APPLICATION NOTE

Improved Picture Quality Module MK10

AN 99022

Philips
Semiconductors

PHILIPS

Improved Picture Quality Module MK10

Application Note
AN 99022

Abstract

The Improved Picture Quality (IPQ) module MK10 is an application PC-board designed to evaluate the SAA4978H and demonstrate its features.

The SAA4978H is a video processing IC providing analog and digital interfacing, video enhancing features, memory controlling and an embedded 80C51 µC core. It can be used in a $1f_H$ environment but is especially applicable for $1f_H$ to $2f_H$ scan conversion using one or two 2M9 field memories.

Purchase of Philips I^2C components conveys a license under the Philips I^2C patent to use the components in the I^2C system, provided the system conforms to the I^2C specifications defined by Philips.

APPLICATION NOTE

Improved Picture Quality Module MK10

AN 99022

Authors:

Peter Kelting, Heinrich Waterholter
Systems Laboratory Hamburg,
Germany

Keywords

IPQ
SAA4978H
SAA4955
SAA4956
Time base correction
Scan rate conversion
Sample rate conversion
Noise reduction
Histogram modification

Date: April 20, 1999

Improved Picture Quality Module MK10

Application Note AN 99022

Summary

The SAA4978H is a video processing IC providing analog and digital interfacing, video enhancing features, memory controlling and an embedded 80C51 µC core including RAM and ROM. It is especially applicable for $1f_H$ to $2f_H$ scan conversion using one or two 2M9 field memories, but it can also be used in a conventional 50 Hz or 60 Hz environment. All data busses are 9 bits wide and support the 4:1:1 and 4:2:2 data format.

This application note gives an overview of the functions of the SAA4978H and describes an application board designed to evaluate the IC and demonstrate its features. The current evaluation software gives access to the SAA4978H´s internal registers and supports basic scan conversion in A-A-B-B mode.

CONTENTS

List of Figures

List of Tables

1. Introduction

The MK10 module is a 100 Hz scan converter based on the video processing IC SAA4978H (PICNIC). The module can be run in an analog environment as it supplies A/D converters at the input and D/A converters at the output.

The SAA4978H incorporates the control logic to run the attached field memories in field rate conversion mode (50 Hz to 100 Hz or 60 Hz to 120 Hz), and it also offers a variety of picture improvement functions. An overview is given in the feature list below. More features are added if the MK10 board is extended by the SAA4990H (PROZONIC), SAA4991WP (MELZONIC) or the SAA4992H (FALCONIC) and additional field memories. These ICs are connected to the interface BUS_C / BUS_D. The SAA4978H offers another interface (BUS_A / BUS_B) for additional features in the 50 Hz domain, e. g. PALplus (SAA4996H / SAA4997H). This interface is only partly supported on the MK10 module. There is a connector at BUS_B, so digital data can be input.

A rough application overview is given in fig. 1.

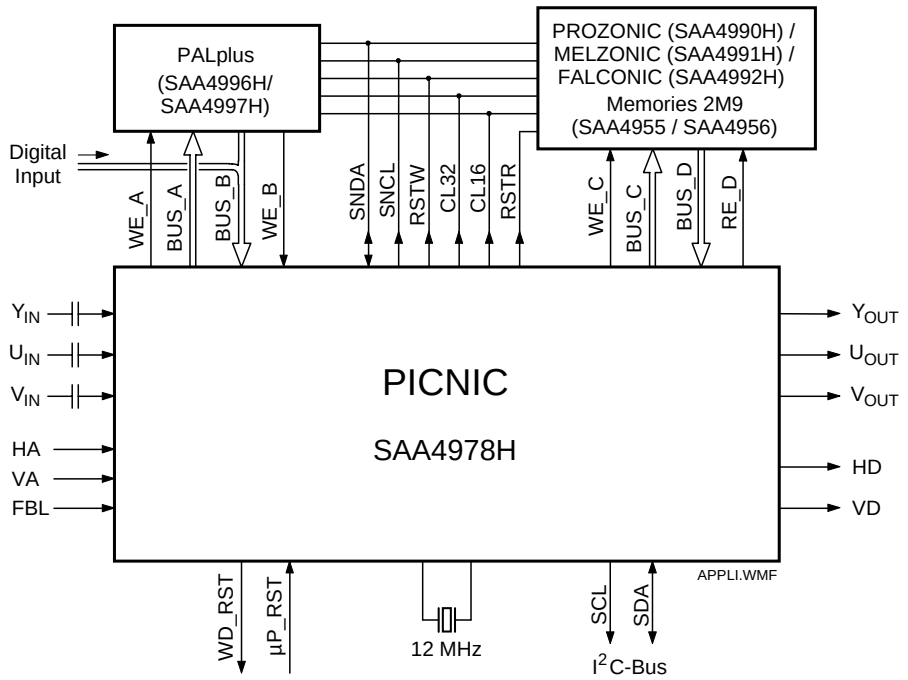


Fig. 1 General application overview of the SAA4978H

2. Features of the MK10 module

In the following the features of the basic MK10 module equipped with a memory for simple A-A-B-B scan conversion are listed.

- Clamping
- Analog AGC
- Triple YUV 9 bit A/D Converter
- Triple YUV analog anti alias filter (can be bypassed)
- 4MHz notch filter
- Non Linear Phase filter after A/D Converter

- 4:1:1 / 4:2:2 digital processing
- 4:1:1 / 4:2:2 selectable I/O interface
- Asynchronous digital input
- Time base correction
- histogram analysis
- histogram modification
- subtitle detection
- blackbar detection
- line-memory based noise reduction (spatial)
- noise level measurement
- clamp noise reduction
- dynamic peaking
- energy measurement
- MPIP decimation
- DPCM data compression and decompression for chrominance
- 2D-peaking and coring
- Non Linear Phase filtering before D/A Conversion
- DCTI
- Triple 10 bit D/A converter
- Triple analog reconstruction filter (can be bypassed)
- Embedded control microprocessor
- Programmable signal positioner
- SNERT interface
- I^2C user control interface
- Boundary Scan Test (BST) supported
- Scan conversion in A-A-B-B mode (field repetition)

3. Functional description of the SAA4978H

The internal functions of the SAA4978H can be divided into a signal processing part and a control part, see fig. 2. Signal processing includes the analog input processing, digital processing at $1f_H$, digital processing at $2f_H$ and analog output processing. The control part comprises the microprocessor around an 80C51 core and a programmable signal positioner generating the controls to run the field memories.

The block 'Digital Processing at $1f_H$' is connected to the digital input/output YUV_A/YUV_B which allows to connect additional picture processing ICs in the $1f_H$ domain, like the PALplus ICs SAA4996H / SAA4997H for example. However, the input YUV_B can also be used to input a digital video signal from external sources, e. g. a data stream from an MPEG decoder.

The data output YUV_C and input YUV_D represent the interface between the $1f_H$ and $2f_H$ domain. In the simplest case only one memory is connected here for scan conversion. Reading to the memory occurs at double the speed than writing. For better 100 Hz quality and extra features picture improvement ICs like the SAA4990H (PROZONIC), SAA4991H (MELZONIC) or SAA4992H (FALCONIC) can be added.

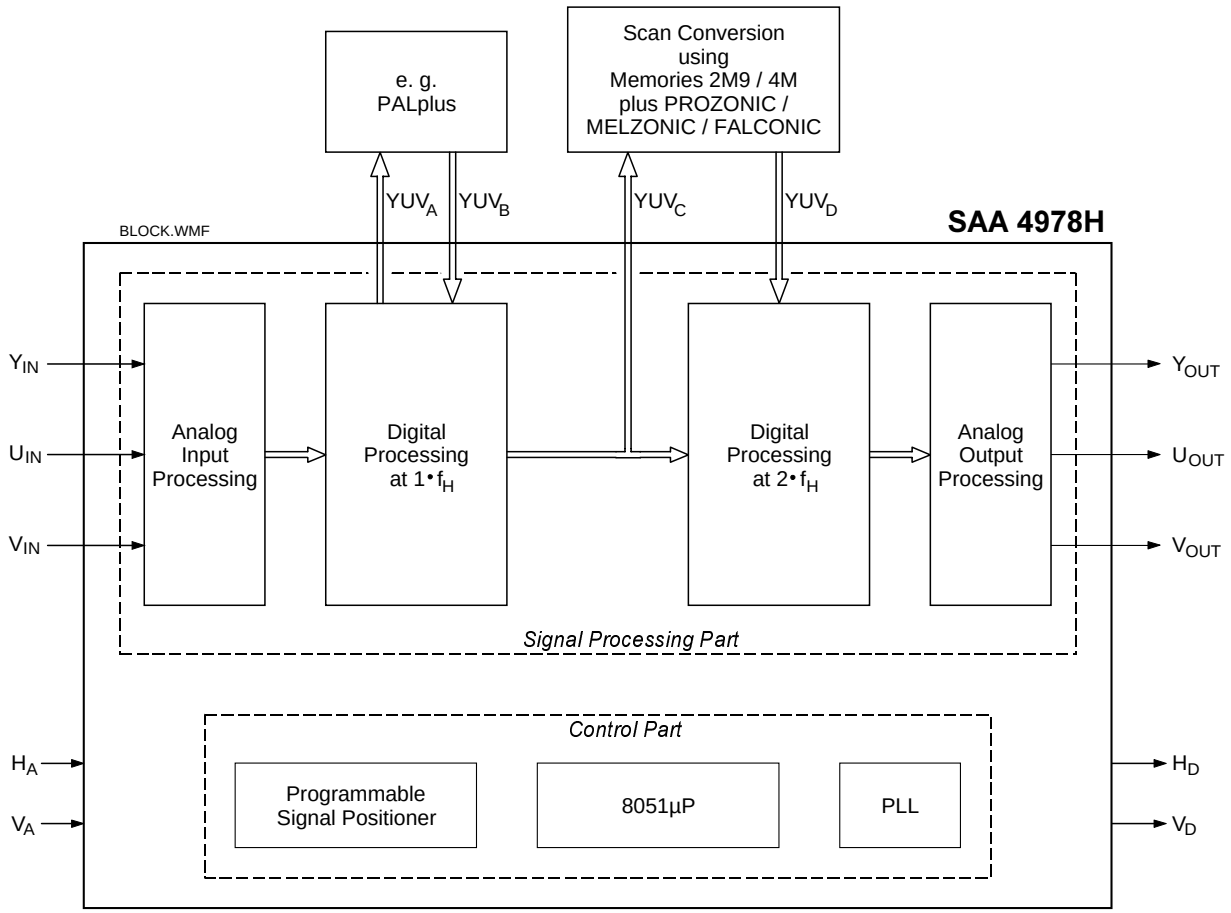


Fig. 2 Block diagram of the SAA4978H

3.1 Analog input processing

3.1.1 Clamping

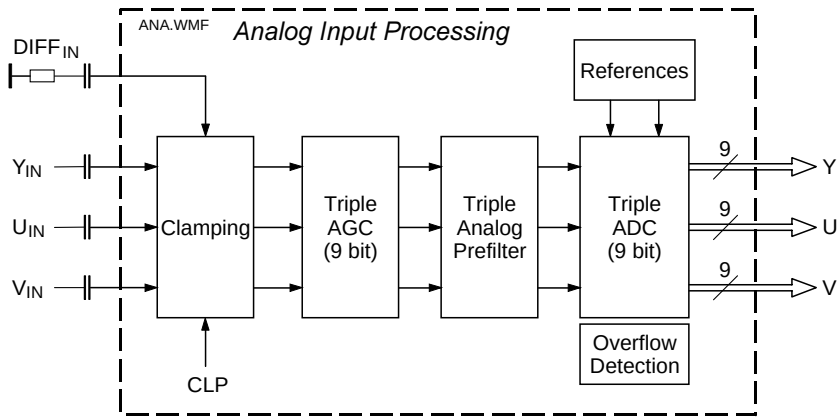


Fig. 3 Analog input processing

The analog signals Y, U and V are supplied to the IC via capacitors. At each input a clamping circuit is used to adjust the DC level such that the luminance A/D converter generates value 32 at black level (at a total signal

range of 9 bits) and the chrominance A/D converters output center level (value 0 in 2´s complement data format) during sync backporch. The clamping pulse is generated on-chip by the block PSP (Programmable Signal Positioner), its start and stop position can be adjusted by the parameters *CLAMP_START* and *CLAMP_STOP*.

In order to improve common mode rejection it is recommended to connect at the $DIFF_{IN}$ input to ground the same source impedance as used at the Y_{IN} input.

3.1.2 Gain control

For each signal a variable amplifier is used to adjust the signal amplitude to the A/D converter range. For nominal input amplitudes of

Y_{IN} = 1.00 V_{PP}
U_{IN} = 1.33 V_{PP}
V_{IN} = 1.05 V_{PP}

the amplifier gain should be set to 0 dB. For larger or smaller signals the gain setting can be adjusted from -3 dB to +6 dB. The adjustment is controlled by data words from the µC which are converted to an analog control signal by an 8 bit D/A converter. 8 bits give a step size of 0.4%/step which is hardly visible in the picture. The parameters to control gain are *AGC_GAIN_Y*, *AGC_GAIN_U* and *AGC_GAIN_V*.

3.1.3 Analog prefilter

The SAA4978H does not require external anti-aliasing filters at its analog inputs. Third order linear phase filters are supplied at each of the Y, U and V channels. These filters have a notch at the clock frequency (16 MHz) to prevent aliasing of low frequencies. The 3-dB-bandwidth is 5.6 MHz. The filters can be bypassed if external filtering with other characteristics is desired.

3.1.4 A/D converters and overflow detection

Three identical 9 bit A/D converters are used to convert Y, U and V at a data rate of 16 MHz. The converters have a two bit overflow and a one bit underflow detection for each channel. So there are three ranges of overflow and one indication of underflow. Underflow indication is used for U and V only.

overflow = 00:	signal in linear range
overflow = 01:	overflow range 0 .. 1 dB
overflow = 10:	overflow range 1 .. 2 dB
overflow = 11:	overflow range 2 .. 3 dB

Only one A/D converter can be overflow-tested at a time, this is selected by the register *YUV_SELECT*.

YUV_SELECT = 00:	select overflow of A/D converter Y
YUV_SELECT = 01:	select overflow of A/D converter U
YUV_SELECT = 10:	select overflow of A/D converter V
YUV_SELECT = 11:	select underflow

The occurrences of the over/underflow events are counted for one field (or a definable window within a field) and stored in 2 8-bit registers for each overflow range.

register:	*overflow detection:*	*underflow detection:*
OVERFLOW_11_HIGH	level 11 high byte	- not defined -
OVERFLOW_11_LOW	level 11 low byte	- not defined -
OVERFLOW_10_HIGH	level 10 high byte	underflow U high byte
OVERFLOW_10_LOW	level 10 low byte	underflow U low byte
OVERFLOW_01_HIGH	level 01 high byte	underflow V high byte
OVERFLOW_01_LOW	level 01 low byte	underflow V low byte

These registers can be read by the µC and used for automatic gain control.

3.2 Digital processing at $1f_H$ level

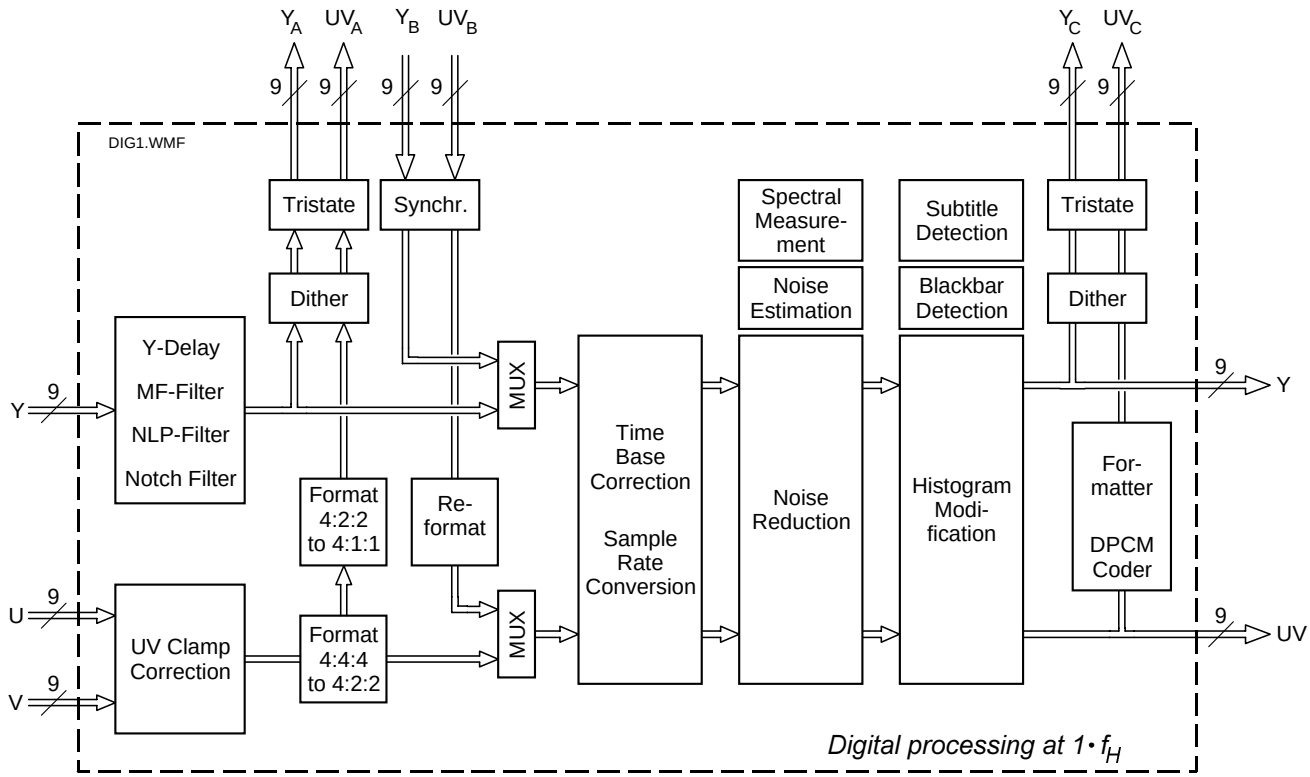


Fig. 4 Digital signal processing in the $1f_H$ domain

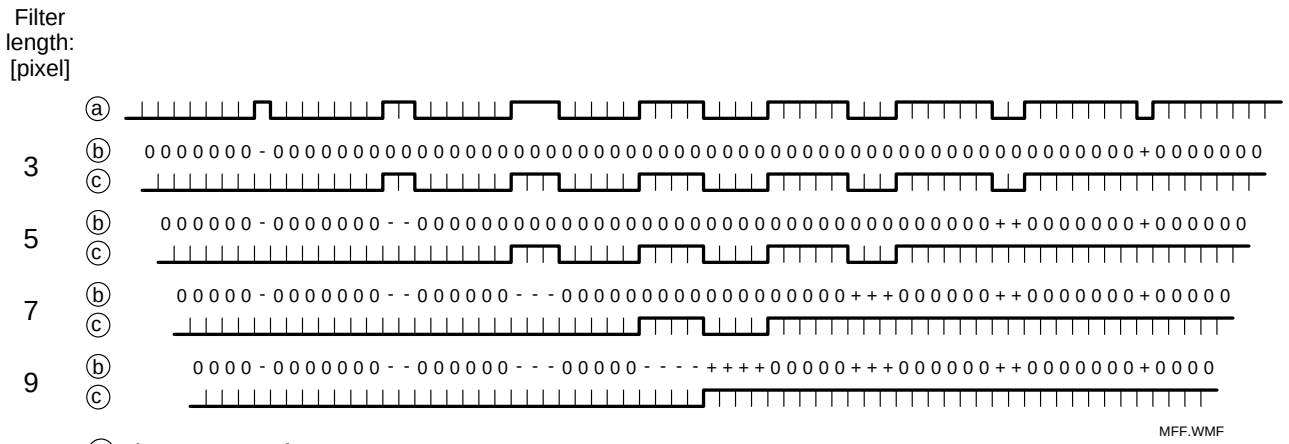
3.2.1 Luminance delay

The luminance data Y from the A/D converter can be delayed by -1 .. +2 positions with respect to the U and V samples (parameter *Y_DELAY*). This delay allows to compensate for a possible delay difference in front of the A/D converters. The delay setting of zero is meant for the nominal case of aligned input data according to the interface format standard.

3.2.2 Majority following filter

The majority following filter (MFF) reduces possible multi-step trip level noise generated by the A/D converter. When applied to a peaking circuit this noise can cause disturbing quantization effects in smooth pictures like faces and slowly changing backgrounds.

For each pixel this filter compares the neighboring pixels to +1 or -1 LSB difference. If the majority of these differences is +1 then 1 is added to the actual pixel. If the majority is -1 then 1 is subtracted from the actual pixel. The number of pixels included in the filter can be selected between 1 (bypass), 3, 5, 7 and 9 (parameter *MFF_WIDTH*). Transients, ramps and all signal changes of 2 or more LSBs are not affected. Fig. 5 shows the function of filter. The input at a) is a slowly changing signal (within 1 LSB), b) shows the decision of the filter after evaluating the neighboring pixels, and c) shows the output of the filter. With a filter length of 3 only single deviating pixels are straightened out. If the filter length is increased, up to 4 diverging pixels in a row (at filter length 9) are changed.



(a) input pattern of MFF

(b) result of majority evaluation:
0 . . . don't change input value
\+ . . . add 1 to input value
\- . . . subtract 1 from input value

(c) output pattern of MFF

Fig. 5 Transient noise suppression by a majority following filter

3.2.3 Nonlinear phase filter A/D

The nonlinear phase filter A/D (NLP-Filter) is designed to compensate for nonlinear filtering and bandwidth loss in front of the A/D converter. The filter can be adjusted by two parameters: λ defines the highpass amplitude, and μ determines the overshoot behavior. Settings are provided for λ = 0, 1/16, 2/16 and 3/16 (parameter *NLP_L_AD*) and μ = 0, 1/4 and 1/2 (parameter *NLP_U_AD*).

In fig. 6 the transfer and group delay curves are given for the different combination of λ and μ. In each plot the upper group of curves represent the group delay, the lower group of curves give the amplitude response. The left three plots show the behavior of the digital filter itself, the plots on the right side show the superposition of the digital filter and the analog prefilter.

3.2.4 Notch filter

This filter provides a zero at 4 MHz (1/4 of the sample frequency of 16 MHz). The -3dB bandwidth is from 3 MHz to 5 MHz with a slight peaking of 0.8 dB at 1.6 and 6.4 MHz. The filter gives an extra suppression at the color subcarrier frequency. It is also very useful as a downsampling prefilter in multi-PIP mode where the zero at 4 MHz prevents alias. The transfer curve is shown in fig. 7. The notch filter can be bypassed without influencing the group delay.

3.2.5 UV clamp correction

During the time when the analog clamp circuit is active the output of the U and V A/D-converters is expected to be zero (in 9 bit signed data format). Any error or static offset here effects the respective signal level and could result in a color shift or hue error. The clamp correction circuit reduces or eliminates this error.

During the active clamping time 32 samples are measured and the error is accumulated and lowpass filtered. Then a vertical recursive filter is used to further lowpass filter the error value. This value can be read by the microprocessor or directly used to correct the clamping error by subtracting it from the U and V data respectively.

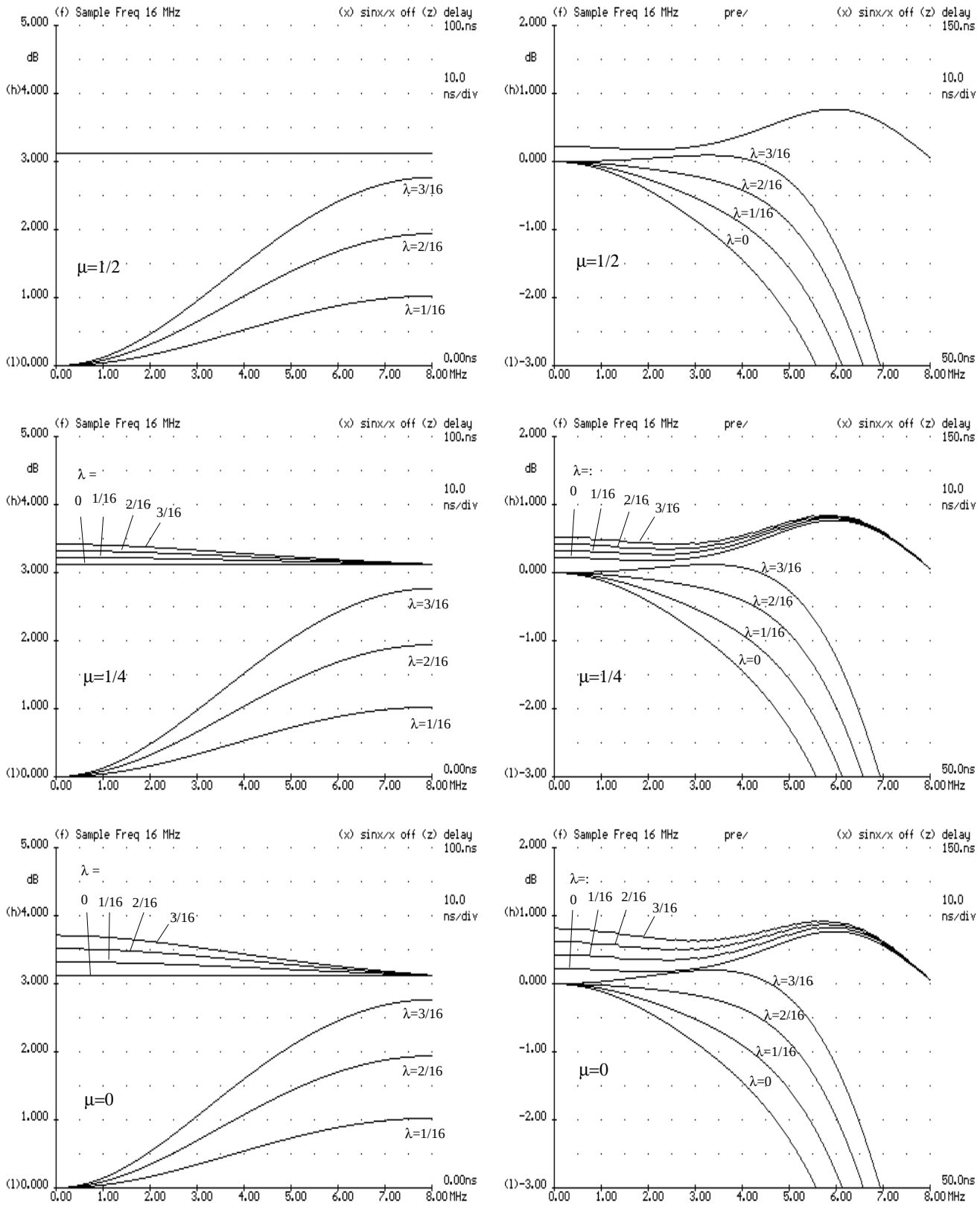


Fig. 6 Group delay and transfer curves of the NLP A/D filter

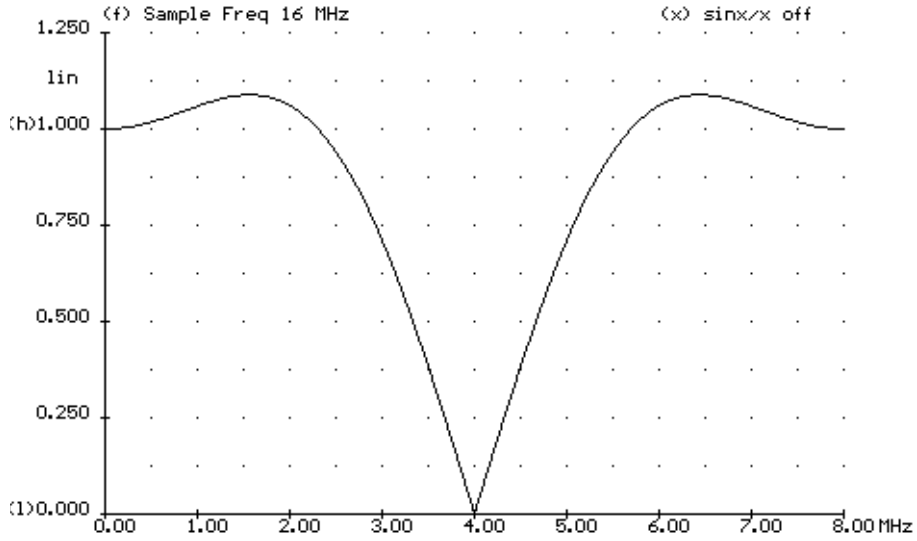


Fig. 7 4 MHz notch filter

The time constant of the vertical filter can be adjusted between 4, 9, 19 and 39 lines by the parameter *UV_TAU*. *UV_COR_MODE* selects between internal and external control of clamp correction. In external mode the fixed values supplied by the parameters *U_CLAMP_COR_FVAL* and *V_CLAMP_COR_FVAL* are used.

3.2.6 Formatting

The color difference signals U and V are sampled at 16 MHz, the same frequency as the luminance signal. This data format of 4:4:4 at the output of the A/D converters is adapted to the bandwidth of the U and V signals and converted to the 4:2:2 or 4:1:1 format for further processing.

In the process of downsampling to 4:2:2 a first lowpass filter is used to limit the bandwidth (-3dB at 3.5 MHz). If further downsampling to 4:1:1 is selected a second lowpass filter reduces the bandwidth to 1.75 MHz. In the 4:2:2 format a slight coring can be selected to reduce offsets around zero. The curves are shown in fig. 8. The amount of coring must be seen in relation to the local wordlength of 11 bits signed. So based on the 9 bit signed output data width the selectable coring settings by the parameter *UV_CORING* are: coring off / 0.5 LSB / 1 LSB / 2 LSB.

3.2.7 Dithering

Before data is output at BUS_A (Y_A and UV_A) the internal datawidth of 11 bits is reduced to 9 or 8 bits. 8 bit is selected if the attached processing system only supports 8 bits (like the current PALplus ICs). In case of 9 bit the data is rounded, in case of 8 bits rounding or dithered rounding can be selected. Dithering is a way of maintaining some of the information contained in the lost bits by high-frequency modulation of the bits to be truncated. Before truncation 0.25 LSB (relative to 8 bit) is added every odd output sample and 0.75 LSB every even sample. In this way on the average 0.5 LSB is added and correct rounding is done (no DC shift). For low frequency signals this gives a resolution increase by a factor of 2. The phase of dithering can be switched by 180^o from line to line, field to field or frame to frame, in order to decrease the visibility of the dithering pattern. Parameter *UV_BUS_A_8BIT_ROUND* selects 8 or 9 bit and rounding or dithering, parameter *UV_BUS_A_DITHER* determines the sequence of dithering, see table 1.

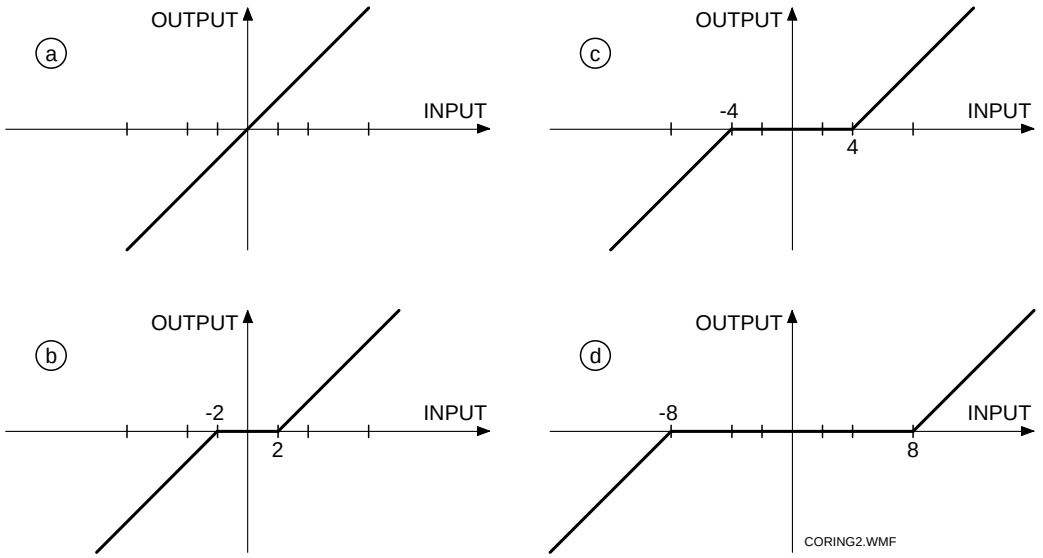


Fig. 8 UV coring

UV_BUS_A_ 8BIT_ROUND	Output mode		UV_BUS_A_ DITHER	Dither sequence
0 0	9 bit rounded		0 x 0	dither sequence F1 L1
0 1	9 bit rounded		0 x 1	dither sequence F1 L2
1 0	8 bit **dithered**	see dither sequence ⇒	1 0 0	dither sequence F2 L1
1 1	8 bit rounded		1 0 1	dither sequence F2 L2
			1 1 0	dither sequence F4 L1
			1 1 1	dither sequence F4 L2

	Dither sequence
L1	all lines the same dither phase
L2	dither phase changes every line
F1	all fields the same dither phase
F2	dither phase changes every field
F4	dither phase changes every two fields

Table 1: Output modes of BUS_A and dither sequence

3.2.8 BUS_A output

The data enable signal WE_A determines the phase of the sequence in 4:1:1 or 4:2:2 data format. If WE_A is in *qualifier* mode the first data word after its rising edge determines the start of the sequence, in *prequalifier* mode

the second data word is the start of the sequence. Selecting the appropriate mode is done by the parameter *WE_A_QUALIFIER*.

BUS_A supports (like all other data I/O data busses of the SAA4978H) 9 bits for luminance and chrominance. Depending on the mode part of the data bits can be turned inactive, however, or all of BUS_A can be turned to tristate mode. Control of the output mode is done by the parameters *FORCE_BUS_A_TRI*, *Y_BUS_A_8BIT_ROUND*, *UV_BUS_A_8BIT_ROUND* and *SEL_422_OUT*, see the following table 2.

FORCE_ BUS_A_ TRI	Y_ BUS_A_ 8BIT_ ROUND	UV_ BUS_A_ 8BIT_ ROUND	SEL_422_ OUT	YA [8:1]	YA [0]	UVA [8:5]	UVA [4:1]	UVA [0]	WE_A
1	x	x	x	tri	tri	tri	tri	tri	tri
0	0	0	0	active	active	active	tri	active	active
0	0	0	1	active	active	active	active	active	active
0	0	1	0	active	active	active	tri	tri	active
0	0	1	1	active	active	active	active	tri	active
0	1	0	0	active	tri	active	tri	active	active
0	1	0	1	active	tri	active	active	active	active
0	1	1	0	active	tri	active	tri	tri	active
0	1	1	1	active	tri	active	active	tri	active

tri . . . tristate

Table 2: BUS_A tristate modes

3.2.9 BUS_B input

At the inputs YB and UVB external digital video data can be supplied to the SAA4978H. These data can be processed data from output BUS_A, for example by a PALplus decoder, but they can also come from any other source. BUS_B can handle any of the following data formats:

4 : 1 : 1 serial
4 : 2 : 2 parallel
4 : 2 : 2 double clock

By the parameter *SEL_INPUT_FORMAT* one of the above formats or the internal input can be selected. The internal format is always 4:2:2 and 9 bit data width. Any external data is reformatted to this internal format. In case of 8 bit wide input data the LSB should be connected to a fixed logic level. A complete list of all possible BUS_B input modes is given in the following table 3, table 4 gives the data format.

The external data sources must provide a write enable signal WE_B to define the data frame of the 4:1:1 and 4:2:2 signal format. The first resp. second data word after the rising edge of WE_B then determines the start of the data frame, depending upon whether qualifier or prequalifier mode of WE_B is selected (parameter *WE_B_QUALIFIER*).

The input data can be either synchronous to the system clock or asynchronous. In case of asynchronous data the clock has to be input at pin CLK_AS. This clock may not be faster than the internal system clock of 32 MHz. Asynchronous mode is activated by parameter *SEL_ASYNCHRONOUS*.

Adaptation to different input formats can be made by the parameters *INV656* and *UV_INV*. *INV656* inverts the MSB of the UV data. This inversion is used for 656 format input data of which the UV component is unsigned. *UV_INV* inverts the UV data, so any polarity can be handle.

External / Internal	4:2:2 / 4:1:1	9 bit / 8 bit	Synchronous / Asynchronous	Double clock / Normal clock
internal	4:2:2	9 bit	synchronous	normal clock
external	4:1:1	8 bit	synchronous	normal clock
external	4:1:1	9 bit	synchronous	normal clock
external	4:1:1	8 bit	asynchronous	normal clock
external	4:1:1	9 bit	asynchronous	normal clock
external	4:2:2	9 bit	synchronous	normal clock
external	4:2:2	8 bit	synchronous	normal clock
external	4:2:2	9 bit	asynchronous	normal clock
external	4:2:2	8 bit	asynchronous	normal clock
external	4:2:2	9 bit	synchronous	double clock
external	4:2:2	8 bit	synchronous	double clock
external	4:2:2	9 bit	asynchronous	double clock
external	4:2:2	8 bit	asynchronous	double clock

Table 3: Y and UV input modes of BUS_B

I/O Pin	4 : 1 : 1 Format				4 : 2 : 2 Format		4 : 2 : 2 Format Double Clock				4 : 2 : 2 DPCM Format	
Y*x*8	Y07	Y17	Y27	Y37	Y07	Y17	U07	Y07	V07	Y17	Y07	Y17
Y*x*7	Y06	Y16	Y26	Y36	Y06	Y16	U06	Y06	V06	Y16	Y06	Y16
Y*x*6	Y05	Y15	Y25	Y35	Y05	Y15	U05	Y05	V05	Y15	Y05	Y15
Y*x*5	Y04	Y14	Y24	Y34	Y04	Y14	U04	Y04	V04	Y14	Y04	Y14
Y*x*4	Y03	Y13	Y23	Y33	Y03	Y13	U03	Y03	V03	Y13	Y03	Y13
Y*x*3	Y02	Y12	Y22	Y32	Y02	Y12	U02	Y02	V02	Y12	Y02	Y12
Y*x*2	Y01	Y11	Y21	Y31	Y01	Y11	U01	Y01	V01	Y11	Y01	Y11
Y*x*1	Y00	Y10	Y20	Y30	Y00	Y10	U00	Y00	V00	Y10	Y00	Y10
Y*x*0	Y0L	Y1L	Y2L	Y3L	Y0L	Y1L	U0L	Y0L	V0L	Y1L	Y0L	Y1L
UV*x*8	U07	U05	U03	U01	U07	V07	-	-	-	-	UC03	VC03
UV*x*7	U06	U04	U02	U00	U06	V06	-	-	-	-	UC02	VC02
UV*x*6	V07	V05	V03	V01	U05	V05	-	-	-	-	UC01	VC01
UV*x*5	V06	V04	V02	V00	U04	V04	-	-	-	-	UC00	VC00
UV*x*4	-	-	-	-	U03	V03	-	-	-	-	-	-
UV*x*3	-	-	-	-	U02	V02	-	-	-	-	-	-
UV*x*2	-	-	-	-	U01	V01	-	-	-	-	-	-
UV*x*1	-	-	-	-	U00	V00	-	-	-	-	-	-
UV*x*0	U0L	-	V0L	-	U0L	V0L	-	-	-	-	-	-

Table 4: Y and UV data formats

3.2.10 Time base correction / Sample rate conversion

The block TBC provides a delay for the incoming data and/or changes its sample rate in order to achieve the functions below. Fig. 9 shows examples in the use of the TBC unit.

- Horizontal shift:
  - static, for horizontal displacement of the visible picture

    Horizontal shift is done by simply delaying or advancing the incoming lines. The range for this shift is from -1/2 line to +1/2 line (plus the nominal delay of 1/2 line). It is user accessible by the parameters *H_SHIFT_LOW* and *H_SHIFT_HIGH*.
  - varying from line to line, for time base correction

    Time base correction is a horizontal shift that changes from line to line, but is constant within a line. The amount of required shift is taken from the PLL and is equal to the measured phase error. To this varying horizontal shift a static shift (see above) can be added.
- Compression or expansion:
  - static, for sample rate conversion

    For input data running on an external clock the TBC block is used to convert the data to the internal system clock (CL16). The external clock must have a frequency less than, or equal to, the instantaneous frequency of CL16. Synchronisation of the input samples with the system clock is done in a FIFO. The output sample rate of the TBC is equal to CL16.
  - static, for horizontal zoom

    With a zero order variation of the delay a linear compress or expand function is obtained. The range for the compression factor is 0 to 2, meaning infinite zoom to a compression factor of 2. The amount of compression or expansion is determined by the parameter *C0* (zero order control). In fig. 9 a the shaded area is equal to the total mount of input samples converted to output samples.
  - dynamic during a line, for panorama mode or its inverse (amaronap)

    With a second order variation of the delay a parabolic compression or expansion is obtained. This means that the lines are geometrically expanded at the sides and slightly compressed at the centre (see fig. 8b). This mode is especially useful for display of 4:3 pictures with full width on a 16:9 screen, whereby the geometry in the centre is more or less correct (see fig. 10).

    The amplitude of the parabolic modulation is determined by the parameter *C2* (second order control) whereas the compression factor at the centre of the line is controlled by *C0*.

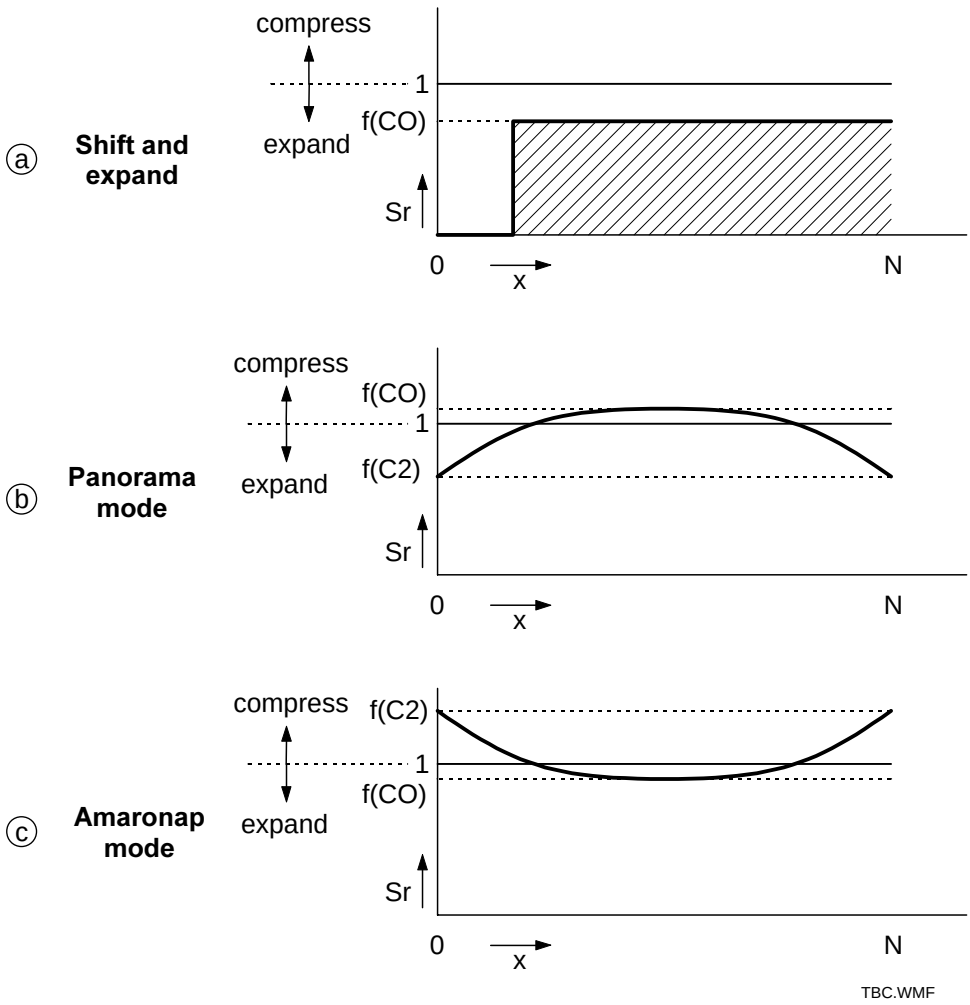
    The amaronap mode (see fig. 9 c) is the inverse of the panorama mode. It can be used for full width display of 16:9 pictures on a 4:3 screen.

3.2.11 Noise Reduction

Noise reduction is done only in the luminance signal Y. The chrominance signals U and V are bypassed and provided with a delay equivalent to the processing delay of the luminance signal.

The noise reduction block consists of the following subblocks, see fig. 11.

- Clamp noise reduction — this is a temporal filter for clamp noise, it adapts automatically to the clamp noise
- Band split — this filter separates the high frequency and low frequency parts in the video signal
- Noise estimation — in this block the noise level of the video signal is measured



$$Sr(x) = \frac{fs_{input}}{fs_{output}}$$

fs_{input} ... input sample frequency

fs_{output} ... output sample frequency

Fig. 9 Examples of TBC control

- Spatial noise reduction — this filter is for Gaussian noise, it automatically adapts to the noise level
- 2D-peaking & coring — here peaking and coring is done on the HF part of the video signal
- VHF energy — here the energy in the HF part of the video signal is measured

Clamp noise reduction

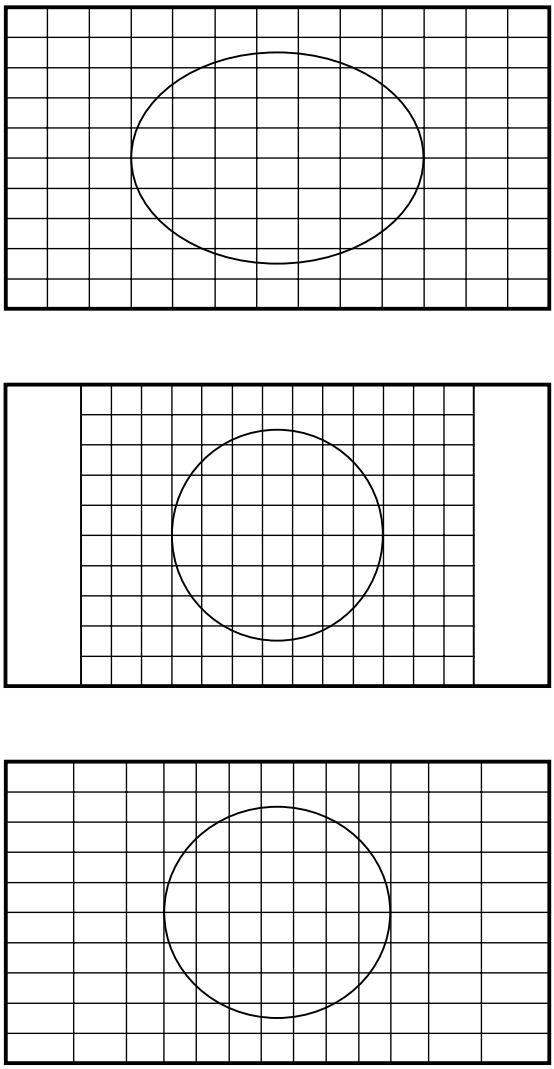
Clamp noise reduction is used to correct for clamping errors within the video signal which cannot be removed by conventional clamping on the back porch of the video signal. Detecting clamping errors is done by an adaptive temporal recursive filter. Fig. 12 shows the block diagram.

Each line is divided into 7 segments with the pixel values for each segment being accumulated and averaged. This average is compared to the average stored in the memory for this segment, the difference then is subjected to a multiplication by K. The K-factor depends on the difference (DIF) and can further be influenced by the param-

4:3 picture on a 16:9 screen

4:3 picture on a 16:9 screen, compressed

4:3 picture on a 16:9 screen, in panorama mode



PANORAM.WMF

Fig. 10 Nonlinear compression/expansion in panorama mode

eters *K_ONE* and *K_SCALE*. *K_ONE* = 1 will force K to its max. value. *K_SCALE* is a scaling factor and can be set to any of 8 values: 4, 2, 1, 1/2, 1/4, 1/8, 1/16, 1/32. K influences the recursion:

$M_{IN} = K \bullet A + (1 - K) \bullet F$

M_{IN} input to memory
A actual average of segment
F stored average of segment from previous field

Low values of K will preserve the memory contents while high values of K get the memory contents updated quickly. Due to its dependency on DIF the K-value can vary with each segment and each line. The amount of updating can be limited by the parameter *CLINIC_MAX_DIFF*.

As fig. 12 shows, the final correction value COR to modify the luminance data is generated from the actual segment average A and the vertically filtered average of M_{IN} and the memory data from the previous line. In order to get a constant COR value for each line, the SORT block picks out representative segments whose value is taken to generate the correction value which then is added to each pixel of the line. By turning the parameter *CLINIC_OFF* to 1 the correction value can be set to zero and thus the clamp noise reduction be disabled. In the

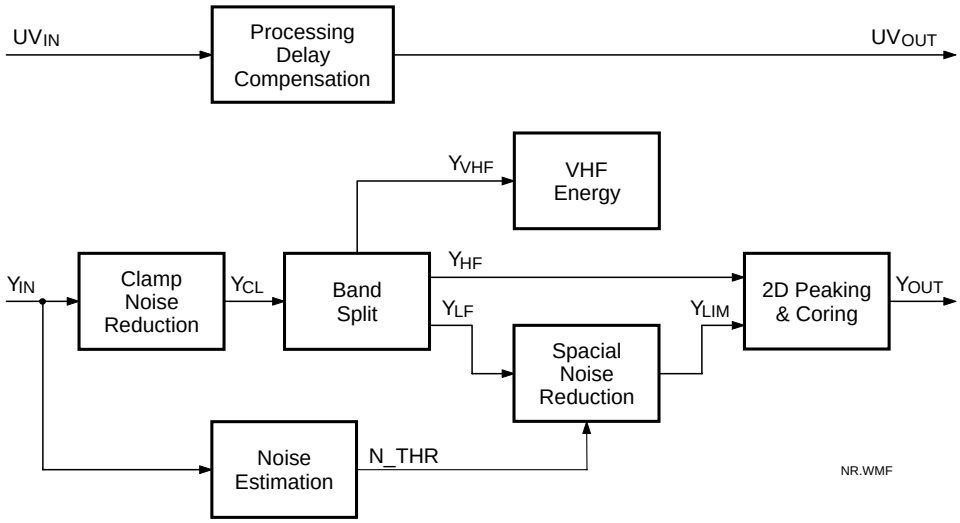


Fig. 11 Noise reduction block diagram

block LIMIT the word size is reduced from the internal size to the output width of 9 bit, to reduce rounding errors the dithering function can be enabled by the parameter *DITHER* = 1.

Once per line the value DIF of representative segments is compared to a threshold, and every time this threshold is exceeded a counter is incremented. The counter value *NBR_EVENTS* can be read by the microprocessor. It is an indication of motion in the picture.

The absolute correction values are accumulated for the complete field. This sum needs a 19 bit register which can be read by the microprocessor. It is divided into 3 8-bit values: *TOT_COR_H*, *TOT_COR_M*, *TOT_COR_L.*

Band split

The band splitting filter (see fig. 13) separates the luminance signal into a low pass and a high pass part. While the high pass signal Y_{HF} is fed to the peaking and coring block, the low pass signal Y_{LF} is processed in the spatial noise reduction circuit.

The 2D lowpass filter is a [1 2 1] filter in both the horizontal and vertical direction. 2D-highpass filtering is done by taking the centre tap and subtracting the 2D lowpass output (/16) from it. Also added to the 2D-highpass signal is the vertical highpass data ($Y_{(H+V)HF}$). It has an extra gain adjustment to enable vertical peaking. The 3-bit parameter *V_GAINSTR* can be set from 0 to 7/8 in steps of 1/8.

2-D peaking and coring

The 2D-highpass data Y_{HF} from the bandsplit filter is multiplied by the factor 0/4, 2/4, 3/4, .. 8/4 (user selectable by the parameter *2D_PEAK_COEFF*) and cored before added to the 2D lowpass branch for the 2D peaking function. The high frequency signal Y_{HF} is bypassing both the low frequency temporal and the spatial noise reduction, therefore sharpness in the high frequencies is not reduced by the noise reduction parts. The factor 0 on the Y_{HF} signal yields a pure 2D lowpassed signal on the output. Multi Picture In Picture (multi-PIP) with pure subsampling of this signal yields a much better result than without the lowpass operation.

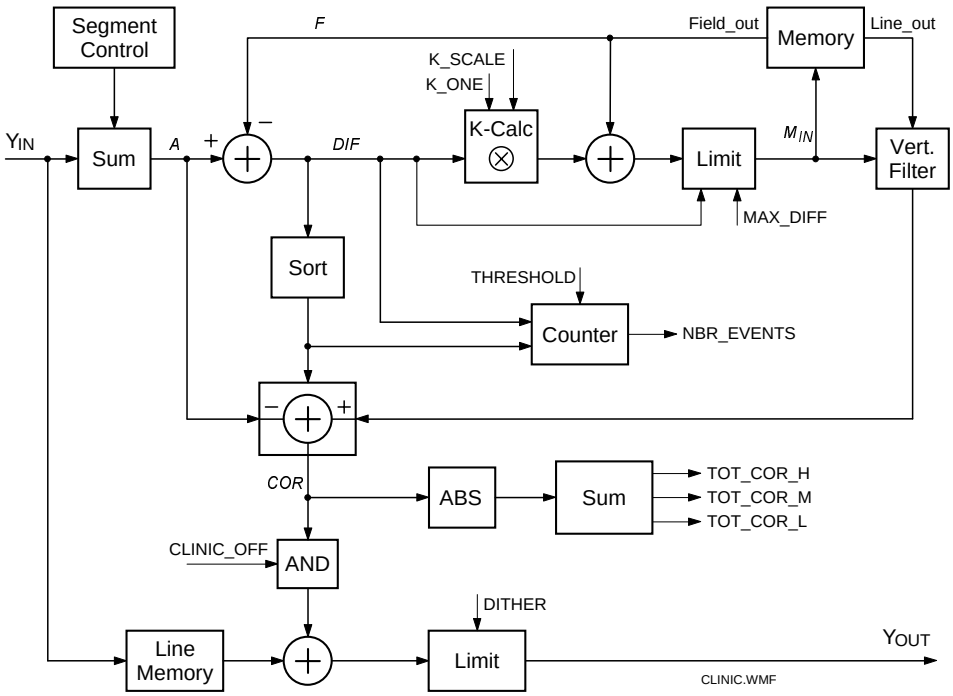


Fig. 12 Block diagram of clamp noise reduction

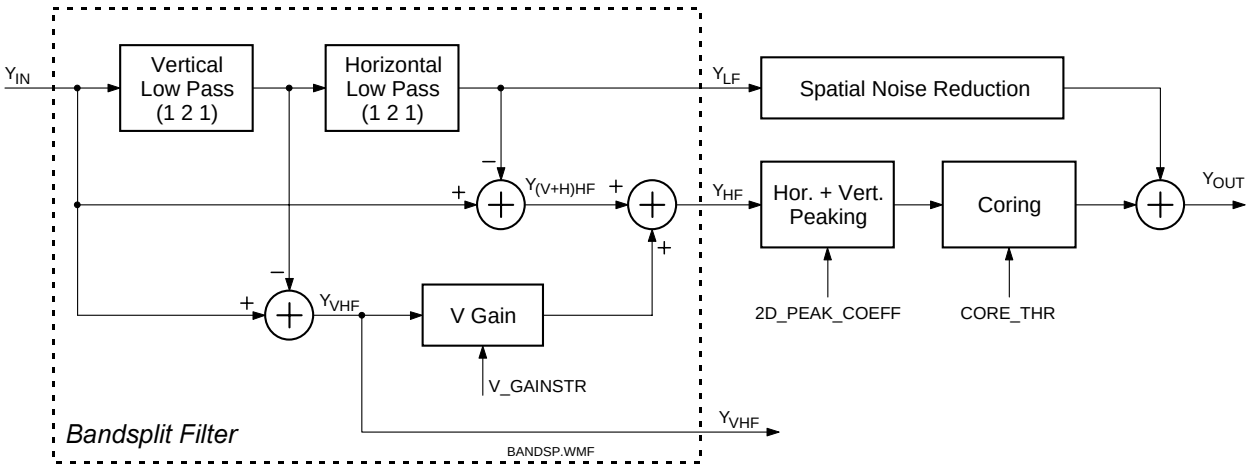


Fig. 13 Bandsplit filter, 2D-peaking and coring

The coring function compares the absolute value of the input signal to the 4-bit control parameter *CORE_THR*. All values which are less than or equal to *CORE_THR* are set to zero. The other values are shifted up (for negative values) or down (for positive values) by the amount of *CORE_THR*, see fig. 14.

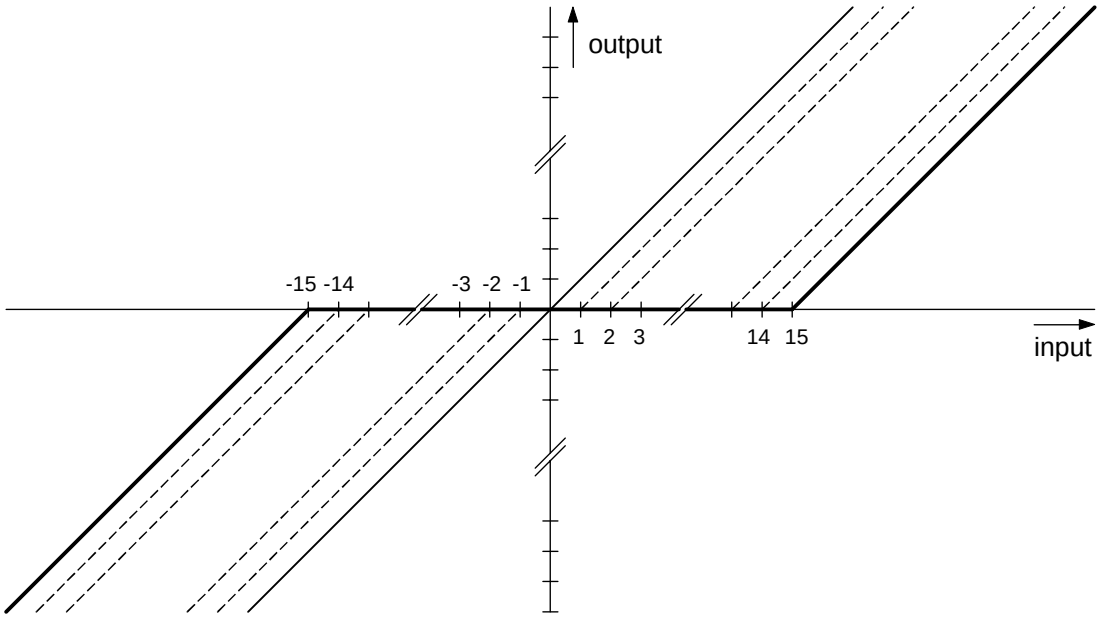


Fig. 14 2D coring

Noise estimation

Measuring noise in a video signal would preferably be done in a part without picture content like the vertical or horizontal blanking period. However, measurement here is not reliable because of possible artificial signal content, e. g. sync insertion at VCR playback. So in the SAA4978H noise measurement is carried out within the active video signal. Because noise is hard to detect in moving parts of a picture with a high degree of detail information, the task is to find those parts of a picture that have almost no detail (flat areas).

A block diagram of the noise estimator is given in fig. 15. In the PREFILTER block where the interesting part of the spectrum is boosted, a first comparison is made between two groups of neighboring pixels on different positions in the same field. These differences are output as Y_{FIL}. Furthermore in this block a selection is made on the format of the incoming data. Parameter *INPUT8BIT* = 1 indicates that Y_{IN} is in 8 bit format placed in a 9 bit word. In this case bit 0 (LSB) is a copy of bit 1. No change is made for *INPUT8BIT* = 0. With Y_{FIL} being the output signal of the filter, parameter *PREFILTER_SCALING* has the following influence:

PREFILTER_SCALING = 0: scale = 1
PREFILTER_SCALING = 1: scale = 1/2
PREFILTER_SCALING = 2: scale = 1/4
PREFILTER_SCALING = 3: $Y_{FIL} = Y_{IN}$ (Filter off)

The idea of the noise estimator is to find flat areas in the picture and to determine the noise there. In order to find these areas each pixel is compared in amplitude to its neighboring pixels. In the block SAD calculation (SAD = Sum of Absolute Difference) the absolute values of the differences between the actual prefilter output Y_{FIL} and the four previous ones are summed up. These sums are then compared to a lower and upper bound. Each sum within these limits increments the event counter.

The counter can be disabled depending on the result of the SOB (SOB = sum over block) calculation and comparison. This function permits to detect black (level below 65) or white areas (level above 940) in the picture (e. g. black bars or side panels) which may have a signal content not representative for the picture. Parameter *SOB_NEGLECT* = 1 means that the result of the SOB comparison is not used and noise measurement is carried out in the complete picture, *SOB_NEGLECT* = 0 turns off measurement around the white and black level.

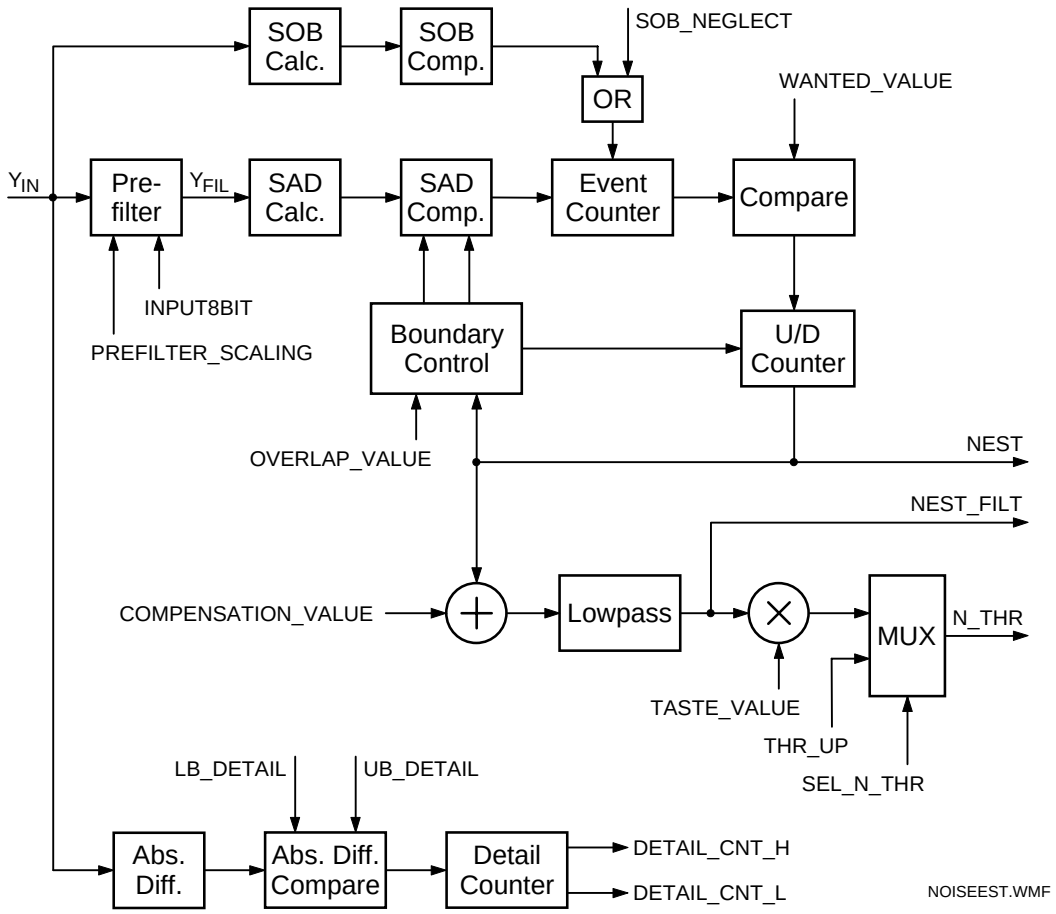


Fig. 15 Block diagram of noise estimator

At the end of every field the value of the event counter is stored and the counter is reset. The stored number of events is compared to the user defined *WANTED_VALUE*. If the number of events is smaller than *WANTED_VALUE* then a 4-bit up/down counter is incremented, otherwise it is decremented. The contents of the up/down counter is the noise estimator value *NEST* and can be read by the microprocessor. It is also used as input to the boundary control block where the lower and upper limits for the SAD comparison are adjusted. In this way the control loop is closed and the sensitivity can be influenced by *WANTED_VALUE*. The lower and upper limits are furthermore adjusted by the parameter *OVERLAP_VALUE* which determines the difference between these two limits:

lower boundary > *OVERLAP_VALUE*: upper boundary = 1.5 • lower boundary + 1
lower boundary <= *OVERLAP_VALUE*: upper boundary = lower boundary + 1

It is difficult to avoid that scenes with a large amount of detail result in a higher noise estimate compared with a scene of less detail but the same amount of noise. Therefore a function to measure the detail is incorporated. The difference between every two adjacent pixels is taken and compared to *LIMERIC_LB_DETAIL* and *LIMERIC_UB_DETAIL*. The number of times in a picture where this difference falls in between these boundaries is counted and can be read by the microprocessor in a two byte value: *DETAIL_CNT_H* and *DETAIL_CNT_L*. The microprocessor evaluates these data and calculates a correction factor *COMPENSATION_VALUE* for the noise estimation.

The noise estimate *NEST* is also available in a lowpass filtered version: *NEST_FILT*. The lowpass filter generates a moving average over the last 16 *NEST* values. *NEST_FILT* is then multiplied by *TASTE_VALUE* to generate the noise threshold value N_THR. Alternatively a microprocessor generated *LIMERIC_THR_UP* can be selected by turning *SEL_N_THR* to 1. The signal *N_THR* is used to control the noise filter, see fig. 21.

Spatial noise reduction

In the noise filter neighboring pixels from the current line and the previous line are taken and compared to the actual pixel. The absolute differences are compared to *N_THR* and coefficients c are generated according to fig. 16. Absolute differences above *N_THR* will not be taken into account (c = 0) while particularly the small differences below *N_THR*/4 are heavily weighted (c = 4). The same weight of 4 is also put the actual picture P(i). All selected pixels of the current and the previous line are multiplied with their resp. coefficients, the products are added and divided by the sum of the coefficients. This results in a kind of averaging in flat areas (small absolute differences) while for more detailed areas more coefficients will be zero and filtering is turned off.

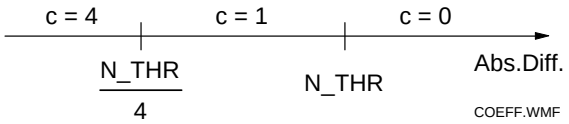


Fig. 16 Coefficient weighting

The selection of pixels taken into account for this filtering can be influenced by several parameters. For the current line *N_DIST* will control the distance of the two neighboring pixels, for the previous line this is controlled by the parameters *PC_DIST* for the center two pixels and *PE_DIST* for the external two pixels, see fig. 17 to 19.

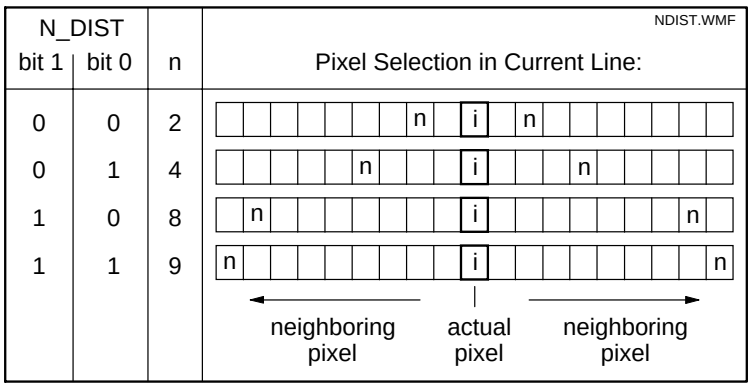


Fig. 17 Pixel selection in current line

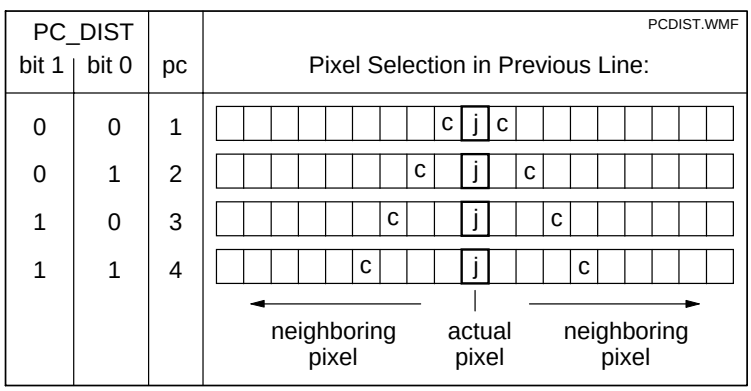


Fig. 18 Selection of center pixel in previous line

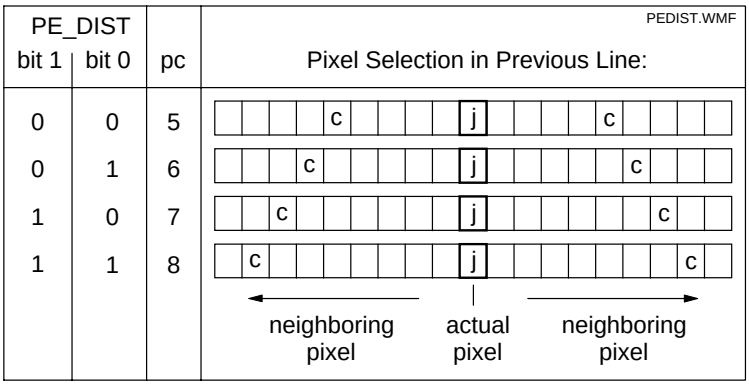


Fig. 19 Selection of external pixel in previous line

In order to avoid a possible pattern in the noise reduced signal a weave function can be turned on. This function gives an alternating shift to the five pixels of the previous line. It repeats every four lines, and the shift sequence is 0 / -1 / 0 / +1, see fig. 20. The sequence is made using a two-bit counter. The counter will start at position 0 in even fields and at position 2 in odd fields.

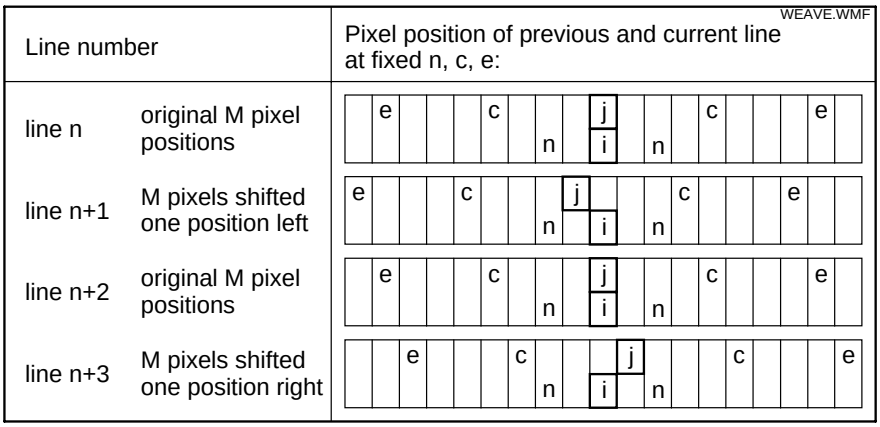


Fig. 20 Weave sequence

VHF energy

In this block the amount of vertical high frequency energy is calculated. The absolute values of the vertical high frequency data from the bandsplit filter are accumulated over one field and the result is stored in a register. The upper two bytes can be read by the microprocessor: *VHF_ENERGY_SUM_H* and *VHF_ENERGY_SUM_L*. In a third register the maximum vertical peak energy of the field can be read: *VHF_ENERGY_MAX*.

3.2.12 Histogram

The histogram system of the SAA4978H is intended to improve the picture quality of a TV set by dynamically controlling contrast. The optimum contrast depends on the scene which is being displayed. Therefore the properties of the scene are continually measured and an optimum video processing curve is calculated. These calculations are done in a microcontroller. So the complete histogram function consists of an analysis part (hardware), calculation of the optimal transfer curve (software) and programming this curve into the transfer function (hardware).

Fig. 22 gives an overview of the analysis part. The actual histogram data are generated only from the luminance data, the optimized transfer curve however equally effects the color difference signals U and V. Filters in the luminance channel are used in order to eliminate noise and aliasing somewhat before the histogram measurement is

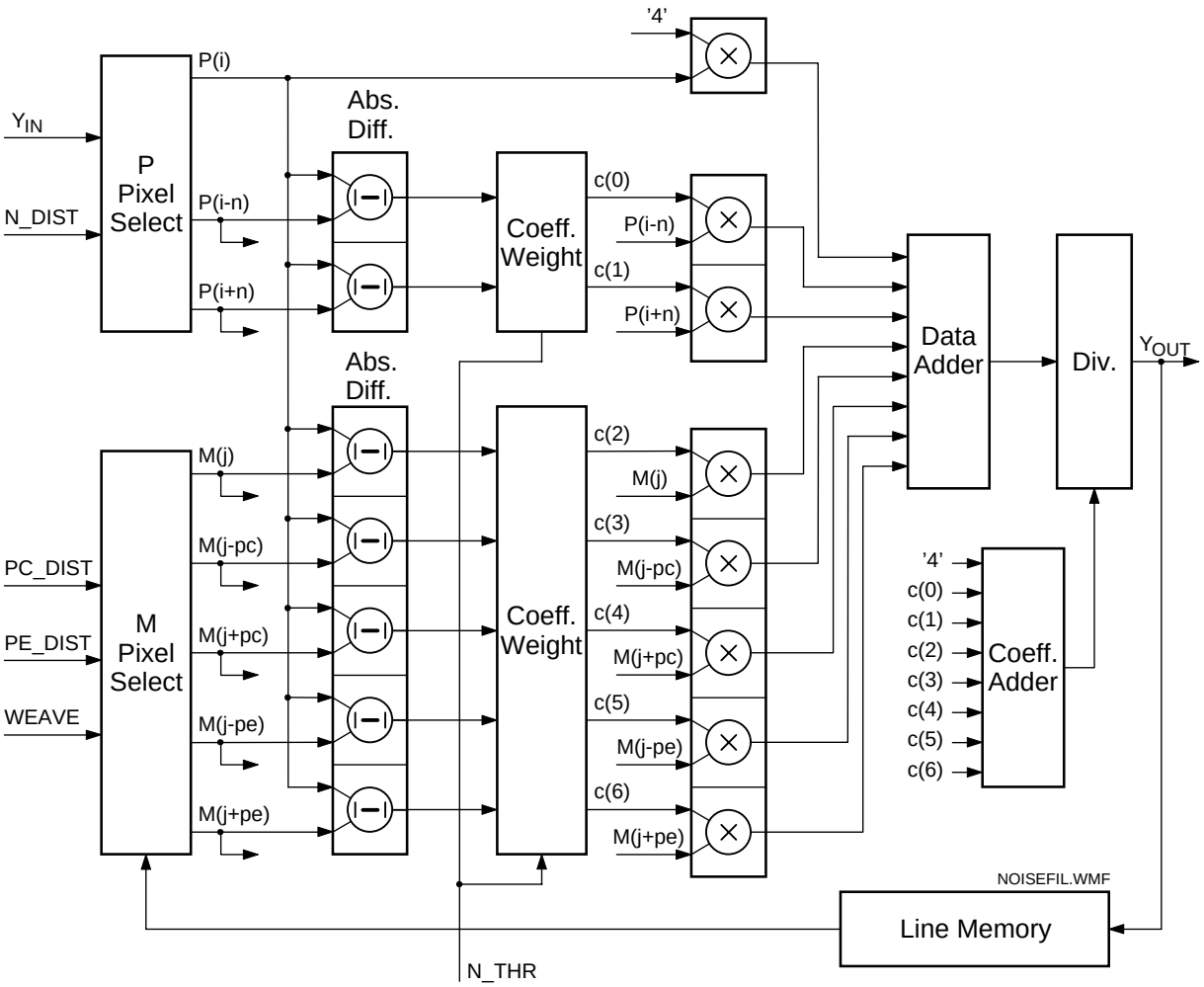


Fig. 21 Block diagram of noise filter

done. All measurements (histogram, min/max detection and smart black detection) are done in a rectangular window which forms a part of the visible picture. The window is defined by the parameters

HGM_WINDOW_H_START,
HGM_WINDOW_H_STOP,
HGM_WINDOW_V_START and
HGM_WINDOW_V_STOP.

The horizontal parameters are defined in multiples of 4 samples which gives a resolution of 250 ns at the nominal clock frequency of 16 MHz. The vertical resolution is 1 line. So the vertical start and stop values need a 9 bit word to define. If the ´on´ value is equal to the ´off´ value then the window is closed.

The min/max detection circuit determines the minimum and maximum values of the Y, U and V samples in the window of one field. They can be read from the output registers *Y_MIN*, *Y_MAX*, *U_MIN*, *U_MAX*, *V_MIN* and *V_MAX* after the last active line has passed. Only the upper 8 bits are output.

The smart black detector is a special minimum detector. From a number of consecutive samples the maximum one is taken, then from these maximum values the minimum one is determined. In this way noise and short low pulses are eliminated. The result can be read in the *SMART_BLACK* register. Only the upper 8 bits are output.

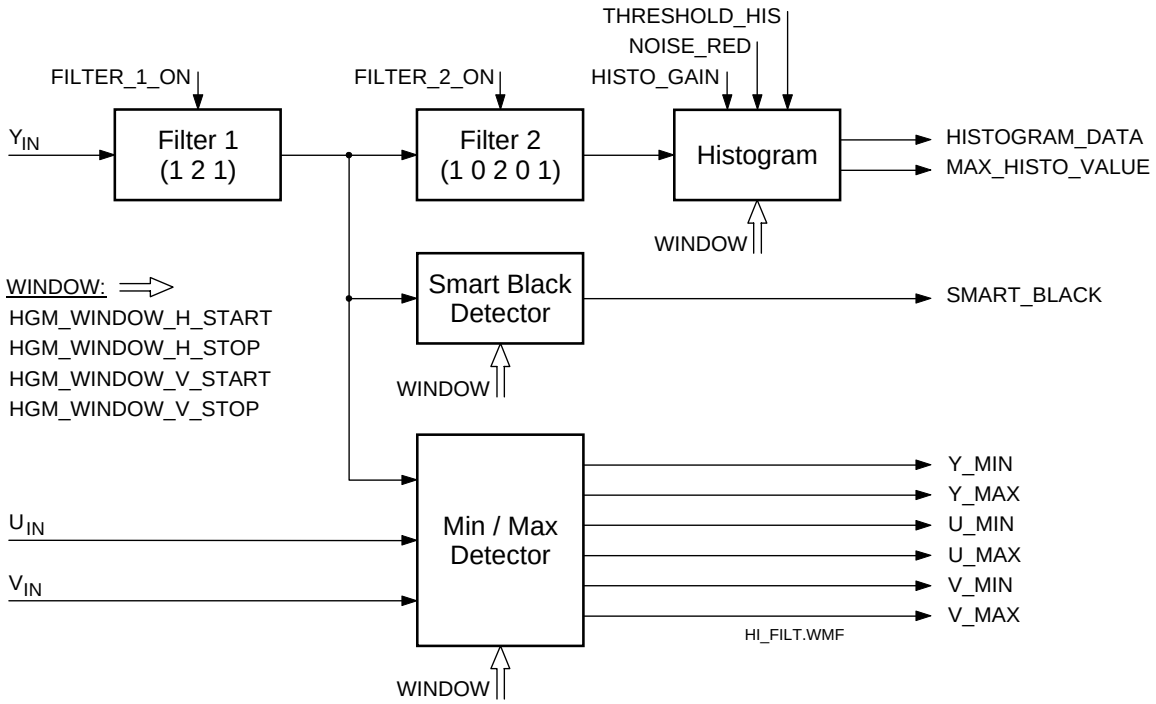


Fig. 22 Block diagram of histogram analysis part

Before the luminance data is passed to the histogram block it is low pass filtered by filter 2 which processes every second sample. The following histogram circuit uses only every fourth sample thus reducing the amount of data to be handled in the histogram. From line to line a 2 sample offset is used on the subsample pattern. The luminance samples are collected and sorted according to size. For an accurate histogram each luminance level would need its own register (in the following also referred to as basket) which would mean 512 registers for a 9 bit signal. In order to reduce the amount of data only 32 baskets are used. This means that 16 luminance levels fall into one basket. If the luminance level is equal to the center of the basket range this increases the register by 8. If the luminance level is near the edge between two baskets this would increase both, like 3 in one and 5 in the other one, see fig. 23. By this method, the quantization distortion from having a discrete set of baskets is overcome. This quantization noise reduction can be turned on by setting *NOISE_RED* to 1. If this control bit is off then each of the 16 luminance levels has the same weight in the basket and no neighboring ones are increased if the level is near an edge.

If no measures are taken all luminance samples are rated equal as they are sorted into the histogram baskets. A way to underrepresent flat areas is given by the parameter *THRESHOLD_HIS*. As shown in fig. 24 each luminance sample is compared to its predecessor. If the absolute difference is larger than *THRESHOLD_HIS* then the output signal STORE_Y enables this sample to be stored in the histogram. Setting the threshold to zero will get all samples stored while a higher value will prevent those samples from contributing which have a similar value compared to their preceding neighbor. At the beginning of a line the value of the preceding pixel is initialized to the value of the first pixel.

The histogram is collected in 32 registers of 18 bits each. Since only the relative distribution of signal levels is of interest, it is sufficient for the microprocessor to read only the upper 8 bits of the registers. Depending on the picture content and the setting of *THRESHOLD_HIS* the distribution can be anything from rather flat with low register values to something irregular with high peaks. Before the microprocessor reads the upper 8 bits, a gain adjustment takes place in a way that the MSB of the highest register value will be set to 1. This gain then is used for all 32 registers. It is set by the microprocessor or user by the parameter *HISTO_GAIN*. In order for the micro to find the proper setting a *MAX_HISTO_VALUE* can be read from the histogram. This value returns the top 8

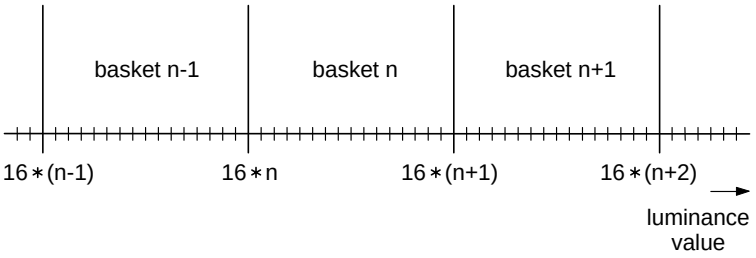
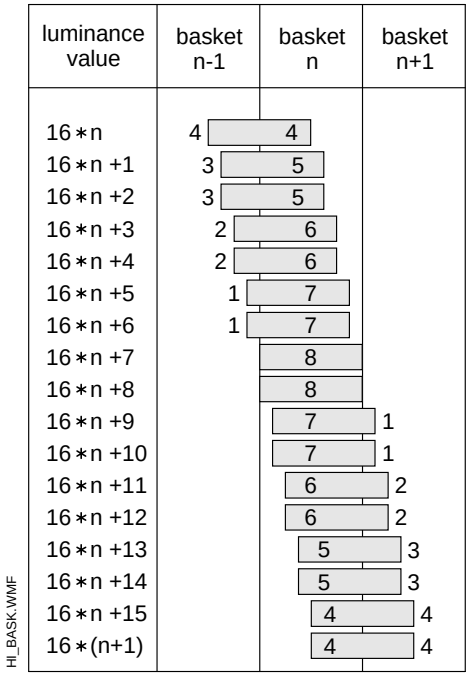


Fig. 23 Distribution of luminance levels among a discrete set of baskets



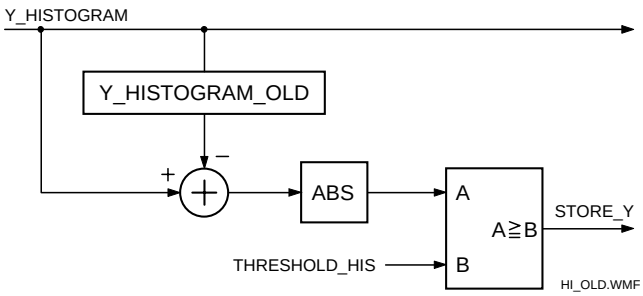


Fig. 24 Input processing of histogram measurement

bits of the maximum value of the histogram registers in the previous field. In fig. 25 the situation is depicted. On the left the histogram register with the highest value is shown. A gain setting of 1 would leave the upper four bits of the read byte at zero, and registers with lower values would return even smaller values, some possibly even zero. A gain setting of 16 (or a shift of 4 bits) is optimal in this case as it makes use of the full range of the 8 bits the micro can read. If gain is increased further this creates an overflow and all bits are set to 1. This clipping gives the smallest error.

After the collected histogram data are evaluated a luminance transfer function is defined by the microprocessor and the luminance transfer curve is programmed into the LUT (look up table). Fig. 26 shows how the luminance transfer function is realized.

To the input signal Y_{IN} a *BLACK_OFFSET* is added. Normally this is a negative number resulting in a brightness adjustment to darken the picture. In the LUT the differences from a linear transfer curve are programmed for 32 points at equidistant intervals. For values between these points a linear interpolation is done. The differences are then added to the original signal (input of the LUT). At the output a switch can be activated to turn the histogram processing on or off. Programming the switch position in the visible area allows a direct comparison between the original (left side of the screen) and the histogram-modified picture (right side of the screen).

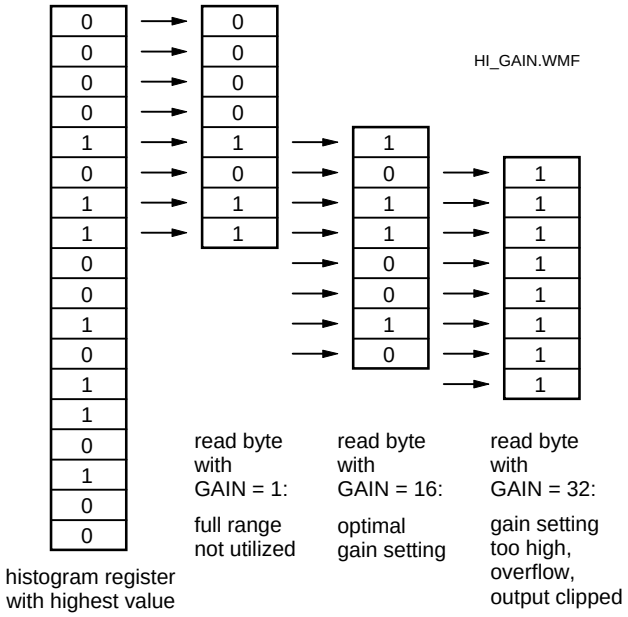


Fig. 25 Histogram register gain adjustment for microprocessor reading

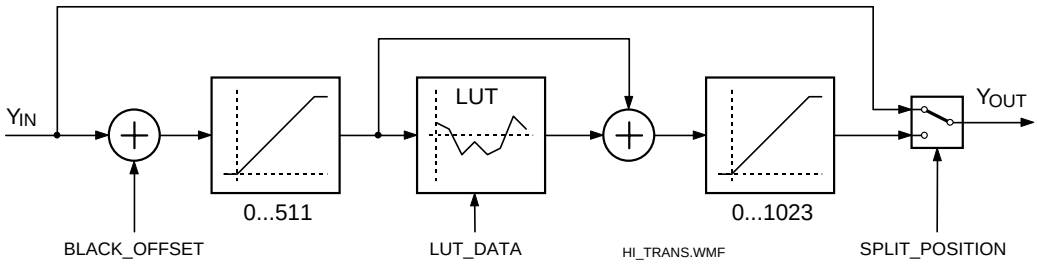


Fig. 26 Luminance transfer curve

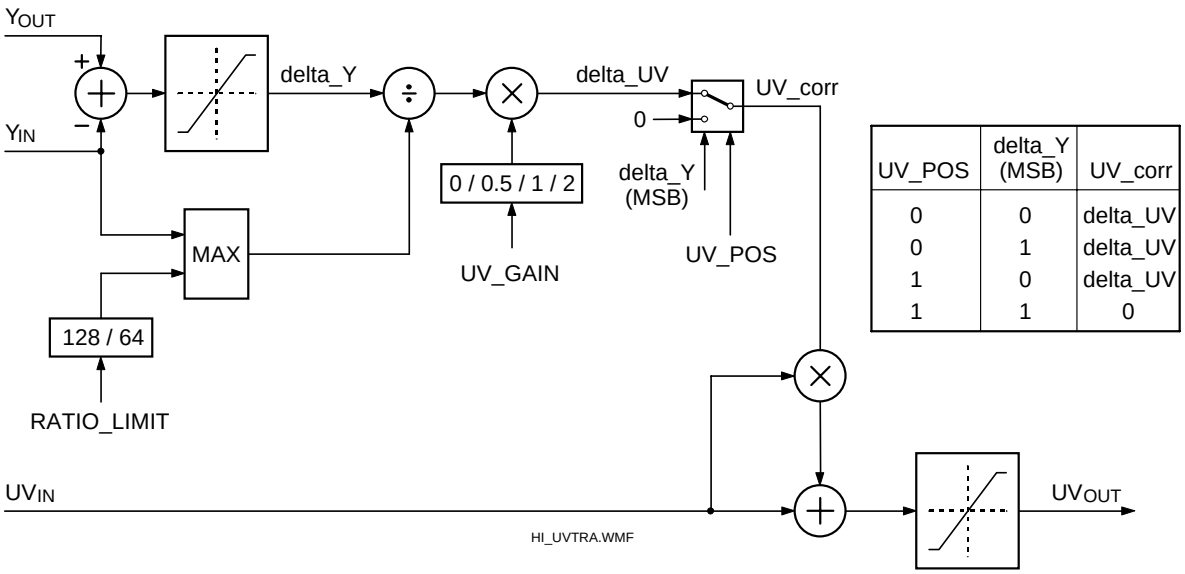
In order to maintain proper saturation of the picture the U and V levels must be adjusted in relation to the modified luminance histogram according to the formulas

$$U_{OUT} = U_{IN} + \frac{Y_{OUT} - Y_{IN}}{Y_{IN}} \cdot U_{IN}$$

$$V_{OUT} = V_{IN} + \frac{Y_{OUT} - Y_{IN}}{Y_{IN}} \cdot V_{IN}$$

This involves a division of Y_{OUT} - Y_{IN} by Y_{IN}. For very small input values Y_{IN} the quotient can have a high value. Therefore the quotient is limited by limiting the denominator. As can be seen in fig. 27 the circuit picks the maximum value of Y_{IN} and a lower limit of either 128 or 64 as denominator. The limiting value can be set by parameter *RATIO_LIMIT*. After the division the amount of saturation correction done on the U and V signals can be selected by parameter *UV_GAIN*. In cases where *delta_Y* (= Y_{OUT} - Y_{IN}) is negative (MSB of *delta_Y* = 1), a negative correction signal *UV_corr* can be prohibited by setting parameter *UV_POS* = 1. This increases the color saturation when the luminance signal is attenuated. At the output the UV signal is adjusted by multiplication and addition of the correction signal:

$$UV_{OUT} = UV_{IN} \cdot (1 + UV_{CORR})$$



UV_POS	delta_Y (MSB)	UV_corr
0	0	delta_UV
0	1	delta_UV
1	0	delta_UV
1	1	0

Fig. 27 UV transfer function

3.2.13 Display bars

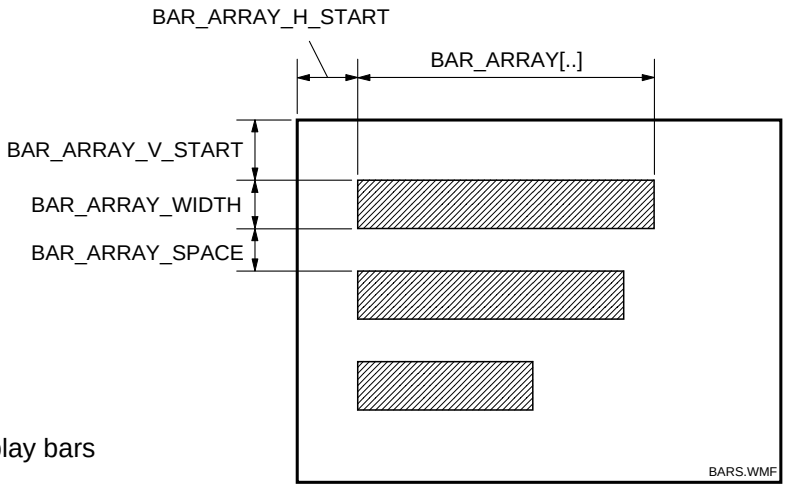


Fig. 28 Display bars

For demonstration and debug purposes artificial video signals can be inserted into the datapath. These can be used to indicate the position of a window, the histogram of a scene or the transfer curve. The basic shape of these signals is rectangular. It is possible to define 32 bars on a screen. For a bar display the following registers must be defined:

- *BAR_ARRAY_Y* / *BAR_ARRAY_U* / *BAR_ARRAY_V* for showing luminance or color difference values. Only the 8 most significant bits are used, the LSB is forced to 0.
- *BAR_ARRAY_H_START* is the horizontal start position for all bars.

- *BAR_ARRAY_V_START* is the vertical start position of the first bar. Line number 0 is the first line in the active video area.
- *BAR_ARRAY_WIDTH* is the width of a bar. All bars have the same width. If this value is 0 no bars will be displayed.
- *BAR_ARRAY_SPACE* is the distance between two bars.
- *BAR_ARRAY* defines the length of each bar. There is only one *BAR_ARRAY* register which sequentially steps through the array when writing to it. At the beginning of each field the write pointer is reset.

There are three more bits controlling the bar display. *BAR_ARRAY_ON* turns the bar display on or off. *BAR_ARRAY_TRANS* selects the way the bars are displayed on the screen. If *BAR_ARRAY_TRANS* = 0 the output signal is replaced by the bar signal (superimpose mode), if *BAR_ARRAY_TRANS* = 1 then only the even numbered pixels are substituted, so the original scene is still visible through the bars. This is the transparent mode. *BAR_ARRAY_RESOLUTION* defines the resolution used for positioning and displaying the bars, see table 5.

	BAR_ARRAY_RESOLUTION =	
	0	**1**
BAR_ARRAY_H_START	4 pixel	2 pixel
BAR_ARRAY_WIDTH	2 lines	1 line
BAR_ARRAY_SPACE	2 lines	1 line
BAR_ARRAY	4 pixel	2 pixel

Table 5: Bar array resolution

3.2.14 Subtitle detection

The subtitle detector is able, with the assistance of a microcontroller, to detect subtitles in a TV picture by detecting events or transitions. Within a defined window the number of events per TV line is stored in an array. The size of this array is 128 registers, so the window can comprise a maximum of 128 TV lines. The array can be read by a microprocessor which is then able to calculate the lines in which subtitles occur.

The subtitle window is defined by the parameters *SUBT_WINDOW_H_START*, *SUBT_WINDOW_H_STOP*, *SUBT_WINDOW_V_START* and *SUBT_WINDOW_V_STOP*. The horizontal start and stop parameters are 8 bits wide, so the actual borders are 4 * *SUBT_WINDOW_H_START* and 4 * *SUBT_WINDOW_H_STOP*. The vertical start and stop parameters are 9 bits wide and thus have a resolution of one line.

The event detector evaluates signal levels and transitions. There are two modes: BETWEEN LEVELS MODE and EVENT MODE. The modes are controlled by parameter *EVENT_MODE*. In the BETWEEN LEVELS MODE the number of occurrences that a sample is greater or equal to *THRESHOLD_LOW* * 2 or lower or equal to *THRESHOLD_HIGH* * 2 is counted. (Parameters are multiplied by 2 because of the 9 bit signal level.) In fig. 29 the video signal between the two thresholds is shown in bold.

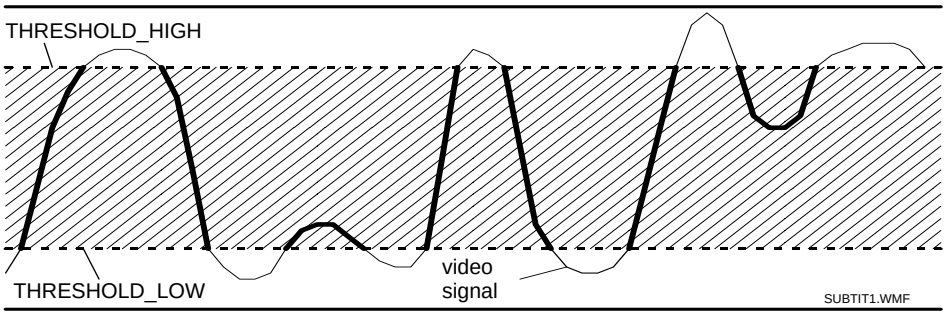


Fig. 29 Detecting video signal levels between two thresholds

In EVENT MODE an event is detected whenever the luminance signal is greater than *THRESHOLD_HIGH* * 2 for more than *HIGH_TIME* samples or if the signal drops below *THRESHOLD_LOW* * 2 for more than *LOW_TIME* samples. For each line of the defined window the number of these events is counted. If parameter *RESET_EVENTS* = 1 the result is stored in the array, if *RESET_EVENTS* = 0 then the result is added to the result of the previous field. The number of events per line is limited to 255.

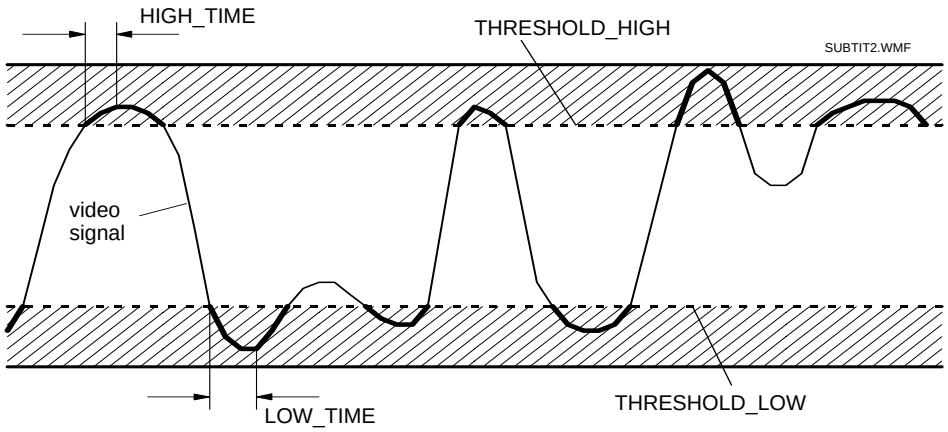


Fig. 30 Detecting peak high or low signal levels

Also incorporated in the subtitle detection block is a peak detector which determines the peak luminance signal within the subtitle window. If parameter *RESET_PEAK* = 1 then the peak value measured during the actual field can be read in register *PEAK_Y*. If RESET_PEAK = 0 then the actually measured peak value is returned only if it is greater than or equal to the value of the previous picture. If it is smaller then the peak value returned is that of the previous field minus 1. The returned value comprises only the 8 most significant bits.

3.2.15 Blackbar detection

Wide screen transmissions ("letterbox format") displayed on a conventional 4:3 screen will result in black bars at the top and bottom, see ⓐ in fig. 31. If no measures are taken, they will also leave black bars on a wide 16:9 screen, see ⓓ in fig. 31. On a 4:3 screen this is acceptable because the proper aspect ratio is maintained. On a 16:9 screen it appears distorted however, which is less acceptable. In fig. 31 some measures are pointed out of what to do with a letterbox signal.

On a 4:3 display the picture can be expanded horizontally and vertically to fill the whole screen, this results in some parts of the picture getting lost (left and right side, see part ⓑ in fig. 31). Another way is to expand the picture vertically and activate the inverse panorama mode ("amaronap mode") which compresses the left and right side so everything fits on the screen (ⓒ in fig. 31).

On a 16:9 screen the only desirable action would be to expand the picture vertically. This would fill the whole screen and restore a proper aspect ratio (ⓔ in fig. 31).

When there is no information transmitted about the picture format, the display has to be adjusted manually. An automatic mode though becomes available if the blackbar detection of the SAA4978H is activated and its results are evaluated.

Fig. 32 shows the block diagram of the black bar detection. All measurements are done in a rectangular window which is defined by the four parameters *BBD_WINDOW_H_START*, *BBD_WINDOW_H_STOP*, *BBD_WINDOW_V_START*, and *BBD_WINDOW_V_STOP*. The horizontal start and stop position can be programmed in steps of 4 pixels, the vertical position in steps of one line.

There are two detectors which operate alike. However each one of them has its own slicing level to recognize black level. The aim of the detector is to determine the first and last non-black line in the picture. At the beginning of a field a temporary register *first_videoline* is incremented every line as long as the line is found to be black. The incrementing stops with the first non-black line. This one represents the top of the letterbox picture. The register content can be read as *BBD_FIRST_VIDEOLINE1* or *BBD_FIRST_VIDEOLINE2* resp. The bottom of the

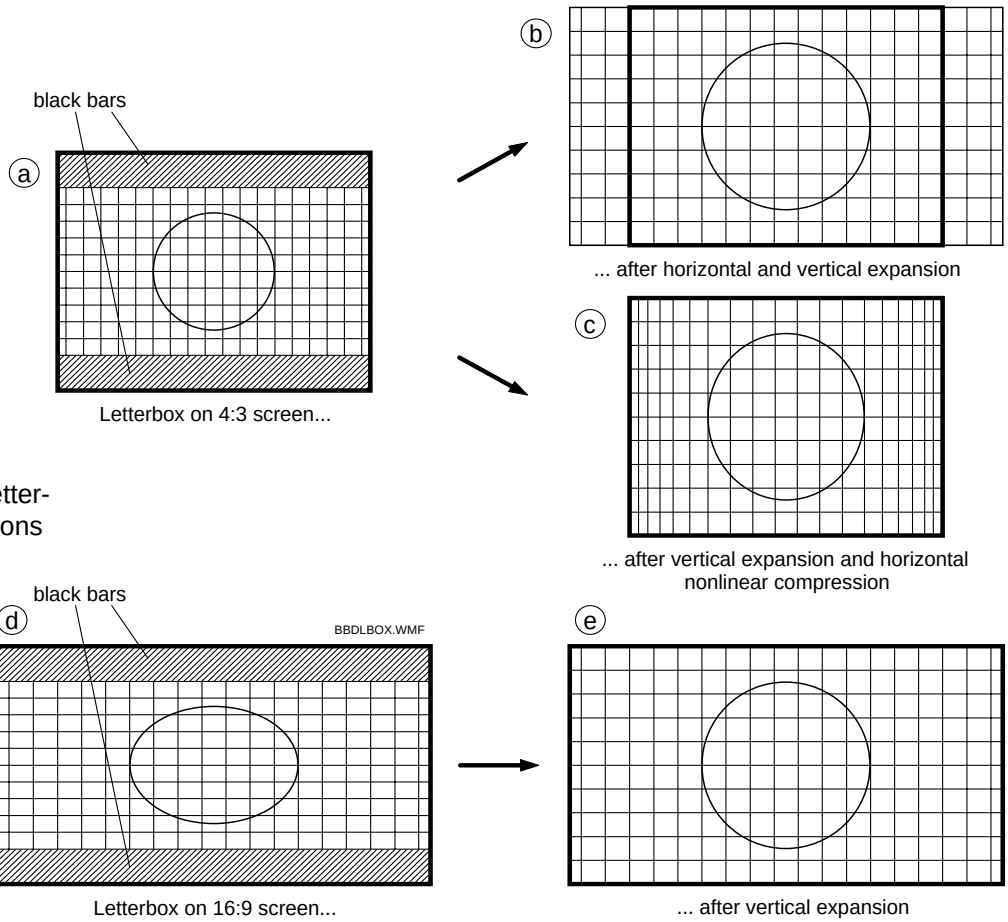


Fig. 31 Dealing with letter-box transmissions

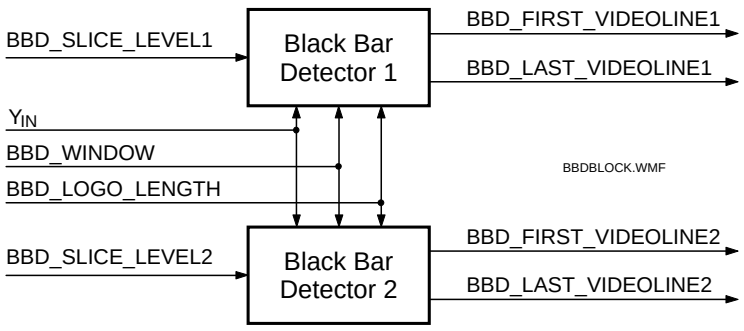


Fig. 32 Block diagram of the black bar detection

letterbox picture is found in a similar way. Incrementing of a temporary register *last_videoline* stops with the last non-black line, and the register content can be read as *BBD_LAST_VIDEOLINE1* or *BBD_LAST_VIDEOLINE2* resp. Only the eight most significant bits of the resp. nine bit line numbers can be read, so the vertical resolution for determining top and bottom of the letterbox is two lines.

Recognizing a black line can be influenced not only by adjusting the window borders, but also by three more parameters. *BBD_SLICE_LEVEL1 and BBD_SLICE_LEVEL2* determine whether a pixel is considered black or

not, and *BBD_LOGO_LENGTH* sets a limit on how many non-black pixels are allowed while that line is still considered to be black.

3.2.16 Bus_C Output

Luminance and chrominance data from the histogram modification block can be output at BUS_C (Y_C and UV_C) for scan conversion purposes. The scan converted data then is input again at BUS_D. If no external conversion is required then Y and UV can internally be passed to the subsequent processing blocks. In this case the full 9 bit data width is always maintained.

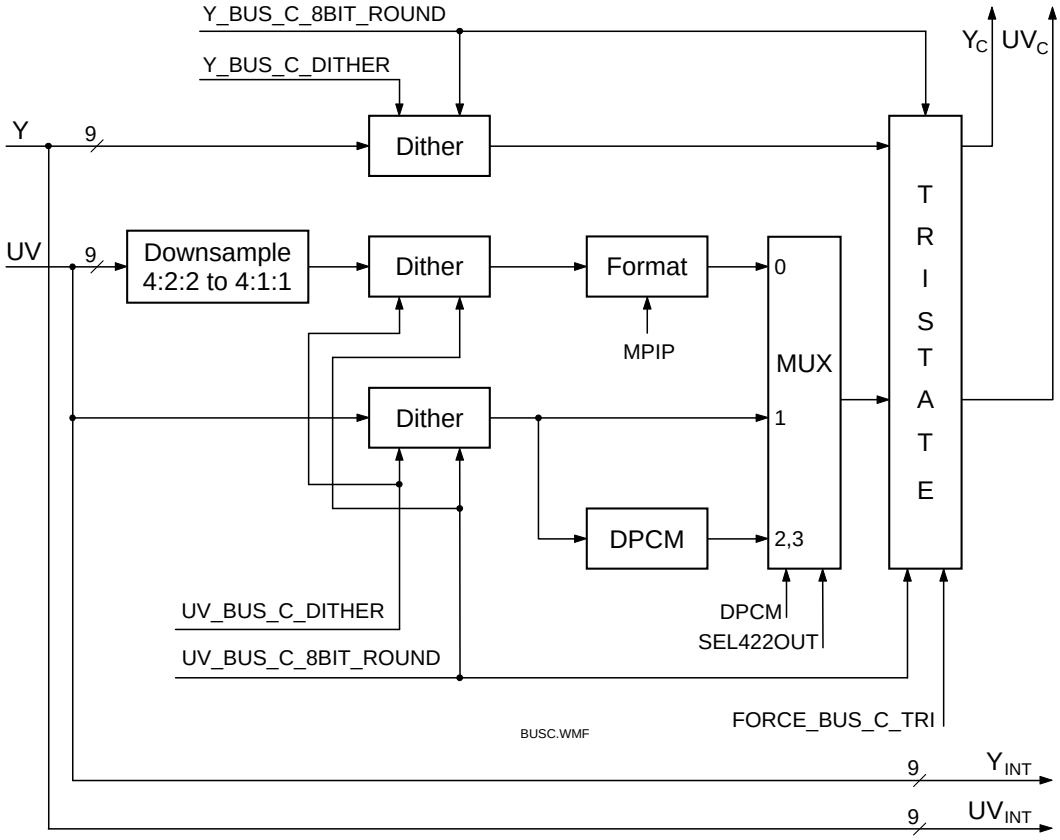


Fig. 33 Block diagram of Bus_C output processing

Fig. 33 gives a detailed block diagram of the data processing at the output of BUS_C. Several data formats are supported. Luminance data width can be selected to be 9 or 8 bits wide. In 8 bit mode dithering can be turned on to regain some of the lost bit for low frequency signals. Luminance data format and width is controlled by the parameters *Y_BUS_C_8BIT_ROUND* and *Y_BUS_C_DITHER*. Dither control is the same as for BUS A (see paragraph 3.2.7 on page 16 and table 1). Dithering can be activated for the 4:1:1 format as well as for the 4:2:2 format.

UV data can be output in 4:2:2 or 4:1:1 format. For the 4:1:1 format the 4:2:2 data is downsampled in a 7-tap FIR filter and then subsampled in the FORMAT block. With parameter *MPIP* = 0 the standard 4:1:1 data is generated, *MPIP* = 1, 2 or 3 define multi-PIP decimation modes for 2 x 2, 3 x 3 or 4 x 4 pictures on the screen. Decimation of the Y signal is done by subsampling using the WE control signal. The U and V subsampling must fit in the WE control pattern in such a way that the serial format remains valid.

For the UV signal also DPCM processing can be selected. In this mode the color bandwidth of a 4:2:2 format can be maintained with only a 4 bit (instead of 8 or 9 bit) wide UV data format. Together with 8 bit wide luminance data this allows to handle a 4:2:2 quality signal on a 12 bit wide data path, e. g. to use conventional 12 bit wide field memories for scan conversion.

In a multiplexer the wanted output format 4:1:1, 4:2:2 or DPCM is selected by the parameters *DPCM* and *SEL422OUT* according to table 6.

Output Format	DPCM	SEL422OUT
4:1:1	0	0
4:2:2	0	1
DPCM	1	x

Table 6: Output format selection

When BUS_C outputs are not required in an application, setting bit *TRISTATE* to high sets the outputs of this bus to tristate. With the *TRISTATE* bit at low the state of the outputs depends on the output configuration according to table 2.

FORCE_ BUS_C_ TRI	Y_ BUS_C_ 8BIT_ ROUND	UV_ BUS_C_ 8BIT_ ROUND	SEL422 OUT	DPCM	YC [8:1]	YC [0]	UVC [8:5]	UVC [4:1]	UVC [0]	WE_C / IE_C
1	xx	xx	x	x	tri	tri	tri	tri	tri	tri
0	0x				active	active				active
0	1x				active	tri				active
0		0x	0	0			active	tri	active	active
0		xx	x	1			active	tri	tri	active
0		0x	1	0			active	active	active	active
0		1x	0	0			active	tri	tri	active
0		1x	1	0			active	active	tri	active

tri . . . tristate

Table 7: BUS_C tristate modes

3.3 Digital processing at $2f_H$ level

3.3.1 BUS_D input

In Fig. 34 a diagram of the processing blocks at $2f_H$ level is given. If the SAA4978H is not used for scan conversion, the blocks will also process signals at the $1f_H$ level. The signal data can be input via BUS_D (Y_D and UV_D), or it is supplied internally from the previous blocks (only in case of $1f_H$ processing). The internal bus is always in 4:2:2 format and 9 bits wide for both luminance and chrominance. The input format can be selected by parameter *SEL_INPUT_FORMAT*.

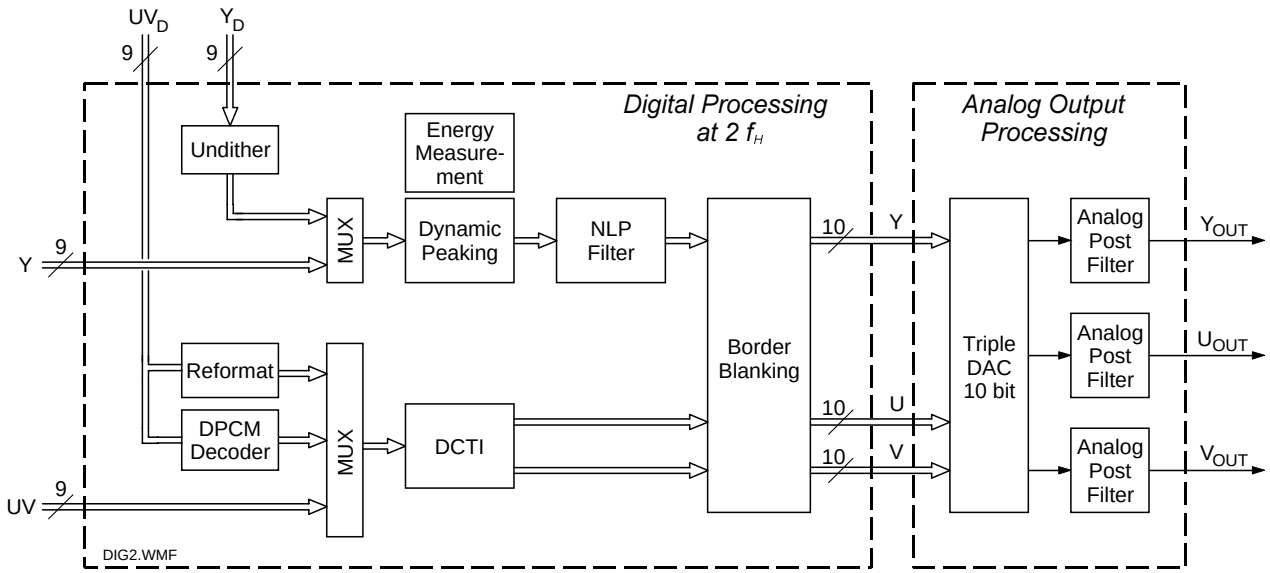


Fig. 34 Digital processing in the $2 \cdot f_H$ domain and analog outputs

For 8 bit wide input formats data can be undithered if it has been dithered at the output of BUS_C. The undither circuit removes the $f_S/2^1$ modulation and restores the ninth bit for low frequency signals as can be seen in the example in fig. 35: the input sequence (only even numbers, representing 8 bit) is transformed into a 9 bit sequence (odd numbers added).

input: .... 2 2 2 2 4 2 4 2 4 2 4 4 4 4 4 4 6 4 6 4 6 4 6 6 6 6 6 ....
output: .... 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5 6 6 6 6 6 ....

Fig. 35 Example of undithering signal data

3.3.2 Reformatting and DPCM Decoding

In the REFORMAT block the 4:1:1 UV input data is converted to the 4:2:2 format. This envolves upsampling from 8 to 16 MHz which in a first step is done by duplicating the input samples (fig. 36). In a second step a 4-tap FIR filter is used for smoothing the output signal. The frequency response is shown in fig. 37.

DPCM decoding takes place in the block DPCM DECODER. It is activated whenever the chrominance data at the output of BUS_C are DPCM coded.

1. f_S ... sample frequency

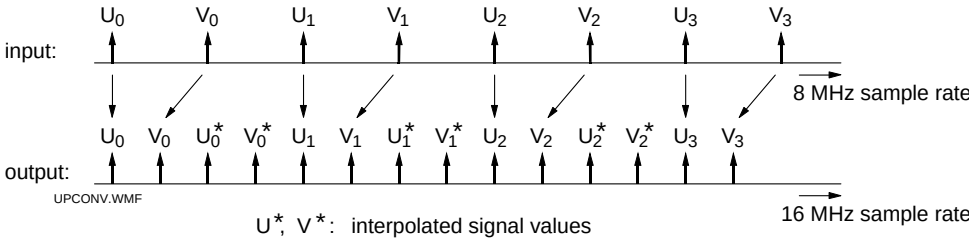


Fig. 36 UV upconversion from 8 to 16 MHZ

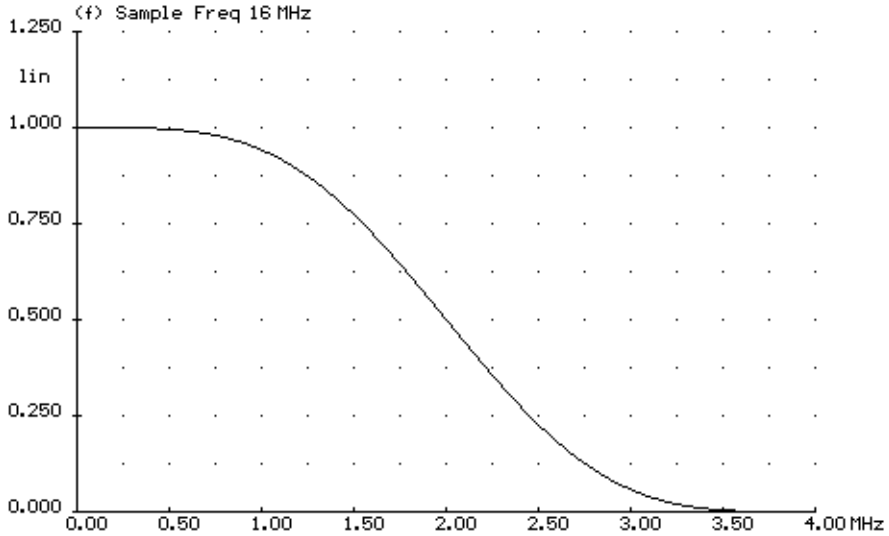


Fig. 37 Frequency response of the upsampling filter

The following MUX is activated according to the wanted input signal by the parameter *SEL_INPUT_FORMAT*. The setting are: 422 external / 411 external / 422 internal / DPCM external. Data width is always 9 bits, in case of externally provided 8 bit wide data the 9th bit (LSB) has to be connected to GND.

3.3.3 Luminance Peaking

The luminance signal Y is processed by the peaking circuit in order to boost the higher frequency ranges. A block diagram of the circuit is shown in fig. 38. The circuit uses a combination of two band pass filters and a high pass filter. The first band pass filter has the coefficients [-1 0 2 0 -1] and gives a maximum throughput at $f/f_c = 0.25$ (4 MHz)[2]. The second band pass filter is a convolution of the two filters [-1 0 0 2 0 0 -1] and [1 2 1] giving a peak at approx. $f/f_c = 0.15$ (2.4 MHz). The high pass filter is made with [-1 2 -1] coefficients with a maximum throughput at $f/f_c = 0.5$ (8 MHz). The summed output of the filters is processed by a coring circuit and then added to the original luminance signal.

The influence of each of the filters can be adjusted in eight steps from 0 to 8/16 (7/16 omitted), each step representing about 2 dB of gain difference at the center frequency. In fig. 39 to 41 the frequency response of each filter is given for different values of *TAU* (band pass 1), *ALPHA* (band pass 2) and *BETA* (high pass). Fig. 42 gives an example of two transfer curves having different center frequencies.

The peaking filters will boost higher frequency signals regardless of their amplitude. For structured small signals this will lead to unwanted coring. In order to prevent this the signal is fed through the above mentioned coring cir-

2. f_c ... clock frequency (16 MHz)

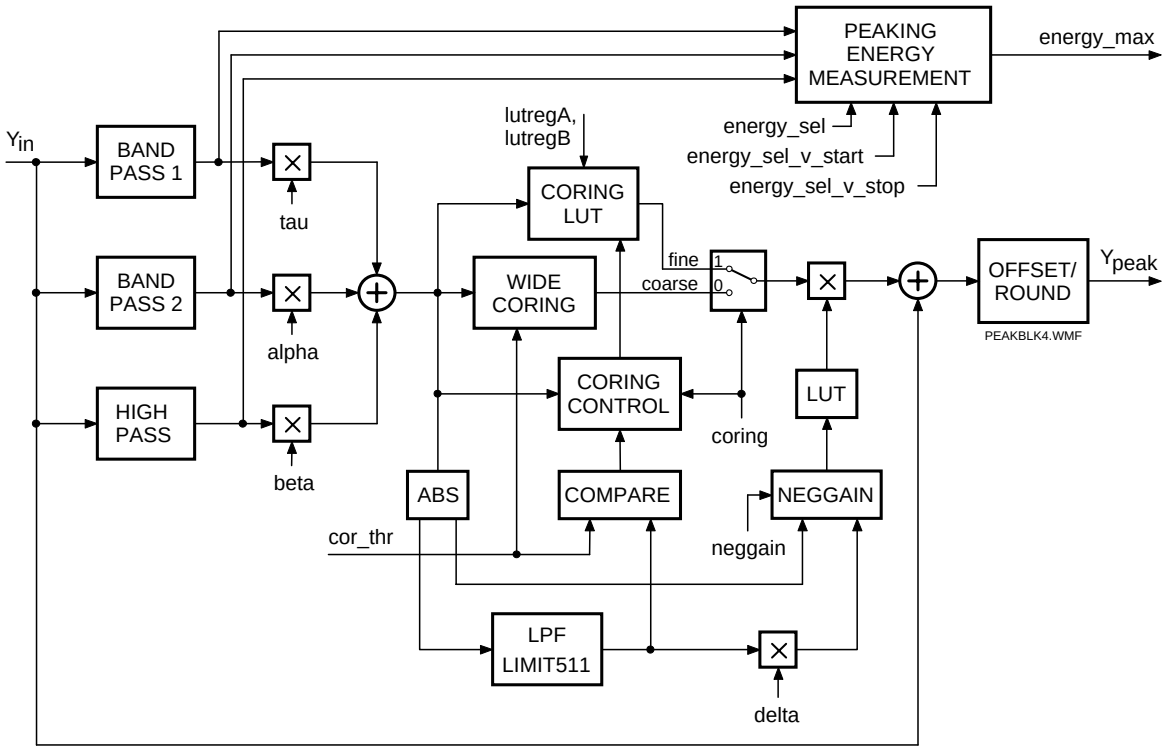


Fig. 38 Peaking block diagram

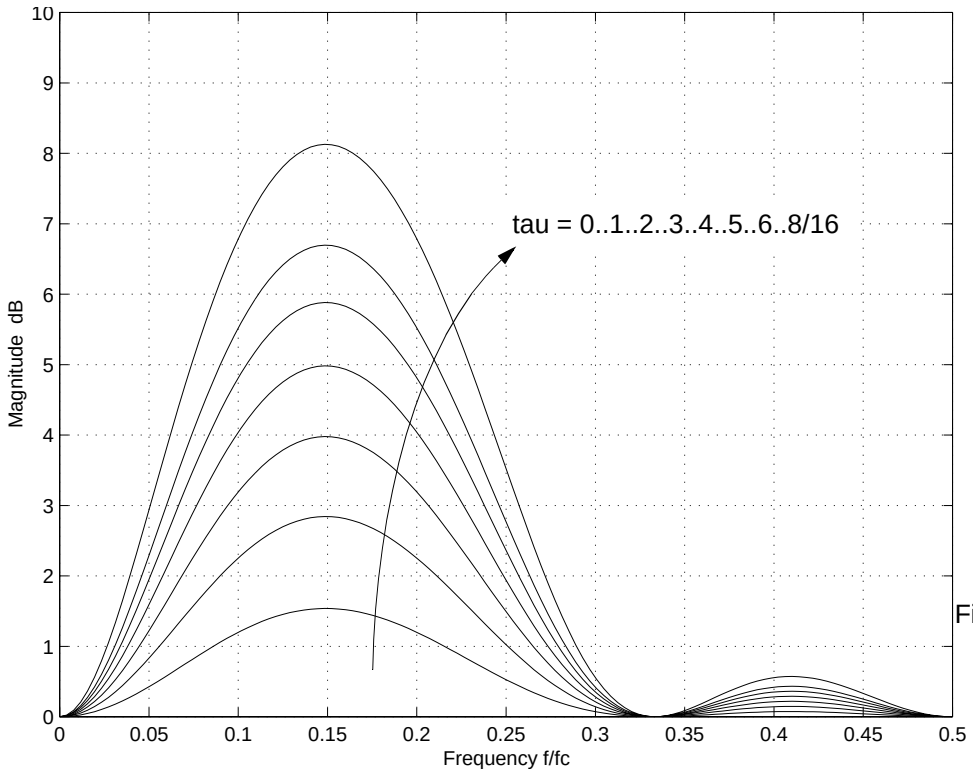


Fig. 39 Frequency response of the peaking band pass filter 1

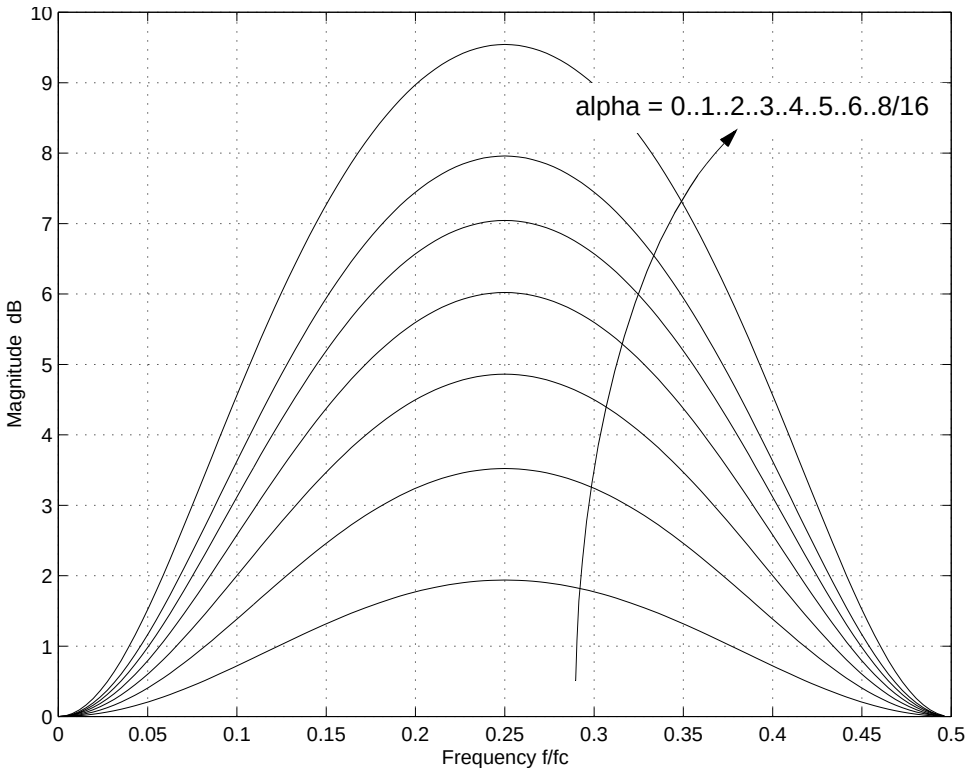


Fig. 40 Frequency response of the peaking band pass

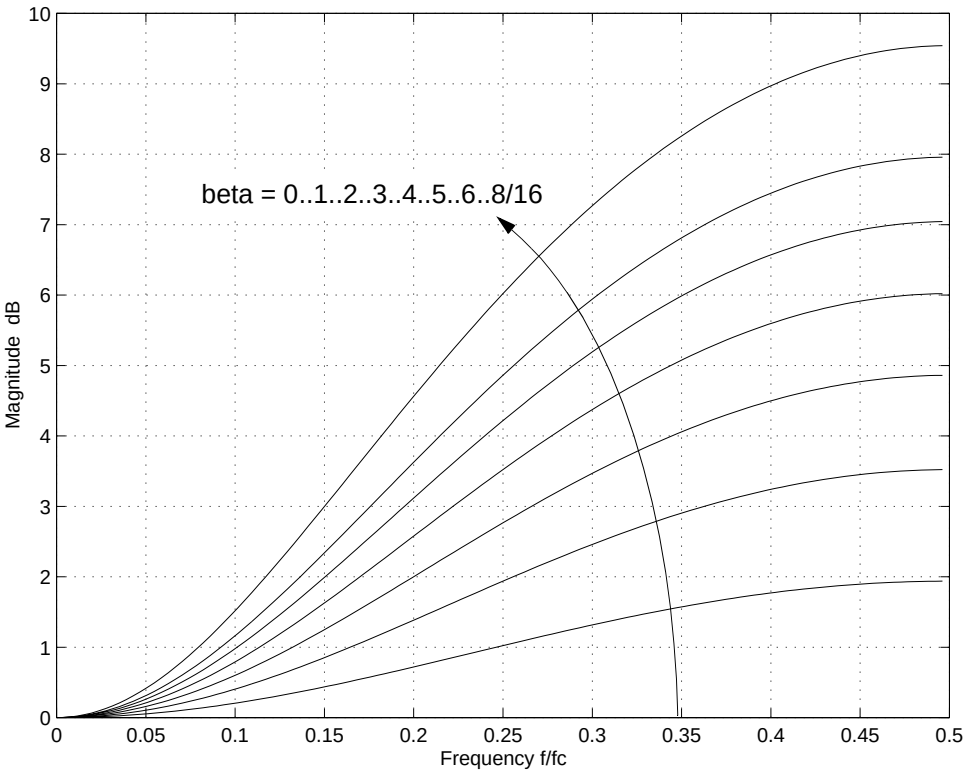


Fig. 41 Frequency response of the peaking high pass fil-

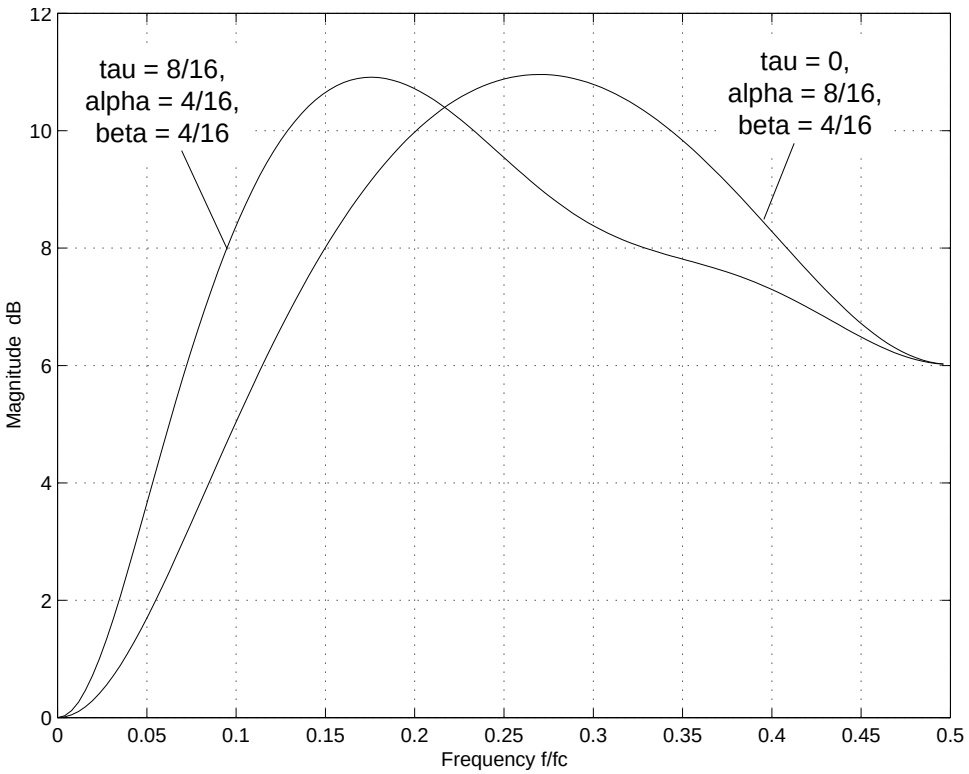


Fig. 42 Variation of peaking center fre-

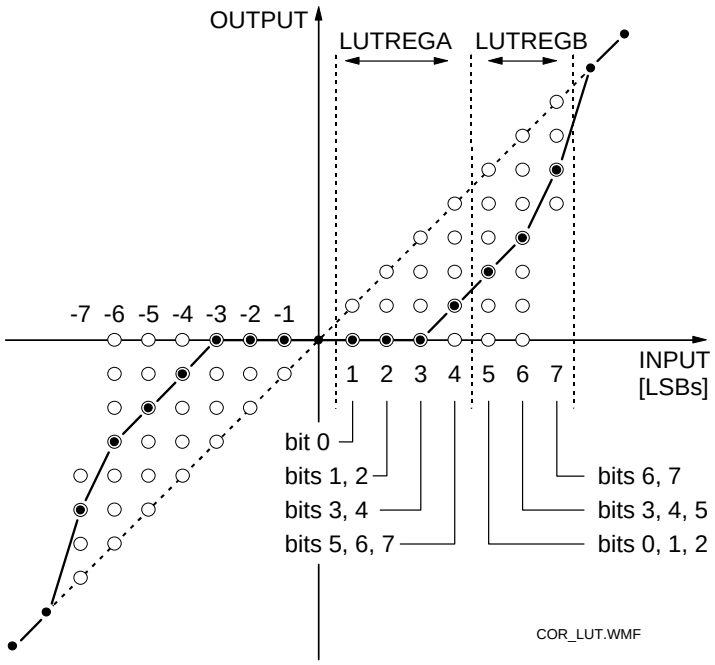
cuits. These circuits can suppress or lower the gain for low amplitudes, so the original luminance signal is less influenced.

There are two coring circuits in parallel: fine coring by use of a look-up-table (LUT), and wide (or coarse) coring. The transfer curve of the fine coring circuit is user definable and comprises the range from -7 to +7 LSBs. Signal amplitudes within this range are replaced by values from a lookup table, see fig. 43. The LUT is defined by the two registers *LUTREGA* and *LUTREGB*. The bits of these registers refer to the resp. LSBs of the filtered input luminance amplitude. So for the amplitude of 1 LSB the LUT can be defined to output either 0 or 1, thus one bit is sufficient. The input amplitude of 2 LSBs can be defined to output 0, 1 or 2, so two bits are supplied in *LUTREGA*, etc. For the LUT point 7 only two bits are supplied, so values from 4 to 7 can be defined. In this way output amplitudes can sometimes be defined which are greater than the input amplitude, these are contradictory to the wanted coring function and should be avoided.

The transfer curve of the coarse coring circuit is depicted in fig. 44. Small input changes are not transferred to the output. The limit up to which this suppression occurs is user definable by the parameter *COR_THR*. (*COR_THR* also serves the purpose to dynamically control the coring function, see below!). Selecting the fine or coarse type of coring is done by the parameter *CORING* (0 = coarse, 1 = fine).

The peaking function can be dynamically controlled in order to provide less gain on large details and edges. For this purpose the filtered luminance signal is lowpass filtered so the high pass energy is stretched to neighboring pixels in order to decrease decision noise in the attenuator. The output of the low pass filter is controlled by the parameter *DELTA* which can have the values 0, 1/4, 1/2 and 1. For *DELTA* = 1 attenuation is fully active, for *DELTA* = 1/2 and 1/4 it is reduced and for *DELTA* = 0 it is turned off.

In the following block NEGGAIN the negative going edges can be controlled separately. The parameter *NEGGAIN* can have the values 0, 1/4, 1/2 and 1. For *NEGGAIN* = 1 attenuation is fully active (same attenuation for positive and negative going edges), while for *NEGGAIN* = 1/2 and 1/4 it is reduced and for *NEGGAIN* = 0 it is turned off. *NEGGAIN* factors of less than 1 mean that in the output signal undershoots are larger than overshoots.



The sample curve on the left would be defined by:

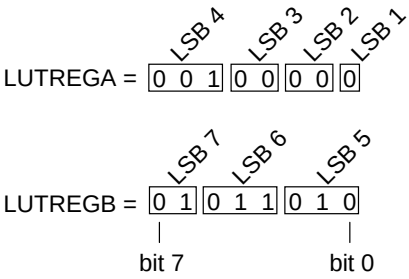


Fig. 43 Luminance fine coring using a lookup table

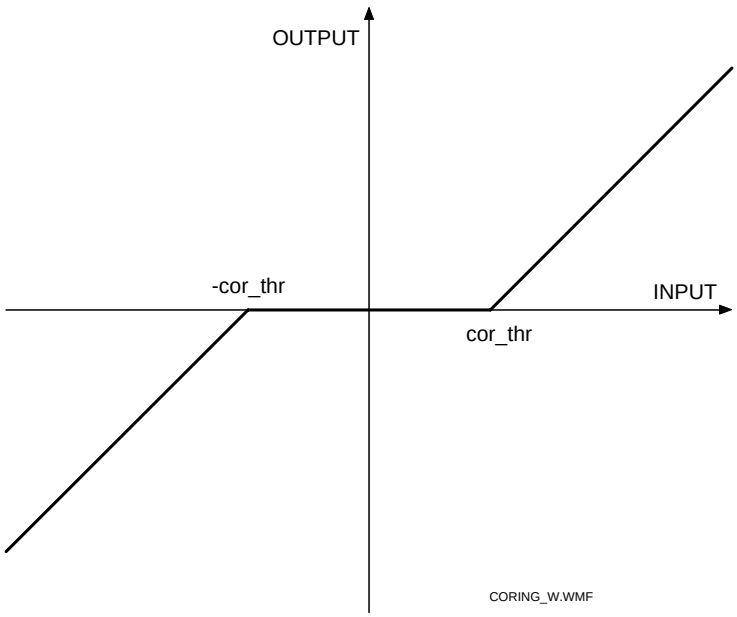


Fig. 44 Luminance coarse coring

The output of the NEGGAIN block is fed to a lookup table which delivers an 8-bit attenuation factor. The output of the LUT for different values of *DELTA* is given in fig. 45. The value of 128 is equal to no attenuation. Maximum attenuation is achieved at value 24.

The data used for the dynamic control of the attenuator is also used for automatic control of the fine coring function. This means that if the high frequency energy exceeds a programmable threshold then fine coring is switched off. The threshold is user definable by the parameter *COR_THR*. The effect of this circuit is that coring is only applied on steady areas like faces where contouring is very critical.

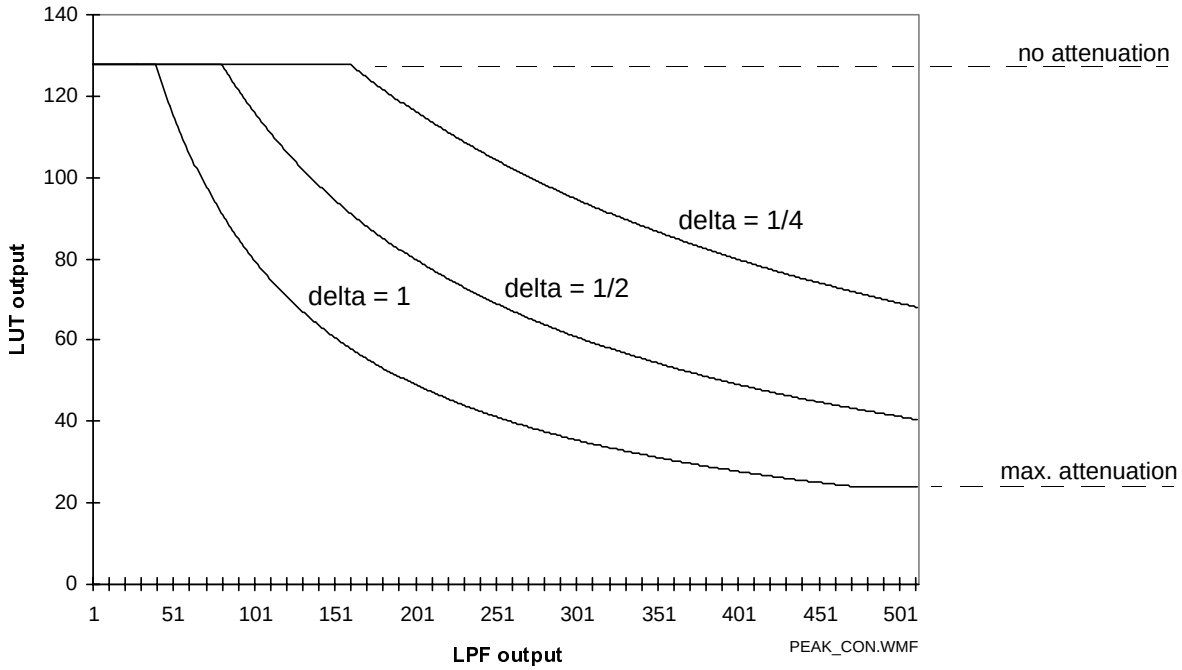


Fig. 45 Dynamic peaking control

The energy measurement circuit supplies data on the spectral content of the actual picture. From each of the three frequency bands it is possible to monitor the peak energy. In order to avoid false data generated by subtitles a vertical window can be defined where the energy measurement takes place. This window is defined by the parameters *ENERGY_SELECT_V_START* and *ENERGY_SELECT_V_STOP*. Per field only one filter output can be monitored, this is controlled by *ENERGY_SEL*. For measuring the generally lower high pass content of a signal a gain factor of 4 can be turned on.

ENERGY_SEL	SELECTION
0	high pass x 4
1	band pass middle
2	band pass low
3	high pass

Table 8: Filter selection for energy measurement

3.3.4 Nonlinear Phase Filter D/A

The nonlinear phase filter D/A (NLP-Filter) is designed to compensate for nonlinear filtering and bandwidth loss at the output of the IC as well as for sin x/x compensation. The filter can be adjusted by two parameters: λ defines the highpass amplitude, and μ determines the overshoot behavior. Settings are provided for λ = 0, 1/8, 2/8 and 3/8 (parameter *NLP_L_DA*) and μ = 0, 1/4 and 1/2 (parameter *NLP_U_DA*). Preshoots are generated for μ = 0, symmetry is obtained for μ = 1/2.

In fig. 46 the transfer and group delay curves are given for the different combination of λ and μ. In each plot the upper group of curves represent the group delay, the lower group of curves give the amplitude response. The left three plots show the behavior of the digital filter itself, the plots on the right side show the superposition of the digital filter and the analog postfilter.

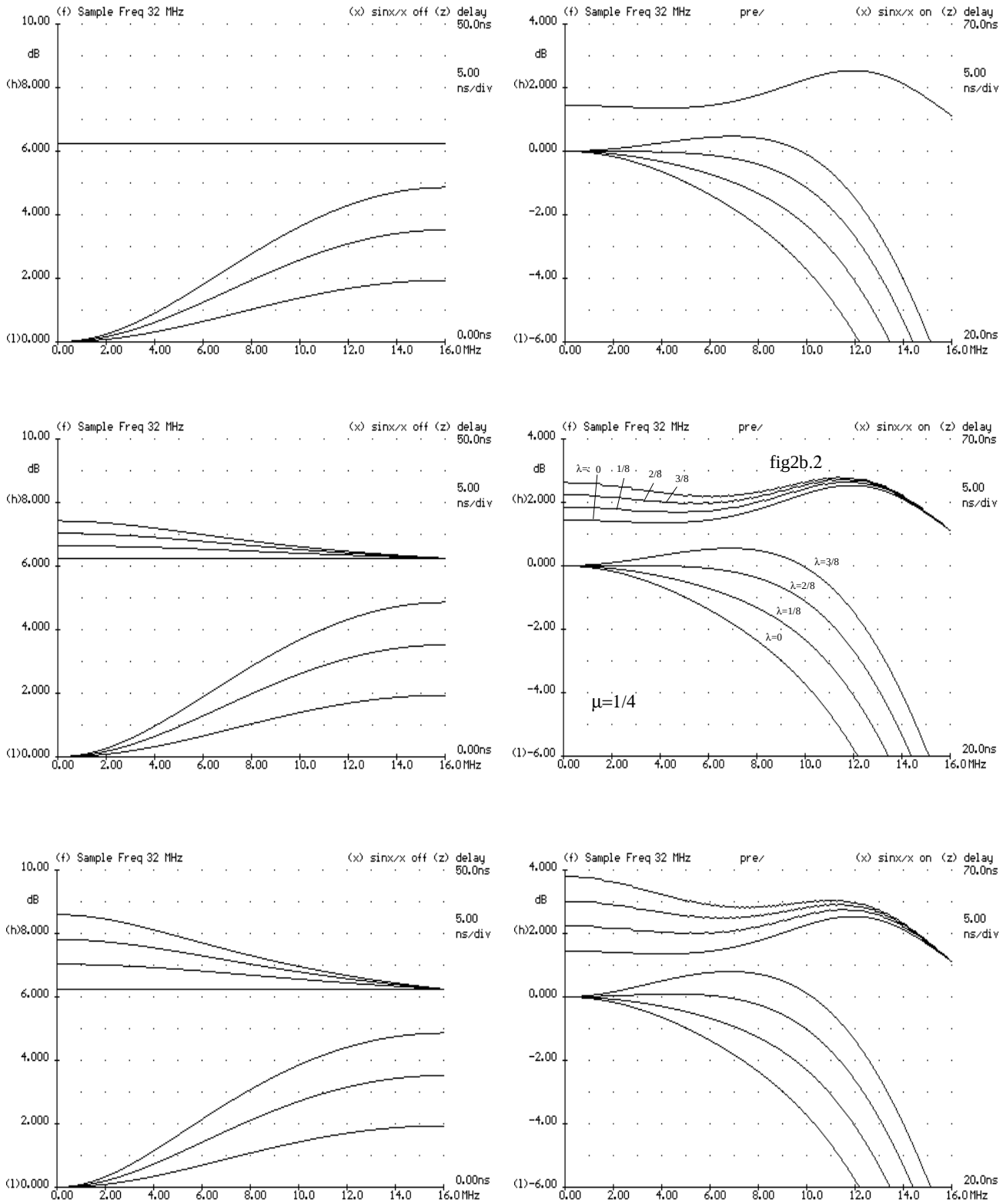


Fig. 46 Group delay and transfer curves of the NLP D/A filter

The NLP filter has four settings of parameter NLP_L_DA which give the following gain factors at 10 MHz:

NLP_L_DA	λ	gain [dB]
0	0	0
1	1/8	1.3
2	2/8	2.6
3	3/8	3.7

Table 9: NLP D/A gain settings

Setting 3 almost completely compensates the sin x/x and postfilter loss at 10 MHz, if more gain is wanted then this should be done in the dynamic peaking block.

3.3.5 Digital Color Transient Improvement (DCTI)

The Digital Color Transient Improvement (DCTI) is originally intended for U and V signals originating from a 4:1:1 source but 4:2:2 will also benefit from this circuit. The basic principle is to detect horizontal transients and improve their steepness without generating overshoots. This principle is depicted in fig. 47. U and V data always enter the block in 4:2:2 format but regarding their bandwidth they may stem from a 4:1:1 source. During the process upsampling to the 4:4:4 format occurs.

The idea is to vary the data path delay on the basis of a function of the second derivative of the U and V signal. Positive and negative transients are treated alike, the output of the first differentiator therefore is taken as absolute value. The signal is differentiated again and the output used to control the momentary data path delay. The effect at an edge is that during the first half the data path delay is higher than nominal and in the second half it is lower than nominal. This will make the edge much steeper. The control signal is amplified by a gain setting that is user defined. The parameter *CTI_GAIN* allows eight settings in the range of 0 .. 7/8. In case of *CTI_GAIN* = 0 the CTI function is turned off. Increasing this parameter results in a steeper transient.

The first differentiating filter offers two transfer curves which can be selected by the parameter *CTI_DDX_SEL*. The transfer curves are given in fig. 48.

The second differentiator runs at 16 MHz sample clock and generates interpolated values. The original U and V signals also are upsampled to 16 MHz, so the output of the CTI circuit has a resolution equal to that of the Y signal.

The DCTI function can be controlled mainly by adjusting the parameters *CTI_GAIN* and *CTI_LIMIT*. While *CTI_GAIN* influences the resulting steepness of the output signal, *CTI_LIMIT* affects the maximum amount of data path delay. User definable values for this parameter are 0, ±4, ±8 and ±12. Modifications of these parameters are depicted in the fig. 49 and fig. 50 using a maximum amplitude color transient as input signal. Both *CTI_GAIN* and *CTI_LIMIT* must be greater than zero for DCTI to be active.

An artifact of this processing becomes apparent when two edges are close together in the video signal. During the second half of the first edge a delay is chosen that will collect video data from where the second edge is already active. The same is valid for the second edge. The result of this processing on a video pulse, which is looking like a hill, is that of a hill with one or two bumps on it. To prevent this from happening, the positions where the first derivatives in U and V change sign, are marked and used to limit the range of the relative delay. This function is called ‘over-the-hill protection’. It can be turned on and off by the parameter *CTI_PROTECTION*. Fig. 52 and 53 show the effect of the DCTI function with and without ‘over the hill protection’ when applied to a hill-shaped video pulse. In order to detect a hill the second derivative of the input function is checked for a sign change (zero crossing), see fig. 51. When a hill is detected the distance to that hill for each directly surrounding pixel is calculated. The drive signal will be dynamically limited to this distance for each pixel. The result is that DCTI is prevented from ‘looking over the hill’. For hill detection a threshold can be set by the 4-bit parameter *CTI_SUPERHILL*.

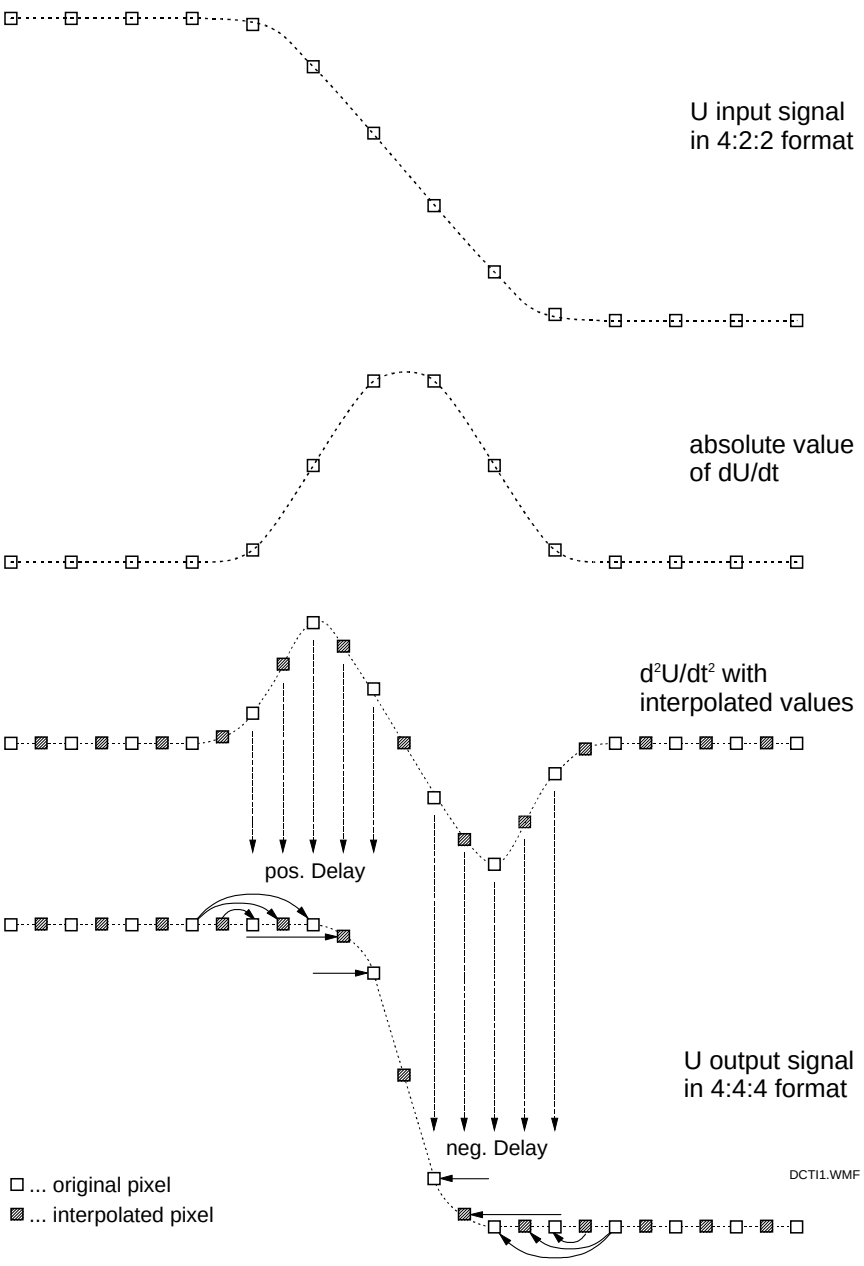
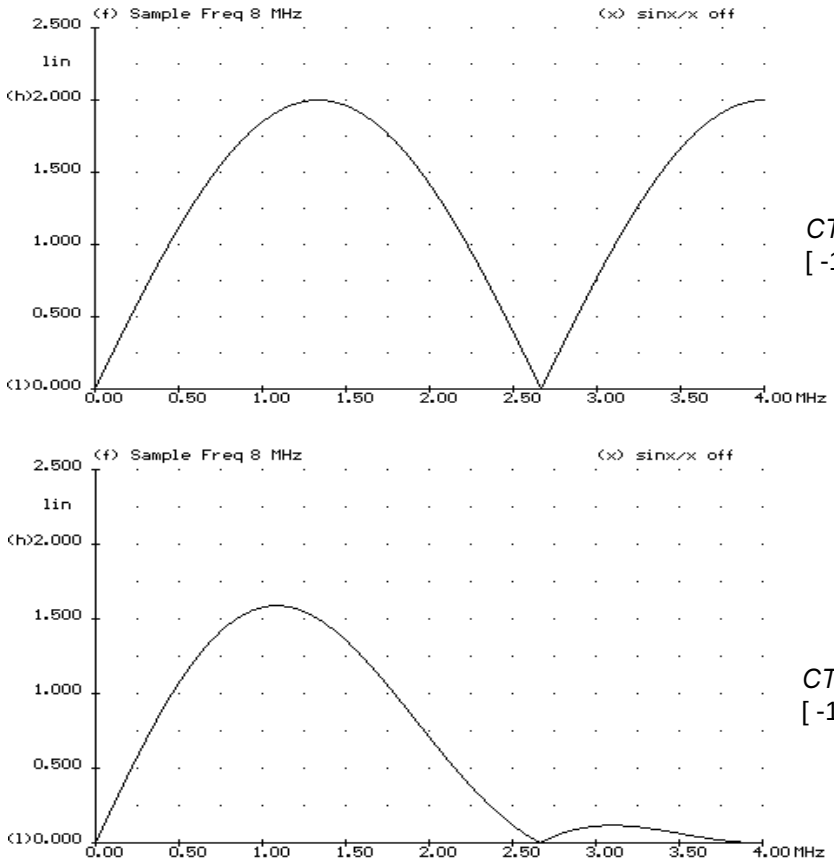


Fig. 47 DCTI basic operating principle

The ‘hill protection‘ function still produces artefacts for signal transitions where the first derivative does not change sign, i. e. two (or more) positive (or negative) steps following each other. Signals of this kind are handled properly if ‘superhill-protection‘ is turned on by parameter *CTI_SUPERHILL* = 1. The behavior of DCTI with active and inactive ‘superhill protection‘ is shown in fig. 54 and 55. Slight overshooting occurs if the postfilter is turned on.

The postfilter is used to correct upsampling in case DCTI is not activated (*CTI_GAIN* = 0 and/or *CTI_LIMIT* = 0). In this case upsampling uses only linear interpolation and the output signal shape can be improved by turning on the postfilter. The transfer characteristic is given in the upper curve of fig. 56. The lower curve gives the corrected upsampling characteristic with the postfilter turned on.



CTI_DDX_SEL = 0:
[-1 0 0 -1]

CTI_DDX_SEL = 1:
[-1 -2 -1 1 2 1]

Fig. 48 Transfer curves of the first differentiating filter

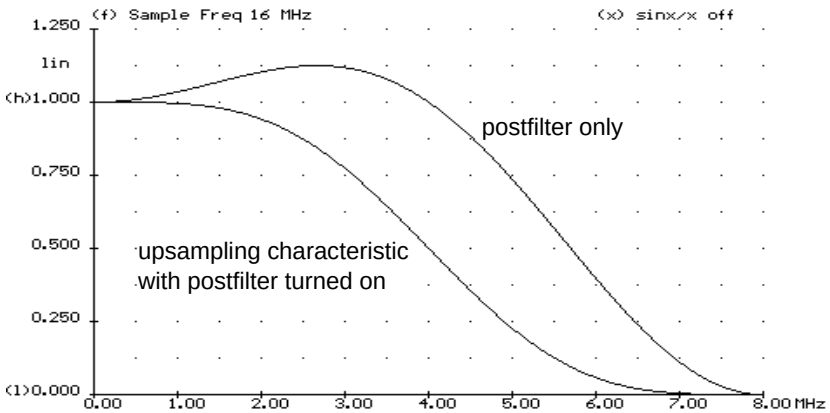


Fig. 56 Transfer curve of postfilter

The DCTI function can further be controlled by the parameter *CTI_SEPARATE* in regard to whether both signals U and V are processed together or each one separately. In case of *CTI_SEPARATE* = 0 (off) a steep transition in either signal is sufficient to activate the data path delay variation. This setting is based on the fact that most color

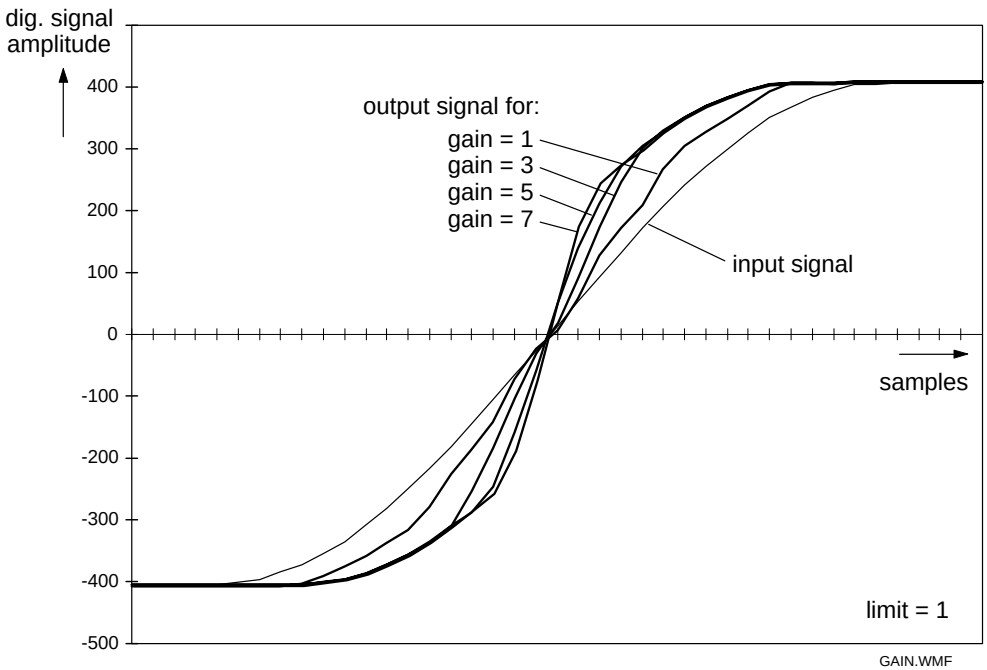


Fig. 49 DCTI with variation of gain for a limit setting of 1

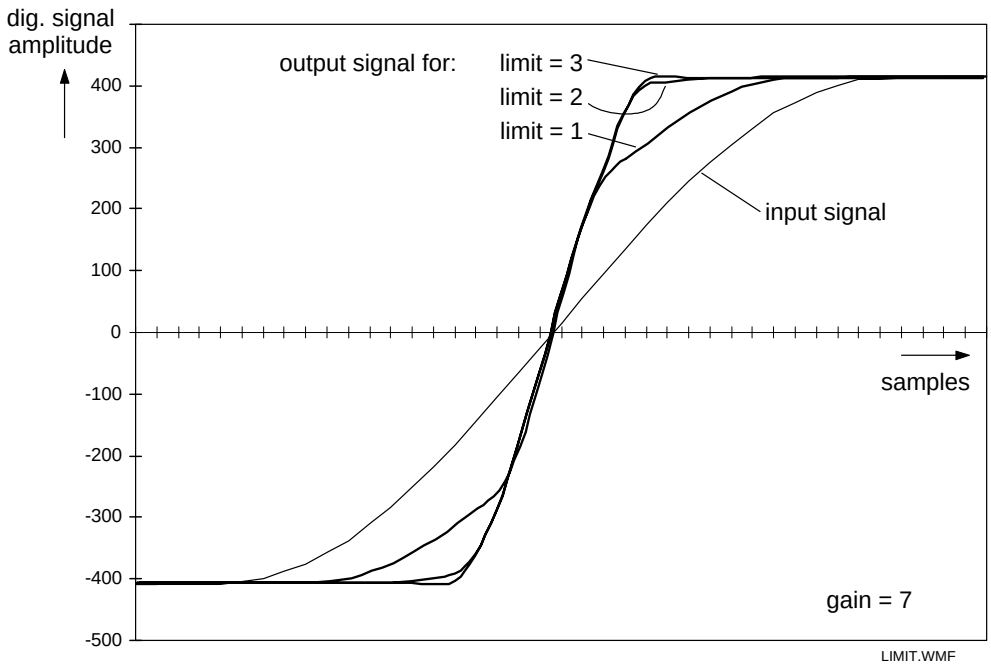


Fig. 50 DCTI with variation of limit for a gain setting of 7

transients involve both signals U and V. And if one of the signals stays constant, a data path variation would do no harm.

In case of *CTI_SEPARATE* = 1 (on) each signal is processed separately. This setting is favorable if the transitions in both signals do not occur at the same time. Common processing then would give false colors which can be annoying. An example for processing such signals is given in fig. 57 and 58.

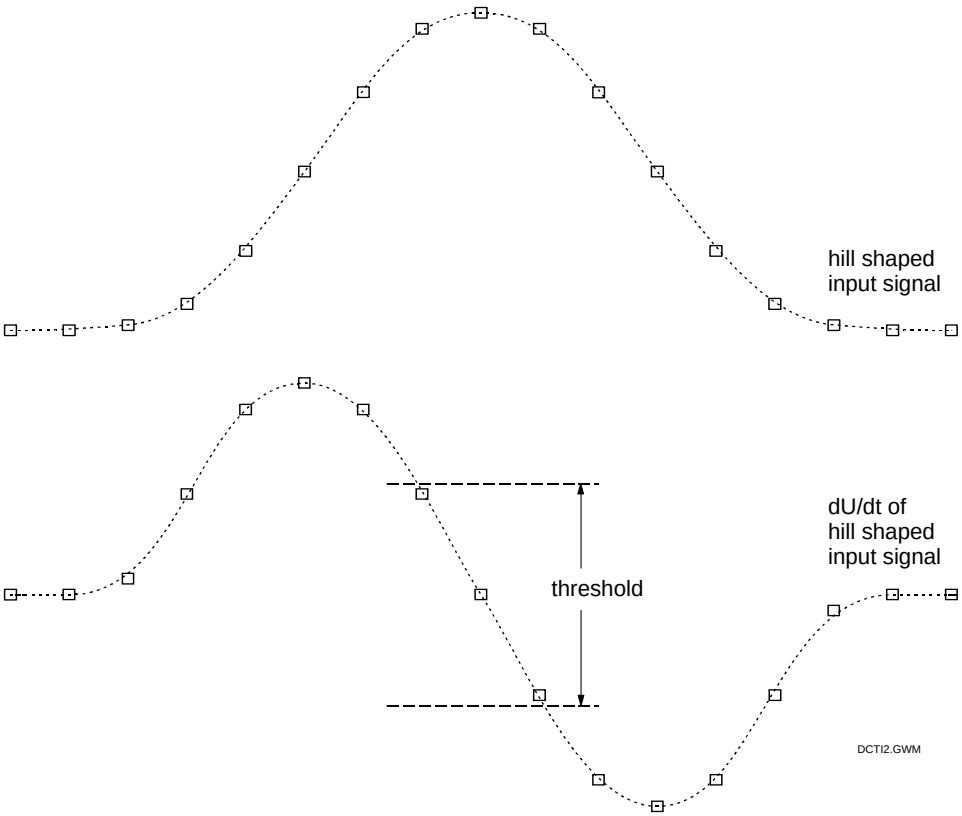


Fig. 51 Principle of hill detection

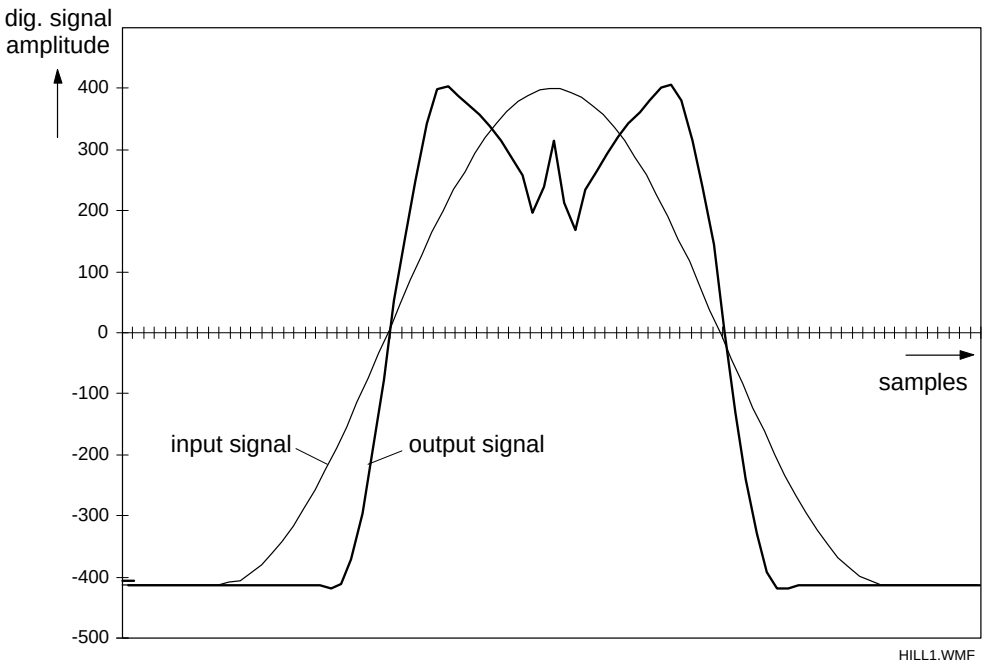


Fig. 52 DCTI without ´over-the-hill protection´

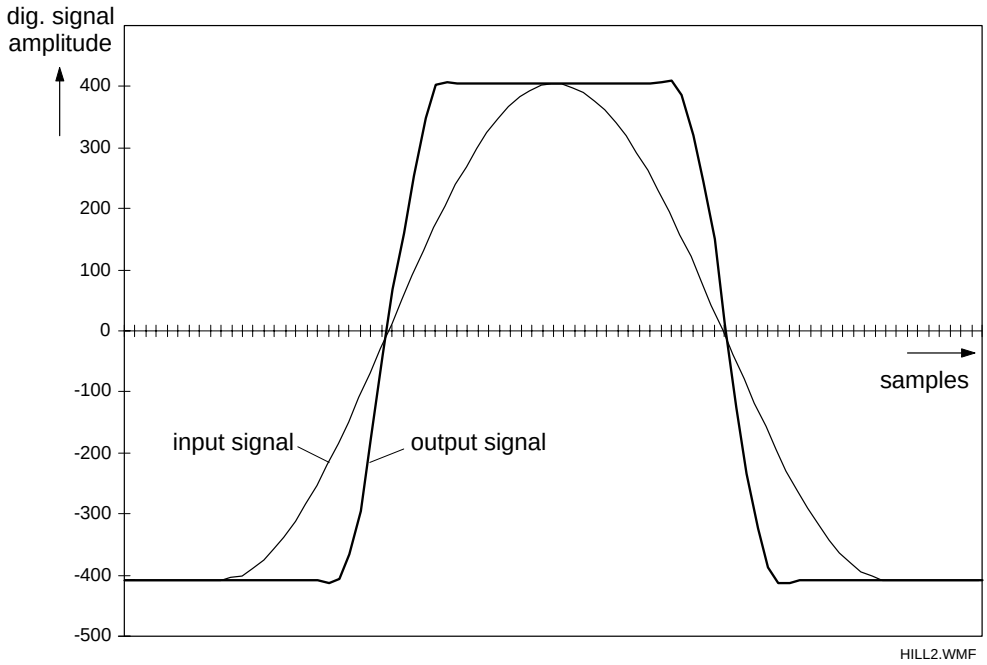


Fig. 53 DCTI with over-the-hill-protection

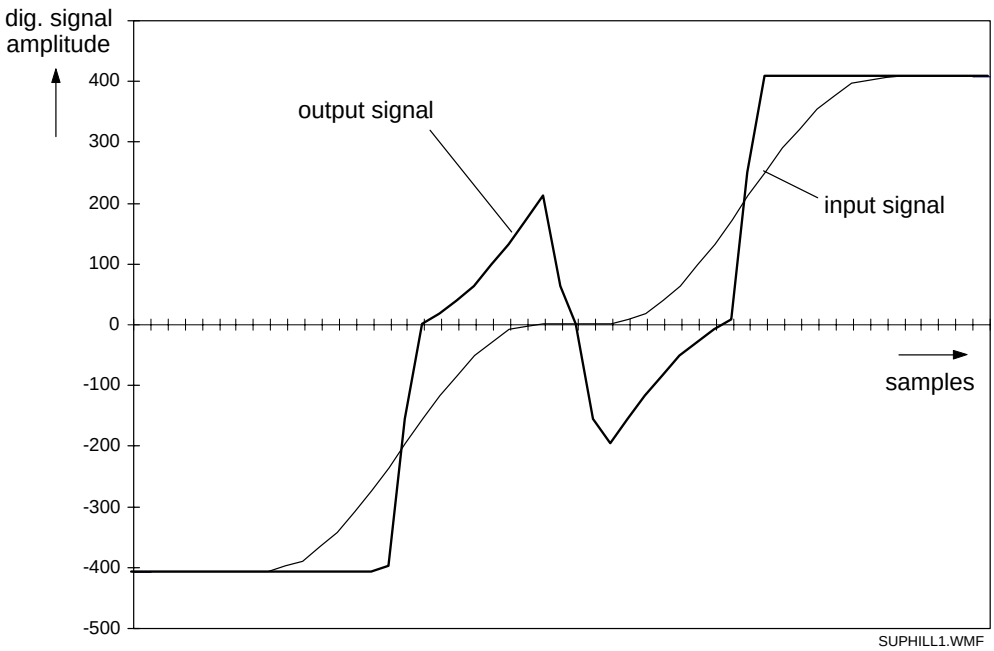


Fig. 54 DCTI with superhill-protection off

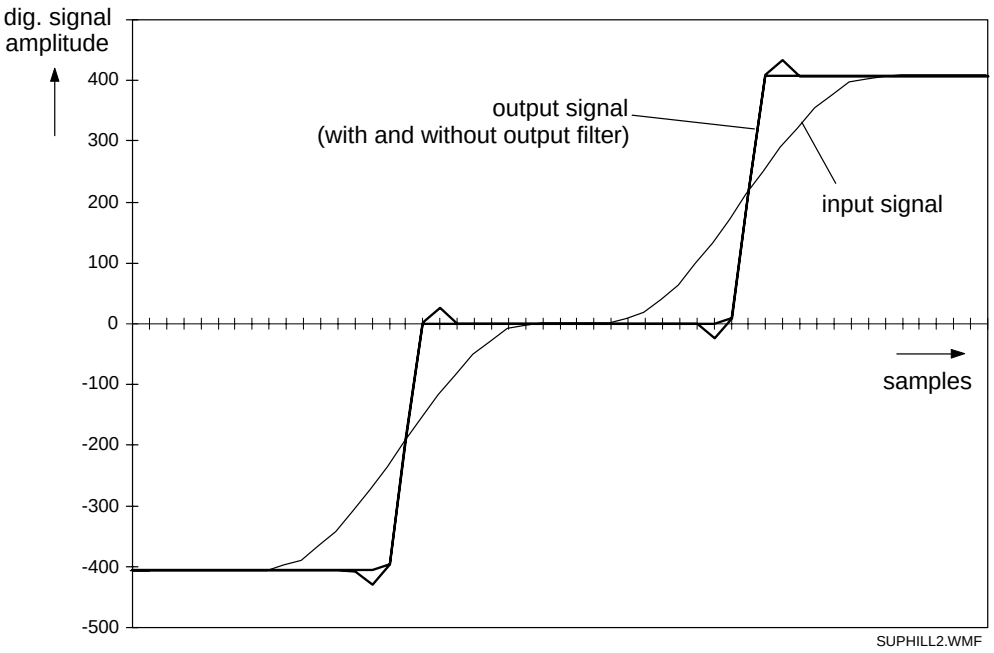


Fig. 55 DCTI with superhill-protection on

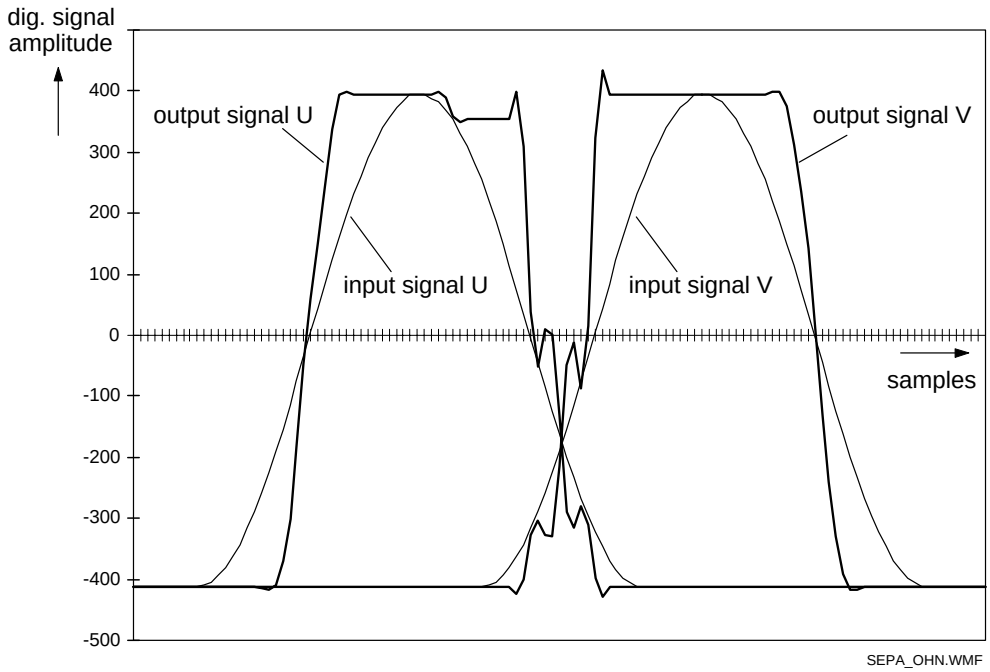


Fig. 57 DCTI with common processing of both signals (*CTI_SEPARATE* = 0)

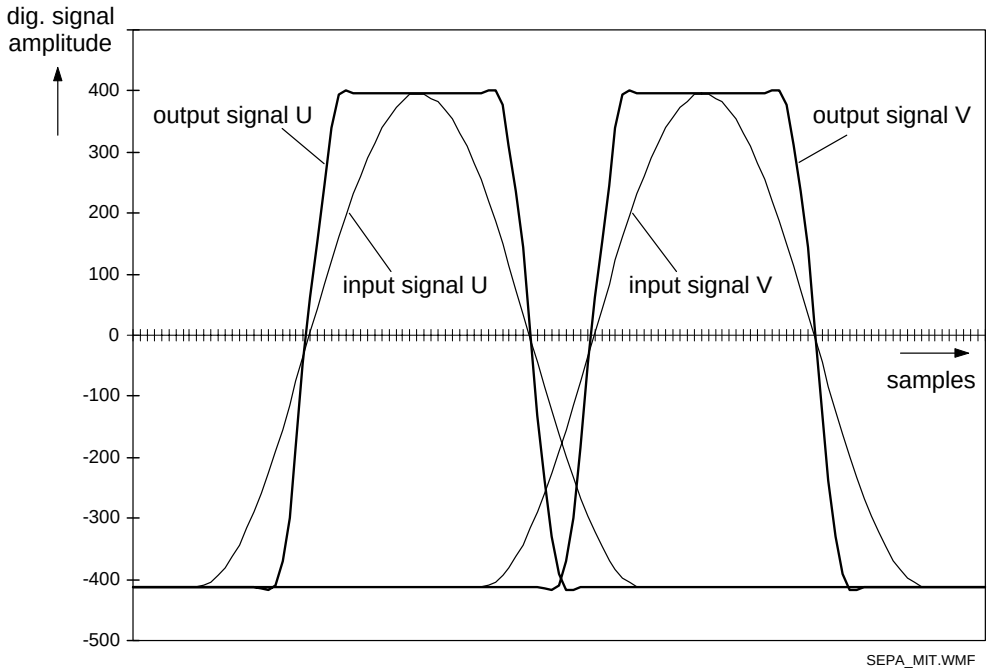


Fig. 58 DCTI with separate processing of both signals (separate = 1)

3.3.6 Border blanking

The border and blanking processing is operating at the 4:4:4 level, just in front of the D/A converters. Here a blanking window can be generated and within this window a border window. The blanking window is used to blank the non-visible part of the output to the clamp level. The border window is the visible part of the video that contains no signal, such as the sides in compression mode, this part can be programmed to display any luminance or color level in 8 bits accuracy, and also pixel repetition is possible here. In case of multi-PIP this block can generate separation borders in horizontal and vertical direction.

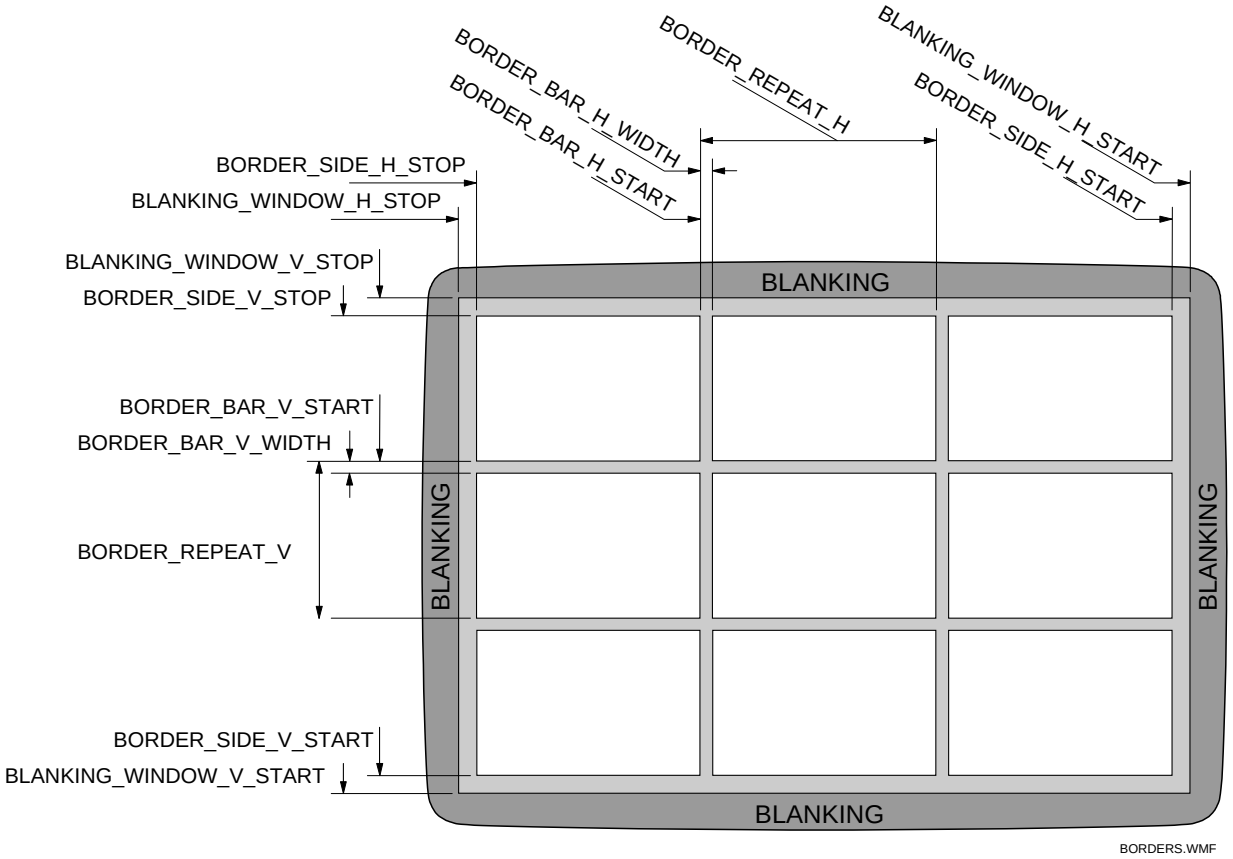


Fig. 59 Generation of blanking, borders and bars

The blanking window is defined in the horizontal direction by the parameters *BLANKING_WINDOW_H_STOP* and *BLANKING_WINDOW_H_START*. Vertically the resp. parameters are *BLANKING_WINDOW_V_STOP* and *BLANKING_WINDOW_V_START*.

Borders are defined similarly. The parameters *BORDER_SIDE_H_STOP* and *BORDER_SIDE_H_START* are used for horizontal definition and the parameters *BORDER_SIDE_V_STOP* and *BORDER_SIDE_V_START* for vertical definition. The starting and stopping points for the border should be defined to be within the non-blanked area for the border to become visible.

Both in horizontal and vertical direction bars can be defined. Horizontally this is done by the parameters *BORDER_BAR_H_START* and *BORDER_BAR_H_STOP* and vertically by the parameters *BORDER_BAR_V_START* and *BORDER_BAR_V_STOP*. The width of the bars is determined by *BORDER_BAR_H_WIDTH* and *BORDER_BAR_V_WIDTH*. If more than one bar is wanted horizontally or vertically, this can be specified by the parameters *BORDER_REPEAT_H* and *BORDER_REPEAT_V*.

For the horizontal registers defining blanking, borders and bars 8 bits are used, so the accuracy is 4 pixels per LSB. The vertical registers are 9 bits wide, so the resolution is 1 line per LSB.

The luminance and color of the borders and bars is defined by the registers *BORDER_Y*, *BORDER_U*, and *BORDER_V*. These are 8-bit registers. *BORDER_U* and *BORDER_V* expect values in 2´s complement. Activating pixel repetition is done by setting bit *PIXEL_REPETITION* to "1". In this case the last luminance and color value of the line is frozen, so the right and left side of active video contain the rightmost pixel column as picture content.

In order for blanking, borders and bars to become active the respective enable bit must be set in the display control byte: *ENABLE_RESET_BLANK*, *ENABLE_BORDER_H_SIDE*, *ENABLE_BORDER_H_BAR*, *ENABLE_BORDER_V_SIDE*, *ENABLE_BORDER_V_BAR*. Moreover the bit *BAR_ARRAY_TRANS* must be set to "0" (mashing). In case of pixel repetition this mode has to be enabled by the bit *PIXEL_REPETITION*.

4. Controlling

Besides the signal processing part the SAA4978H contains a control part the main blocks of which are the PLL for clock generation, the programmable signal positioner (PSP) to generate the internal and external timing signals, and an 80C51 type microcontroller. Fig. 60 shows the block diagram.

4.1 PLL

There is only one PLL within the SAA4978H which generates the clocks. Other scan conversion concepts use separate PLLs for the acquisition and display side. In such a case the acquisition PLL is required to be fast-locked to the video source in order to be able to follow any time base changes. The display PLL on the other hand has to be a very slow one in order to generate a stable picture on the screen. Now the concept in the SAA4978H is such that there is only one very slow PLL for a stable display clock, and any time base errors (from a VCR e. g.) are absorbed by a sample rate converter which acts as a variable video delay, see chapter 3.2.10 “Time base correction / Sample rate conversion” on page 20.

The PLL is locked to the input (or acquisition) horizontal sync HA. It consists of phase detector, loop filter, oscillator and divider. A functional block diagram is given in fig. 61.

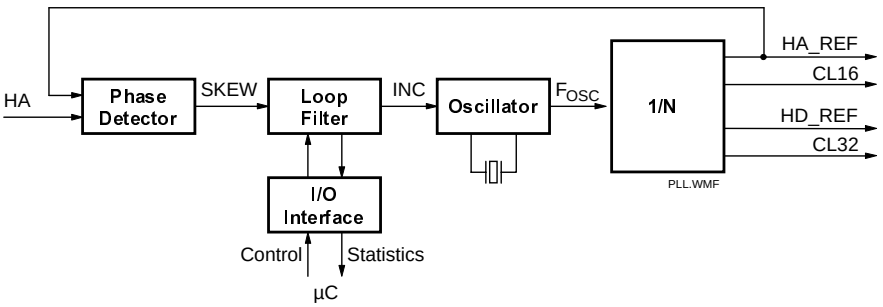


Fig. 61 Block diagram of the PLL

The oscillator generates the system clock F_{OSC}. It is digitally controlled and synthesizes its output frequency from a reference crystal frequency. The clock frequency is divided down to the line frequency HA_REF which is coupled in frequency and phase to the incoming line frequency HA by the phase detector and loop filter. The dynamic behavior of the loop is controlled by independent settings of loop gain (parameter *PLL_CK*) and damping (parameter *PLL_CD*) via the micro controller. Although the time base corrector TBC compensates time base errors it nevertheless is advantageous to have a somewhat faster loop in case of VCR. For PAL+ or normal operation an ultra slow loop is desired. A change of parameters is possible at the beginning of each new frame.

Another control input is *PLL_OFF* by which the phase locking is turned off and the clock is frozen at the momentary frequency. *PLL_OPEN* opens the loop so the oscillator runs at its nominal frequency.

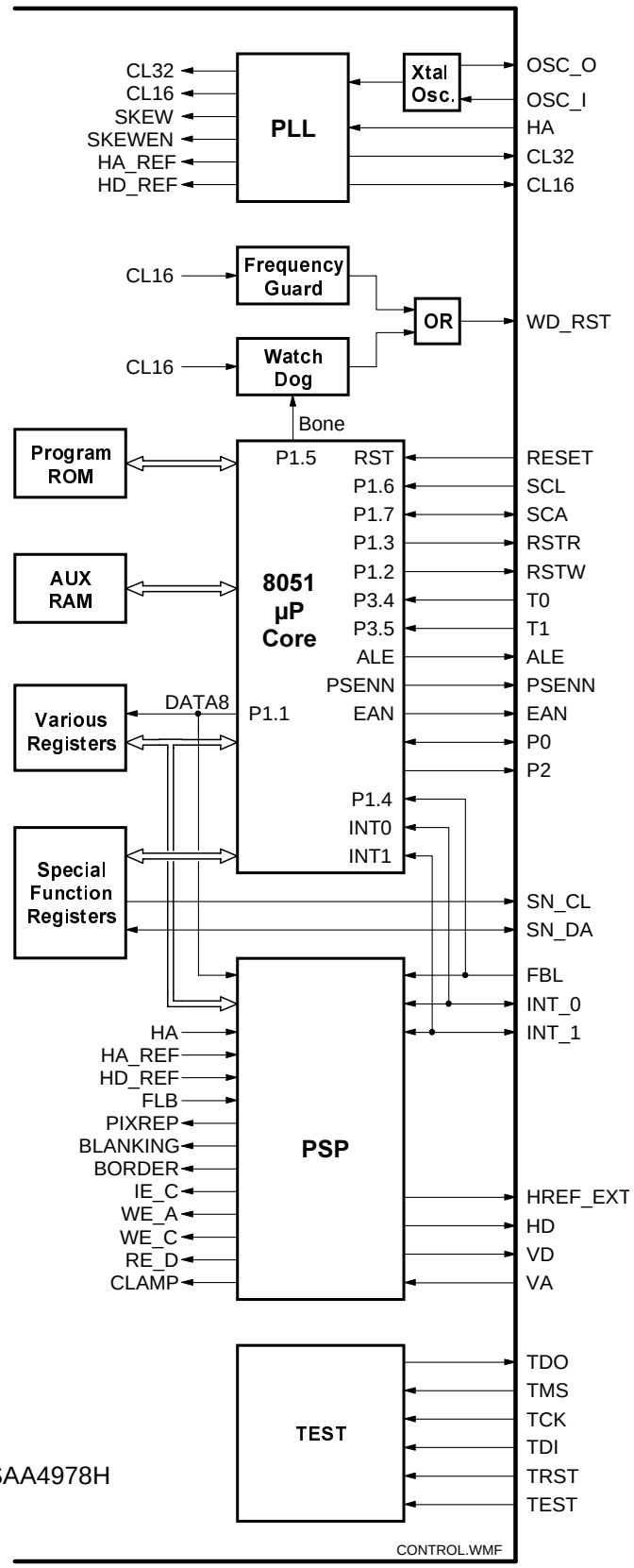


Fig. 60 Control part of the SAA4978H

There are other control inputs which however serve test purposes: *PLL_IDTO* adds an offset to the increment for the oscillator, in this way the loop response can be tested. *DO_SNAP* is an enable signal for monitoring important PLL-signals as serial output data on a special IC-pad. *PLL_SKEW_DELAY* delays the transfer of the skew data in multiples of 512 clocks.

The PLL block allows to retrieve several statistical data which can be evaluated by the microprocessor. *PLL_PE_MAX* and *PLL_PE_MIN* give the maximum and minimum phase error during a field, *PLL_PE_SUM* and *PLL_PE_SABS* give the sum of skew and sum of absolute skew in a field (only the 16 MSBs). In *PLL_INC_OFFSET* the actual increment-offset for the oscillator can be read. In register *PLL_CKA_VALUE* the actual gain value K can be read, and the bit *PLL_ADAPT_STATUS* informs about the current status of the PLL: "1" means that the PLL is still adapting (unlocked state), "0" indicates that the PLL has locked.

4.2 Programmable Signal Positioner

Timing signals that need to be changed depending on input signal, operating mode or user interaction are generated by the Programmable Signal Positioner (PSP). A block diagram is shown in fig. 62. The PSP can be divided into an acquisition part and a display part.

4.2.1 Horizontal Acquisition Timing

The acquisition related signals are based on the acquisition counter. This is a 10 bit counter running on the acquisition clock CL16 of 16 MHz. Each clock cycle is equivalent to one pixel in the horizontal line. The counter is reset by the horizontal reference signal HA_REF at the beginning of each video line. HA_REF is generated by the PLL.

The current value of the horizontal counter is compared to a set of registers which are loaded by the microcontroller. The comparator outputs set or reset flip-flops which generate the various signals.

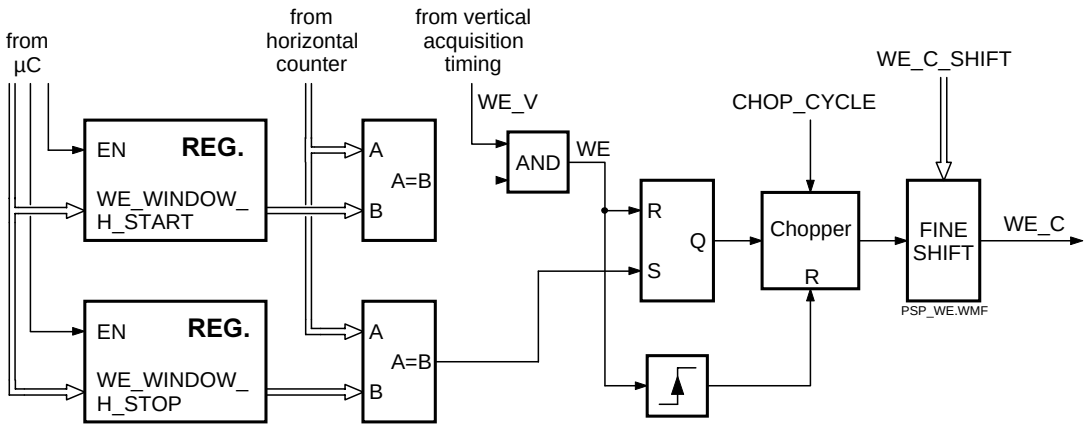


Fig. 63 Generation of the WE_C signal

As an example fig. 63 depicts the generation of the WE_C signal. This signal is used to control the writing of data into the scan-conversion memory. It is in HIGH state during the active part of the video line. The start and stop values of the pulse are loaded into registers which are compared to the running horizontal counter value. If equivalence is detected, an RS-flip-flop is set or reset, its output basically is the wanted pulse. Since in this case the signal has to be LOW during the vertical blanking period, the start of the pulse is controlled by the signal WE_V (generated in the vertical acquisition timing section), which is HIGH only during the active part of the field.

In this case of WE_C a chopper circuit is added which is controlled by the parameter *CHOP_CYCLE*. It is used in multi-PIP decimation mode when the picture has to be reduced in size. For a reduction to half the size the WE-signal is turned on and off clockwise alternating, for a reduction to one third the on/off ratio is 1:2 and for one forth

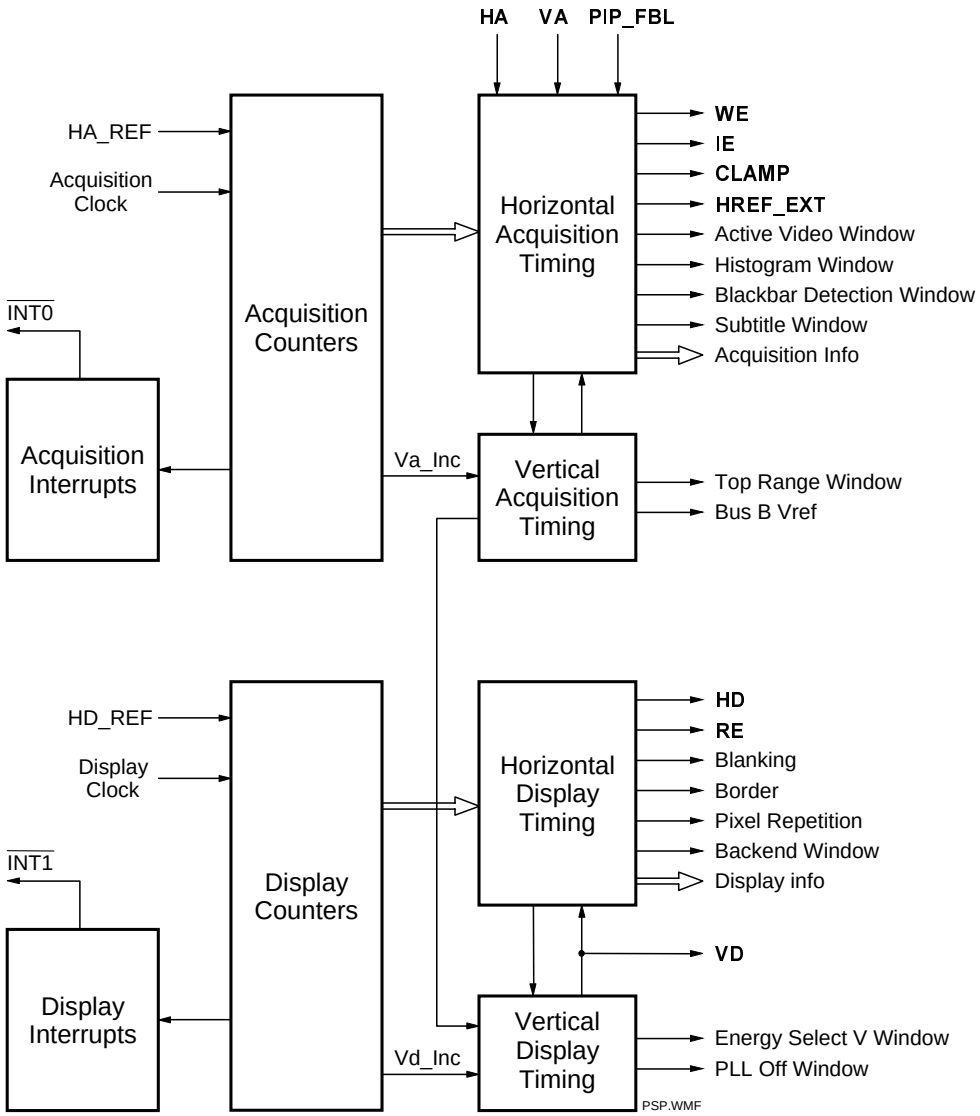


Fig. 62 Block diagram of the PSP (Programmable Signal Positioner)

the original size the ratio is 1:3 (see fig. 64). The chopper sequence is synchronized to the start of the line by detecting the positive-going edge of the WE signal.

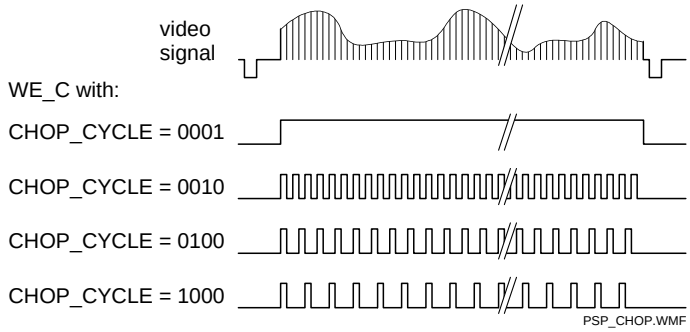


Fig. 64 Chopping the WE_C signal in multi-PIP mode

The start and stop registers for this pulse are 8 bits wide. Therefore the length of the pulse can be defined in increments of 4 clock cycles or pixels only. (At a clock frequency of 16 MHz the line length of 64 µs equals 1024 clock cycles.) In order to have more precision in positioning the WE window a fine delay is added which allows to shift the complete signal for 0..3 clock cycles. This fine delay is defined by the parameter *WE_C_SHIFT*.

WE is used as write enable memory control. It must be HIGH for the memory to operate and have its internal address counter increment. IE (input enable) however defines whether the memory´s input data is stored or the old content of the memory cells is retained. Therefore in normal operation IE can remain in HIGH state for the complete field or should at least be HIGH before WE is turned on. A possible definition of the acquisition window concerning the control of WE and IE is given in fig. 65.

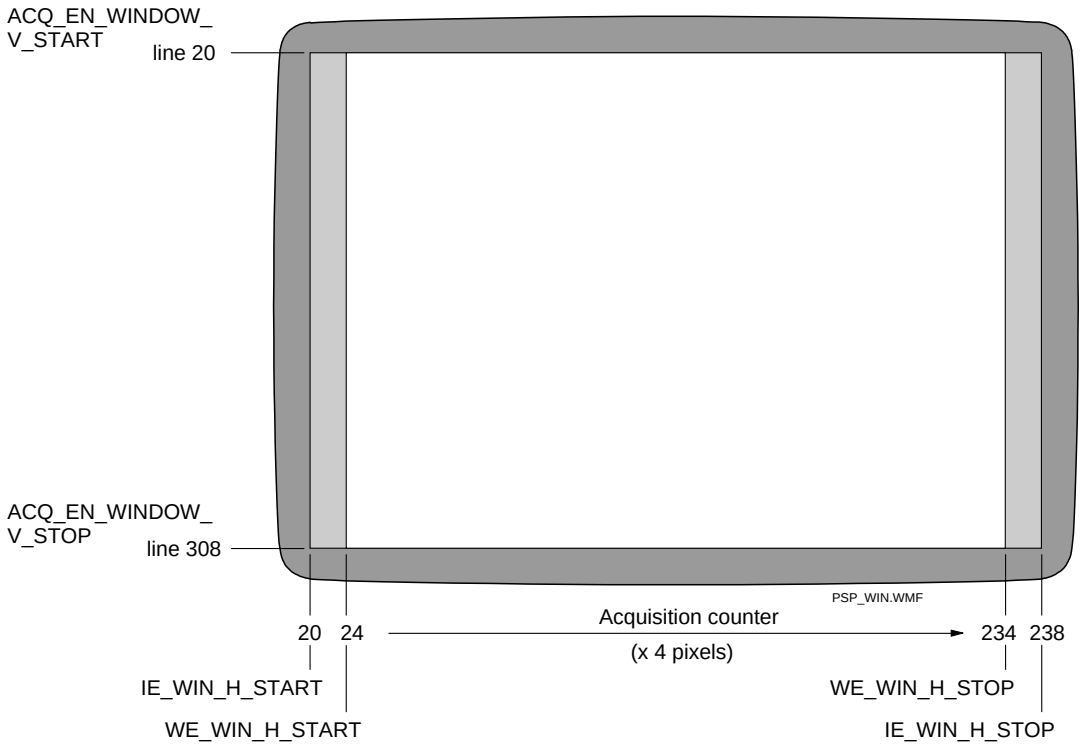


Fig. 65 Defining the acquisition window

In multi-PIP mode IE defines the part in memory that is allocated to the actual PIP-picture. It must be in HIGH state for the new data to be stored in memory and remain LOW where the main picture is to be retained. The start and stop values of IE are set by the parameters *IE_WINDOW_H_START* and *IE_WINDOW_H_STOP*. Both values can be define in steps of 4 clock cycles only, but a fine shift of 0..3 pixels (effecting both the rising and the falling edge) is possible by setting the parameter *IE_C_SHIFT*. IE as well as WE are vertically enabled by setting the parameters *ACQ_EN_WIN_V_START* and *ACQ_EN_WIN_V_STOP*.

HREF_EXT is an output pulse which can be used as horizontal reference for external devices. The signal can be programmed via the registers *HREF_EXT_START* and *HREF_EXT_STOP*.

The signal CLAMP is derived from a separate pixel counter. This counter can be reset by HA or by HA_REF (selectable by the parameter *SEL_HA_CLAMP*). The reason is that the clamping pulse must have a certain delay related to HA (and not HA_REF which is derived from the PLL). Especially in VCR mode there can be a considerable phase difference between HA and HA_REF. In case of a stable input signal (standard TV reception) the counter reset can be switched to HA_REF and thus prevent a possible clock jitter on the clamping pulse. The

counter has 8 bits only and runs on the acquisition clock of 16 MHz, therefore the clamping pulse can be generated only in the first quarter of a video line.

Another output pulse of the horizontal timing block is the Va_Inc signal, it is used to increment the vertical counter.

Besides generating signals the horizontal and vertical counters are also used for phase measurement purposes. The measurement results can be read by the micro controller (acquisition info). As an example fig. 66 shows how the horizontal position of the external FBL pulse is measured. The falling and rising edges are detected and generate an ENABLE pulse each to store the current position of the horizontal counter in a register. The position of the edges can then be read by the microcontroller as *PIP_RISING_EDGE_POS* and *PIP_FALLING_EDGE_POS*.

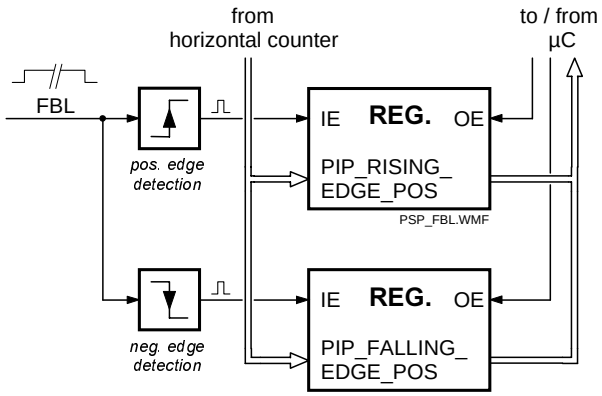


Fig. 66 Measuring pulses by the PSP

4.2.2 Vertical Acquisition Timing

The vertical counter is a 9 bit counter which is reset by the external VA signal. It is incremented once per line by the pulse VA_Inc generated in the horizontal timing part.

Fig. 67 explains the reset function of the vertical counter. In order to ensure stable operation a vertical window is defined and only VA pulses within this window will reset the vertical counter. If VA falls into this window the positive edge will generate the VA_REF pulse which resets the counter. If VA is missing or too late then at the end of the vertical window an artificial VA_REF pulse is generated. Too early or multiple VA pulses are ignored. Start and end of the vertical window are user definable by the registers *VA_SYNC_WINDOW_START* and *VA_SYNC_WINDOW_STOP.*

The vertical counter works like the horizontal counter. Its value is compared to a set of registers loaded by the microprocessor. The comparator outputs are used to generate signals for creating vertical pulses or windows, but which also are used as gating signals for horizontal signals which are to be turned off during the vertical blanking period for example.

4.2.3 Horizontal Display Timing

The display related signals are based on the display clock. If the SAA4978H runs in scan conversion mode then the display clock is CL32 of 32 MHz. In case of $1f_H$ processing the back end part runs on CL16, this can be selected by the parameter *SEL_1FH*.

Horizontal display is controlled by a 10 bit pixel counter running on the display clock. Normally the counter is reset by the rising edge of *HD_REF*, a signal generated by the PLL. This reset however can be disabled by setting the parameter *HD_CNTR_RST_BY_HDREF* to LOW and thus put the display part into a self-contained mode (generator mode). This can be advantageous in case of generating OSD[3] without an input signal being

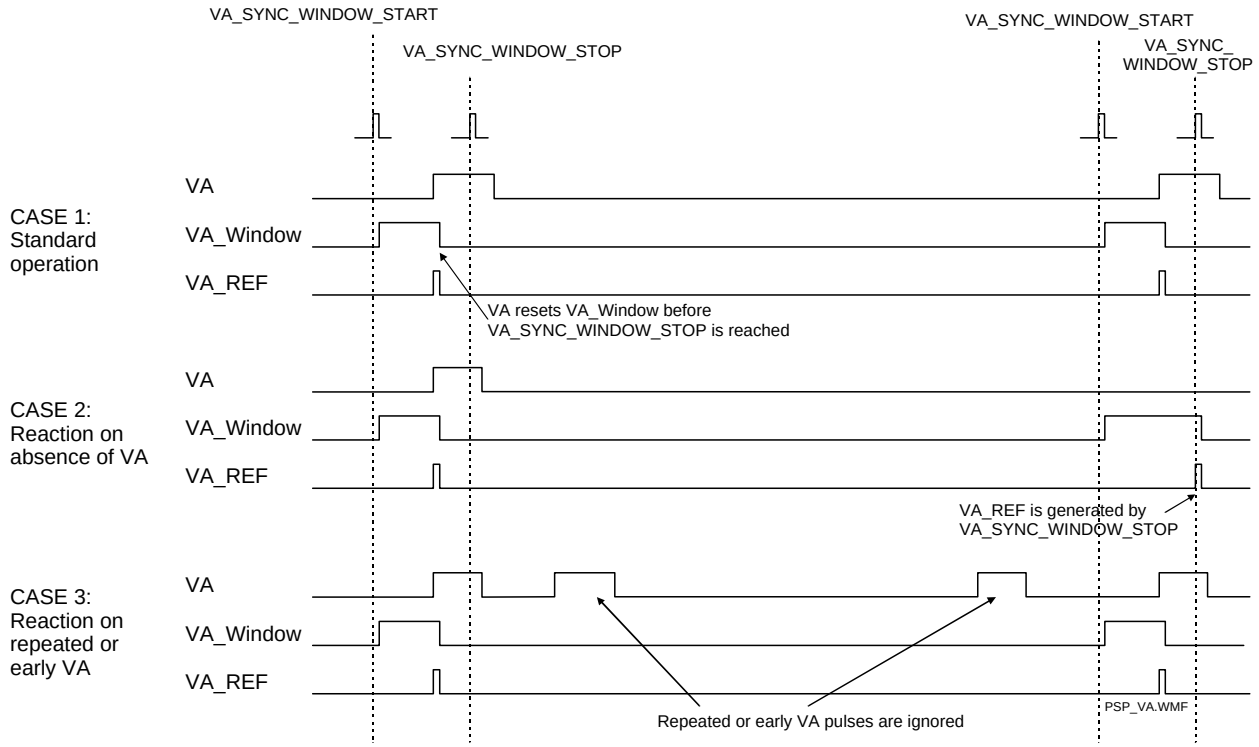


Fig. 67 Vertical counter reset

present at the acquisition side. The line length in generator mode is set by the parameter *H_EXT_POS*. Fig. 68 shows the reset generation of the horizontal display counter.

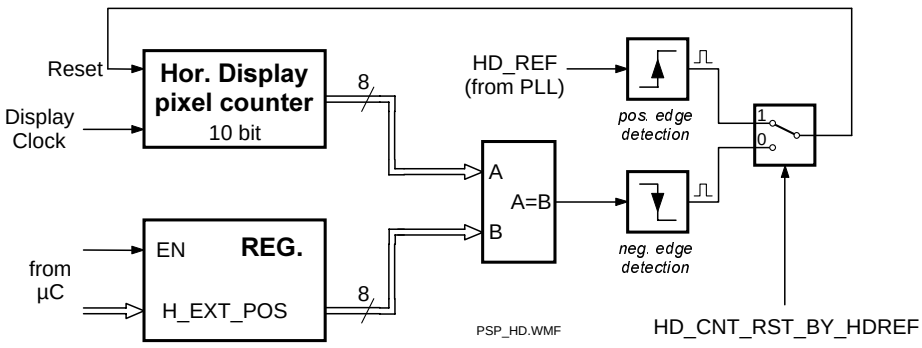


Fig. 68 Reset generation of the horizontal display counter

HD is an external signal used as reference for the horizontal deflection. The pulse is generated by defining the start and stop position by setting the registers *HD_START* and *HD_STOP*.

RE is the memory read enable signal for the BUS_D data from the scan conversion memory. Horizontally it is defined by *RE_WINDOW_H_START* and *RE_WINDOW_H_STOP*. The 8 bit registers allow a definition in steps of 4 pixels only, therefore a fine shift of the complete pulse is possible by the parameter *RE_SHIFT* in the display

3. OSD ... on-screen display

control register. During the vertical blanking period the signal is suppressed, so a vertical window is defined by the parameters *RE_WINDOW_V_START* and *RE_WINDOW_V_STOP*.

4.2.4 Vertical Display Timing

Vertical display is mainly controlled by a 9 bit line counter. This counter is reset by the microcontroller at the beginning of the field and incremented by the pulse VDInc from the horizontal display timing block. For progressive scan for example this pulse can be divided by 2 (VDIncDiv2) by setting control parameter *DIVIDE_VD_INC* to HIGH.

The vertical display timing block generates the vertical display pulse VD. Start and stop are set by the microcontroller by accessing address location $38C_H$ (*TRIGGER_FLYBACK*) and $38D_H$ (*TRIGGER_SCAN*). The rising and falling edges are synchronized by the pulse VD_HOR_POS. This pulse is generated in horizontal display section, and its horizontal position can be programmed by the parameter *VD_HOR_POS* (see fig. 69).

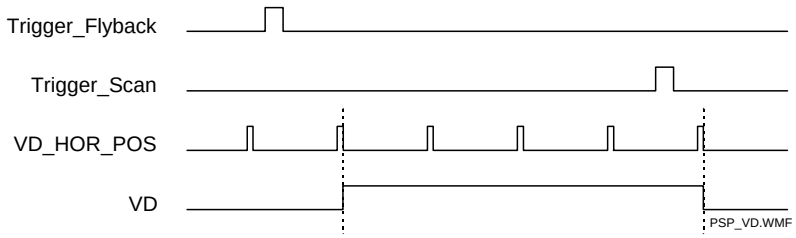


Fig. 69 Generation of the vertical display pulse VD

Other signals created in the vertical display timing block include the vertical enable pulses for RE, blanking, back end window, energy select window, pll_off_window and border generation. These enable signals are combined with the respective signals from the horizontal timing block to form the window signals.

4.2.5 Interrupts

Two interrupts for the processor can be programmed: $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$. $\overline{\text{INT0}}$ is generated by the vertical acquisition timing block. It can either be made coinciding with the LOW going edge of WE (Interrupt_WE) or be activated at any value of the vertical acquisition counter (Interrupt_VA). $\overline{\text{INT1}}$ is generated by the vertical display timing block at a programmed value of the vertical display counter. These interrupts are used to start various internal tasks like determining input field length, interlace, odd/even field or 50/60 Hz system, but also for loading specific registers in order to generate timing signals (e. g. the vertical display pulse VD).

4.3 Microprocessor

The SAA4978H has an 8051 microprocessor core built in which includes RAM and ROM. For development reasons an external ROM with up to 64 kB can be accessed. Also an external emulator can be connected.

The main difference to most existing 8051 derivatives is:

- 768 byte of auxiliary RAM of which 128 bytes are accessible by the subtitle detection unit also.
- Interrupt vector address for I^2C is 33H.
- On chip ROM code protection.
- SNERT at 1 or 2 MBit/s with additional SFRs instead of UART.
- Host interface containing all control registers. Access e.g. via MOVX instruction.

4.4 SNERT

SNERT stands for *Synchronous No parity Eight bit Reception and Transmission*. It is a serial bus used for communication between a micro controller and ICs offering a SNERT interface. In this chapter only an overview is given, for further information consult the application note on this subject.[4]

The SNERT bus follows a master - slave concept. The µC always is the master while the connected devices are the slaves. It uses the µC´s serial interface for transmitting and receiving data. Clock is supplied by pin SN_CL while data is written or read through pin SN_DA (mode 0 of the serial interface). Address and data bytes are transmitted alternatively. There are no device addresses but only register addresses. Each SNERT accessible device has a number of fixed register addresses, and only devices which are not being used together on the same bus can share the same register address. There is a total amount of 256 addresses available.

The bus needs a third signal line to determine the correct address / data sequence as well as to update any readable registers in the devices. This reference signal can either be supplied by the µC, or a generally available signal like the vertical synchronization pulse V_A can be taken.

The bit rate of the SNERT bus is specified as 1 MBit/s for most devices. So this is the default setting of the SAA4978H´s SNERT interface. However the interface can also be set to a bit rate of 2 MBit/s if devices are connected that support this speed. Switching the bit rate is done by the SNERT control bit *MB2*.

There is no special signal indicating data transmission or reception. Rather the register in the device itself is defined as a read or write register. In write mode the µC puts an address and a data word on the bus, in read mode the µC generates only the address and then waits on the data byte from the device.

5. MK10 Application Board

The application board MK10 is designed to test the SAA4978H and demonstrate its features. Fig. 70 gives an overview of the components of the board.

The SAA4978H is connected to a field memory for scan conversion. Since the digital busses are 18 bits wide two field memories of the SAA4955/56 type (12 bit wide I/O) are needed. The memories are fed by the YUV_C bus and return the data at twice the speed at bus YUV_D. The YUV_C/YUV_D busses are split into a two parts. 12 bits are fed through the SAA4956 and a feature connector which - for standard A-A-B-B scan conversion - carries a bridge feeding back the data bit by bit. The remaining 6 bits are processed by the SAA4955 and are returned right on the board.

5.1 Additional features

For additional features like LFR (line flicker reduction) or motion compensation an extra board can be attached to the connector. The feature ICs SAA4990 (PROZONIC), SAA4991 (MELZONIC) and SAA4992 (FALCONIC) all run in 4:1:1 mode and thus only need a 12 bit wide bus. The feature connector therefore only supports data I/O for these 12 bits. If the connector is bridged then the SAA4978H can run in all data modes supported. See table 4 on page 19 for details on data formats.

5.2 Noise reduction of the SAA4956

The SAA4956 is a field memory with an additional noise reduction circuit built in. In contrast to the noise reduction implemented in the SAA4978H this version uses a field based recursion loop. The field memory therefore offers two read ports. One of them runs on the input clock speed, so the data can be used for recursion. The second one runs at twice the speed and outputs the scan converted data.

Fig. 71 shows a block diagram of the SAA4956. The noise reduction function runs in 4:2:2 mode using 8 bits for luminance and 8 bits for chrominance. So any 4:1:1 input data has to be interpolated and formatted to 4:2:2 for processing. At the output of the noise reduction block data is reformatted back to 4:1:1. By this interpolation the

4. Waterholter, Heinrich: The SNERT bus specification, Philips Semiconductors Application Note AN95127, 1996

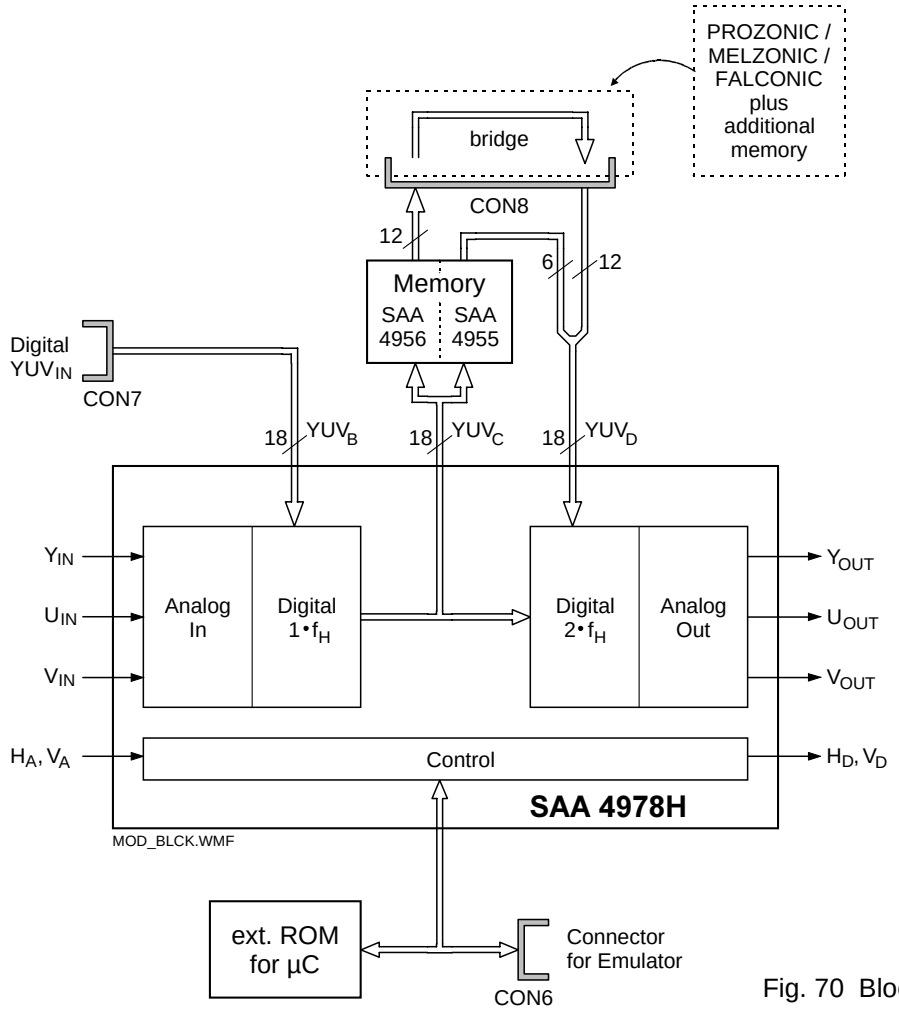


Fig. 70 Block diagram of the MK10 module

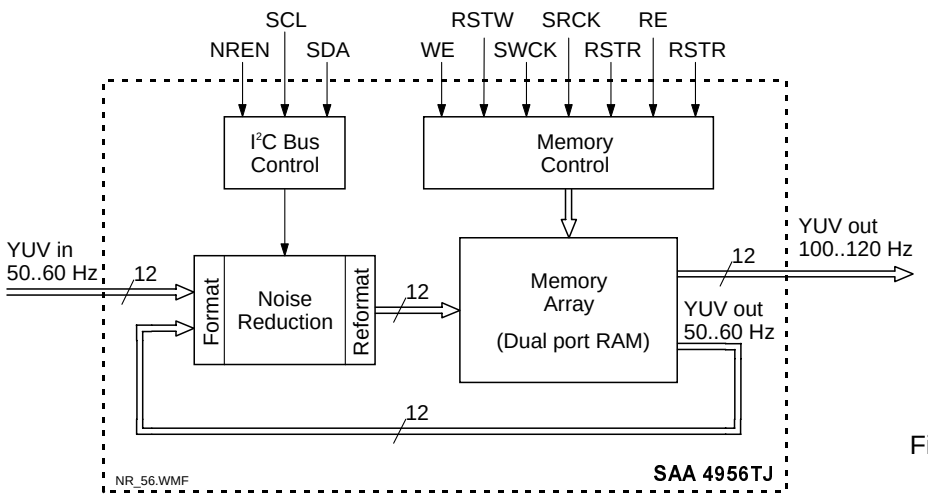


Fig. 71 Block diagram of the SAA4956

chrominance bandwidth is unchanged and stays at 1/4 of the luminance bandwidth. However, the SAA4956 is also able to process higher quality chrominance if it is DPCM coded. So for this mode the noise reduction block includes a DPCM decoder and coder. For further information on the SAA4956 see the data sheet.[5]

5.3 Software

The SAA4978H has an internal program ROM for the µC, however at the time this application note is written the board runs on a program stored in the external ROM. Accessing this external ROM is done by setting a jumper and pulling pin AEN to GND. The software thus supplied is for evaluation purposes. It runs the memories in field repetition mode (A-A-B-B) and gives access to the internal read and write registers by I^2C bus. Future software is planned and will include the control of feature ICs like SAA4990 (PROZONIC), SAA4991 (MELZONIC) or SAA4992 (FALCONIC) which can be connected to the PC board by the feature connector. This software will be accompanied by the resp. User Manual describing its functions.

5.4 Circuit diagram and PC-board

The following pages give the circuit diagram (fig. 72 to 75) and show the layout of the PC-board (fig. 76 to 79). Fig. 80 gives the position of the parts. The SAA4978H has several power pins which are connected to an "analog" supply (+3.3VA) or a "digital" supply (+3.3VD). Decoupling capacitors are spread across a large value range, so a wide-band suppression of supply noise is ensured. The positive supply voltages are fed through an inner layer of the 4-layer board, the other inner layer is completely reserved for ground. This ground layer is depicted in fig. 77 and is shown inversely: copper is white while the black spots represent the holes within the copper layer. The inner two layers (supply and ground) are closely on top of each other form a high-frequency capacitor and thus further improve supply stability.

6. References

[1] SAA4978H, Picture Improved Combined Network IC (PICNIC), Philips Semiconductors datasheet, 1998

[2] SAA4956TJ, 2.9 MBit field memory with noise reduction, Philips Semiconductors datasheet, 1998

[3] Waterholter, Heinrich: The SNERT bus specification, Philips Semiconductors Application Note AN95127, 1996

[4] The I^2C-bus and how to use it, Philips Semiconductors, 1992

5. SAA4956TJ, 2.9 MBit field memory with noise reduction, Philips Semiconductors datasheet, 1998

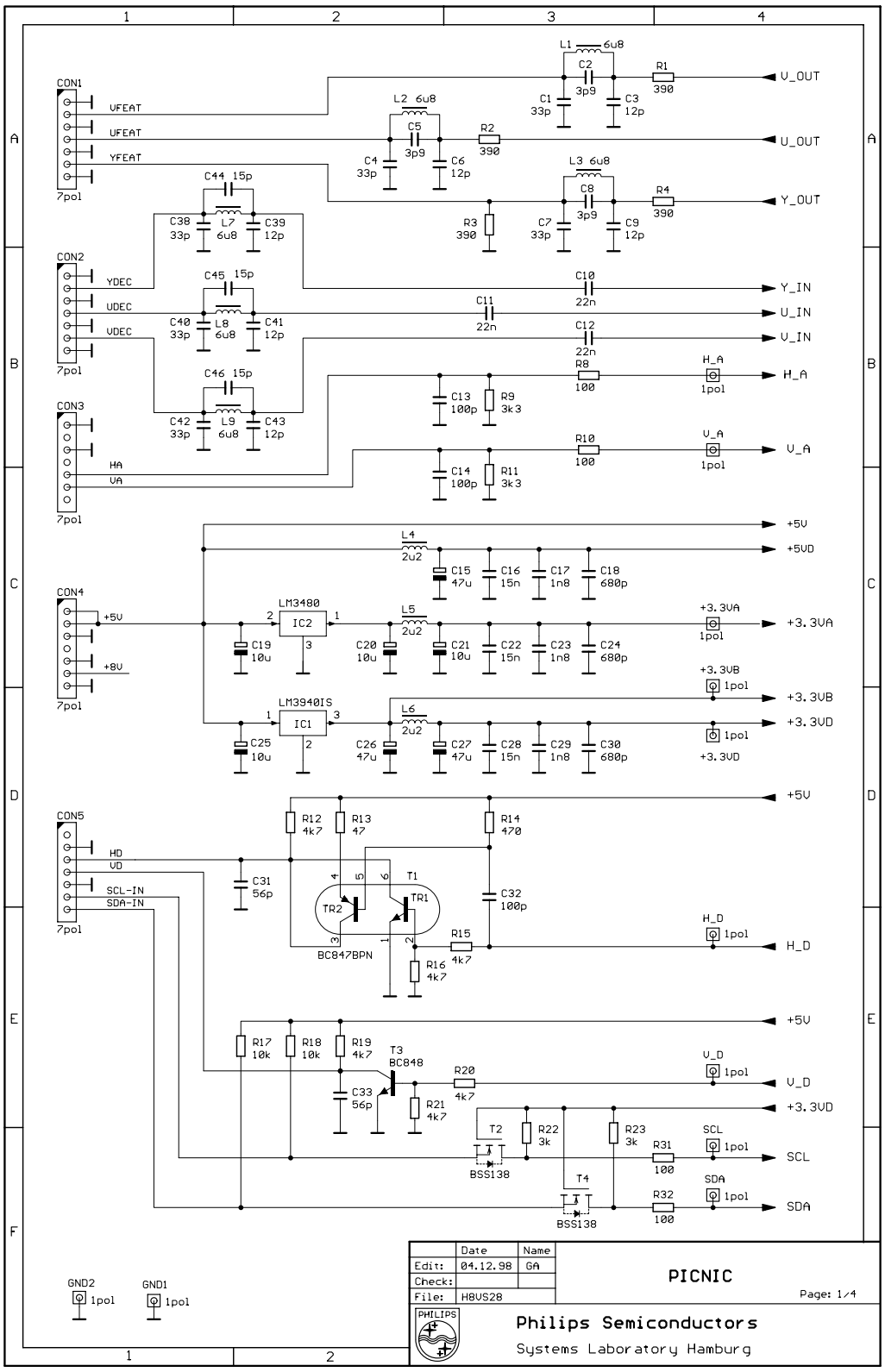


Fig. 72 MK10 module circuit diagram, sheet 1

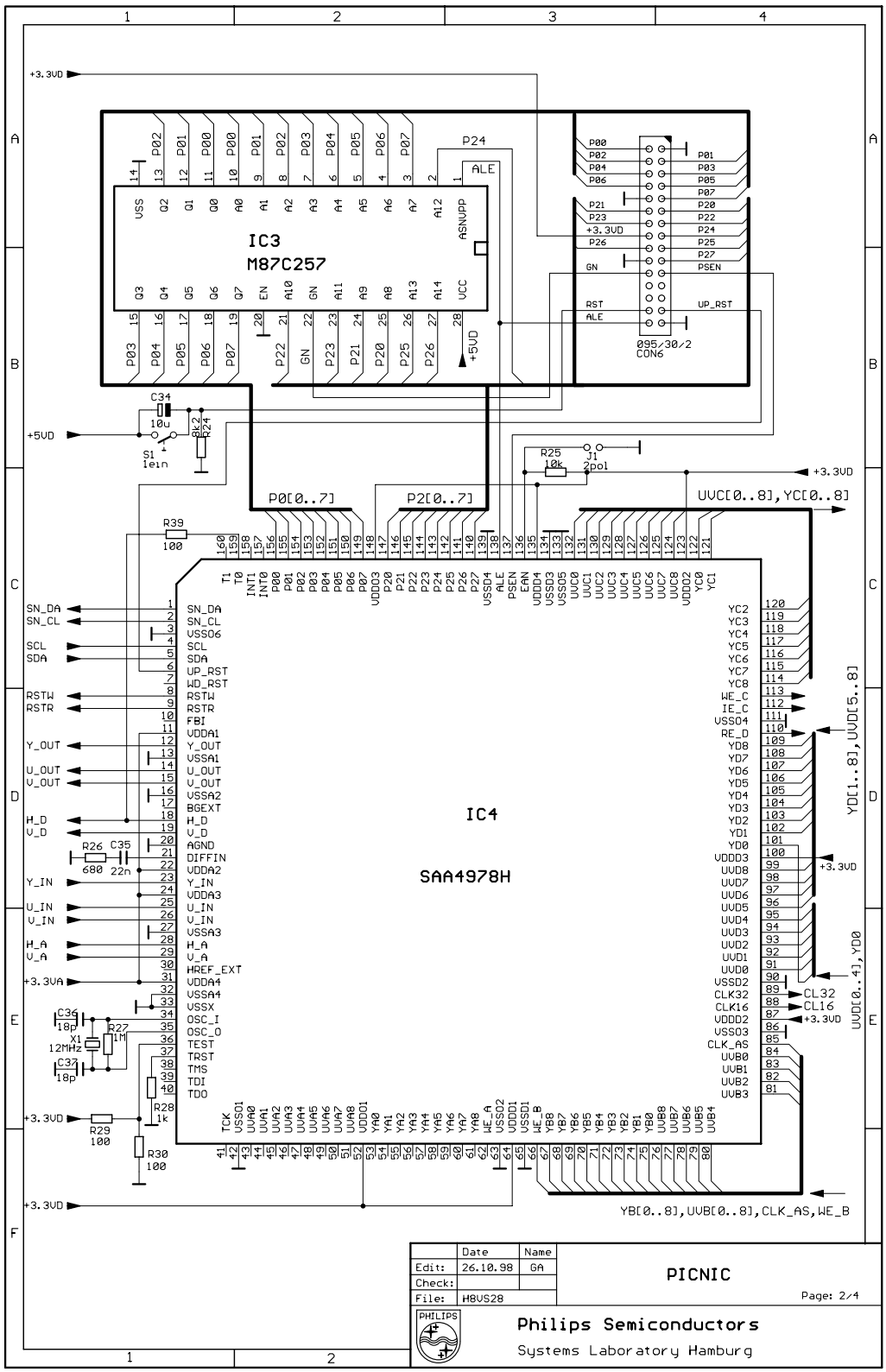


Fig. 73 MK10 module circuit diagram, sheet 2

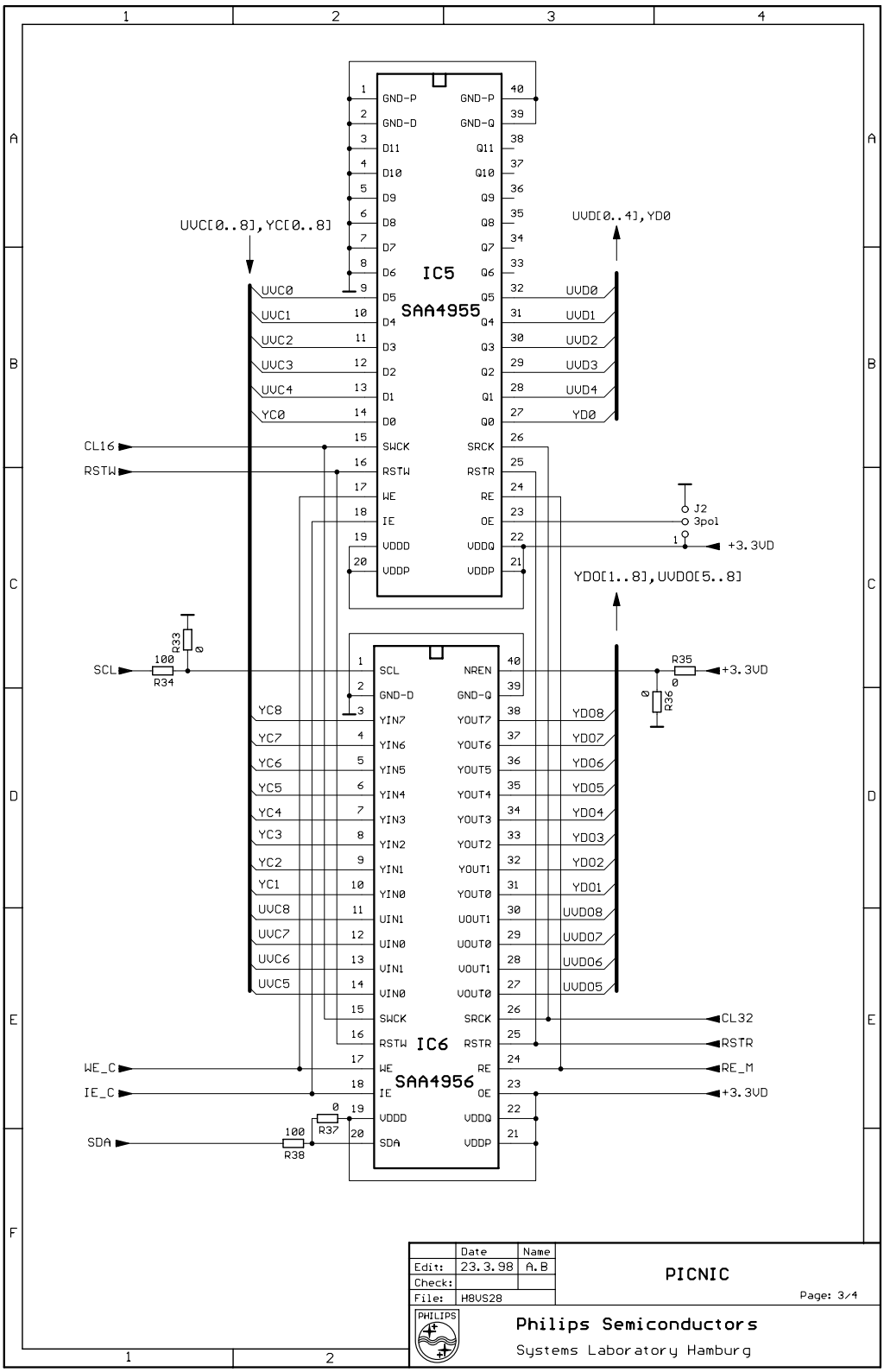


Fig. 74 MK10 module circuit diagram, sheet 3

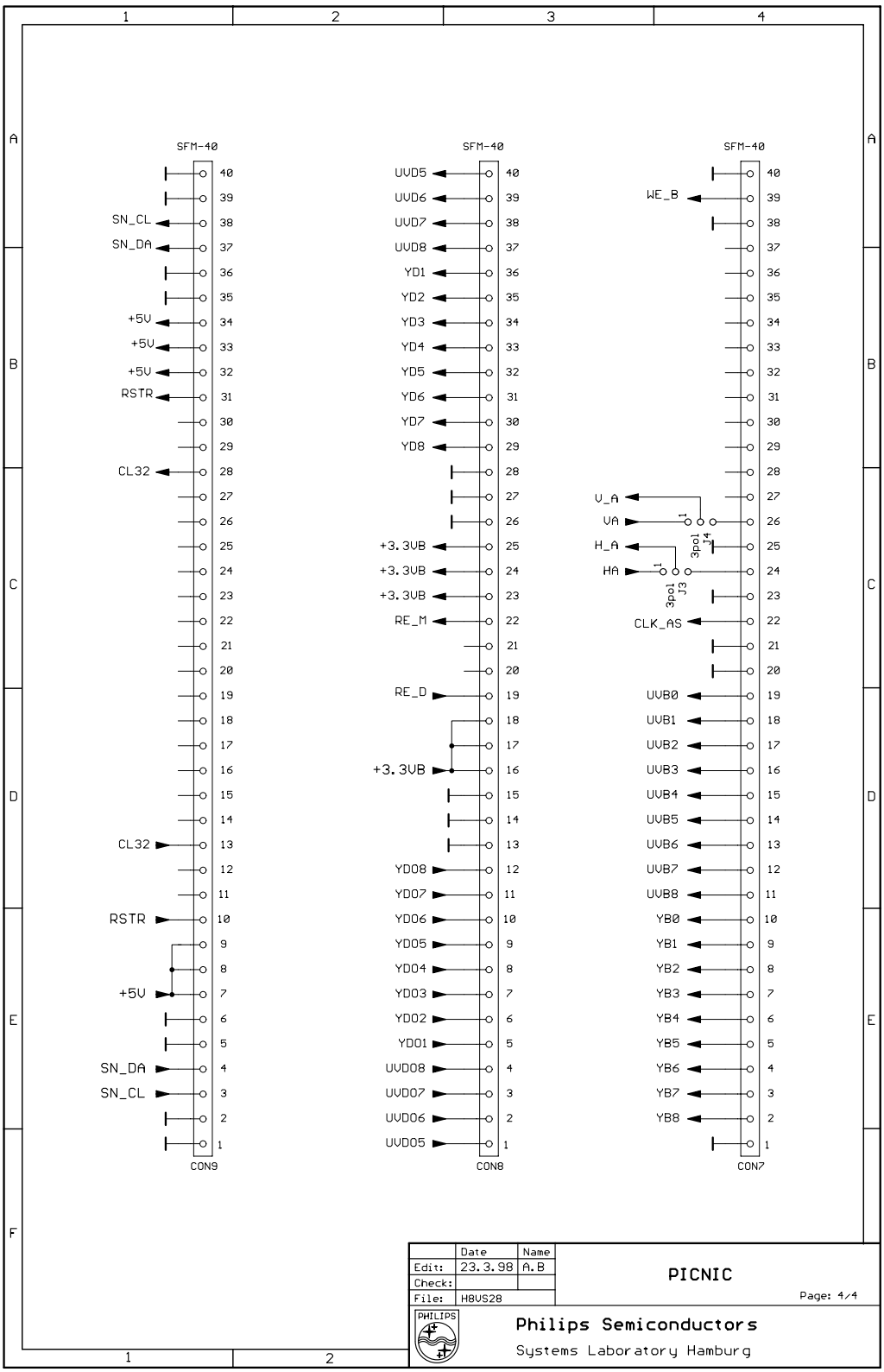


Fig. 75 MK10 module circuit diagram, sheet 4

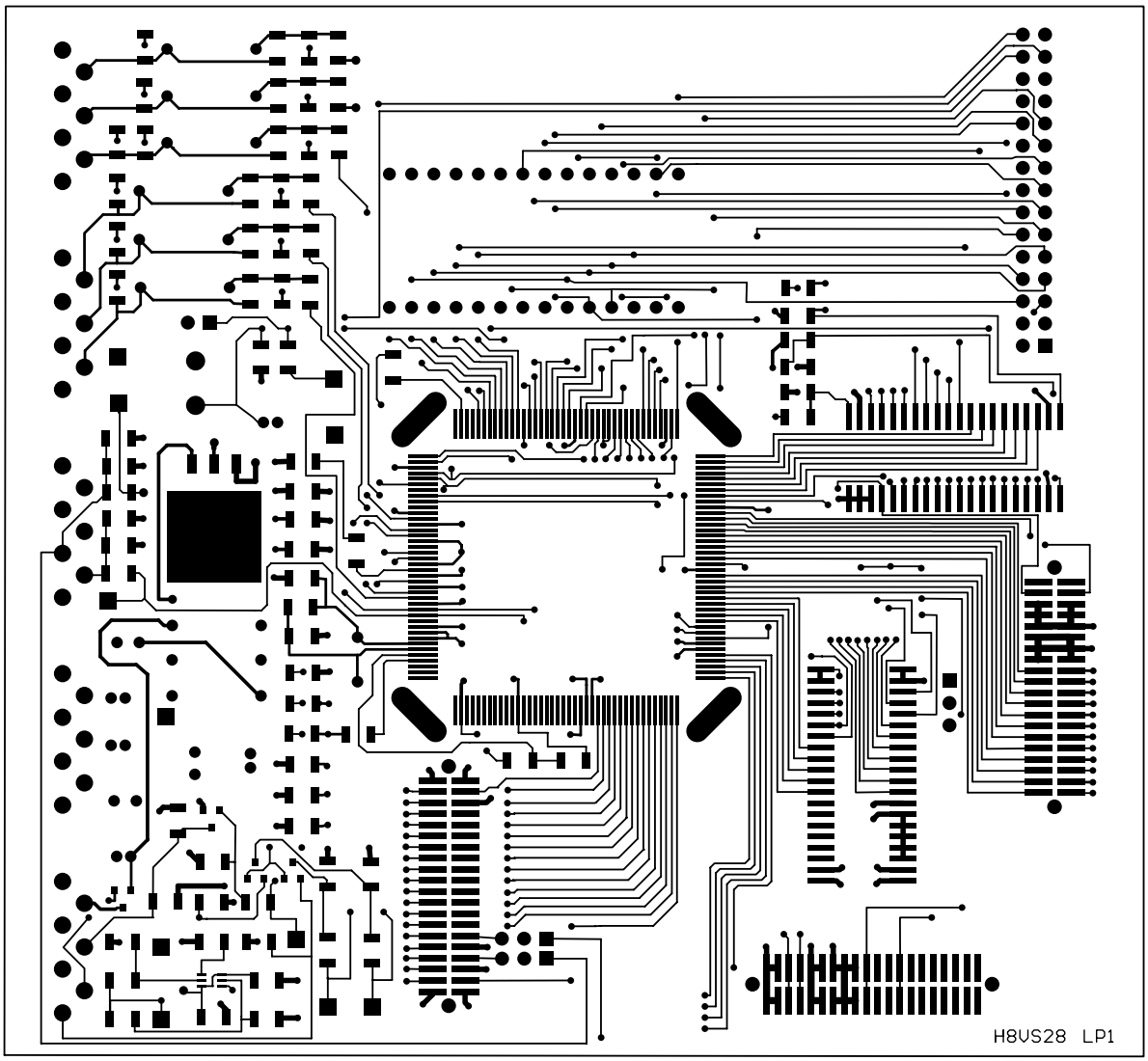


Fig. 76 PC board MK10: layer 1 (top)

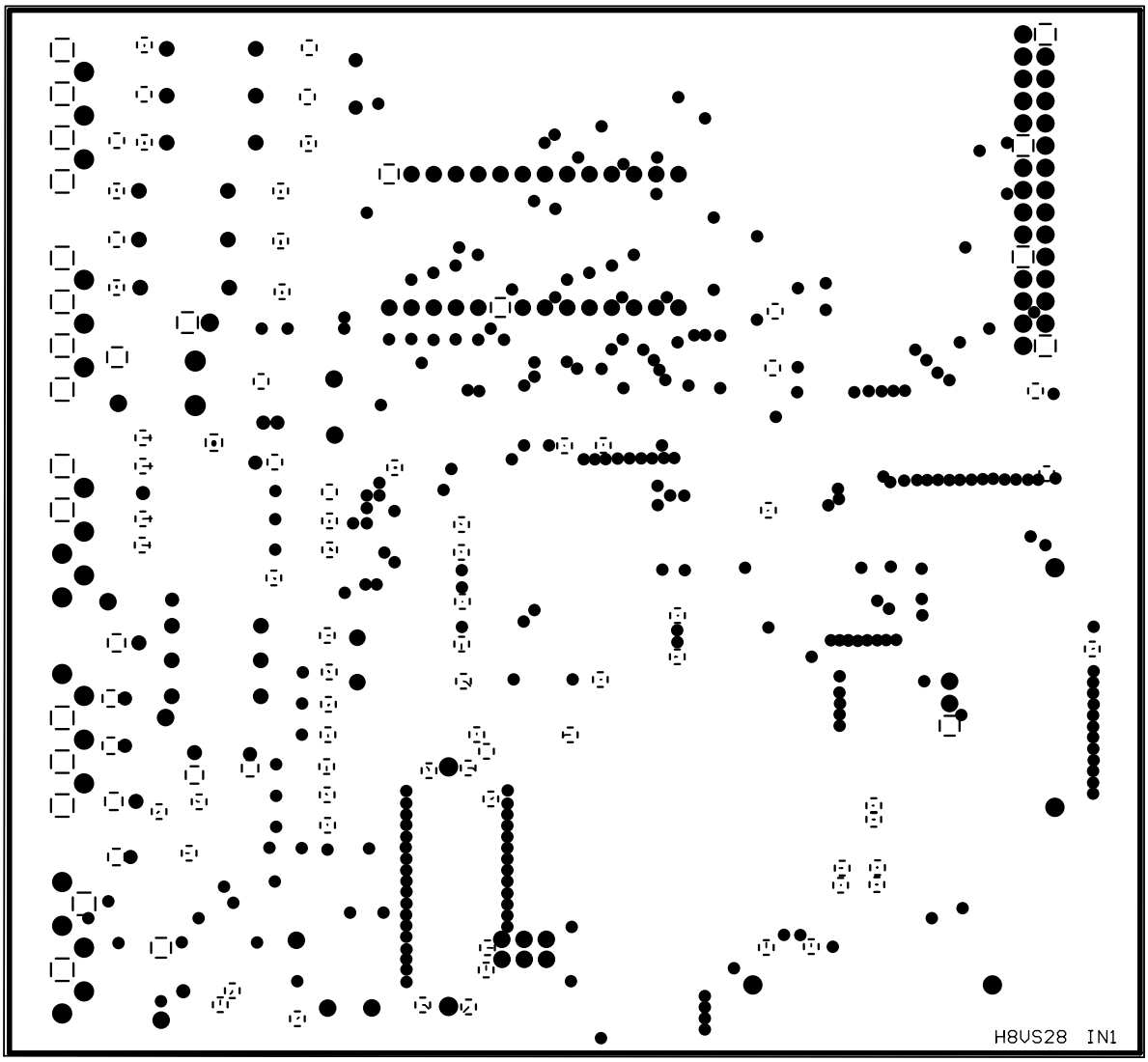


Fig. 77 PC board MK10: layer 2

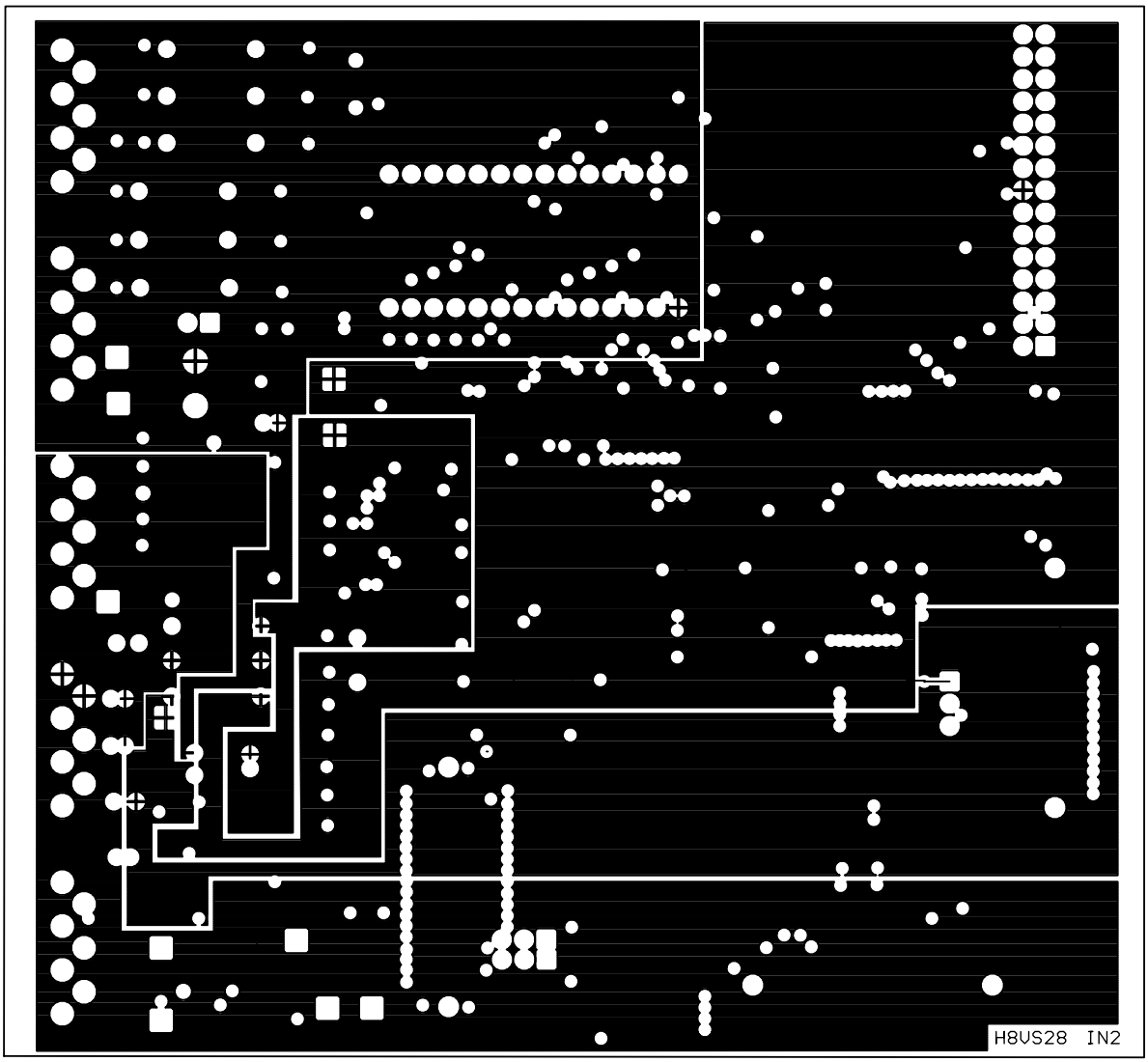


Fig. 78 PC board MK10: layer 3

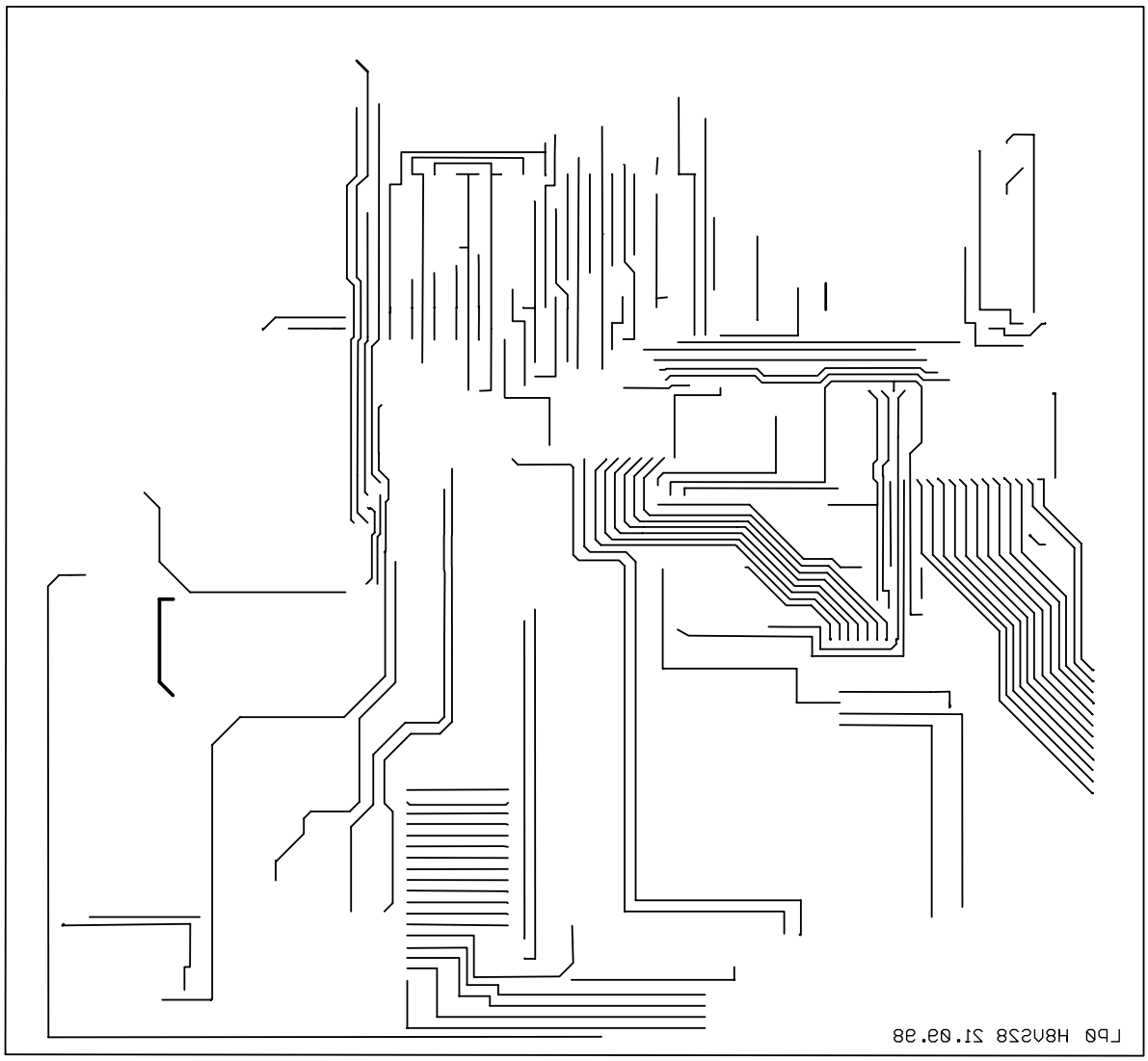


Fig. 79 PC board MK10: layer 4 (bottom)

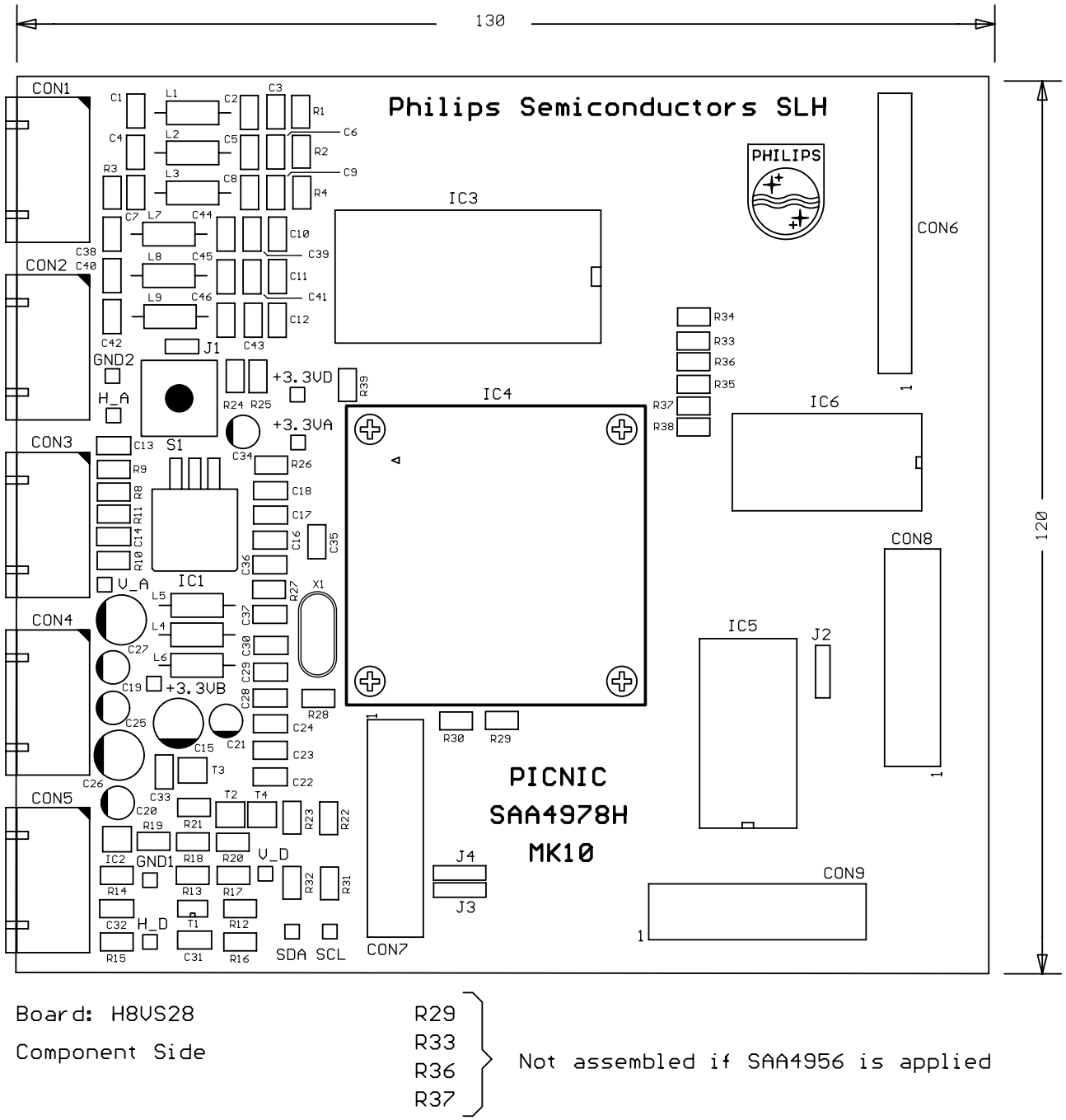


Fig. 80 PC board MK10: position of parts

Index

Entries in italic indicate register names

L

M

N

O

P

R

S