Wipe Generation Using IPM16

Introduction

The IPM16 is a systolic image processing array capable of performing various operations on digital video and digital images. This application note describes how the IPM16 is capable of complex wipe generation that involves multiple mathematical functions and arithmetic operations. The wipe algorithm and efficient implementations for the IPM16 are also described in detail.

Overview

The wipe generation process consists of several modules as shown in Figure 1 below. The current pixel's horizontal and vertical positions are derived from the input video's timing signal in the IPM16's pixel counter, and then processed with frame dimensions and repeating factors to produce normalized coordinates. Repeating wipe patterns can be easily accomplished by setting appropriate repeating factors without reloading configurations.

The wipe pattern generator is capable of most standard wipes in a single configuration. An alpha stream that carries wipe patterns is created from normalized coordinates and wipe specifications. Wipe patterns can be combined with sine waves in the wave generator. Either modified or original alpha is used to blend hard, soft, or bordered wipes with various blending options. The parity control data indicates even or odd regions to the blender for alternating or checkerboard wipe effects.

Besides wipes, the blender also produces dissolve effects. A random number stream is internally generated by IPM16's RNG block and processed by the dissolve generator to produce an alpha stream that carries noise patterns. The random number stream is also sent to the blender for producing noise borders. Generated streams, such as alpha, parity, and random numbers all carry timing signals and will be re-equalized with input video streams before blending.

As is true for all the IPM16 configurations, this application can be combined with other applications to a build large system. An example can be found in Application Note 6 "Configuring Multiple IPM16s for Cascaded Operation From 4:2:2:4 Streams". Full implementation of this application uses one and one half IPM16s. Depending on system requirements, modules can be customized to perform partial functionality with a single IPM16. A single IPM16 generates most wipe and dissolve effects.

Block Diagram

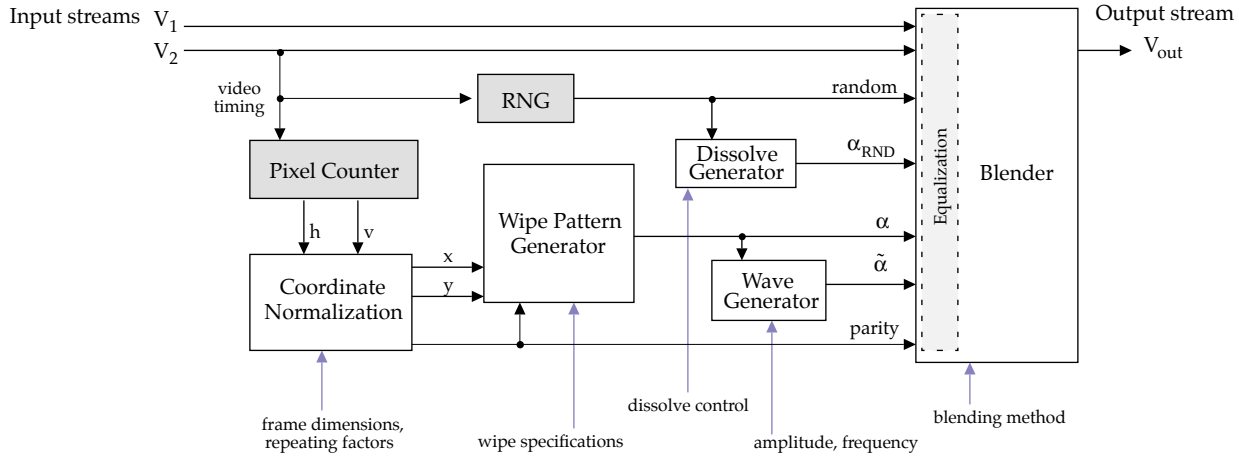


Figure 1: Wipe Generation Process

Coordinate Normalization

Pixel positions are properly scaled by frame dimensions and repeating factors, then replicated by the IPM16's modulo1 function to produce normalized coordinates. The modulo1 function generates multiple linear functions with repeat intervals of length one from a continuous input. To express it mathematically, the result for the modulo1 function of a continuous parameter a is

$$\text{Modulo1}(a) = a - n, \quad \text{for } n \le a < n + 1 \qquad (1)$$

where n is the largest integer smaller than a. The normalized output coordinates are independent of wipe display size, hence simplifying data range and precision settings. Repeat factors can be changed every field for real-time effects.

Parity data is extracted during the process of coordinate normalization. A checkerboard is created by combining horizontal and vertical parity data inside the same arithmetic unit that generates normalized coordinate. Switching between repeat types is made by selecting different parity data to the output.

Wipe Pattern Generation

The wipe pattern generator dynamically calculates alpha values from normalized coordinates and the current field's wipe shape, position, size, and border width. Since most standard wipe patterns are combinations of basic geometric objects such as lines, circles, and ovals, part of the IPM16 configuration is used to produce multiple alpha streams for several of these objects simultaneously. The wipe algorithm is directly implemented in the IPM16 through arithmetic level programming. The other part of the configuration is used to combine simple objects into various shapes and to send out a single alpha stream.

• Lines

A straight boundary that separates two sources in a video frame can be defined by two points $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, and the linear equation

$$(y_2 - y_1)x + (x_1 - x_2)y + (x_2 y_1 - x_1 y_2) = 0 \qquad (2)$$

In the following discussion, the first video source is assumed to be on the positive side of the line and the second on the negative side of the line. Note that for display devices with a top-left origin, the sign assignment and rotation angles are reversed from common mathematical conventions. Inside the IPM16, the following multiplier-efficient line equation is implemented instead of Eq. (2).

$$y = mx + b, \text{ with } m = \frac{y_2 - y_1}{x_1 - x_2},\ b = \frac{x_2 y_1 - x_1 y_2}{x_1 - x_2} \qquad (3)$$

The slope m and the vertical intercept b can be updated every field to produce a moving line boundary. Since any 2-dimensional transformation for a point can be easily divided into a series of basic transformations, Eq. (2) is suitable for computing new line coefficients under such transformations. To create a vertical line that has an undefined slope, the IPM16's optional mask operation on the y input can be activated.

For soft borders, continuous alpha values are generated by computing the point-to-line distance

$$d = (y - mx - b)/(\sqrt{1 + m^2}) \qquad (4)$$

Since the slope m is updated on a field-by-field basis, the square root $\sqrt{1 + m^2}$ is treated as a constant in the IPM16 and its value changes only during vertical blanking. The point-to-line distance is then scaled by transition width d_{tw} and clipped to a range that is proper for the selected border type. Mathematically, a hard border is produced by switching between foreground video and background video based on the sign of the distance d. But this practice produces jagged edges. A better method is to set a very small transition width for hard borders so that edges look smooth on display.

For colored borders, an area of full border color is produced by adding or subtracting line width d_{lw} in the distance calculation. The result is then scaled by transition width for soft transitions between border colors and video sources. Total border width, including the transition area, is twice of d_{tw} plus d_{lw}. A general formula is used in the IPM16 to compute alpha for all border types

$$\begin{aligned} \alpha &= \text{Clip}[(d - d_{lw})/d_{tw}], && d \ge d_{lw} \\ &= \text{Clip}[(d + d_{lw})/d_{tw}], && d \le -d_{lw} \\ &= 0, && |d| < d_{lw} \end{aligned} \qquad (5)$$

The IPM16 configuration for a line wipe includes an arithmetic unit processing normalized coordinates x, y with line coefficients, and a second unit producing soft borders as shown in Figure 2. One of the three functions in Eq. (5) is selected based on intermediate data d' through controlled operations shown in shaded areas. To reduce the number of multiplications, the factor $\sqrt{1 + m^2}$ in Eq. (4) is multiplied to border and transition widths to produce constants d'_{tw} and d'_{lw} used in the arithmetic unit. Besides line coefficients and width

parameters, mask and clip operations can also be updated on a field-by-field basis.

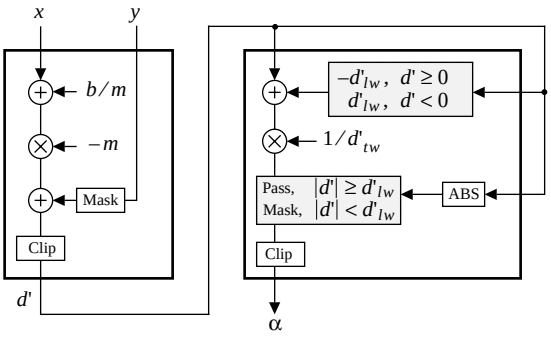


Figure 2: Line wipe configuration. Symbol prime denotes that the quantity is multiplied by $\sqrt{1+m^2}$

• Circle and Oval

Similar to line wipes, circle wipes can be created from normalized coordinates, and the circle's location, size, and border width. The IPM16 uses a quadratic function to calculate output alpha for a circle. For hard borders, the quadratic function yields the same results as exact calculations. For soft borders, good visual results can be made by this method as shown in Figure 8. Oval wipes are similar to circles except that two symmetrical axes have different lengths.

The IPM16 configuration for circle wipes consists of three arithmetic units. The first two units calculate quadratic functions with circle location and size parameters. Soft borders are produced by the third unit that computes Eq. (5). Intermediate data is also used to switch between functions in this unit. The IPM16 configuration for oval wipes has an additional arithmetic unit for processing the ratio between two axes before computing quadratic functions.

• Combine Lines and Curves to Shapes

Two combination methods, comparative and multiplicative, are implemented in the configuration to create shapes from lines, circles, and ovals. The comparative combination performs minimum or maximum comparisons on every intersection, and produces sharp intersections on soft borders as shown in Figure 3. The multiplicative combination smooths the intersection by multiplying properly clipped alpha values. However, the disadvantage is that the clipped alpha output can not be further processed in the wave generator, unless a more complex algorithm is implemented. Both methods produce similar results for small transition widths.

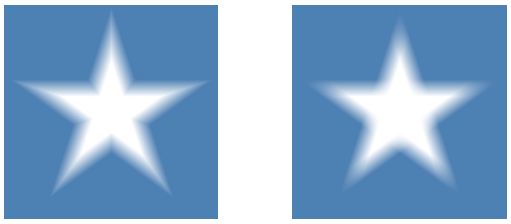


Figure 3: Alpha from comparative (left) and multiplicative (right) combinations

The IPM16 configuration for creating shapes can be customized to a user-specified number of lines or curves. An example of combining three lines into a triangle is shown in Figure 4. Each line is produced by a pair of arithmetic units which are configured as in Figure 2. Both comparative and multiplicative combinations are performed in the same units, and one of the results is selected as the alpha output. The combination units can be cascaded to combine more lines and curves. For example, the 5-point star pattern shown in Figure 3 is created by five pairs of arithmetic units for generating lines, and nine units for processing ten intersections. Similar configurations can be made to combine curves by replacing line-generating units with curve-generating units.

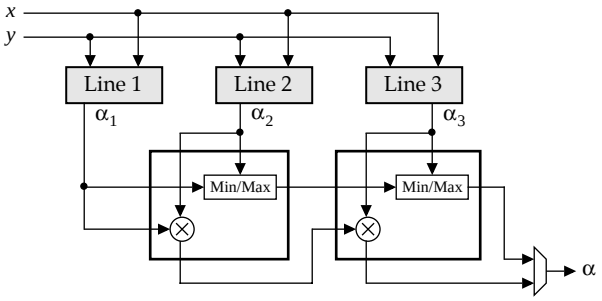


Figure 4: Configuration for the line combination

A general rule for designing a multi-line or multi-curve wipe configuration is to fit the most complex shape required by system. Once the complex configuration is implemented, simple wipes can be created by masking some of the lines or curves without reloading a new configuration.

• Other Wipe Patterns

With a multi-line configuration, a single IPM16 can produce polygons, radial wipes, matrix effects, and other shapes that have less than five lines and ten intersections. Symmetric shapes can be folded to reduce the

INTELLIGENT PARADIGM
10050 N. Wolfe Road, SW2-152 • Cupertino, CA U.S.A. 95014-2528
Tel: 408-725-3810 • Fax: 408-725-3819
www.IntelligentParadigm.com

number of lines. Repeating wipe patterns can have different timing between wipes for the waterfall matrix effects. Interesting curve shapes such as cat eye, heart and round rectangles can be generated by a multi-oval or line-circle configuration.

Wave Generation

Wipe patterns can be modified with time-varying waves by superimposing a sine function on the alpha streams. The output alpha for waves propagating in horizontal direction with amplitude A and period length λ is

$$\tilde{\alpha} = \alpha + A \cdot \sin(2\pi x/\lambda) \tag{6}$$

The sine function can be approximated to less than 0.5% of maximum error by replicating the period throughout the whole video frame and using polynomial expansion within the period. The replication is made by the IPM16's Modulo1 function, with proper scaling on coordinate x. Although the IPM16 can efficiently process higher order terms, smaller polynomials are preferable because higher order terms are still multiplier-consuming even after optimizations. To reduce the size of the polynomial and consequently the number of multiplications, the center of expansion is offset to the middle of the period such that both sides of the expansion are within the valid range. After replication, the center should be offset back to its original value. Mathematically, these descriptions correspond to

$$x' = \text{Modulo1}\left(\frac{x}{\lambda} + 0.5\right) - 0.5 \tag{7}$$

The variable x' is then multiplied by 2π before applying in the sine function. To further reduce the polynomial size, the outer side of the period is folded into the inner side by the trigonometric rule $\sin(2\pi x') = \sin(\pm\pi - 2\pi x')$ to produce variable x'' for the sine function.

$$\begin{aligned} x'' &= 2\pi \cdot x', && |x'| \le 0.25 \\ &= \pi - 2\pi x', && 0.25 < x' \le 0.5 \\ &= -\pi - 2\pi x', && -0.5 \le x' < -0.25 \end{aligned} \tag{8}$$

With the input x'' restricted to the range $-\pi/2$ and $\pi/2$, the sine function can be approximated to the desired accuracy by three terms.

$$\Delta\alpha \approx A \cdot \left(x'' - \frac{x''^3}{3!} + \frac{x''^5}{5!}\right) \tag{9}$$

The IPM16 configuration for wave generation consists of two arithmetic units for computing x'' and six units for the polynomial expansion as shown in Figure 5. Since the modulo1 function is located at the end of the arithmetic unit, the second offset of Eq. (7) is computed with Eq. (8) in the second arithmetic unit. Intermediate data x' is also used to control the constants and perform different computations based on its value. Data streams can be redirected to several arithmetic units for computing higher order terms of the polynomial.

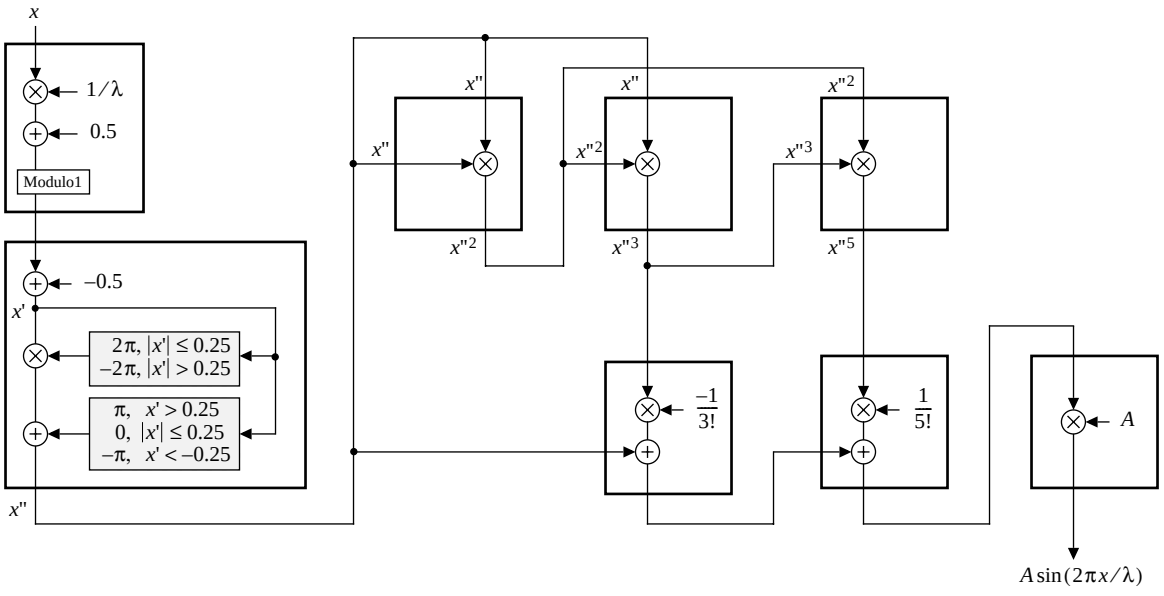


Figure 5: Configuration for wave generation

For a wave propagating in an arbitrary θ direction instead of horizontal direction, the coordinate x is transformed by the rotation $x \rightarrow x\cos\theta + y\sin\theta$. One of the rotation coefficients can be combined with period λ. Only one extra multiplier is required to compute the ratio of two coefficients. Similar to the line wipe configuration, special cases such as $\theta = 90^\circ$ can be properly treated with optional mask operations.

Motion Effects

In order to produce motion effects, a CPU will have to recalculate constants, such as line coefficients, on a field-by field basis. Constants and optional controls are updated in IPM16 during vertical blanking to ensure smooth motion effects. A wipe pattern is transformed by applying a series of scaling, rotation, and translation on points that defines shapes, and then computing new constants from transformed points. Since each line is independently updated in the IPM16, shapes generated by a multi-line wipe can be changed by applying different transformation on lines. Moving curves and waves are also produced by similar methods.

Blending

The blender takes two video streams, each with several components, and an alpha stream carrying wipe patterns. Various borders are created by applying one of the following blending methods regardless whether the alpha is coming from wipe, wave, or dissolve generators. The first four blending methods are basic types in this application. With more resources to process additional data streams and controls, interesting effects such as noise or rainbow borders can be derived from the basic types described below.

• Hard Border

The output video stream switches between two input video streams based on the alpha's sign to produce hard borders. As mentioned in the line wipe section, a hard border can be approximated by a soft border of small transition width to avoid jagged edges.

• Soft Border

The transition between two video streams are linearly proportional to the alpha value, which should be constrained to the range 0 and 1 before being sent to the blender. Full display of video source A or B corresponds to alpha value of 1 or 0, respectively. The blending formula for soft borders is

$$V_{out} = \alpha V_A + (1-\alpha)V_B, \quad 0 \le \alpha \le 1 \tag{10}$$

• Colored Hard Border

The output video switches between video stream A, B, and a constant border color based on the alpha value as shown in Eq. (11). A colored hard border can also be approximated by a colored soft border of small transition width to avoid jagged edges.

$$\begin{aligned} V_{out} &= V_A, \quad \alpha > 0 \\ &= V_B, \quad \alpha < 0 \\ &= C, \quad \alpha = 0 \end{aligned} \tag{11}$$

• Colored Soft Border

The constant border color is linearly blended into input video streams on both sides of the border according to Eq. (12). In order to produce colored borders using the same set of arithmetic units, the alpha value should be constrained to the range -1 and 1 before being sent to the blender. Full display of video source A or B corresponds to the alpha value of 1 or -1, respectively.

$$\begin{aligned} V_{out} &= = \alpha V_A + (1-\alpha)C, \quad 0 \le \alpha \le 1 \\ &= |\alpha| V_B + (1-|\alpha|)C, \quad -1 \le \alpha < 0 \end{aligned} \tag{12}$$

• Colored Hard-Soft Border

The constant border color is linearly blended into one of the input video streams on one side, but hard switched on the other side. This is made by performing either Eq. (11) or Eq. (12) based on the sign of the alpha value, which should be constrained to the range -1 and 1 as well. A colored hard-soft border can also be approximated by a colored soft border with different transition widths on both sides.

• Noise Border

To produce a noise border, the alpha is combined with random numbers generated by IPM16's RNG block, then blended with video streams using the formulas for soft or colored soft borders. The RNG block can output evenly distributed random numbers over several ranges. For this particular effect, a range of -0.5 to 0.5 is chosen such that the random numbers have an average of zero, and the average for the output alpha correctly reflects the blending ratio of the original alpha. The noise can be synchronous or asynchronous to input videos, monochromatic or randomly colored.

INTELLIGENT PARADIGM
10050 N. Wolfe Road, SW2-152 • Cupertino, CA U.S.A. 95014-2528
Tel: 408-725-3810 • Fax: 408-725-3819
www.IntelligentParadigm.com

- **Rainbow Border**

  To produce a rainbow border, the constant border colors are replaced by varying colors. A data stream with smooth data values, such as the soft border's alpha or pixel position, is used to produce varying colors from constant colors. Rainbows are created by processing color components with different functions.

The IPM16 blending configuration includes several arithmetic units, one per video component. Each arithmetic unit is equipped with switchable border settings. Hard and colored hard borders are produced by changing output selections based on the input alpha as shown in Figures 6(a) and 6(c). In Figure 6(b), two video streams are subtracted before being multiplied by the common alpha value to reduce the number of multiplications in soft border calculations. The settings for colored soft borders are a combination of selection and multiplication as shown in Figure 6(d). Output data values can be clipped to appropriate ranges before being sent to the output interface or further processing.

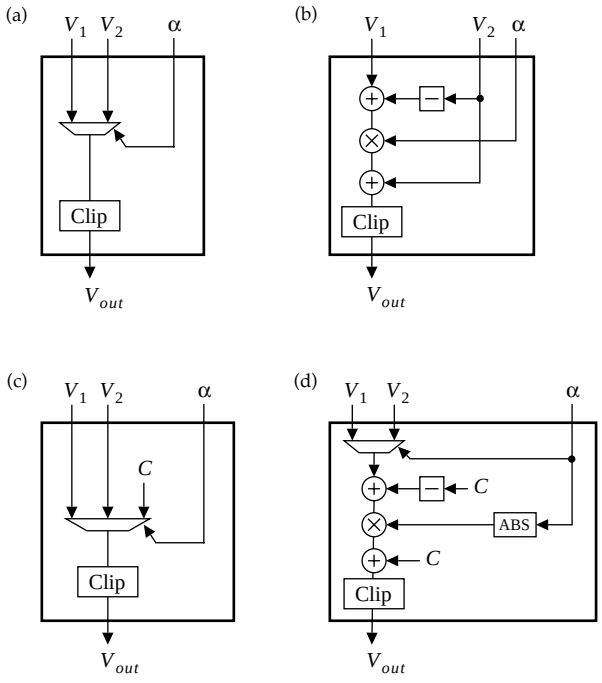


Figure 6: Basic blending configurations
(a) hard (b) soft (c) colored hard (d) colored soft borders

Dissolve Effect

Dissolve effects are created by blending input video streams with noise patterns. In the dissolve generator, random numbers are computed with the video source's dissolve ratio, which can change over frames linearly, quadratically, or with other interesting variations. An alpha carrying noise pattern is produced by the dissolve generator and then sent to the blender. Random number streams coming from IPM16's RNG block can be synchronized to input video streams for steady dissolve effects. Since random numbers are compared against the dissolve ratio ranging from 0 to 1, their distribution should be set to the same range in the RNG block.

The IPM16 configuration for the dissolve generator consists of only one arithmetic unit. Input random numbers are offset by the dissolve ratio and divided by an interval that specifies soft transition. Before being sent to the blender, the alpha is clipped to proper values.

Results

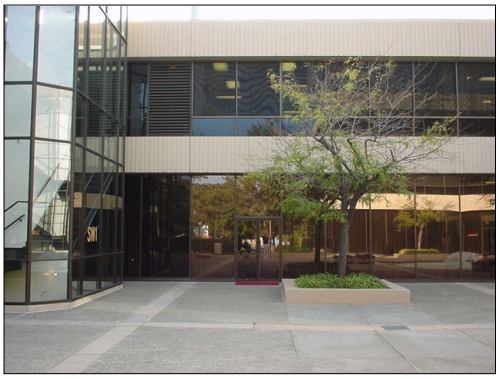
This application note describes a general wipe generation process that produces standard wipes, dissolves, waves, repeat patterns and various border effects. The configuration is divided into several modules of distinct functionality and can be customized to fit available resources.

Wipe patterns are computed according to their mathematical definitions on a pixel-by-pixel basis in the IPM16. Modifications on a pattern are made by setting appropriate parameters and constants in appropriate IPM16 modules. These settings can be updated every field to achieve real-time motion effects and controls.

Highlights for the IPM16's capabilities described in this application note include:

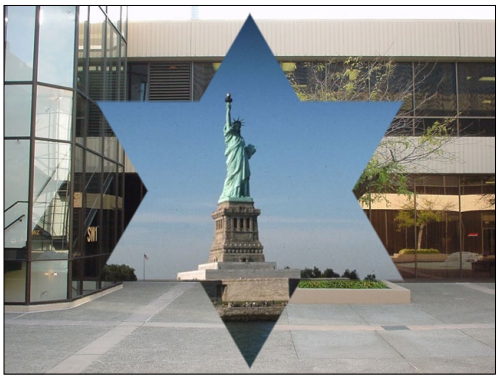
- direct mapping from algorithm to an IPM16 configuration
- flexible data redirection
- switching between functions based on input or intermediate data values
- pixel position detection
- synchronous or asynchronous random number generation with various range settings
- equalization for aligning streams coming from different processing paths
- arithmetic operations on data streams including
  - — minimum and maximum comparisons
  - — absolute value
  - — negation
  - — mask operation activated by certain data values or external controls
  - — clip operation with user selected ranges
  - — modulo1 function

Results of multi-line and multi-oval wipes, as well as various borders, waves, and repeat effects are shown in Figures 7 and 8. Figure 7(a) and 7(b) are the two inputs for the application. Figure 7(c) shows a 6-point star produced by combining three lines and symmetric folding. The hard borders are produced by setting small transition widths in soft border calculations. Figure 7(d) shows a shape similar to a British flag with a colored hard border. The borders are also approximated by small transition widths. Figure 8(a) shows a horizontal waterfall matrix that runs several line wipes from left to right with different timings. This is made by first dividing the frame into several horizontal stripes, then controlling line wipe locations based on the current stripe number. The total number of stripes can be as many as the maximum value allowed in the user specified data precisions. Figure 8(b) shows horizontal-propagating waves added to a diamond with colored soft border. Figure 8(c) shows a rotated oval with colored soft noise border. The amount of noise added to the border is proportional to the distance between pixel location and the middle of the border. Figure 8(d) shows a heart created by mirroring a rotated oval along center vertical line. The soft border's alpha is also used to produce rainbow border colors. Figure 8(e) shows a combination of five circles that look like a circular wave when their boundaries move outward as one video source replaces the other. A soft border is used to make the effect more visible. Finally, in Figure 8(f), two sets of stars are rotated in opposite directions and blended with different border colors, depending on which region of a checkerboard the star is in. The inner sides of stars are set with small transition widths to produce hard-soft borders.



(a) A input

(b) B input

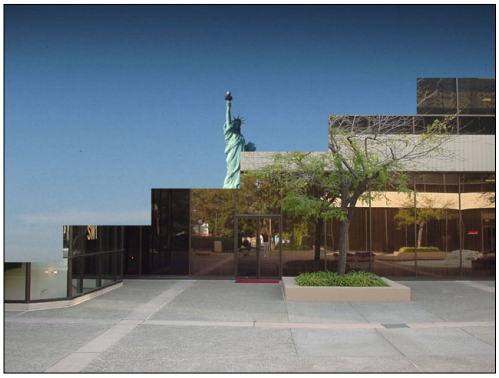


(c) 6-point star by using 3 lines and symmetries; hard border

(d) "British flag" wipe using 4 lines and a colored hard border

Figure 7: Inputs and multi-line wipes

INTELLIGENT PARADIGM
10050 N. Wolfe Road, SW2-152 • Cupertino, CA U.S.A. 95014-2528
Tel: 408-725-3810 • Fax: 408-725-3819
www.IntelligentParadigm.com



(a) Horizontal waterfall left matrix;
hard border

(b) Waving diamond;
colored soft border

(c) Rotated Oval;
colored soft noise border

(d) Heart by mirroring an oval;
soft rainbow border

(e) Circular wave by using 5 circles;
soft border

(f) Checkerboard of stars;
colored soft border on one side, hard on the other

Figure 8: More wipe effects