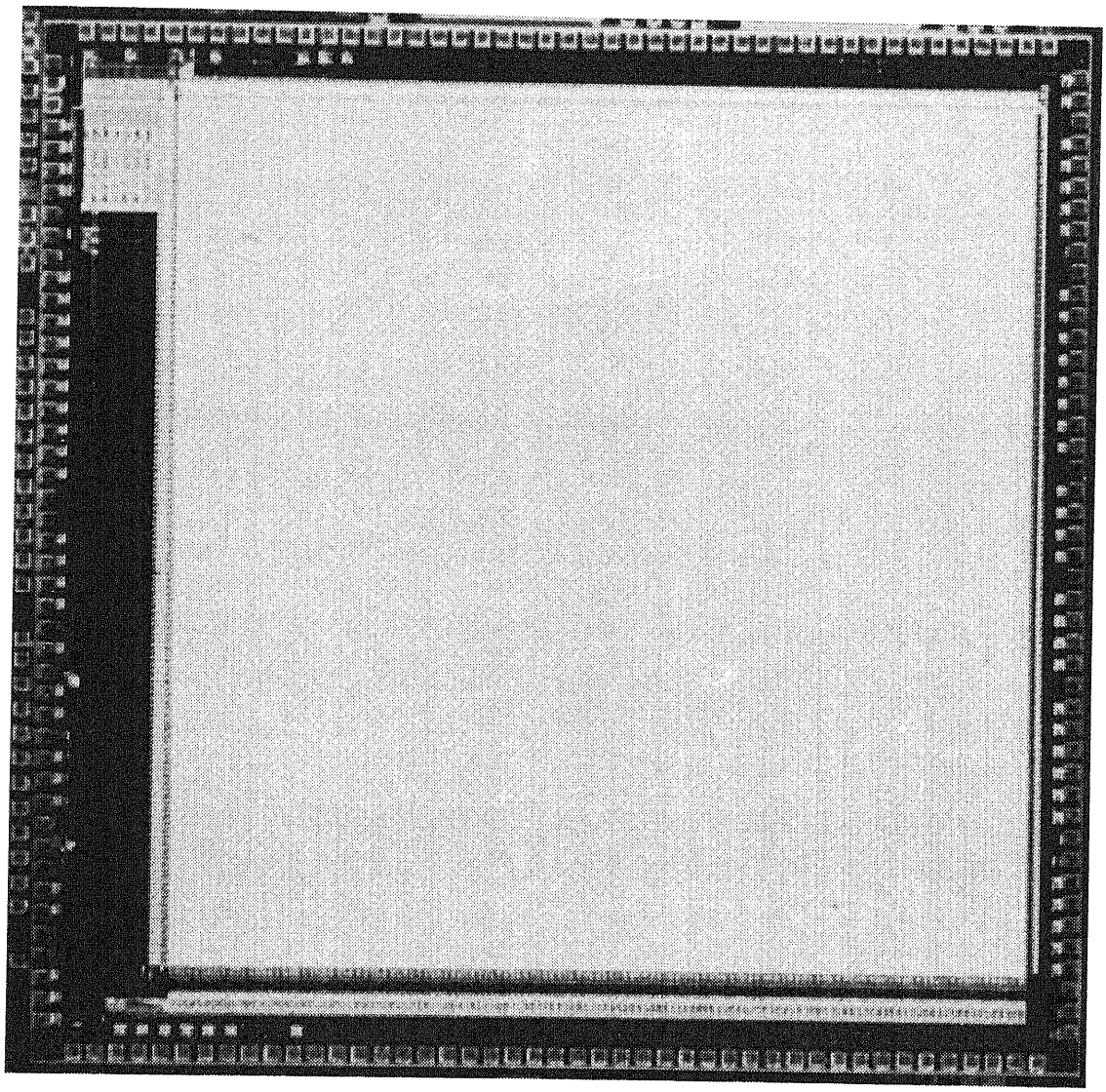


Figure 6.20: Chip photograph of the designed 128×128 1:4 tree sensor using external address decoders, without taking inter-frame difference.

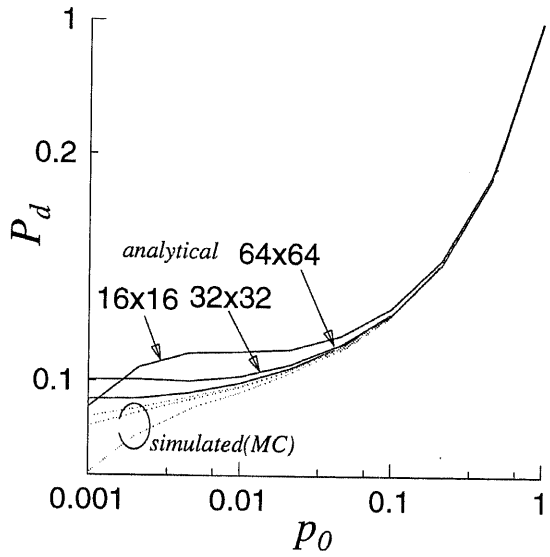


Figure 6.21: Relation of the selection probability of the bit lines or the row select lines, P_s, and the probability of the pixels being "1", p_0

- power consumption if the scan sequence controller, U_{FSM}.

The probability of each bit line B_i or row select line R_j being selected, P_s is derived as dividing the frequency of selection through all the scan steps by the number of scan steps as follows,

$$P_s = \frac{\sum_{l=1}^{N} 2^{N+1-l} p_l}{1 + \sum_{l=1}^{N-1} 4^{N+1-l} p_l}, \tag{6.7}$$

where p_l is the probability of the node at the level l being "1" derived in Equation (3.2) as $p_l = 1 - (1 - p_0)^{4^l}$, where p_0 is the probability of the pixel being "1". The relation of P_s and p_0 is shown in Figure 6.21, with the results of the Monte-Carlo switch level simulation, for the various size of pixel plain, 16×16, 32×32, and 64×64. The relative frequency of the bit lines or row select lines being selected is lower for the larger p_0, since the number of scan steps increases rapidly.

It is also derived the probability of the bit in 1:4 tree code being "1", P_d as dividing the total number of the nodes whose value is "1" by the number of scan steps as follows.

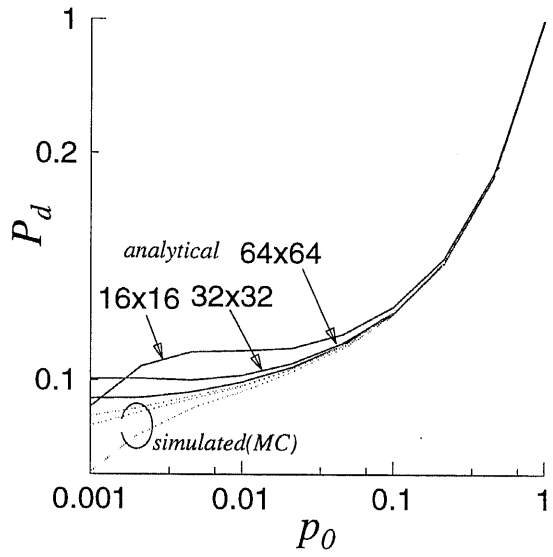


Figure 6.22: Relation of the probability of the bit in 1:4 tree code being "1" and the probability of the pixels being "1", p_0

$$P_d = \frac{\sum_{l=0}^{N} b^{N-l} p_l}{1 + \sum_{l=1}^{N-1} b^{N+1-l} p_l} \tag{6.8}$$

The relation of P_d and p_0 is shown in Figure 6.22, with the results of the Monte-Carlo switch level simulation, for the various size of pixel plain, 16×16, 32×32, and 64×64. The probability of the bit in 1:4 tree code being "1" is smaller for the smaller p_0.

Using these probabilities, the probabilities of charging or discharging signal lines are derived as follows.

- The bit line B_i is discharged when it is selected and at least one pixel at the selected rows is "1", and the probability is expressed as $P_s\{1-(1-p_0)^{nP_s}\}$, since nP_s is the expectation of selected rows.

- The bit line B_i is charged when it is discharged at the previous scan step, and it is also selected at the current scan step, and the probability is expresses as $P_s \cdot P_s\{1-(1-p_0)^{nP_s}\}$.

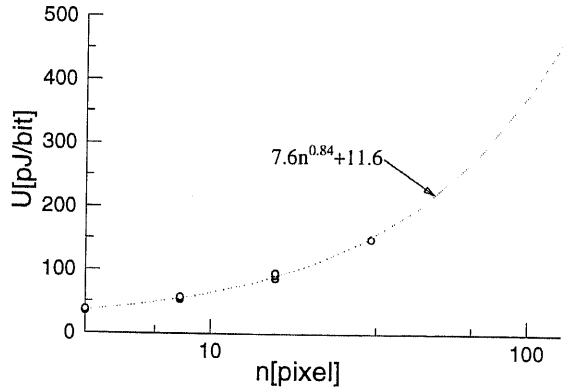


Figure 6.23: Relation of power consumption per scan step and the number of pixels at one edge, for various images

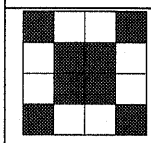
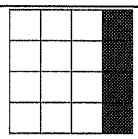
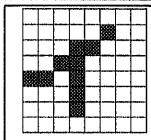
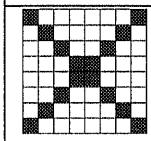
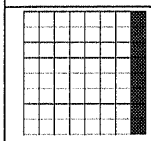
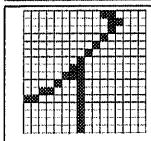
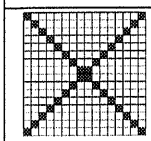
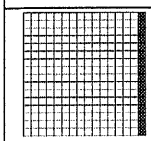
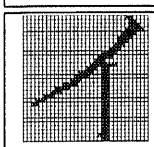
- The row select line R_j is charged and discharged by the probability of selection, P_s at the every scan step.
- The word line W is discharged when the bit of 1:4 tree code is "1", and the probability is expressed as P_d.
- The word line W is charged at the pre-charge cycle if it is discharged at the previous scan step, and the probability is expressed as P_d.

The load capacitance of the bit lines and the row select lines and the word line are proportional to their lengthes, and they are proportional to the number of pixels at one edge, n. Thus the expectation of power consumption at each scan step, P is expressed as the sum of each power consumption as follows.

$$P = n \times \frac{P_s(1+P_s)\{1-(1-p_0)nP_s\}}{2} nC_{B0}V_{dd}^2$$
$$nP_s \times nC_{R0}V_{dd}^2 + P_d \times nC_{W0}V_{dd}^2 + U_{\mathrm{FSM}} \qquad (6.9)$$

The power consumption for some sample images is calculated by `spice` simulation, since the active probability of pixels have spatial correlation in these cases, which can not be derived by the Equation (6.9), that is derived by assuming no correlation among pixels. The load capacitance of bit lines, row select lines, and word line is assumed to be proportional to the number of pixels at one edge using the standard

Table 6.2: Power consumption of 1:4 tree structure using external address decoders per scan step

Image	# of pixels	Code length	U[pJ/bit]
	4×4	21bit	38.5
	4×4	21bit	40.5
	4×4	13bit	43.0
	8×8	41bit	55.1
	8×8	53bit	57.5
	8×8	29bit	62.6
	16×16	93bit	89.5
	16×16	117bit	95.5
	16×16	61bit	101.1
	32×32	317bit	150.3

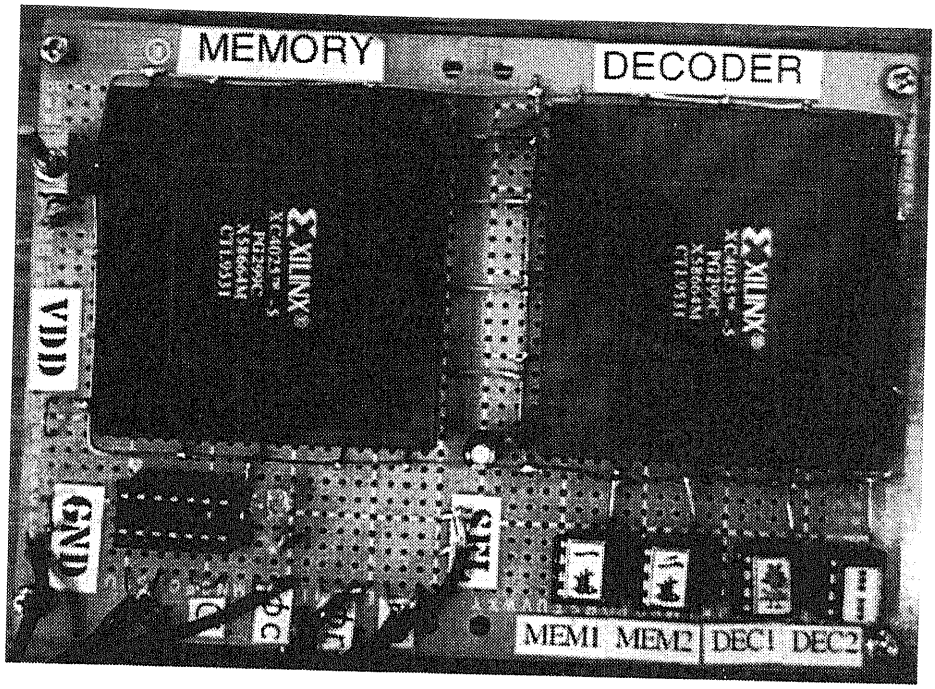


Figure 6.24: Photograph of the developed prototype system for 1:4 tree code decoder.

1.5μm CMOS process. The simulation results are shown in Table 6.2. The relation of power consumption per scan step and the number of pixels at one edge is shown in Figure 6.23, with the curve-fitting results using least mean square (LMS) method. It is expected to be proportional to $O(n^{0.84})$, which is expected to be small enough than that of CCD image sensors for the larger n.

Because of the design mistake in pad assignment suitable for LSI tester, the measurement of the fabricated chip has not been carried out yet.

6.5 Design and Evaluation of Tree-Code Decoder

6.5.1 Prototype implementation of tree-code decoder

We have developed the prototype system of the 1:4 tree code decoder using FPGAs(Field Programmable Gate Array). We have employed the FPGA of XC4025, which is used in section 6.2.1, and one for memory cell arrays, and the other for address decoders and controllers. The number of memory cells is $32 \times 32 = 1,024$, and the decoded image is serially read out.

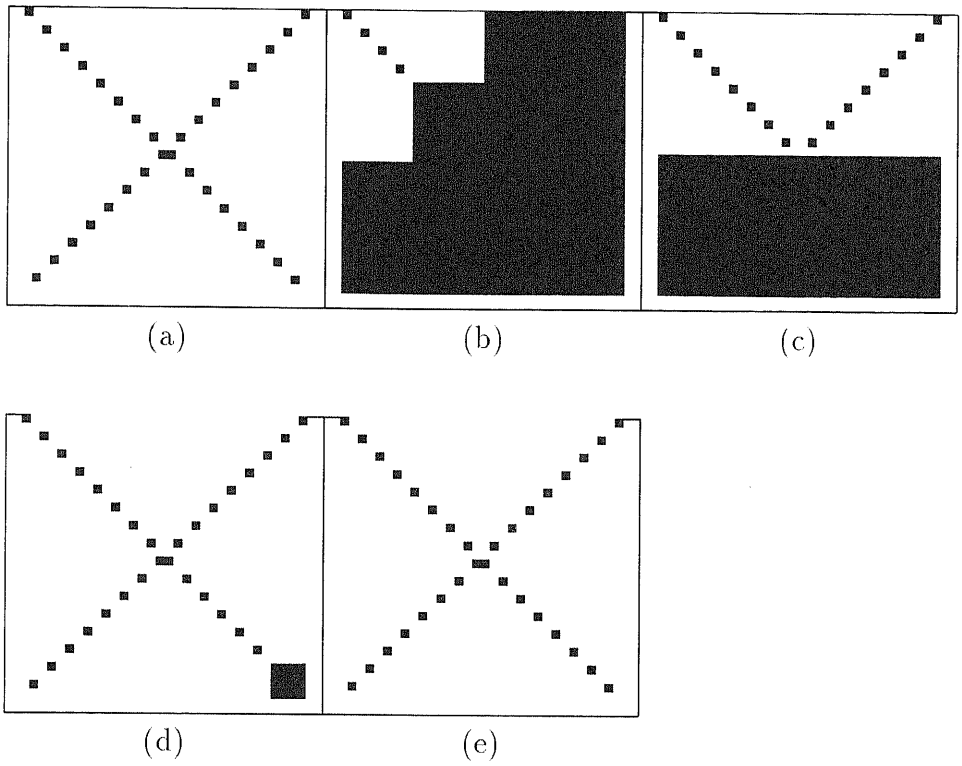


Figure 6.25: Original image(a) and the images under decode(b), (c), (d), and the completely decoded image(e).

The photograph of the prototype 1:4 tree code decoder is shown in Figure 6.24.

The 1:4 tree code to be decoded is given by PC, with the system clocks, and it is decoded to binary image by the developed prototype system. The binary image is again transferred to PC serially. The image under decode procedure can be also read out serially, since each memory cell is set its value according to the decoding step, which is more detail at the decode step for lower levels. Figure 6.25 shows the original image and the images under decode, and the completely decoded image.

The operation frequency of the developed prototype system is up to 1MHz, which is restricted by the speed of PC.

6.5.2 Design of full-custom tree-code decoder

The decoder of 1:4 tree code has been designed, by using the circuits of the controller and the address decoders, which are identical to those used in the 1:4

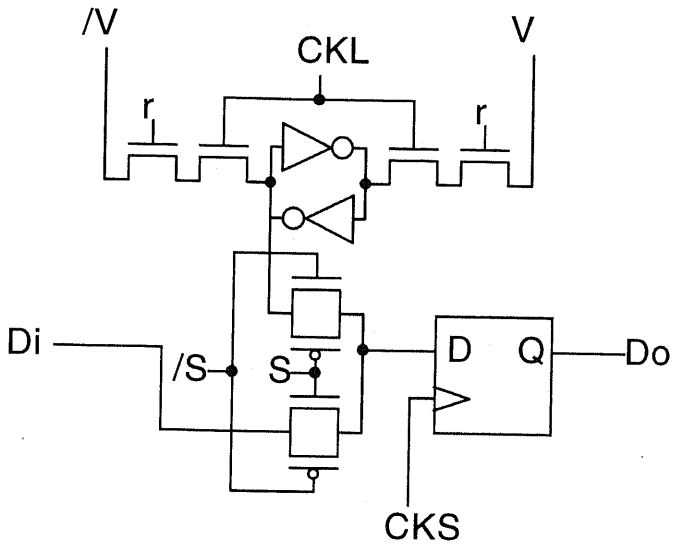


Figure 6.26: Circuit of memory cell for the 1:4 tree code decoder

tree sensor using external address decoder, used in section 6.4. The circuit of the memory cell is shown in Figure 6.26. The memory cell has coupled inverters to store the value of raster image, and one D-flip-flop for the shift register in order to shift out outside the decoded image. The decoded image is given by V and its complementary as /V, and they are stored to the coupled inverters using clock CKL, if the row of this pixel is selected by r. The selection of columns is done by the signal generation circuit placed at each column, as shown in Figure 5.14. The stored image is transferred to D-flip-flop at the positive edge of CKS by keeping S as low and /S as high. The transferred image is shifted out outside by the clock of CKS with keeping S as high and /S as low. Data input Di is connected to the data output of the previous stage in shift register, and data output Do is connected to the data input of the next stage in shift register, respectively. Data output Do of the final state in shift register is the serial output of the decoded image.

The layout of the memory cell used in this decoder is shown in Figure 6.27, which is implemented by the circuit described in section 5.3.

The whole layout of the designed 64 × 64 decoder of 1:4 tree code is shown in Figure 6.28, which has the functions of read out the rasterized image serially, and can be cascaded for the decoding of the larger raster images, as shown in Figure

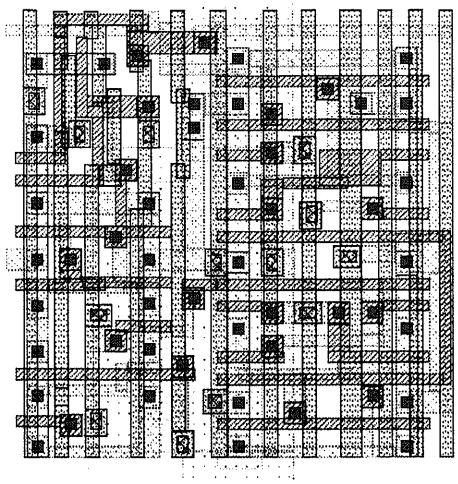


Figure 6.27: Layout of the memory cell for the decoder of 1:4 tree code.

6.29. In this case, one decoder chip is used as master, and the address select signals are send to the other slave chips. The other tree decoder chips are used as slave, where only the address decoders are used to select memory cells. One level of the controller in one of the slave decode chips is also used to generate the address select signals for the extended number of levels. The serial inputs and serial outputs of the decoded images in each chip is connected in order, so as to form a large shift register.

The chip photograph of the designed decoder of 1:4 tree code is shown in Figure 6.30[5]. The number of transistors is 110,024.

There was a fatal design mistake in the fabricated chip, one connection in serial shift registers is lost, and the measure and evaluation of this chip has not been done yet.

6.6 Summary and Conclusion

In this chapter, the designed chips of the 1:4 tree sensors and the decoder of 1:4 tree code and their prototype systems using FPGAs, are described. It is shown

[5]The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo with the collaboration by Nippon Motorola and Dai Nippon Printing Corporation.

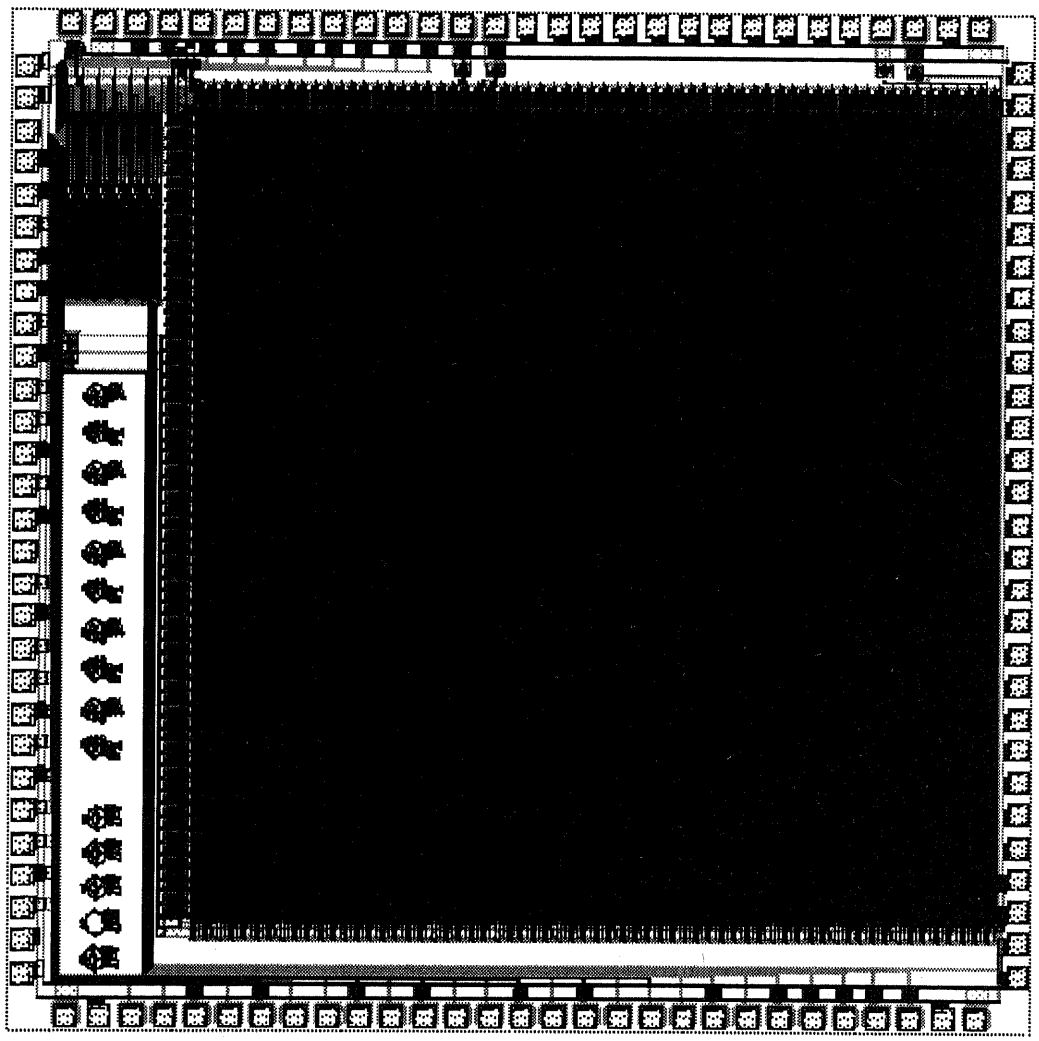


Figure 6.28: Whole layout of the designed 64 × 64 decoder of 1:4 tree code.

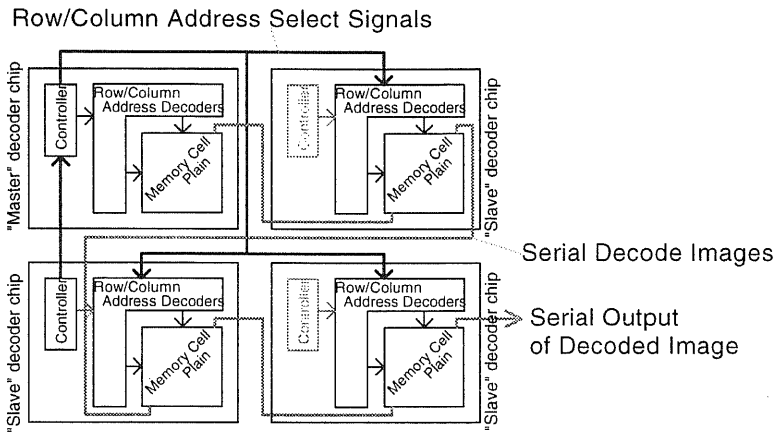


Figure 6.29: Extend connection of designed decoder chips for the larger image

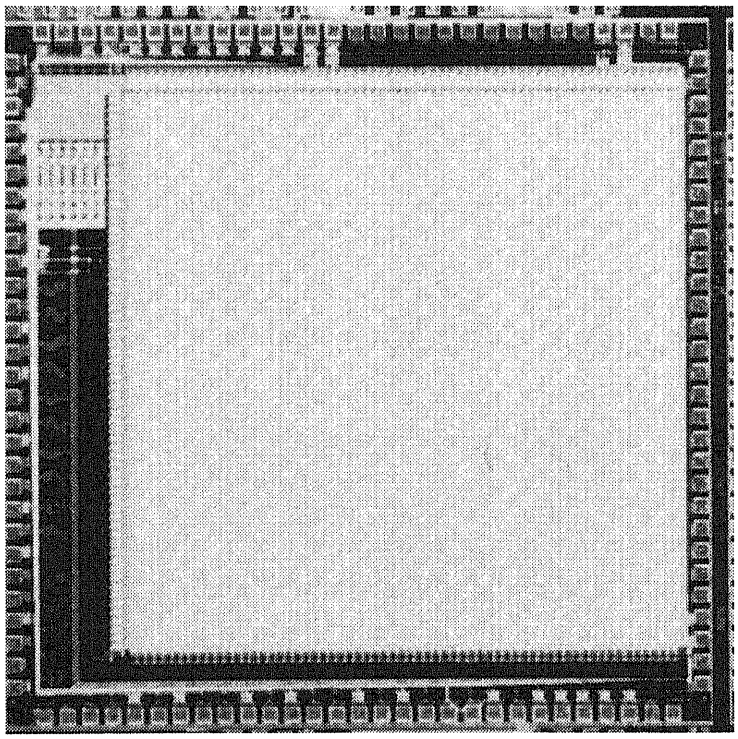


Figure 6.30: Chip photograph of the designed 64×64 decoder of 1:4 tree code.

that the 1:4 tree sensor using the external address decoder can integrate about 16 times pixels in the same size of chip, compared with that using tree structure of automata. It is also shown that the decoder of 1:4 tree code can be implemented by using the external address decoder designed for the 1:4 tree sensor with external address decoders.

The evaluations of them, mainly in terms of the power estimation are also described.

Chapter 7

Conclusions

The followings are the conclusions through this thesis.

Chapter 2

The power consumption in CCD image sensor is estimated as proportional to the cubic of the number of pixels in line, which will be one of the most serious problems in mega-pixel CCDs.

The probabilistic model of power consumption for both combinational logics and sequential circuits are described.

It is also described the power reduction methodologies for both combinational logics and sequential circuits without modification of their function.

The power consumption of combinational logics, 1 bit full adder as an example, is expected to be reduced up to -30% by optimum input assignment for logically symmetric input terminals.

The power consumption of sequential circuits is expected to be reduced up to -10% by the binary state code assignment minimizing the mean number of transition at each clock cycle, against the conventional state code assignment minimizing the complexity of combinational logics. In this case, the number of transistors in combinational logics increases about 10% by minizing power consumption against minizing the complexity of combinational logics, while the total number of transis-

tors increase less than 10%, since the number of flip-flops is the same.

Chapter 3

A method of using tree structure for image signals is proposed, which is effective especially for the image containing many 0s by skipping redundant scan steps for the large area of 0s.

It is also derived that the optimal number of branches in the tree structure is 4, which we call "1:4 tree structure". The expectation of code length using 1:4 tree structure is shorter than that of the raster scan in case of less than quarter of the pixels are "1" in random white noise image, but in the practical image containing about half of "1" pixels can be scanned by 1:4 tree scan with the shorter code length than that of the raster scan.

It is also indicated that the code length by the scan using 1:4 tree structure is shorter than that of run-length compression, especially in case of containing many white pixels.

The methodology for decoding the 1:4 tree code is also described, where the partially decoded images represent the rough images corresponding to the current level of 1:4 tree structure.

Chapter 4

Some concepts and implementations are described for the applications of 1:4 tree sensor for various adaptivities of images as our eyesight has, and the algorithms used in movie compression, and visual tracking system.

The spatial adaptivity for scanning images is implemented by 1:4 tree scan with stopping the scan steps to the lower level, since the sub-areas corresponding to the nodes in the lower level represent the part of image with the higher spatial resolution. The selection whether the rough scan or the detail scan for each sub-area should be executed, is determined according to the interests for it. One of the criteria determining the spatial resolution for each area is the partial uniformity of image,

and the detail scan is executed for the sub-area with low uniformity, while the rough scan is executed for the sub-area of high uniformity. This scan method indicates one possibility of compressing images with maintaining the quality of image.

It is also described the intensional adaptivity for image signals using 1:4 tree structure, since the charge time of junction capacitance by photo current is concerned with the light intensity. The spatial distribution of the pixel whose output voltage reaches to the threshold voltage at each charge time, gathers spatially, which is the case that 1:4 tree scan can be executed efficiently.

The applications of 1:4 tree scan for two important algorithms used in movie compression, inter-frame difference and motion compensation, are described.

The 1:4 tree scan is used for the scan of flag indicating the pixels has a effective difference in successive two frames, which is the case that 1:4 tree structure can scan efficiently, and the analog value of difference can be read out by the other signal path.

The motion compensation within one pixels, which is expected to be suitable assuming high frame rate, can be implemented by adding the interconnections among neighbor pixels and the selector of them to each pixel. It is also notable that the motion compensation using this algorithm can be executed for any size of sub-areas, since the node automata has the mean value of its lower sub-areas.

It is also described the visual tracking algorithm using 1:4 tree structure, which can scan just for the areas containing moving objects with masking the other areas.

Chapter 5

The implementations of 1:4 tree structure in CMOS circuit are described. In the designed node automata, the clock signals are provided just along the scan path, which is expected to be suitable for low power operation. The two dimensional layout of node automata and pixels that can be extended for any number of levels are described, by placing the magnified cross-shaped node automata for the higher level, which is also suitable for minizing the signal delay.

The alternative implementation of 1:4 tree structure is also described, by plaicing

address decoders indicating the pixels to be scanned outside the pixel plain. The 1:4 tree sensor using this architecture is expected to achieve the higher fill factor and higher integration of pixels in image sensor.

The architecture of decoder for 1:4 tree code is also described, which is implementing the identical address decoders and controllers used in the 1:4 tree sensor using external address decoders.

The circuits of pixels are also described, which implement taking inter-frame difference, and gray scale scan considering charge-up time by photo current.

It is also described the concept of light-powered 1:4 tree sensor for ultra low power operation.

Chapter 6

The design and evaluation of 1:4 tree sensors using node automata are described, for both prototype system using FPGAs and full custom chip using 1.5μm CMOS technology is discussed. The designed chip contains 32×32 pixels in 7.2mm$\times$7.2mm chip, and its power consumption at each scan step is expect to be proportional to the edge size, which is smaller enough than CCD image sensors in mega-pixel sensors.

The photo current of photo diodes is also measured, and the photo current of the photo diode with 36μm $\times$ 100μm is about 1μA, and it is proportional to both light intensity and area of photo diode.

The design and evaluation of 1:4 tree sensors using external address decoders are described. The designed 1:4 tree sensor using external address decoders can integrate about 16 times of pixels against using node automata. The power consumption of 1:4 tree sensors using external address decoders is estimated using `spice`, which is proportional to $O(n^{0.84})$, where n is the edge size of pixel plain, that is expected be small enough for mega-pixel image sensors.

The designs of 1:4 tree code decoders are also described, both in prototype systems using FPGAs and full-custom chip using 1.5μm CMOS technology are described, and the operation of the prototype systems is checked.

List of Presentations and Publications

Chapter 2

- J.Akita and K.Asada, "A Method of Reducing Power Consumption of CMOS Logic Based on Probability Model of Signal Transition," *Technical Report of IEICE*, ED93-83, Sep. 1993.
- J.Akita and K.Asada, "A Method for Reducing Power Consumption of CMOS Logic Based on Signal Transition Probability," *Proc. of EDAC-ETC(Europian Design Automation Conference & Europian Test Conference) Euro ASIC*, Mar. 1994.
- J.Akita and K.Asada, "A Method for Power Reduction of Finite State Circuit with Optimum State-code Assignment," *Technical Report of IEICE*, ICD94-104, Sep. 1994.
- H.Hayashi, J.Akita, and K.Asada, "A Method of Optimal State Code Assignment for Reducing Power Consumption in Synchronous Circuits," *Proc. of 1994 IEICE Fall Conference*, A-67, Sep. 1994.
- J.Akita, H.Hayashi, and K.Asada "An Estimation and Reduction of Power Consumption in Clock Line of Synchronous Flip-Flops," *Proc. of 1994 IEICE Fall Conference*, A-68, Sep. 1994.
- K.Asada and J.Akita, "Optimum State Assignment for CMOS Implementation

of Low Power Finite State Machine," *Proc. of IFIP Workshop on Logic and Arch. Synthesis*, Dec. 1994.

- J.Akita and K.Asada, "An Estimation of State Code Assignment for Low Power Finite State Circuit," *Proc. of 1995 IEICE Spring Conference*, A-108, Mar. 1995.

- K.Asada and J.Akita, "A Method for Reducing Power Consumption of CMOS Logic Based on Signal Transition Probability," *IEICE Trans. on Electronics*, Vol.E78-C, No.4, pp.436–440, Apr. 1995.

Chapter 3

- J.Akita and K.Asada, "A Signal Scanning Method of Sensors With Hierarchal Structure of Node Automata," *Technical Report of IEICE*, VLD95-62, Sep. 1995.

- J.Akita and K.Asada, "An Image Scanning Method with Data Compression using Tree Structure of Automata," *Proc. of 1996 IEICE Spring Conference*, A-43, Mar. 1996.

- J.Akita, R.Watabe, and K.Asada, "A Novel Tree Structure of Automata for Selective Scanning of Image Signals," *Proc. of the ITEC'96*, p.33, Jul. 1996.

- K.Asada and J.Akita, "Image Sensor using Tree Structure," *Symposium on Scientific Research on Priority Areas, "Ultimate Integration of Intelligence on Silicon Electronic Systems"*, Mar. 1997.

- J.Akita and K.Asada, "Image Data Compression Efficiency using Tree Scanning and Run-length Coding," *Proc. of 1997 IEICE Spring Conference*, A-6-10, Mar. 1997.

- K.Asada, J.Akita, and R.Watabe, "A Tree Structure of Automata for Selective Image Scanning and Its Implementation," *Comp. & Elec. Eng.* (to be published).

Chapter 4

- K.Asada, J.Akita, M.Nawamin, and R.Watabe, "Intelligent Lower Power Devices and Circuits," *Symposium on Scientific Research on Priority Areas, "Ultimate Integration of Intelligence on Silicon Electronic Systems"*, Mar. 1996.
- K.Asada and J.Akita, "A Selective Image Scanning Method using Tree Structure of Automata and Its Implementation," *Proc. of 1996 IEICE Fall Conference*, ES-3-7, Sep. 1996.
- K.Asada, J.Akita, and R.Watabe, "A Tree Structure of Automata for Selective Image Scanning and Its Implementation," *Proc. of 4th Int. Conf. on Soft Computing (IIZUKA '96)*, pp.113–116, Oct. 1996.

Chapter 5 and 6

- R.Watabe, J.Akita, and K.Asada. "An Implementation on CMOS Circuit of Tree Structure of Automata for Image Scanning," *Proc. of the ITEC'96*, p.35, Jul. 1996.
- J.Akita and K.Asada, "An Implementation of Image Scanning Method with Selective Activation of Tree Structure," *Proc. of 1996 IEICE Fall Conference*, A-54, Sep. 1996.
- J.Akita and K.Asada, "An Image Sensor using Quad Tree for Selective Scanning with Adaptive Resolution," *Proc. of IEEE CCD & AIS Workshop*, p.5-1, 1997.6.
- J.Akita and K.Asada, "An Image Scanning Method with Selective Activation of Tree Structure," *IEICE Trans. on Electronics*, Vol.E80-C, No.7, pp.956–961, 1997.7.
- J.Akita and K.Asada, "A CMOS Image Sensor with Variable Block Access Function," *Proc. of 1997 IEICE Fall Conference*, C-12-39, Sep. 1997.

Acknowledgement

It is a great pleasure to express my gratitude to the dissertation supervisor, Professor Kunihiro Asada, for his heartful advices and hints through the term of this work, and his encouragements for the comfortable circumstances of this work. His help and encouragement in both professional and private manners greatly has influenced me from various points of view.

I would like to give my thanks to Professor Kiyoharu Aizawa of University of Tokyo, and Dr. Takayuki Hamamoto, who is currently a research assistant of Science University of Tokyo, for giving many helpful discussions and hints, in the project of scientific research on priority areas, mainly from the viewpoints of the image processing researchers and image sensor designers.

I am also thankful to all of my past and present colleagues in Asada Laboratory who gave me helpful advices, heartful encouragement, comfortable research circumstances and pleasant time, especially Dr. Makoto Ikeda, who is currently research assistant of VLSI Design and Education Center, for his efforts of comfortable computer circumstances, and Dr. Rimon Ikeno, who is currently in Texas Instruments Tsukuba Research & Development Center Ltd., for his heartful encouragements and useful hints on researches.

I also thank Mr. Hiroyuki Hayashi for his cooperations in the research for the power reduction methodologies of the sequential circuits.

I would like to express my thanks to Mr. Mukdathong Nawamin, Mr. Ryota Watabe, who is currently in Matsushita Communication Industrial Co., Ltd., and Mr. Kenji Hirota, who is currently in KDD Co. Ltd., and Mr. Masashi Hoshino, for their cooperations in the research for development of the prototype systems of

1:4 tree sensor and 1:4 tree code decoder.

I also would like to thank to Mr. Tomohiro Nezuka, and Mr. Kagehiro Mukai, for their cooperations in the research for the intelligent pixel circuits.

I esteem Mr. Shinichi Suzuki, Ms. Noriko Yokochi, and Ms. Makiko Okazaki for their help for my activities in the laboratory.

I would like to thank to Professor Yoshihiro Mizoguchi, who is associate professor of Kyushu Institute of Technology, for his advice on the previous studies on quad tree in image processing from the viewpoint of mathematical science. I also thank to Mr. Hideyuki Suzuki, who is the doctor course student in Department of Mathematical Engineering and Information Physics, University of Tokyo, for his advice on the mathematical methodologies for the two dimensional random images with the desired power spectrum characteristics. I would like to give my thanks and love to all of my friends, especially in Yugen Club offered by Professor Heisuke Hironaka, who is currently the president of Yamaguchi University, through his eager activities for mathematical science education. I would like to express my thanks and love to Naoko Kamiryo in the medical course of Hiroshima University, for her heartful encouragement.

Finally, I would like to express my thanks and love to my father and my mother in my home town, Nagoya, who have been continuously supporting me throughout my life.

Junichi Akita

Bibliography

[1] A.Moini, "Vision Chips or Seeing Silicon,"
`http://www.eleceng.adelaide.edu.au/Groups/GAAS/Bugeye/visionchips/index.html`

[2] M.Yasuda *et al.*, "An Electronic Model of Visual Receptive Fields," Trans. of IEICE(Japanese) Vol.54-C, No.6, pp.514–521, 1971.

[3] C.Mead, "Analog VLSI and Neural Systems," Addison-Wesley Reading, 1989.

[4] A.Moini *et al.*, "An Analog Implementation of Early Vision Processing in Insects," *Proc. of 1993 VLSITSA*, pp.283–287, 1993.

[5] H.Kobayashi *et al.*, "An Active Resistor Network for Gaussian Filtering of Images," *IEEEE Journal of Solid-State Circuits*, Vol.26, No.5, pp.738–748, 1991.

[6] H.Kobayashi *et al.*, "An Analog CMOS Network for Gaussian Convolution with Embedded Image Sensing," *ISSCC Dig. of Tech. Papers*, pp.216–217, 1990.

[7] C.Koch, "Implementing early vision algorithms in analog hardware," *Proc. SPIE*, Vol.1473, pp.2–16, 1991.

[8] A.Gruss *et al.*, "A VLSI Smart Sensor for Fast Range Finding," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, pp.349–358, 1992.

[9] S.E.Kemeny *et al.*, "CCD Focal-Plane Image Recognization Prosessors for Lossless Image Compression," *IEEE Journal of Solid-State Circuits*, Vol.27, No.3, pp.398–405, 1992.

[10] T.Hamamoto *et al.*, "Focal Plane Compression and Enhancement Sensors," *ISCAS'97*, Vol.3, pp.1912–1915, 1997.

[11] T.Nishimura *et al.*, "Three Dimensional IC For High Performance Image Signal Processor," *Proc. IEDM*, pp.111-114, 1987.

[12] A.Yakovleff *et al.*, "A micro-sensor based on insect vision," *Workshop on Conputer Architecture for Machine Perception*, pp.137–146, 1993.

[13] M.Koyanagi *et al.*, "Three-dimensional laminated image processing systems," *Symposium on Scientific Research on Priority Areas, "Ultimate Integration of Intelligence on Silicon Electronic Systems"*, Mar. 1997.

[14] R.H.Dennard *et al.*, "Design for ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, Vol.9, No.5, pp.256–268, 1974.

[15] "Panasonic Data Book of CCD '96," Matsushita Electronics Corp., 1996.

[16] T. Quarles *et al.*, "SPICE 3B1 User's Guide," Jan. 1989.

[17] S.Devadas *et al.*, "MUSTANG: State Assignment for Finite State Machines Targeting Multilevel Logic Implementation," *IEEE Trans. on Computer-Aided Design*, Vol.7, No.12, 1988.

[18] H.Samet, "Region representation: quad trees from binary arrays," *Computer Graphics and Image Processing*, Vol.13, pp.88–93, 1980.

[19] J.van der Spiegel *et al.*, "A foveated retina-like sensor using CCD technology," in *Analog VLSI implementation of neural systems*, Kluwer Academic Publishers, 1989.

[20] T.Yagi *et al.*, "The role of retinal bipolar cells in early vision: an implication with analog networks and regularization theory," *Biological Cybernetics*, Vol.77, pp.163–171, 1997.

[21] T.M.Bennard *et al.*, "A programmable VLSI retina for rough vision," *Machi Vision and Applications*, Vol.7, pp.4-11, 1993.

[22] C.Mead, "A Sensitive Electronic Photoreceptor," *Chappel Hill Conference on VLSI*, pp.463–471, 1985.

[23] V.Ward *et al.*, "VLSI Implementation of Receptive Fields with Current-Mode Signal Processing for Smart Vision Sensors," *Analog Integrated Circuits and Signal Processing 7*, pp.167–179, 1995.

[24] V.Ward *et al.*, "CMOS photodetector with built-in light adaptation mechanism," *Microelectronics Journal*, Vol.24, pp.547–553, 1993.

[25] ISO/IEC 13818-1, 13818-2, 13838-3 International Standard, 1994.

[26] I.Ishii *et al.*, "Target tracking algorithm for 1ms visual feedback system using massively parallel processing," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.2309–2314, 1996.

[27] N.H.E.Weste *et al.*, "Principles of CMOS VLSI design: a systems perspective," Addison-Wesley, 1988.

[28] Y.S.Trisno *et al.*, "Optimization of an Optically Pulsed Photocell Array as a Sensor Power Source," *IEEE Trans. Instrum. Meas.*, Vol.37, No.1, pp.142–144, Mar. 1988.

[29] "The Programmable Logic Data Book," Xilinx Inc., 1994.