SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

THE IMSA110 BACK-END POST PROCESSOR

SUMMARY **Page**

I. INTRODUCTION

The IMSA110 consists of a high performance configurable array of multiply-accumulators (420 MOPs), three programmable length 1120 stage shift registers and a versatile backend post processing unit. All these features are controlled from a microprocessor interface. The comprehensive on-chip facilities ensure that a single device is capable of dealing with many tasks commonly found in the fields of signal and image processing.

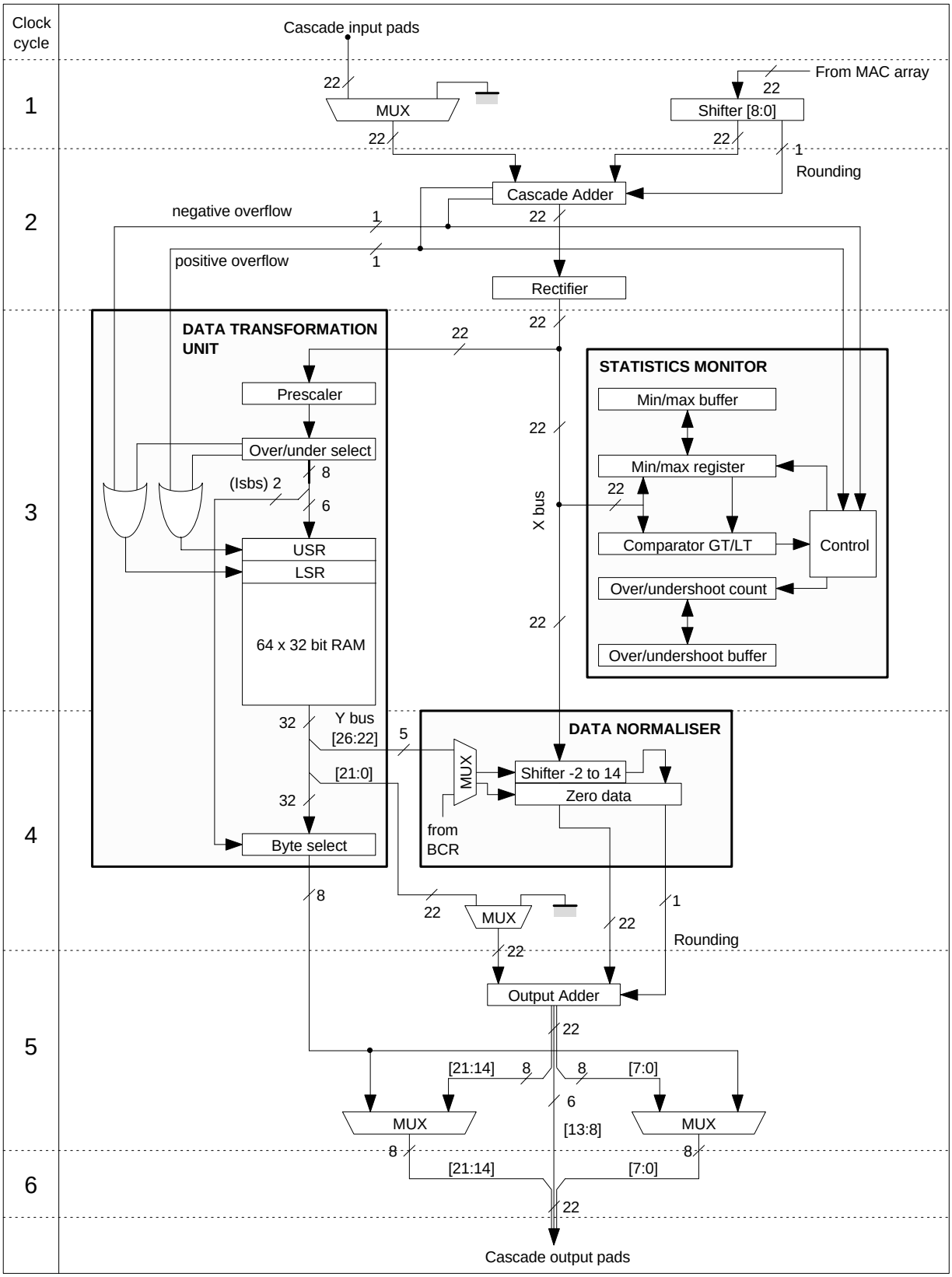
The backend post processing unit gives the IMSA110 a high degree of flexiblility, especially for image processing applications. This document describes by example some of the uses of the back-end post processor.

Unless specified otherwise all the examples considered will be based around image processing applications with 8 bits per pixel being used to represent the image data.

2. DESCRIPTION OF THE BACK-END POST PROCESSOR

Figure 1 shows the functional blocks and interconnections which are present within the backend post processor of the IMSA110.

Figure 1 : Detailed Block Diagram of the Back-end Post Processing Unit



This diagram can be broken down into 4 main sections, the input block, statistics monitior, data conditioning unit and output block. A brief description of each of these major sections is given below, for full details reference should be made to the data sheet.

2.1 Input Block (shifter, cascade adder and rectifier)

Data from the MAC array encounters the shifter when it enters the input block. The shifter is capable of up to 8 arithmetic shifts in either direction. When shifting left it is possible for an overflow to occur. Such an overflow is not detected by the device, hence it is left to the user to ensure that unintentional overflows do not occur. When shifting right rounding is applied to improve the accuracy of the device. The magnitude and direction of the shift are controlled by BCR0[5..1] as described in the data sheet.

The output data from the shifter is fed into the cascade adder. Here it is added to both the rounding bit generated by the shifter and the data applied to either the cascade input bus or zero depending on the setting of BCR0[0]. Should the result of the 22 bit signed addition be greater than 2^{21}-1 then a positive overflow is generated. Similarly if the result is less than -2^{22} a negative overflow is generated.

The output from the cascade adder can be optionally full or half wave rectified depending on the setting of BCR0[7..6]. The output of the rectifier drives the X bus. Note that when full wave rectification is being used and the output of the cascade adder is -2^{21} then the output from the rectifier remains as -2^{21}.

2.2 Statistics Monitor

The statistics monitior allows the X bus to be monitored for certain conditions. Four different modes of operation are possible and these are tabulated below:

Mode	BCR1[1]	BCR1[0]
Max Register	0	1
Min Register	0	0
Overshoot Counter	1	1
Undershoot Counter	1	0

When configured to be in max register mode and the X bus exceeds the current threshold in the MMR (max/min register), then the MMR is loaded with the value on the X bus and the counter (OUC) is incremented. If the threshold is not exceeded then no action is taken. Thus assuming the MMR was initially set to -2^{21} its value at some later time is the maximum value which has appeared on the X bus in that period, and the OUC has been incremented by the number of times the threshold has been updated.

If configured to be in min register mode the threshold is updated and the counter incremented whenever the X bus is less than the current threshold. Note that when operating in max/min register mode if a positive or negative overflow occurs then the threshold is not updated since this could leave a misleading value in the MMR.

As an overshoot counter the statistics monitor operates by incrementing the OUC every time the value on the X bus exceeds the threshold in the MMR or if a positive overflow occurs. The OUC is unsigned and will not wrap around, thus behaving as a saturating counter. Similarly when configured to be in undershoot counter mode the OUC is incremented every time the value on the X bus is less than the current threshold.

When overflows occur this is recorded in bits 22 and 23 of the MMR. Positive overflows cause bit 22 to be set while negative overflows cause bit 23 to be set. These bits may be cleared by writing to the MMB copy location.

Direct access to the MMR and OUC via the microprocessor interface is not possible. Instead the reading and writing of these registers is performed by making use of the MMB, CMM, OUB and COU registers. Full details may be found in the data sheet.

2.3 Data Conditioning Unit (data transformation unit and data normaliser)

2.3.1. DATA TRANSFORMATION UNIT

The data transformation unit contains a prescaler, an under/over select detector, a look up table and a byte selector. It may be used on its own to provide arbitrary data mappings of an 8 bit segment of the X bus, or in conjunction with the data normaliser to implement sophisticated dynamic range compression functions.

The prescaler allows an 8 bit field to be selected from anywhere within the 22 bits of the X bus. This 8 bit field is used as an address to the LUT. The way in which the address is treated is defined in SCR[6]. If this bit is set to zero then the address is signed and runs from -128 to 127. Alternatively if this bit is set to one then the address is unsigned and runs from 0 to 255. The over/under select detector monitors the operaton of the prescaler to

ensure that all the significant bits and the sign of the X bus are included within the 8 bit field. If this is not the case then an overselect or underselect signal is generated depending on whether the X bus is positive or negative respectively.
The LUT consists of sixty four 32 bit words. In addition there are a further two 32 bit locations known as the upper and lower saturation registers (USR, LSR). The most significant 6 bits of the address field are used to select one of the 32 bit registers in the LUT. This 32 bit output is known as the Y bus. The least significant 2 bits of the address field are then used to control a byte select on the output. Thus the LUT may be used to provide arbitrary 8bit - 8bit data transformations.
Positive overflows on the X bus or overselects in the prescaler cause the LUT to access the USR overriding the address supplied by the prescaler. Similarly negative overflows and underselects cause the LUT to access the LSR. When such conditions occur the byte select control is also overridden thus causing the most significant byte (byte 3) of the appropriate saturation register to appear on the byte wide output of the data transformation unit.
The LUT is programmed via the memory interface. The addressing for the LUT corresponds directly to the 8 bit field, assuming that the byte selector is being used. To enable access to the LUT, USR and LSR from the microprocessor interface the LUT access control bit ACR[1] must be set to zero. This forces the Y bus to zero and causes the normaliser to be controlled by BCR3[7..3] regardless of the setting of the dynamic normalisation bit. Once the LUT has been programmed the LUT access control bit may be reset to one thus allowing the LUT to be used in the data transformation unit.

2.3.2 DATA NORMALISER

The data normaliser contains a shifter followed by a zero data unit. The shifter is capable of right shifts of up to 14 bits and left shifts of up to 2 bits. Any amount of shift outside this range invokes the zero data unit which zeros the output of the data normaliser. The amount of shift is specified by one of two 5 bit sources. These are either BCR3[7..3] or bits 26 to 22 of the Y bus. The source currently selected is determined by the setting of BCR3[2].

2.4 Output Unit (output adder and output multiplexers)

2.4.1 OUTPUT ADDER

The output adder takes one of its inputs from the data normaliser (including the rounding bit). The other input is either the least significant 22 bits of the Y bus or zero depending on the setting of BCR3[1]

2.4.2 OUTPUT MULTIPLEXERS

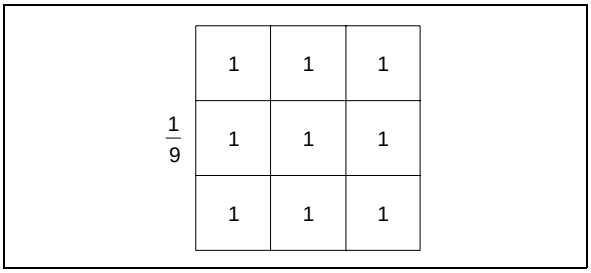
The output multiplexers allow the selected byte from the LUT to be optionally selected to drive either the most or least significant 8 bits of the cascade output pins. This feature is controlled by the setting of BCR2[5..6]. Any cascade output pins not being driven by the selected byte are driven by the appropriate bits of the output adder.

3. USING THE BACKEND OF THE IMS A110s

3.1 Local area averaging

Local averaging is the one of the simplest image filtering operations. A typical local averaging filter may be seen in Figure 2. Although this filter looks very simple to implement on IMSA110s there is one slight problem and that is how to achieve the divide by nine operation. The operation is necessary to ensure that the output image data requires the same number of bits to represent it as the input data.

Figure 2 : Local Averaging Filter Kernel



The IMSA110 is capable of dividing by integer powers of two. Using this capability the $\frac{1}{9}$ could be replaced with $\frac{1}{16}$. Although this would adequately restrict the magnitude of the output data a significant loss of dynamic range could occur. A better solution is to generate an approximation to $\frac{1}{9}$ in the form shown below. Where x represents the coefficient and y the number of righ shift below :

$$\frac{x}{2^y} \approx \frac{1}{9}$$

It may be simply shown that the closest approximation which may be used with IMS A110s is:

$$x = 57$$
$$y = 9$$

By using these values the local averaging kernel to be programmed into the IMSA110 is as shown below:

Figure 3 : Modified Local Averaging Filter Kernel

$\frac{1}{2^9}$

57	57	57
57	57	57
57	57	57

AN548-03.EPS

The division by 2^9 can't be performed by the shifter in the input block since it is only capable of right shifting up to 8 places. The shifter in the normaliser however is capable of right shifting the required nine places.

To configure an IMSA110 so that it performs the local averaging operation used in the above example the following values would have to be programmed into the coefficient and control registers:

Coeff Register	0	1	2	3	4	5	6
CR0a	57	57	57	0	0	0	0
CR0b	57	57	57	0	0	0	0
CR0c	57	57	57	0	0	0	0

Registers	Data msb .. lsb							
SCR	0	x	x	1	1	1	x	0
ACR	0	0	0	0	0	0	x	0
BCR0	x	x	0	0	0	0	0	1
BCR1	0	0	0	0	0	0	x	x
BCR2	0	0	0	x	x	x	x	x
BCR3	0	1	0	0	1	0	0	0

x : indicates don't care.

Exactly the same technique may be applied to other filter kernels which require an awkward division. For example the edge enhancement operation shown in Figure 4 requires a division by 5 operation. A modified version of the kernel which may be easily implemented is shown below.

Figure 4 : Edge Enhancement Filter Kernel

$\frac{1}{5}$

0	- 1	0
- 1	5	- 1
0	- 1	0

AN548-04.EPS

Figure 5 : Modified Edge Enhancement Filter Kernel

$\frac{1}{2^6}$

0	- 13	0
- 13	64	- 13
0	- 13	0

AN548-05.EPS

3.2 Histogram equalization

Histogram equalization is one example of the wider field of histogram modification [1]. All such operations manipulate the grey levels within an image to generate a new image with a modified grey level histogram. The histogram equalization technique attempts to manipulate the grey levels within an image so that an even spread is obtained across the entire range of intensities. Details of the technique are widely available in the technical press [1] so an in depth discussion will not be provided here.

There are two distinct stages in performing a histogram equalization the second of which IMSA110s are capable of performing. The first stage is the calculation of the transfer function which maps the original image onto the histogram equalized image. The main computational cost involved in this stage is the determination of the original histogram. The second stage requires the implementation of the transfer function to map the grey levels in the input image to the equalized grey levels in the output image.

The transfer function is implemented by making use of the arbitrary 8bit-8bit mapping ability of the LUT present within the IMSA110. The offset of each location in the LUT may be regarded as one of the original grey levels and the value programmed into that location is the transformed grey level after equalization.

For example suppose that it was desired to use an IMSA110 to perform a histogram equalization on 8 -bit image data applied to the cascade input port with the MAC coefficients programmed to zero. The table below shows the values which would have to be programmed into the main control registers. The output data would appear on the lower 8 bits of the cascade output port.

Register	Data msb .. lsb							
SCR	0	1	x	1	x	x	x	x
ACR	0	0	0	0	0	0	A	x
BCR0	x	x	x	x	x	x	x	0
BCR1	0	0	0	0	0	0	x	x
BCR2	0	1	0	0	0	0	0	0
BCR3	1	0	0	0	0	0	0	0
LUT n	D	D	D	D	D	D	D	D

x : Indicates don't care.
A : Set to 0 to program LUT, set to 1 to allow IMSA110 LUT access.
D : Program with the mapping n ⇒ D[7..0].

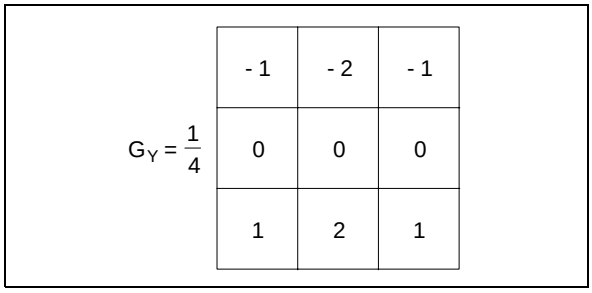
By modifying the transfer function programmed into the LUT many other operations are possible including thresholding and image contouring which are described in sections 3.3 and 3.7 respectively.

3.3 Edge detection and enhancement

3.3.1 EDGE DETECTION

Edge detection is a very important image processing operation since it is often the first stage in feature recognition. For example consider the horizontal edge detector shown in Figure 6. This filter is actually the y component of the Söbel operator. The output (H(x,y)) from the filter when convolved with an image is a measure of the change of intensity in the y direction at each point.

Figure 6 : Y Component of the Söbel Operator



The output at any given point may be positive or negative depending on the direction of the intensity gradient vector at that location. Often when using such a filter to detect vertical edges only the magnitude of the gradient vector is of interest (i.e. its direction is irrelevant). The results may be modified to simply indicate the magnitude by processing the output as shown below.

$$F[x,y]=|H(x,y)|$$

The modulus operation is an ideal example of the use of full wave rectification. The tables below show the configuration of the coefficient and control registers necessary to calculate |H(x,y)|.

Coeff Register	0	1	2	3	4	5	6
CR0a	-1	-2	-1	0	0	0	0
CR0b	0	0	0	0	0	0	0
CR0c	1	2	1	0	0	0	0

Registers	Data msb .. lsb							
SCR	0	x	x	1	0	1	x	0
ACR	0	0	0	0	0	0	x	0
BCR0	1	0	0	0	0	1	0	1
BCR1	0	0	0	0	0	0	x	x
BCR2	0	0	0	x	x	x	x	x
BCR3	0	0	0	0	0	0	0	0

x : Indicates don't care.

Typically once an edge detection operator has been convolved with an image it is necessary to make some sort of decision based on the magnitude as to whether an edge exists at each point of the output. The method usually used is known as thresholding [1].

The threshold operation involves mapping all points with a grey level greater than a given threshold to one value (typically 255), and all other points to another value (typically 0). The lookup table as described in section 3.2 provides the ability to perform just such an arbitrary mapping. By modifying the control registers presented above it is possible to do not only the edge detection operation and the full wave rectification, but also to apply an arbitrary threshold all within a single device. The updated table of control registers is shown below:

Registers	Data msb .. lsb							
SCR	0	1	x	1	0	1	x	0
ACR	0	0	0	0	0	0	A	0
BCR0	1	0	0	0	0	1	0	1
BCR1	0	0	0	0	0	0	x	x
BCR2	0	1	0	0	0	0	0	0
BCR3	1	0	0	0	0	0	0	0
LUT n	D	D	D	D	D	D	D	D

x : Indicates don't care.
A : Set to 0 to program LUT, set to 1 to allow IMS A110 LUT access.
D : Set to 0 for n less than or equal to the threshold, set to 1 otherwise.

3.3.2 EDGE ENHANCEMENT

Edge enhancement is often applied to images to either counteract blurring or to produce a sharper looking image which is sometimes aesthetically more pleasing. One filter kernel which gives an edge enhancement may be seen in Figure 5. When this filter is convolved with an image it is possible to generate not only valid positive image data but also negative values under some circumstances. One solution would be to apply full wave rectifica-

tion to the result however it is generally more acceptable if half wave rectification is applied.

To implement such a filter on an IMSA110 the coefficient and control registers would have to be set up as shown in the following tables.

Coeff Register	0	1	2	3	4	5	6
CR0a	0	-13	0	0	0	0	0
CR0b	-13	64	-13	0	0	0	0
CR0c	0	-13	0	0	0	0	0

Registers	Data msb .. lsb							
SCR	0	x	x	1	0	1	x	0
ACR	0	0	0	0	0	0	0	0
BCR0	0	1	0	0	1	1	0	1
BCR1	0	0	0	0	0	0	x	x
BCR2	0	0	0	x	x	x	x	x
BCR3	0	0	0	0	0	0	0	0

x : Indicates don't care.

3.4 Feature recognition

By using the statistics monitor it is possible to get the IMSA110 to see if a given pattern was present within an image. To enable this process to take place a number of things have to be done:

- The MAC coefficients must be configured as a pattern detector for the pattern which is being searched for. If the pattern is large a number of devices can be cascaded [2] to achieve the required window size.
- The statistics monitor must be configured so that it is in max register mode.
- The MMR must be programmed with -2^{21} at the start of the search period (typically at the start of a frame).

As one or more images are processed the MMR register is continually updated to indicate the highest MAC output which has occured so far. When the pattern detector encounters the pattern that it is designed to search for the MAC output should generate a very large output which exceeds a given threshold. This output will be recorded in the MMR. By examining the MMR at the end of the search period (typically at the end of the frame) it is possible to see if the threshold has been exceeded. If this is the case then it is possible to say that the pattern probably occurred somewhere within the data that was processed. The setting of the threshold to achieve reliable operation requires system teaching using known sets of data.

In a similar fashion it is possible to perform feature recognition with the statistics monitor configured as an overshoot counter. In this mode of operation the detection of the desired pattern is indicated by an increase in the value of the OUC (care must be taken to ensure that it does not saturate). The method of setting the threshold at which the overshoot counter is incremented is identical to the description given in the previous paragraph. At first sight it may appear that this method enables the number of occurences of a given pattern to be counted. Unfortunately this is unlikely to be the case for the following reason.

When the pattern being searched for is encountered it is possible for the OUC to be incremented more than once. This is caused by a combination of uncertainty about the pattern and the properties of pattern detectors as decribed below:

- In a typical pattern matching application the pattern is rarely perfect. Degradations from the ideal may be caused by additive noise, distortion of the object, changing lighting conditions etc. To take this into account the threshold is normally set to a value which is low enough to increment the OUC for all likely occurences of the pattern.
- Due to the nature of pattern detectors a large output is not only generated when the detector is coincident with the pattern but quite large outputs can also be generated when it is just off centre.

The combination of these two problems means that each occurence of the pattern could increment the OUC one or more times thus damaging any indication the change in OUC could give about the number of occurences of a pattern.

3.5 Changing conditions compensation

The front end of many automated image processing systems will experience slowly changing input conditions. These may occur due to changing light levels, drifting component tolerances etc. The inclusion of the max/min register modes of the statistics monitor allows the system to automatically compensate for these changes. For example consider a system which uses daylight to illuminate the field of view. As the day proceeds the output from the camera will change. By spending periods of time monitoring both the maximum and minimum levels in the data stream it is possible to adapt the system to take these changes into account.

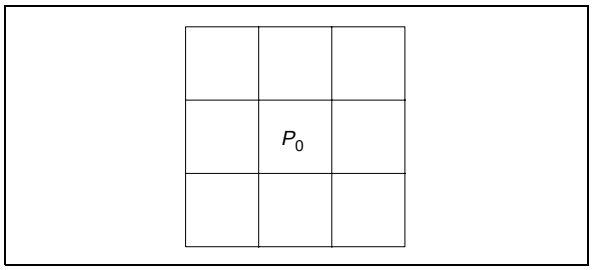
3.6 Binary image procesing

A binary image is one which contains only two grey levels. Typically a binary image is the result of a thresholding operation as described in section 3.3. By making use of the MAC and the backend it is possible to implement a wide variety of different operations some of which are summarised below:

- Isolated pixel removal — removal of all pixels which have no identical neighbour.
- Line linking — bridging of small gaps between pixels.
- Encoding according to connectivity — coding of pixels depending on their connectivity with respect to surrounding pixels.
- Binary thinning including staircase elimination — [3] [4] [5] [6] [7]
- Feature growth — opposite of the above.
- Conway's game of life — the oldest computer game known to man.

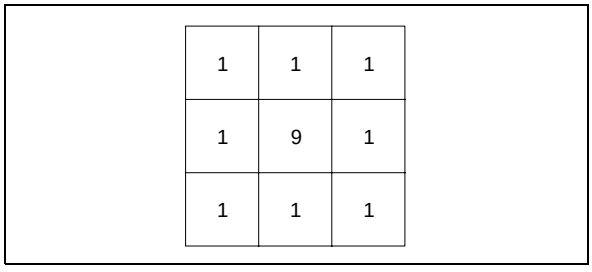
As an example of the techniques involved isolated pixel removal will be examined in more detail. Consider a pixel with its 8 surrounding neighbours as shown in Figure 7. It is assumed that active and inactive pixels are represented by 1 and 0 respectively.

Figure 7 : A Pixel and its 8 closed neighbours



If the central pixel is in the opposite state to all its surrounding neighbours then the value of the central pixel must be toggled. In order to perform the transformation it is necessary to develop a filter kernel which will give a unique output for each of these two condition. One such kernel is shown in Figure 8 below:

Figure 8 : Filter Kernel for Isolated Pixel Removal



By programming the MAC with this kernel the outputs generated when the binary image is applied will range from 0 to 17 inclusive. The two particular cases of special interest are 8 and 9 which correspond to a 0 surrounded by 1s and a 1 surrounded by 0s respectively.

To convert from the output of the MAC to a binary image in the original format use may be made of the LUT. The complete mapping for the LUT and the setting of the main control registers for this example are tabulated below:

Coeff Register	0	1	2	3	4	5	6
CR0a	1	1	1	0	0	0	0
CR0b	1	9	1	0	0	0	0
CR0c	1	1	1	0	0	0	0

Registers	**Data msb .. lsb**							
SCR	0	1	x	1	1	1	x	0
ACR	0	0	0	0	0	0	A	0
BCR0	x	x	0	0	0	0	0	1
BCR1	0	0	0	0	0	0	x	x
BCR2	0	1	0	0	0	0	0	0
BCR3	1	0	0	0	0	0	0	0
LUT 0-7	0	0	0	0	0	0	0	0
LUT 8	0	0	0	0	0	0	0	1
LUT 9	0	0	0	0	0	0	0	0
LUT 10-17	0	0	0	0	0	0	0	1

x : Indicates don't care.
A : Set to 0 to program LUT, set to 1 to allow IMS A110 LUT access.

3.7 Multilevel thresholding - image contouring

Often it is desired to highlight a number of areas within a single image. Providing that each of the areas occupies a different region of the grey scale then this can be achieved by multi level thresholding (sometimes known as image contouring). Typically such a technique is often used in medical work. For example consider an X-Ray taken of a patient which may well contain three very distinct regions:

- Clear regions: representing bone.
- Intermediate regions: representing major body organs.
- Dark regions: representing regions where the X-Rays met little resistance.

By using the LUT to provide arbitrary 8bit-8bit data mappings as descibed in sections 3.2 and 3.3 it is possible to assign each of these three regions a separate value. As a further enhancement external hardware could be used to colour each of the three regions. Such colouring can greatly simplify the comprehension of some types of image.

3.8 Dynamic range compression

Consider image data which requires 12 bits to represent each pixel. If it is desired to display such an image on a system which uses only 8 bits per pixel then some form of range compression is required. One solution is to discard the lower 4 bits of each pixel. This would leave the 8 most significant bits for display. If however, the image was dark the lower 4 bits would contain a large proportion of the image data. To throw away the lower 4 bits in such a situation would almost certainly be unacceptable. A better solution in this case would be to use the nonlinear tranformation shown in Figure 9. Using this transformation values between 0 and 63 are unchanged; values between 64 and 1023 are mapped into the range 64 to 183 and values between 256 and 4095 are mapped into the range 184 to 232.
The IMSA110 is capable of performing just such a nonlinear transformation by making use of both the data transformation unit and the data normaliser. The mode of operation which is required is known as dynamic normalisation, this is selected by setting BCR3[2] (enable dynamic normalisation). In this mode the prescaler selects a 6-bit field anywhere within the X bus. This is used as an address to the LUT. Bits 22 to 26 of the output of the LUT are used to control the normaliser block so that the input to the normaliser is dynamically scaled. The output of the normaliser is then added, in the output adder, to the least significant 22 bits of the output of the LUT.

The operation can be viewed as :

$$output = (input \times scale) + offset$$

where the scale is provided by bits 22 to 26 and the offset is provided by bits 0 to 21 of the LUT.

To define the transformation function shown in Figure 9 it is necessary to carefully calculate the values to be placed in the LUT. The first stage in this calculation is deciding which slice of the X bus the prescaler is going to select. In this example it will be set so that bits 4 through to 11 are selected. This means that bits 6 to 11 are used as the address for the lookup table. Bearing this in mind it may be seen that in the first segment of the transfer function the LUT address is zero. Since in this segment the scale is 1 (0 right shifts) and the offset is 0 the following four bytes of data must be programmed into the first 32 bit location of the LUT.

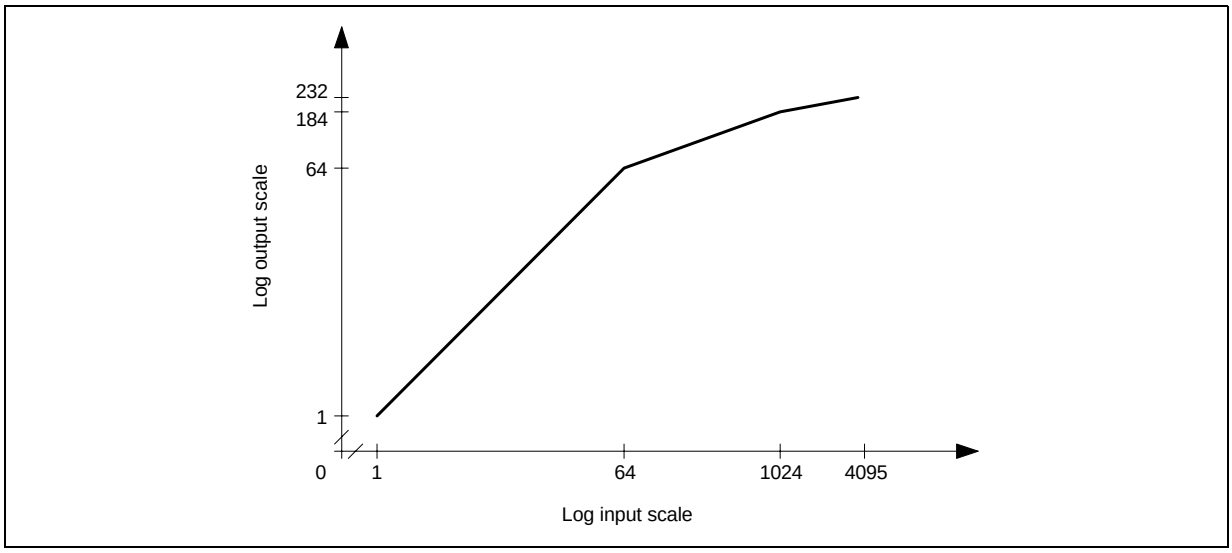
	BYTE 3	BYTE 2	BYTE 1	BYTE 0
LUT 0	00	00	00	00

The second segment of the transfer function occurs between LUT addresses 1 to 15. In this segment the gradient is 1/8 (3 Right shifts). To ensure that the first and second segment line up correctly it is important to set the offset of the second segment to the correct value.

It may be easily shown that in this case the offset is 56. Thus the data to be programmed into the 15 LUT locations from addresses 1 to 15 is:

	BYTE 3	BYTE 2	BYTE 1	BYTE 0
LUT 1	00	C0	00	38
LUT n	00	C0	00	38
LUT 15	00	C0	00	38

Figure 9 : Typical Dynamic Range Compression Function



In exactly the same manner the LUT data for the third and final segment of the transfer function may be shown to be:

	BYTE 3	BYTE 2	BYTE 1	BYTE 0
LUT 16	01	80	00	A8
LUT n	01	80	00	A8
LUT 63	01	80	00	A8

The settings of the other main control registers to perform the example transform on data applied to the cascade input port are:

Coeff Register	0	1	2	3	4	5	6
CR0a	0	0	0	0	0	0	0
CR0b	0	0	0	0	0	0	0
CR0c	0	0	0	0	0	0	0

Registers	Data msb .. lsb							
SCR	0	1	x	1	x	x	x	0
ACR	0	0	0	0	0	0	A	0
BCR0	x	x	x	x	x	x	x	0
BCR1	0	0	0	0	0	0	x	x
BCR2	0	0	0	0	0	1	0	0
BCR3	x	x	x	x	x	1	1	0

x : indicates don't care.
A : Set to 0 to program the LUT, set to 1 to allow IMS A110 LUT access.

4. SUMMARY

This document has attempted to describe by example some of the many ways in which the backend post processor of the IMSA110 may be used. It has only been possible to scratch the surface of a handful of applications but hopefully the examples discussed should have provided an insight into both the flexibility and capability of this section of the device.

5. REFERENCES

[1] R. C. Gonzalez, P.Wintz — Digital Image Processing, Addison Wesley.

[2] R. Whitton — Cascading IMS A110s, INMOS.

[3] R. Stefanelli, A. Rosenfeld — Some Parallel Thinning Algorithms For Digital Pictures. Comm ACM 18, 2.

[4] H.E. Lu, P.S.P. Wang — A Comment On Fast Parallel Algorithms For Thinning Digital Patterns. Comm ACM 29, 3.

[5] C.M. Holt, A. Stewart, M. Clint, R.H. Perrott — An Improved Parallel Thinning Algorithm. Comm ACM 30, 2.

[6] R.W. Hall — Fast Parallel Thinning Algorithms: Parallel Speed And Connectivity Preservation. Comm ACM 32, 1.

[7] Z. Guo, R.W. Hall — Parallel Thinning With Two Subiteration Algorithms. Comm ACM 32, 3.

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.