# VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera Design

**S. P. Mohanty, N. Ranganathan and R. K. Namballa**

Dept. of Computer Science and Engineering

University of South Florida

smohanty, ranganat, rnamball@csee.usf.edu

# Outline of the Talk

- Introduction
- Related Works
- Watermarking Algorithms
- Proposed Architecture
- Prototype Chip Implementation
- Conclusions

# Digital Watermarking ?

Digital watermarking is defined as a process of embedding data (watermark) into a multimedia object to help to protect the owner's right to that object.

Types of Watermarking

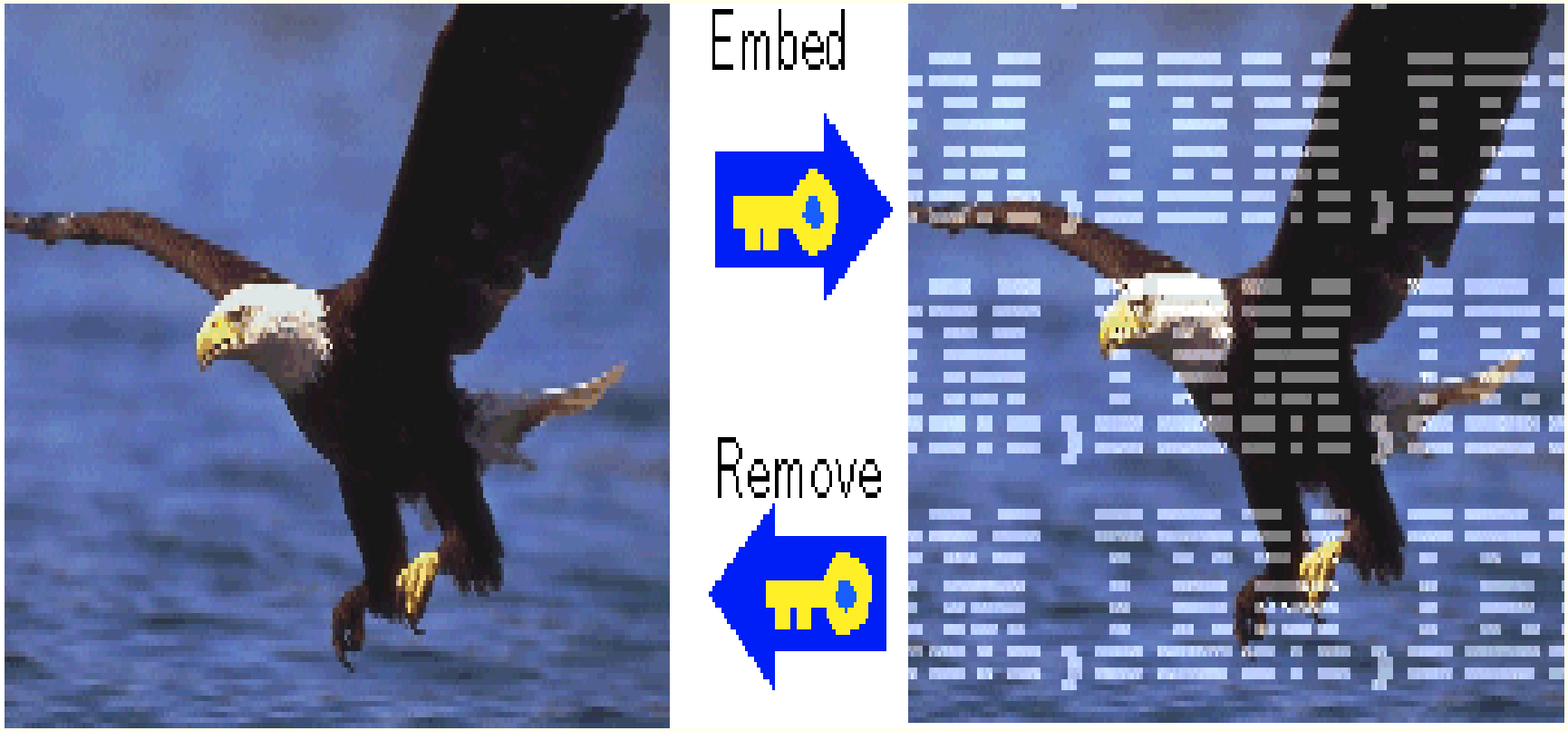
- Visible and Invisible
- Spatial, DCT and Wavelet domain
- Robust and Fragile

# Digital Watermarking ? (Visible Vs Invisible)

- A visible watermark is a secondary translucent overlaid into the primary image and appears visible to a casual viewer on careful inspection.
- The invisible-robust watermark is embedded in such a way that modifications made to the pixel value is perceptually not noticed and it can be recovered only with appropriate decoding mechanism.
- Both visible and invisible watermarking have their applications and hence are equally important.

**NOTE**: We are focusing on visible watermarking.

# Digital Watermarking : Visible Examples




(Source: IBM)

# Watermarking : General Framework

- **Encoder**: Inserts the watermark into the host image
- **Decoder**: Decodes or extracts the watermark from image
- **Comparator**: Verifies if extracted watermark matches with the inserted one

**Encoder implementation is necessary for visible watermarking.**

# Related Works
# (Hardware Systems for Watermarking)

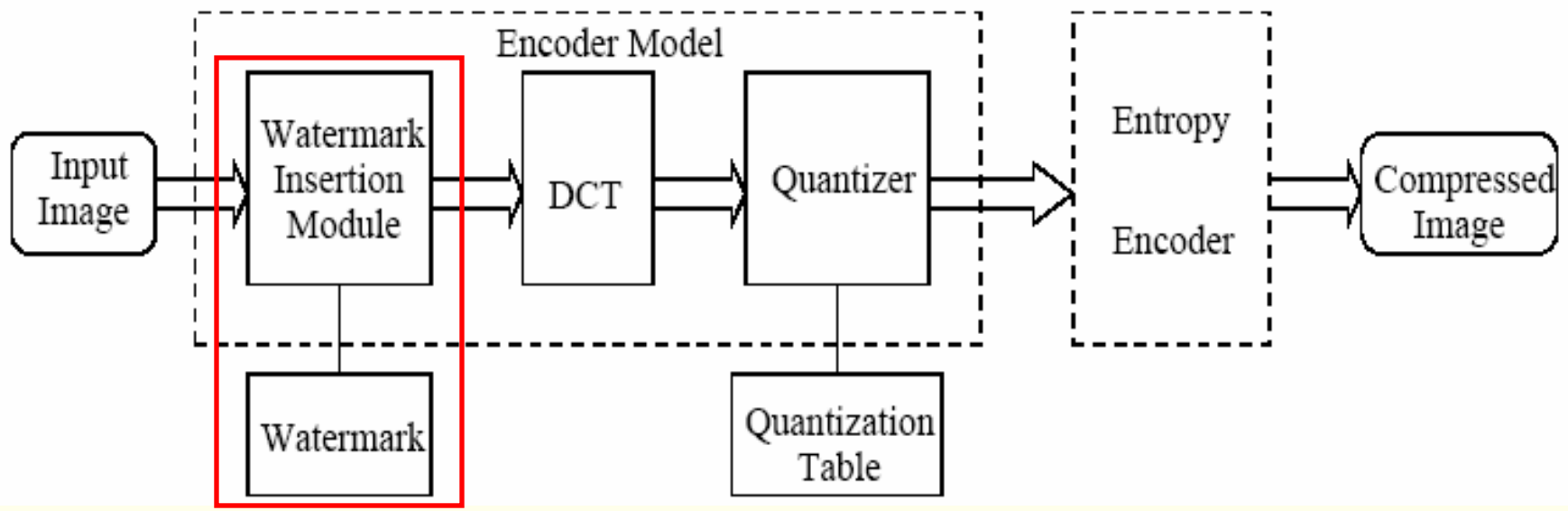
| Work | Type | Target Object | Domain | Technology | Chip Power |
|---|---|---|---|---|---|
| Strycker, 2000 [4] | Invisible Robust | Video | Spatial | NA | NA |
| Tsai and Lu 2001 [6] | Invisible Robust | Video | DCT | 0.35µ | 62.8 mW |
| Mathai, 2003 [5] | Invisible Robust | Image | Wavelet | 0.18µ | NA |
| Garimella, 2003 [7] | Invisible Fragile | Image | Spatial | 0.13µ | 37.6 µW |
| Mohanty 2003 [8] | Robust Fragile | Image | Spatial | 0.35µ | 2.05 mW |

# Why Hardware Implementation ?

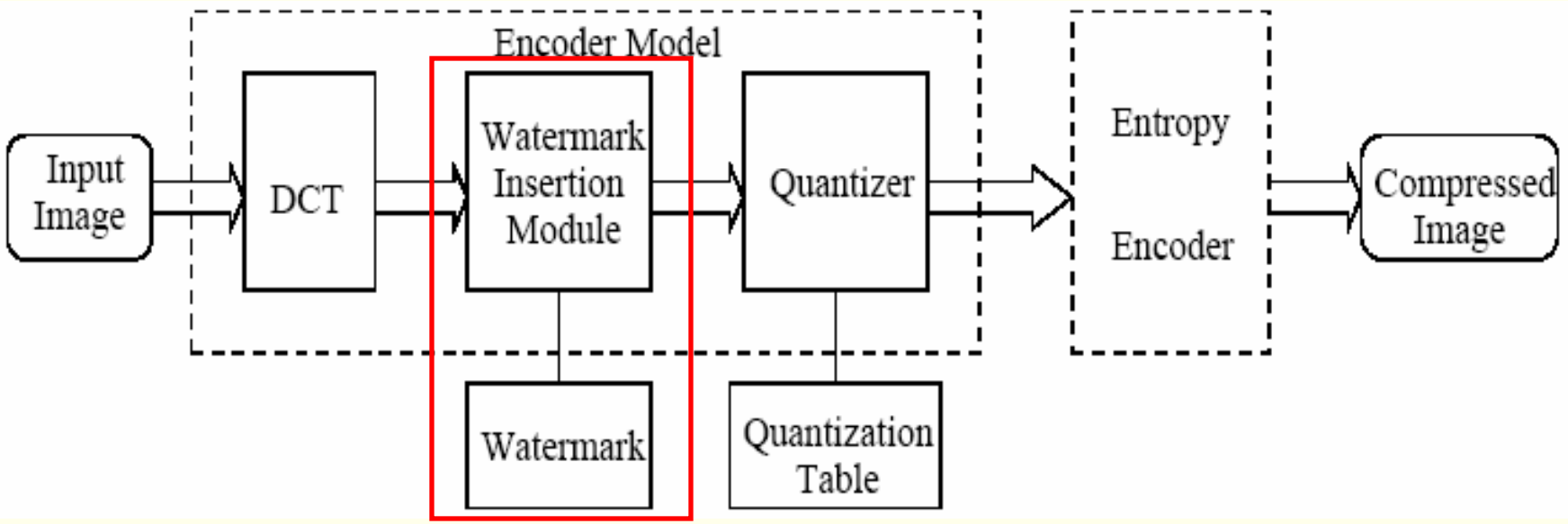
Hardware implementations of watermarking algorithms necessary for various reasons:

- Easy integration with multimedia hardware, such as digital camera, camcorder, etc.
- Low power
- High performance
- Reliable
- Real time applications

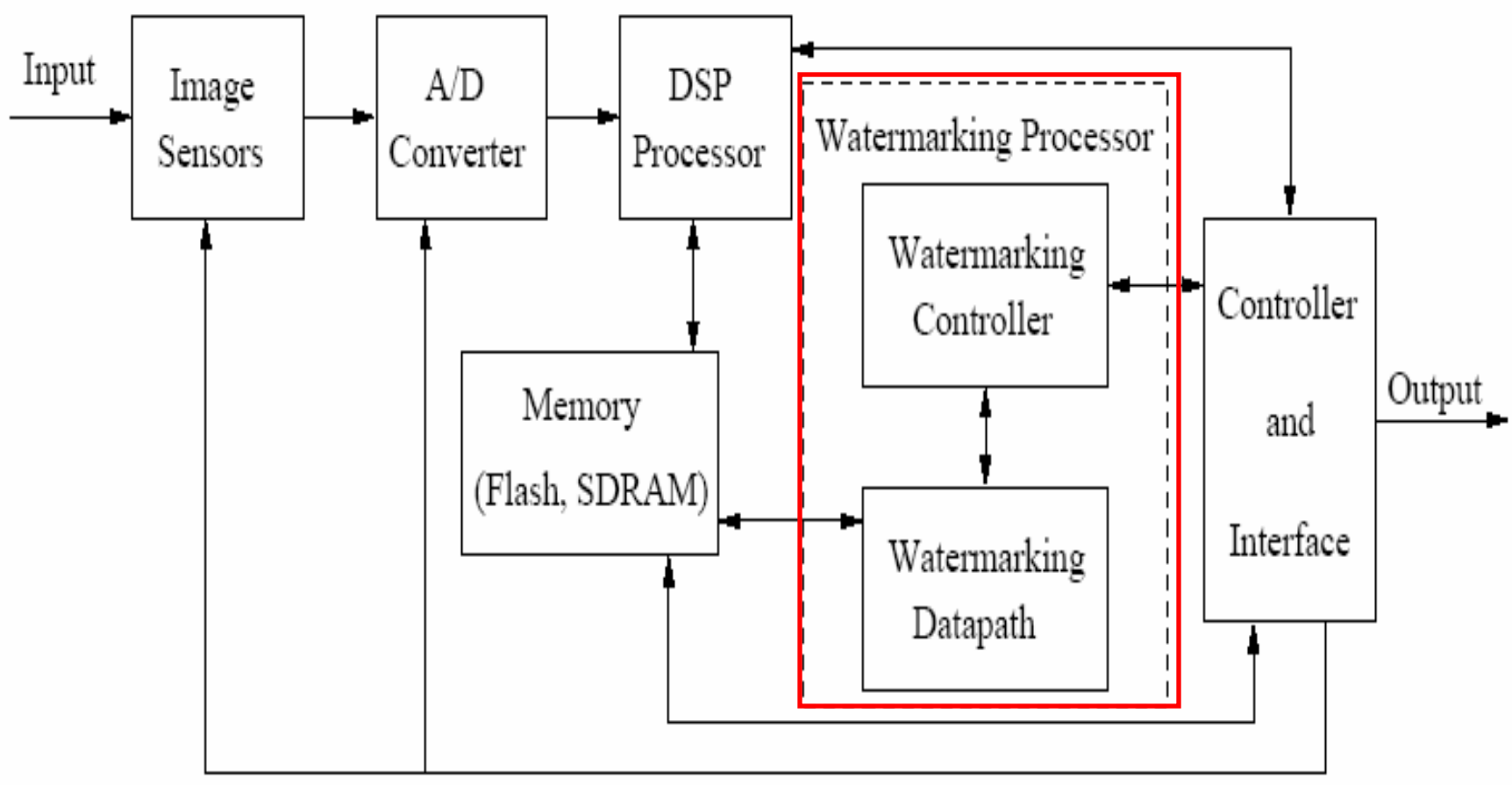
# Secure JPEG Encoder (Spatial Domain)



# Secure JPEG Encoder (DCT domain)



# Secure Digital Still Camera



# Goals of the Visible Watermarking Algorithm

- A visible watermark should be obvious in both color and monochrome images.
- The watermark should be spread in a large or important area of the image.
- The visible watermark should identify the ownership.
- The visual quality of the host image should be preserved.
- The watermark should be difficult to remove from the host image.

**NOTE**: The above conflicting requirements make watermarking a challenging problem.

# Notations used in Algorithm Description

| | |
|---|---|
| $I$ | : Original (or host) grayscale image |
| $W$ | : Watermark image (a grayscale image) |
| $(m, n)$ | : A pixel location |
| $I_W$ | : Watermarked image |
| $N_I \times N_I$ | : Original image dimension |
| $N_W \times N_W$ | : Watermark image dimension |
| $i_k$ | : $k^{th}$ block of the original image $I$ |
| $w_k$ | : $k^{th}$ block of the watermark image $W$ |
| $i_{W\,n}$ | : $k^{th}$ block of the watermarked image $I_W$ |
| $\alpha_k$ | : Scaling factor for $k^{th}$ block |
| $\beta_k$ | : Embedding factor for $k^{th}$ block |
| $\mu_I$ | : Mean gray value of the original image $I$ |
| $\mu_{k\,I}$ | : Mean gray value of image block $i_k$ |
| $\sigma_{k\,I}$ | : Variance of the original image block $i_k$ |
| $\alpha_{max}$ | : The maximum value of $\alpha_k$ |
| $\alpha_{min}$ | : The minimum value of $\alpha_k$ |
| $\beta_{max}$ | : The maximum value of $\beta_k$ |
| $\beta_{min}$ | : The minimum value of $\beta_k$ |
| $I_{white}$ | : Gray value corresponding to white pixel |
| $\alpha_I$ | : A global scaling factor |
| $C_1, C_2$ | : Linear regression co-efficients |
| $C_3, C_4$ | : Linear regression co-efficients |

# Visible Watermarking Algorithm 1 [9]

- The original algorithm proposed in [9]

$$I_W(m,n) = \begin{cases} I(m,n) + W(m,n)\ \left(\frac{I_{white}}{38.667}\right)\ \left(\frac{I(m,n)}{I_{white}}\right)^{\frac{2}{3}} \alpha_I & \text{for } \frac{I(m,n)}{I_{white}} > 0.008856 \\ I(m,n) + W(m,n)\ \left(\frac{I(m,n)}{903.3}\right)\ \alpha_I & \text{for } \frac{I(m,n)}{I_{white}} \leq 0.008856 \end{cases}$$

- Assuming $I_{white} = 256$, simplified to:

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{6.0976}\right)\ W(m,n)\ (I(m,n))^{\frac{2}{3}} & \text{for } I(m,n) > 2.2583 \\ I(m,n) + \left(\frac{\alpha_I}{903.3}\right)\ W(m,n)\ I(m,n) & \text{for } I(m,n) \leq 2.2583 \end{cases}$$

- Fitting piecewise linear model and regression co-efficient:

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{903.3}\right)\ W(m,n)\ I(m,n) & \text{for } I(m,n) \leq 2 \\ I(m,n) + \left(\frac{\alpha_I C_1}{6.0976}\right)\ W(m,n)\ I(m,n) & \text{for } 2 < I(m,n) \leq 64 \\ I(m,n) + \left(\frac{\alpha_I C_2}{6.0976}\right)\ W(m,n)\ I(m,n) & \text{for } 64 < I(m,n) \leq 128 \\ I(m,n) + \left(\frac{\alpha_I C_3}{6.0976}\right)\ W(m,n)\ I(m,n) & \text{for } 128 < I(m,n) \leq 192 \\ I(m,n) + \left(\frac{\alpha_I C_4}{6.0976}\right)\ W(m,n)\ I(m,n) & \text{for } 192 < I(m,n) < 256 \end{cases}$$

# Visible Watermarking Algorithm 2 [3]

- Watermark insertion is carried out block-by-block using: $i_{W\,k} = \alpha_k\, i_k + \beta_k\, w_k \qquad k = 1, 2...$
- The scaling and embedding factors are found out as,

$$\begin{aligned} \alpha_k &= \frac{1}{\hat{\sigma}_{I\,k}}\, exp\left(-(\hat{\mu}_{I\,k} - \hat{\mu}_I)^2\right) \\ \beta_k &= \hat{\sigma}_{I\,k}\, \left(1 - exp\left(-(\hat{\mu}_{I\,k} - \hat{\mu}_I)^2\right)\right) \end{aligned}$$

- Values are scaled to proper range :

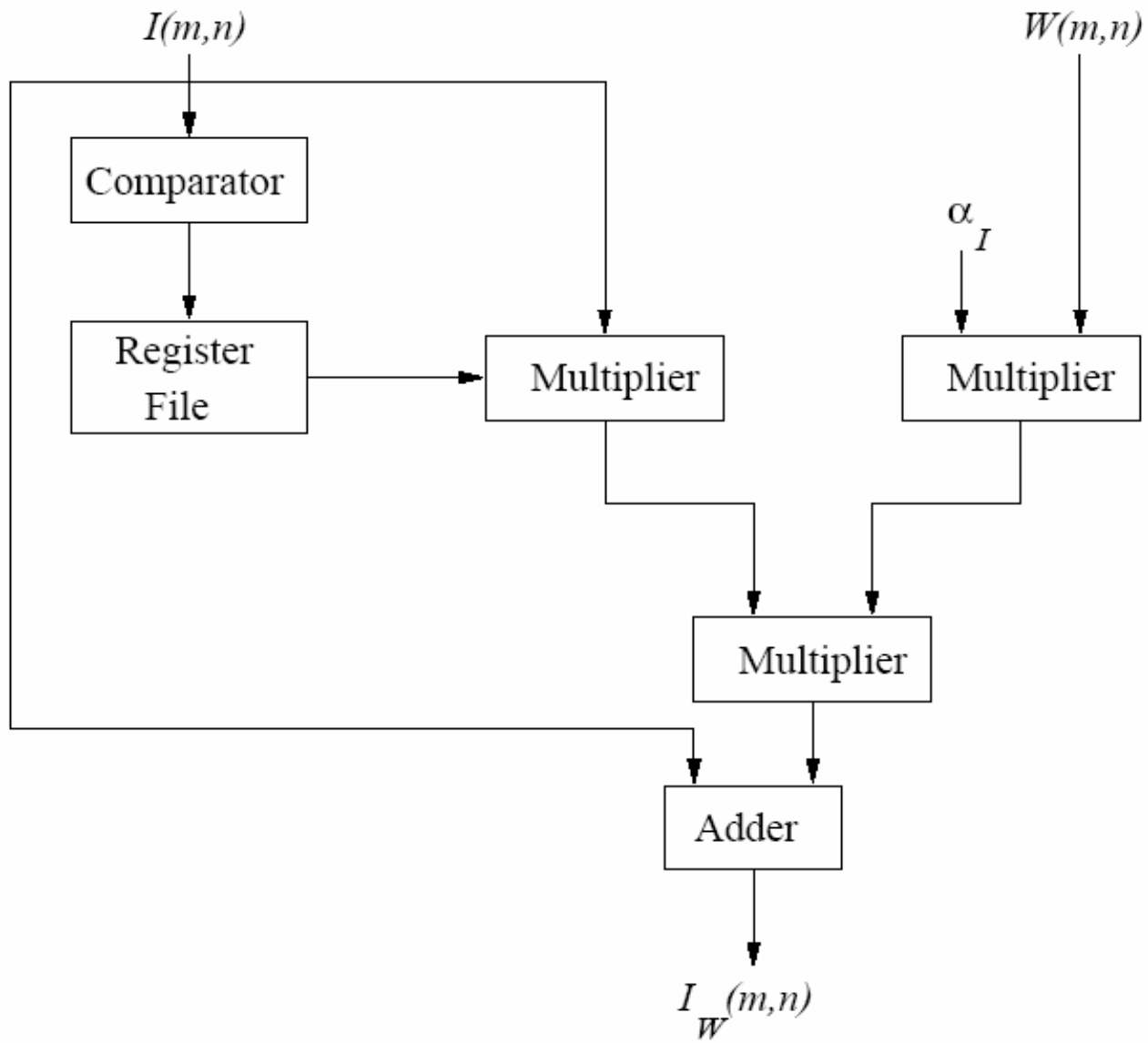
$$\begin{aligned} \alpha_k &= \alpha_{min} + (\alpha_{max} - \alpha_{min})\, \frac{1}{\hat{\sigma}_{Ik}}\, exp\left(-(\hat{\mu}_{Ik} - \hat{\mu}_I)^2\right) \\ \beta_k &= \beta_{min} + (\beta_{max} - \beta_{min})\, \hat{\sigma}_{Ik}\, \left(1 - exp\left(-(\hat{\mu}_{Ik} - \hat{\mu}_I)^2\right)\right) \end{aligned}$$

- Tyler series is used to approximate the exponential.

# Visible Watermarking Algorithm 2 [3] …

- For edge blocks the scaling and embedding factor is assumed to be $\alpha_{max}$ and $\beta_{min,}$ respectively to preserve the edges of the image.
- First order derivates are used to detect an edge.
- When the mean amplitude of an block exceeds a user defined threshold, then the block is declared as an edge-block.

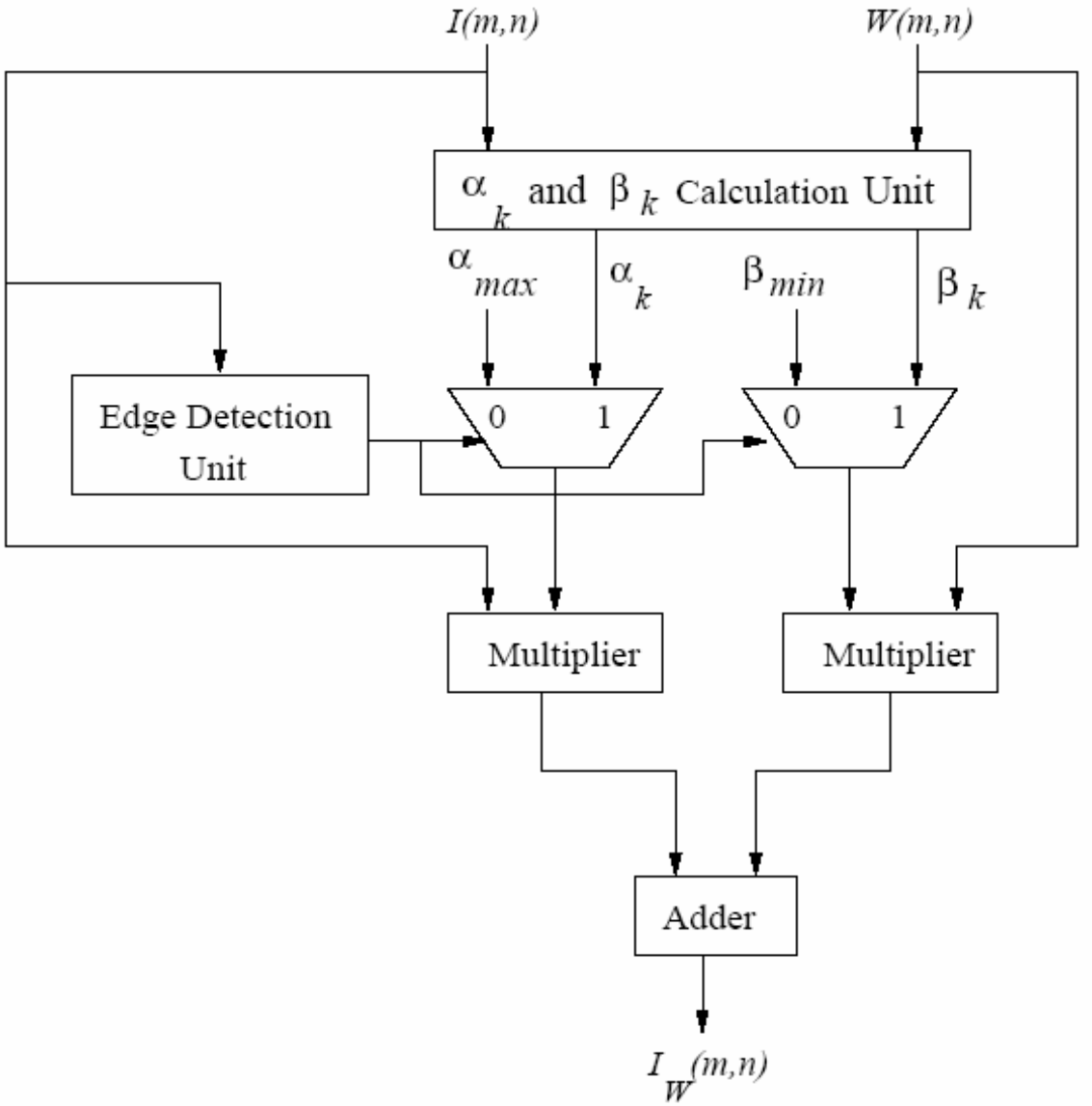
# Architecture for Algorithm 1




Constants are stored in the register file.

$\alpha_I$ is taken as user input.

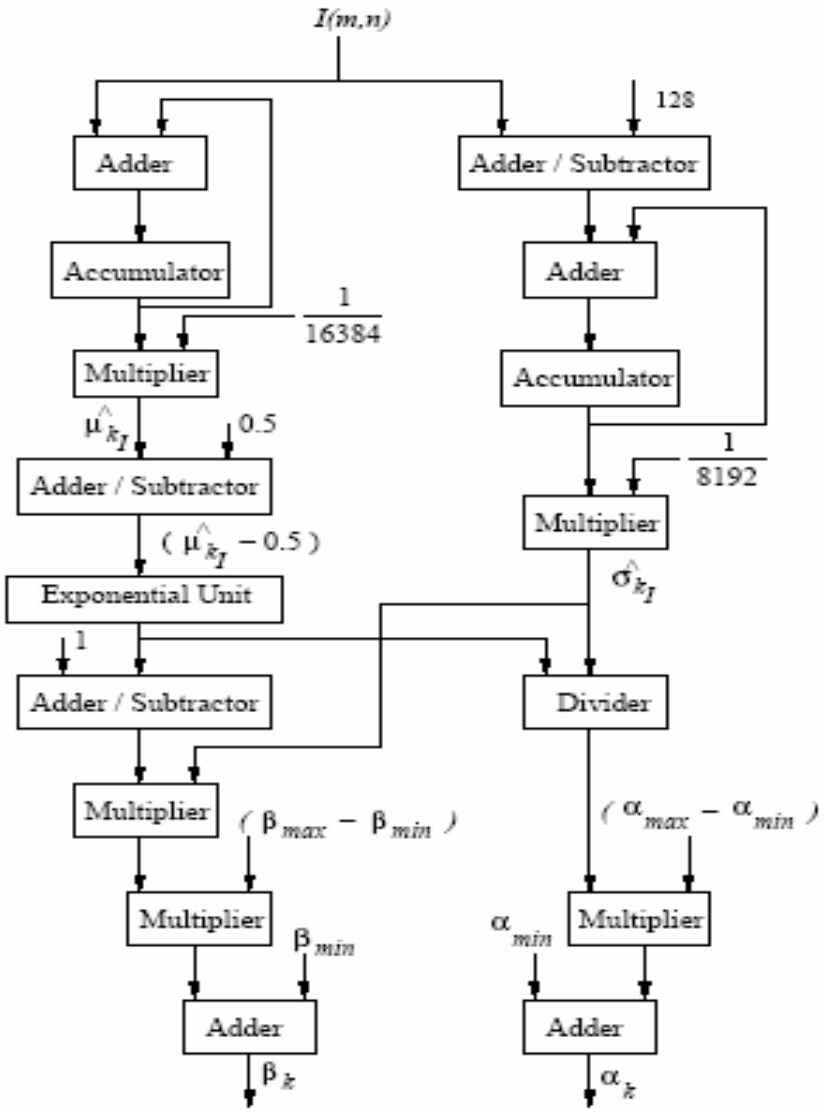
# Architecture for Algorithm 2




The "$\alpha_k$ and $\beta_k$ calculation unit determines the scaling and embedding factors.

The "edge detection unit" determines the edge and non-edge blocks.

# Architecture for Algorithm 2 ($\alpha_k$ and $\beta_k$ calculation unit)

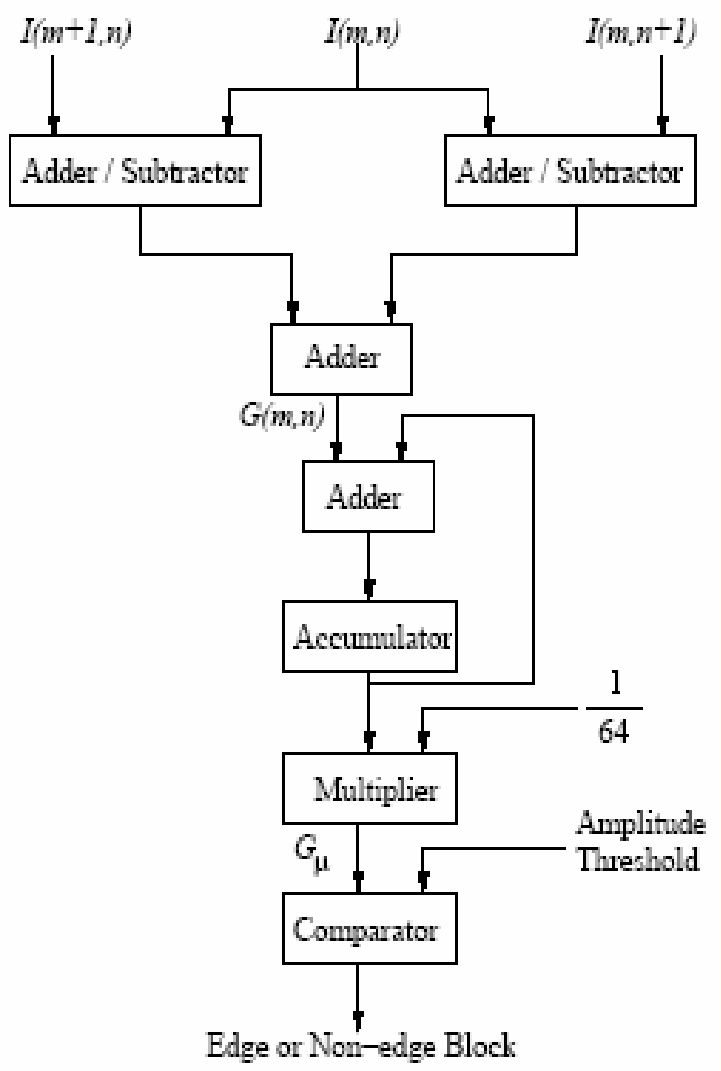



The block size is assumed to be 8 x 8.

$I_{white}$ is assumed as 256.

The deviation of the mean block gray value is calculated form mid intensity $I_{white}/2$ to accelerate the hardware performance.

# Architecture for Algorithm 2 (Edge detection unit)

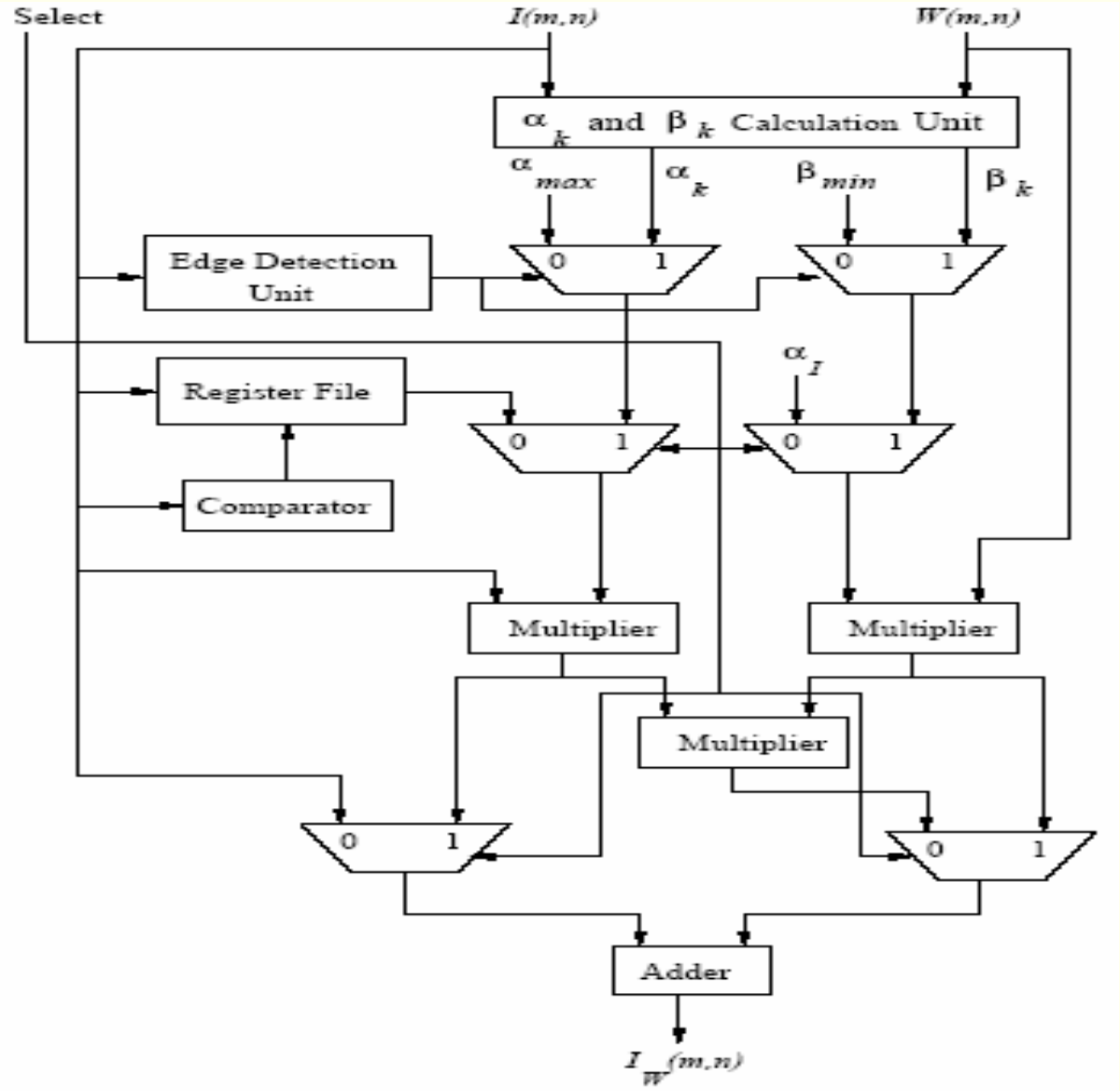



The absolute values of horizontal and vertical gradients are calculated using the adder/subtractor.

G(m,n) is the amplitude of an edge.

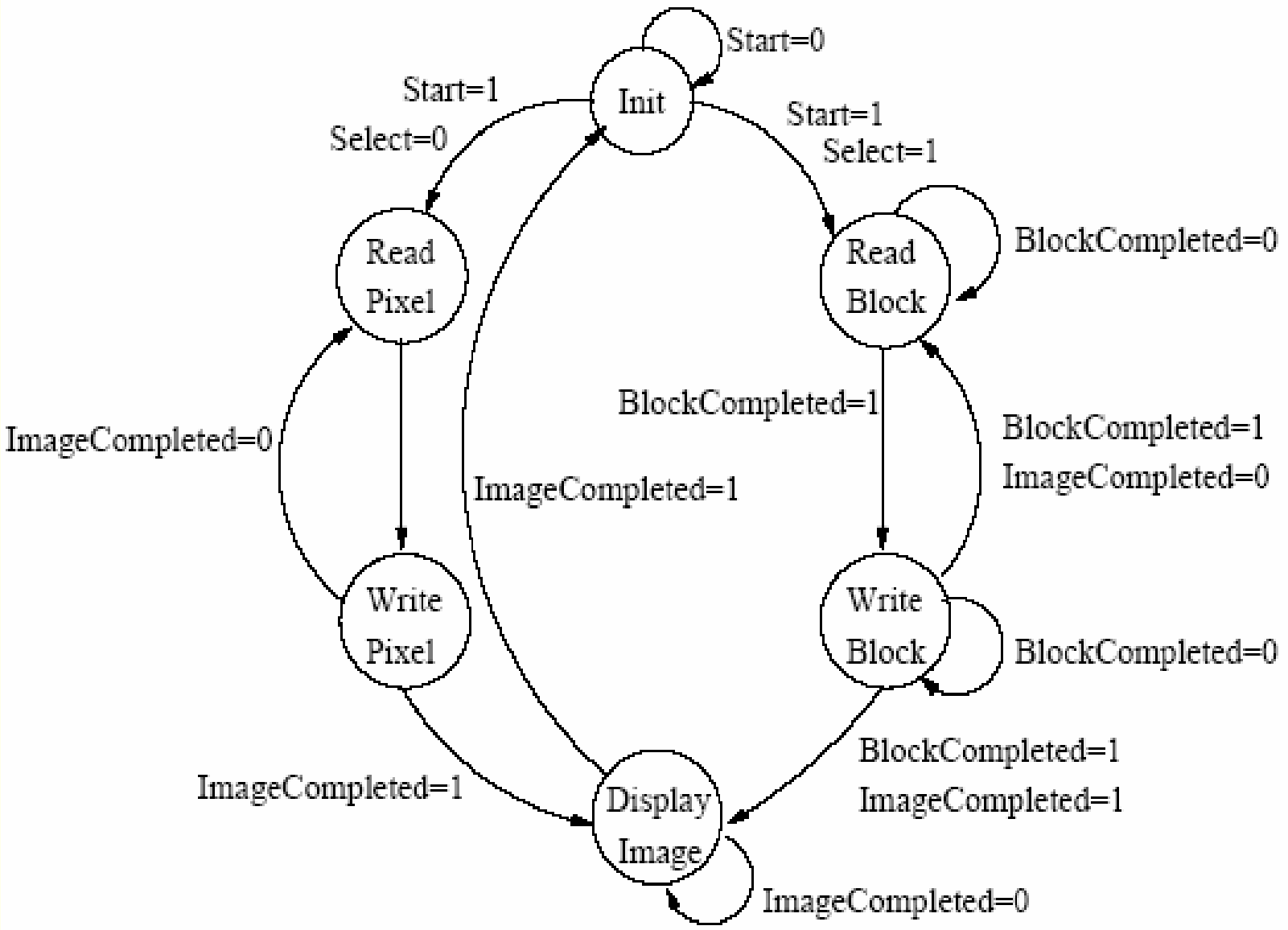
Accumulator-adder unit calculates the amplitude for a block.

# Combined Overall Datapath Architecture




Multiplexers are used to stitch the two datapaths.

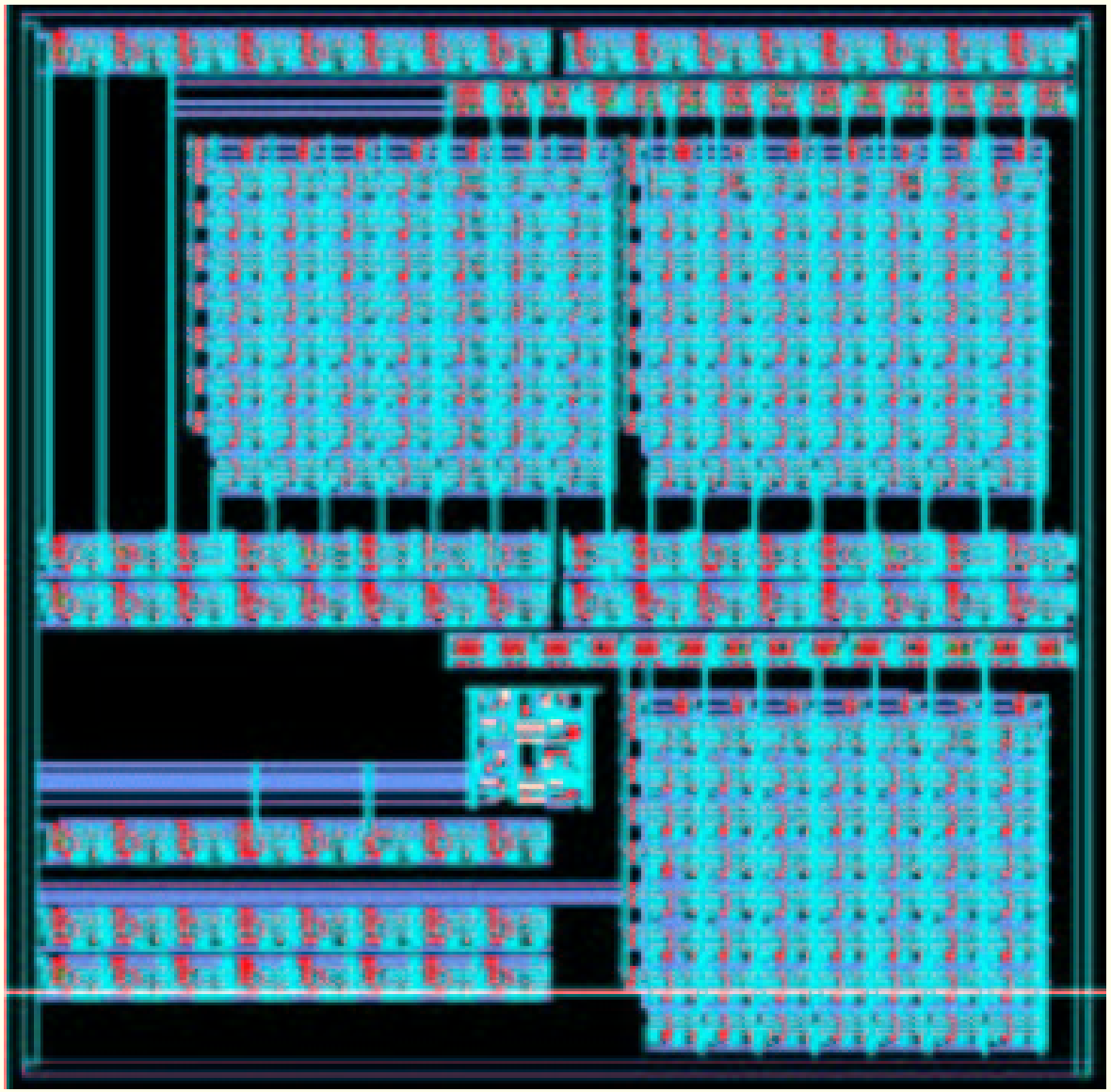
# The Controller for the merged Datapath



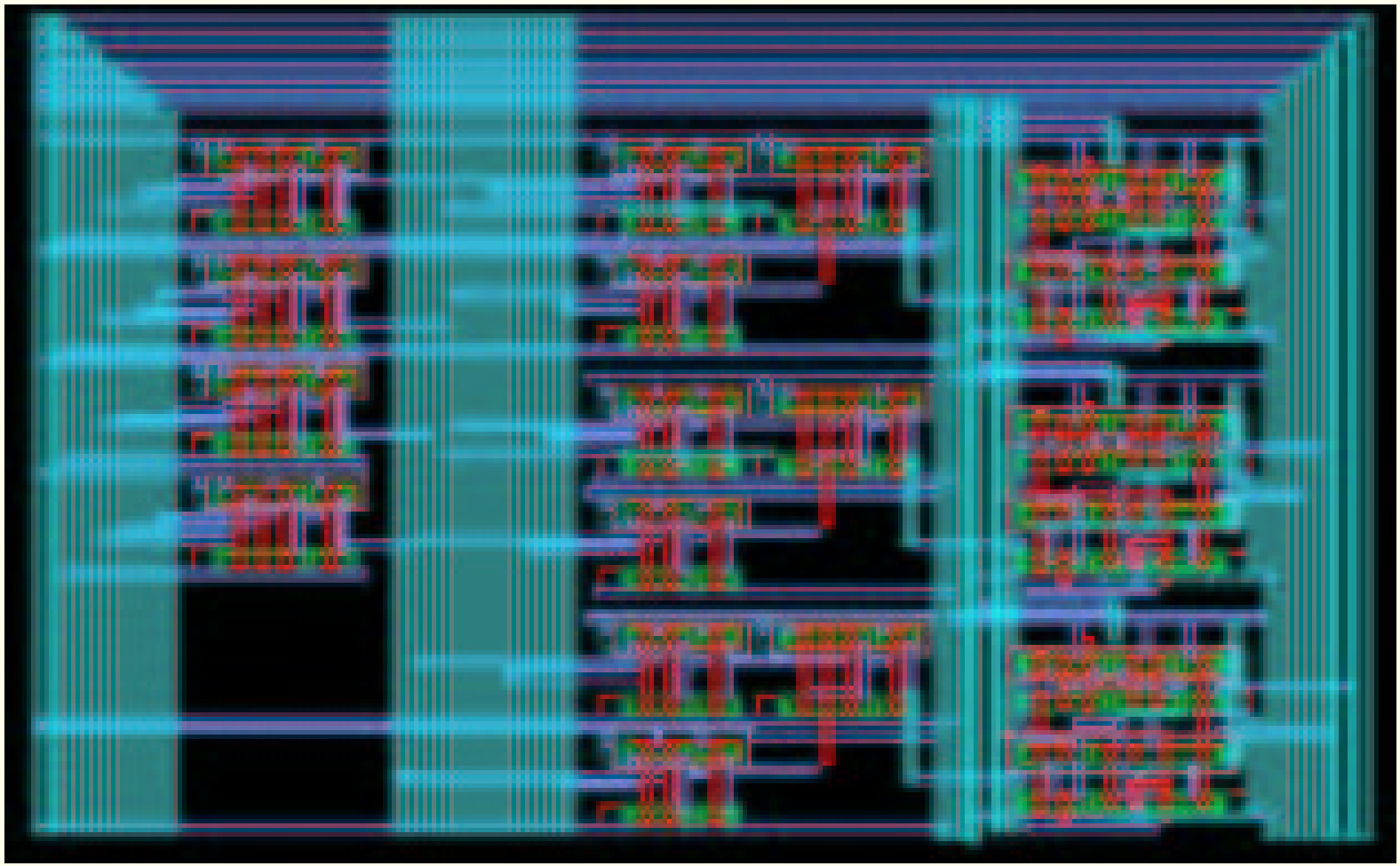
# Prototype Chip Implementation

- The implementation of the chip was carried out in the physical domain using the Cadence Virtuoso layout tool using bottom-to-top design approach.
- We designed our own standard cell library containing basic gates, such as AND, OR, NOT using 0.35µ technology.
- The fundamental functional units are 8-bit adders, 8-bit multipliers and 8-bit adder/subtractor.
- Adder is a ripple-carry manner and the multiplier is a 8-bit parallel array multiplier.
- The divider is implemented using the shift and subtract logic for the division.

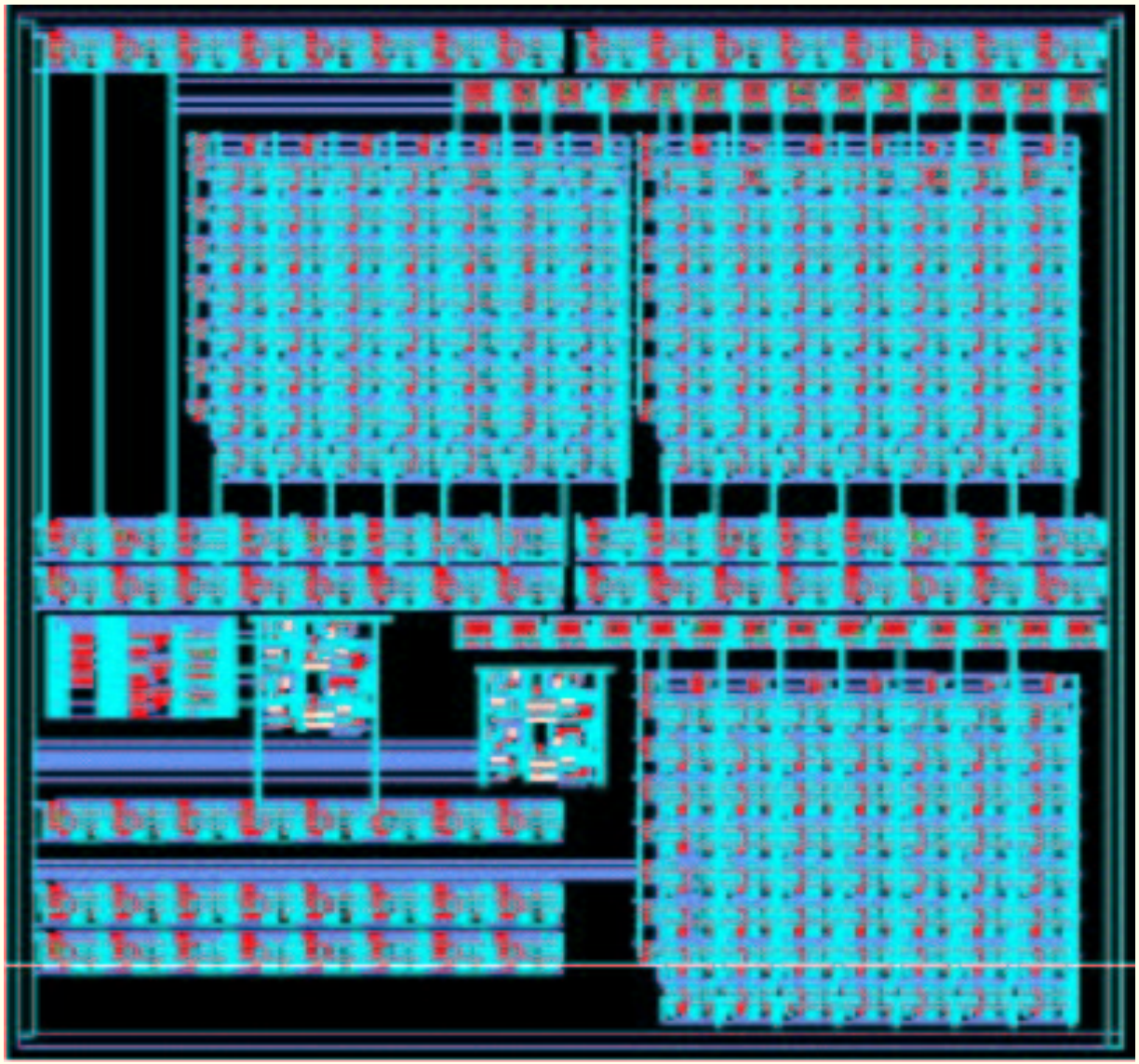
# Datapath Layout



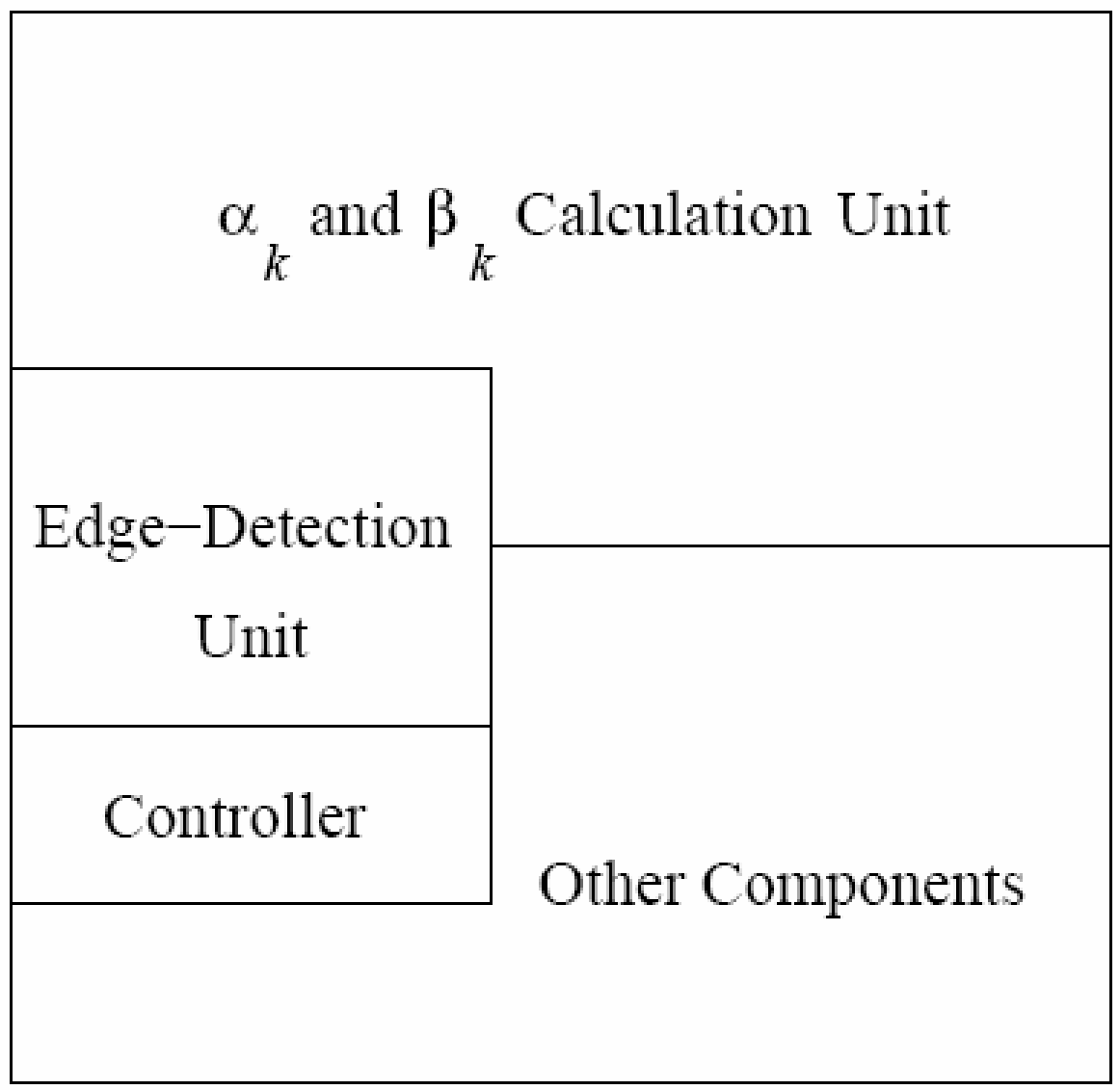
# Controller Layout



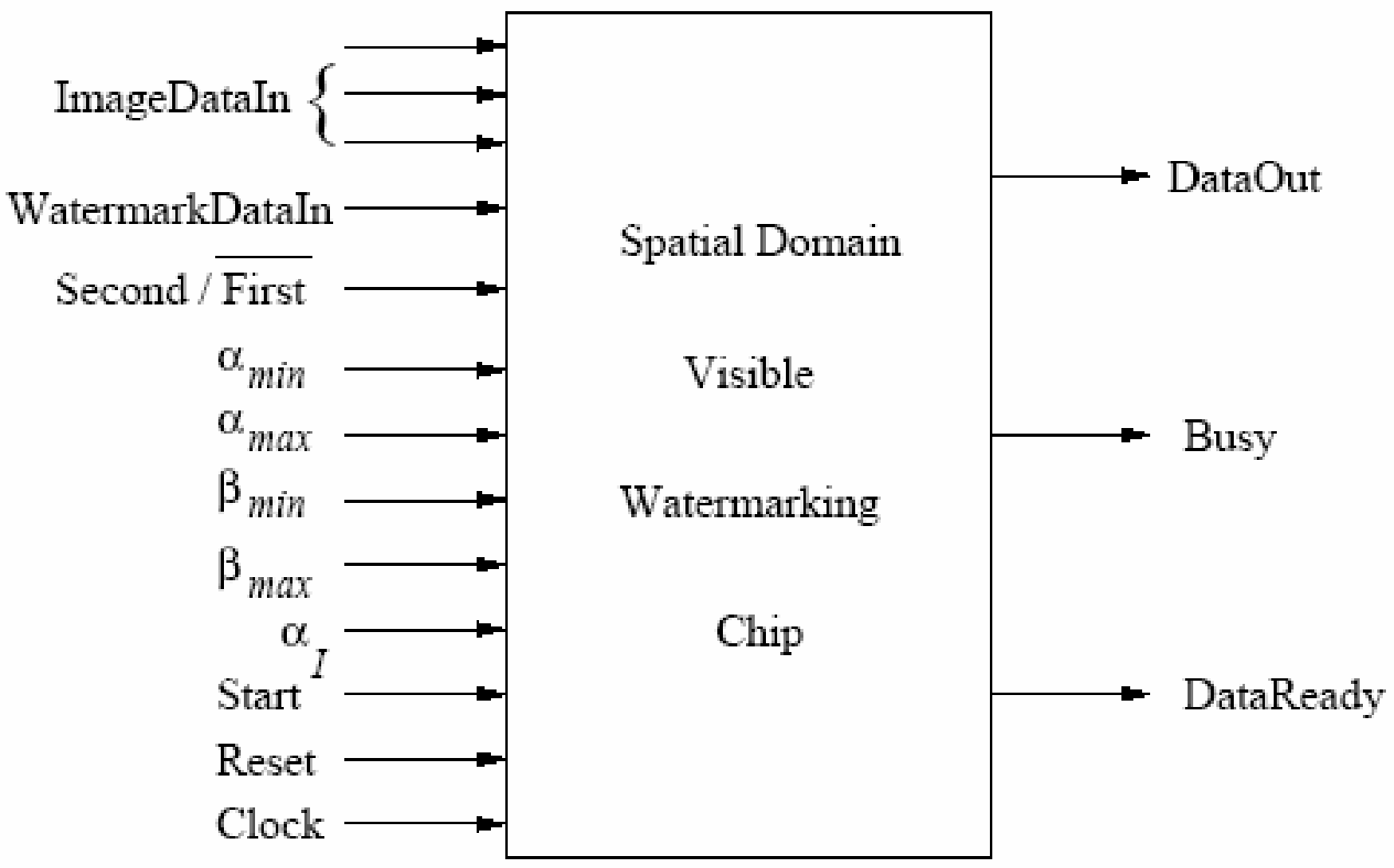
# Overall Chip Layout



# Chip Floorplan



# Chip Pin Diagram



# Power and Area of different Units (0.35 μ)

| Modules | Gate Count | Power $(mW)$ | Delay $(ns)$ |
|---|---|---|---|
| Exponential unit | 2370 | 1.2314 | 0.8981 |
| Edge detection unit | 3599 | 1.4137 | 1.0967 |
| $\alpha_k$ and $\beta_k$ calculation unit | 16279 | 3.444 | 2.0241 |
| Controller | 163 | 0.0034 | 0.3201 |

# Overall Statistics of the Chip

| | |
|---|---|
| Area | $3.34 \times 2.89 mm^2$ |
| Number of gates | 28469 |
| Clock frequency | $292.27 MHz$ |
| Number of I/O pins | 72 |
| Power | $6.9286 mW$ |

# Visible Watermarked Test Images

(a) Lena

(b) Bird

(c) Nuts and Bolts


(d) Watermark

Original Images and Watermark

(a) Lena

(b) Bird

(c) Nuts and Bolts

NOTE: Similar watermarked images are obtained using algorithm2. The difference lies in the SNR.

Watermarked Images using Algorithm 1

# Conclusions

- It is observed that the results of hardware based watermarking schemes are comparable to that of software.
- The first algorithm does pixel-by-pixel processing, whereas the second on does block-by-block processing. If the first algorithm can be converted to a block-by-block one, then chip operating at higher frequency can be developed.
- Each of the functional units, such as adders, multipliers, etc., can be optimized.
- A low power implementation of the chip can be done.
- Pipelined and parallel implementations also possible.

# Thank you