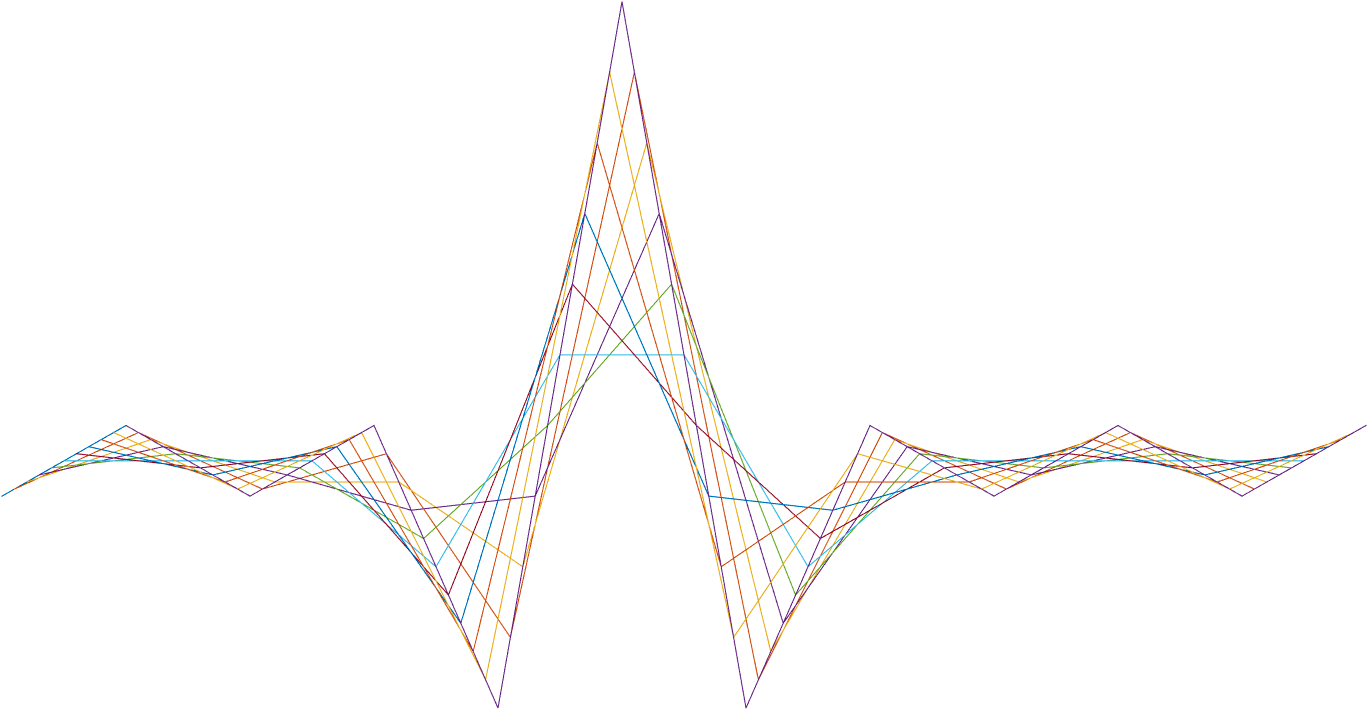
IVCAM2.0 3D Imaging Camera



ASIC A0 Depth Estimation specification

5 February 2017

Revision 0.8.0

Intel Top Secret

Table : Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matlab Version | Revision Number | Revised by | Description | Revision Date |
| TBD | 0.3.0 | Ohad Menashe | Initial Release | 24 Feb 2016 |
| TBD | 0.4.1 | Ohad Menashe | DEST design concept change | 28 Mar 2016 |
| TBD | 0.4.2 | Ohad Menashe | Erez’s review | 19 Apr 2016 |
| TBD | 0.4.3 | Ohad Menashe | Alex’s review | 21 Apr 2016 |
| TBD | 0.4.4 | Ohad Menashe | Disparity support, delay(Ir,phi),confidence, depth, registers, ambiguity length. | 20 May 2016 |
| TBD | 0.51 | Ohad Menashe | Trigonometric LUT sizes updated | 20 May 2016 |
| TBD | 0.52 | Ohad Menashe | -name change: conf\_dc🡪duty\_cycle  -bugfix: mssing 2\* in smoothing filter  -Confidence out: remove alpha blending, output max\_value and duty\_cycle | 17 Jun 2016 |
| TBD | 0.53 | David Silver | * PSLR change from 6-bit to 8-bit | 27 Jun 2016 |
| TBD | 0.53 | Ohad Menashe | * Registers * Angle calcs | 9 Jul 2016 |
| TBD | 0.53 | Omer Sella | Added documentation on confBlock (was in a different document until now). | 11 Jul 2016 |
| TBD | 0.54 | Ohad Menashe | Confidence calculation (re)added | 21 Jul 2016 |
| TBD | 0.54 | Omer Sella | Made the following corrections to confBlock:   1. Changed register names (mainly JFIL to DEST). 2. Added bit shifting registers. 3. Added an explanation on the activation functions. 4. Corrected the flow chart according to reejections and remarks. | 24 July 2016 |
| TBD | 0.54 | Ohad Menashe | Replace zNorm with ZMaxSubMMExp (zNorm = 2^ZMaxSubMMExp) | 24 July 2016 |
|  | 0.54 | Omer Sella | Rewrote confBlock | 03/08/2016 |
|  | 0.55 | Omer Sella | confBlock changes after review with Ohad | 07/08/2016 |
|  | 0.55 | Ohad Menashe | Confblock reorg | 08/08/2016 |
|  | 0.56 | Ohad Menashe | Conf block is now signed  Remove location propagation | 14/08/2016 |
|  | 0.57 | Ohad Menashe | -Invalid depth handling  -remove invalid from NN | 22/08/2016 |
|  | 0.57 | Ohad Menashe | -added ambiguity filter  -added IR to NN (instead of deprecated input) | 25/08/2016 |
|  | 0.58 | Yoni Chechik | -added pipe flags | 14/09/2016 |
| 0.7.39 | 0.59 | Ohad Menashe | -Multifocal support (tx frequency may change between pixels)  -Ambiguity removal missing threshold  -added LUT (float to float) implementation  -rxPD and txPD LUT are now both integer to float | 22/09/2016 |
| 0.7.41 | 0.7 | Ohad Menashe | Depth estimation bug fixes | 05/10/2016 |
| 0.7.42 | 0.7 | Ohad Menashe | Input/output alignment | 13/10/2016 |
| 0.7.44 | 0.7 | Ohad Menashe | -Added txpd/rxpd registers  Range finder registers name change |  |
| 0.7.45 |  | Yoni Chechik | Tx/rxPWRpdScale registers added | 5/11/16 |
| 0.7.46 |  | Ohad menashe | Offset\_index chaged from 10b to 8b | 09-nov-2016 |
| 0.7.46 |  | Ohad Menashe | Trigo function update | 10-Nov-2016 |
| 0.7.47 | 0.7.2 | Vitaly Surazhsky | bypass | 16 Nov 2016 |
| 0.8.49 | 0.8.0 | Ohad Menashe | Range finder x/y switch | 29 Nov 2016 |
| 0.8.5 | 0.8 | Ohad Menashe | Temperature model | 22 Dec 2016 |
| 0.9 |  | Yoni Chechik | corrDebug | 29 Dec 2016 |
| 0.9 |  | Ohad Menashe | Large FOV support:  trigo func support 360deg  unproject bypass  confidence issing offsets | 17 Jan 2017 |
| 0.91 |  | Ohad Menashe | Rx/tx LUT sizes alignment  Alternate IR | 5-feb-2017 |
| 0.91 |  | Ohad Menashe | Added sinatan/cosatan hex values | 12-Feb-2017 |

7

Contents

[1 Introduction 4](#_Toc462570504)

[2 Interfaces and memory 5](#_Toc462570505)

[2.1 Interfaces 5](#_Toc462570506)

[2.1.1 Input 5](#_Toc462570507)

[2.1.2 Output 5](#_Toc462570508)

[2.2 Memory 6](#_Toc462570509)

[2.2.1 Trigonometric functions 6](#_Toc462570510)

[2.2.2 IR dependent propagation delay 6](#_Toc462570511)

[2.2.3 Vertical angle dependent propagation delay 6](#_Toc462570512)

[3 Detailed description 7](#_Toc462570513)

[3.1 Peak locator 7](#_Toc462570514)

[3.1.1 Peak location extraction 7](#_Toc462570515)

[3.2 Index to depth 9](#_Toc462570516)

[3.2.1 Index to round trip distance 9](#_Toc462570517)

[3.2.2 Removing ambiguity 9](#_Toc462570518)

[3.2.3 Ambiguity filter 10](#_Toc462570519)

[3.2.4 Round trip distance to range 11](#_Toc462570520)

[3.2.5 Range to depth 11](#_Toc462570521)

[3.2.6 Low level Z (depth) calculation 12](#_Toc462570522)

[3.3 max\_val calculation 14](#_Toc462570523)

[3.4 Confidence level calculation 14](#_Toc462570524)

[3.4.1 Input 14](#_Toc462570525)

[3.4.2 Output 14](#_Toc462570526)

[3.4.3 Detailed description 14](#_Toc462570527)

[4 Registers 18](#_Toc462570528)

[5 Test plan 20](#_Toc462570529)

Tables

[Table 1: Revision history 1](#_Toc450024955)

[Table 2 : common operational modes 4](#_Toc450024956)

[Table 4: DEST memory and bandwidths 5](#_Toc450024957)

[Table 3: zNorm values 10](#_Toc450024958)

[Table 3: Registers 12](#_Toc450024959)

[Table 3: Test plan 13](#_Toc450024960)

Introduction

The purpose of this block is to find the location of the correlation peak in sub-sample accuracy, convert it to absolute depth, and output the depth with the corresponding IR and confidence level.

The input data is given over a scan line stream generated by the RAST block, processed by the DCOR block, providing fine correlation samples in the vicinity of the coarse peak and the location index of the coarse peak. The correlation segment is then convolved with a Gaussian kernel for smoothing, and depth data is extracted using quadratic function fitting around the maximum value. After the index location is converted to depth, an intensity dependent propagation delay is added to the result.

A separate path calculates the confidence level according to the received peak strength.



**Figure 1: DEST block diagram**

**Table 2 : common operational modes**

Interfaces and memory

Interfaces

Input

* *xy position*: 12 bit + 11 bit unsigned
* *IR*: 12bit
* offset\_index - 8bit, location of coarse max value. Around this value the fine correlation is calculated
* cor\_seg: vector of N bins, each with K bits (N and K are mode dependent. See Table 2 : common operational modes for possible values).  
  Data structure is as a shared memory element with DCOR block.
* Duty\_cycle: 4bit, coarse confidence, represents the distance from 50% duty cycle
* Psnr – 6b psnr value
* pxl\_flags – 4 bits

Output

* *xy position*: 12 bit + 11 bit unsigned
* *IR*: 12bit
* depth: 16 bit, pixel distance from focal plane
* confidence – 4 bit unsigned

Bypass

When RegsDESTbypass is set the block outputs *IR*, *xy\_position* from the block input, while *depth* and *confidence* of the ouput pixels are all outputted as 0.

Memory

Trigonometric functions

LUTs for / functions are also calculated, hard coded on the interval of over 2048 points. LUT size is **hard coded**

Cosatan:

3F800000 3F7FFDF8 3F7FF7E0 3F7FEDB9 3F7FDF84 3F7FCD46 3F7FB6FD 3F7F9CAD 3F7F7E5C 3F7F5C0D 3F7F35C8 3F7F0B8C 3F7EDD63 3F7EAB56 3F7E7564 3F7E3B9D 3F7DFE04 3F7DBCA3 3F7D7781 3F7D2EAB 3F7CE229 3F7C9204 3F7C3E49 3F7BE702 3F7B8C38 3F7B2DFD 3F7ACC57 3F7A6757 3F79FF05 3F799373 3F7924AB 3F78B2BE 3F783DB7 3F77C5A6 3F774A99 3F76CC9D 3F764BC5 3F75C81B 3F7541B2 3F74B898 3F742CDB 3F739E8C 3F730DBB 3F727A76 3F71E4CF 3F714CD4 3F70B297 3F701624 3F6F778E 3F6ED6E3 3F6E3434 3F6D8F91 3F6CE90A 3F6C40AC 3F6B9688 3F6AEAAE 3F6A3D2F 3F698E15 3F68DD74 3F682B5B 3F6777D5 3F66C2F4 3F660CC7 3F655558 3F649CBA 3F63E2FA 3F632823 3F626C45 3F61AF6E 3F60F1AB 3F60330A 3F5F7395 3F5EB35B 3F5DF267 3F5D30C7 3F5C6E86 3F5BABB0 3F5AE850 3F5A2474 3F596024 3F589B6C 3F57D657 3F5710F0 3F564B41 3F558551 3F54BF2E 3F53F8E1 3F533271 3F526BE9 3F51A552 3F50DEB4 3F501818 3F4F5186 3F4E8B04 3F4DC49E 3F4CFE57 3F4C383B 3F4B724E 3F4AAC99 3F49E721 3F4921ED 3F485D04 3F47986B 3F46D42B 3F461046 3F454CC3 3F4489AA 3F43C6FD 3F4304C1 3F4242FD 3F4181B5 3F40C0ED 3F4000AB 3F3F40F1 3F3E81C5 3F3DC32A 3F3D0523 3F3C47B6 3F3B8AE4 3F3ACEB1 3F3A1322 3F395837 3F389DF7 3F37E461 3F372B7A 3F367344 3F35BBC0 3F3504F3

Sinatan:



IR dependent propagation delay

The IR dependent propagation delay is calibration based LUT, saved as 128 bins of 32 bits (float), resulting in a LUT size of 4096. The input to this LUT arrives from the IR data (12b), and the output is 32bit float.

Vertical angle dependent propagation delay

Due to eye safety consideration, the system’s TX output power change is respect to the scan speed, hence vertical location. This change in the output power value creates a different propagation delay in the laser driver / diode for each setting. This delay is calibrated as saved as a 128 bins of 32 bits. The input to this LUT arrives from the Y location (11b), and the output is 32bit float.

Detailed description

## Peak locator

The purpose of this block is to perform a quadratic fit over the peak location in the correlation vector. It is performed as a linear interpolation of the zeros crossing of the gradient around the highest correlation value.  
The Data received from the DCOR block consists of a N bin, K bits each. This data represents a small part of the full correlation vector (which is not computed due to high computational cost).  
In order to reduce noise levels, the input correlation vector is convolved using a smoothing kernel of size 3,5or 9.

The size of the output after the filter depends on the size of the input (e.g. 33,17,9), and the size of the filter (3,5,9), such that:

Filter is implemented as 4 serial filters, each with three taps. The kernel of each block is given by the registers RegsDESTsmoothA/RegsDESTsmoothB/RegsDESTsmoothC/ RegsDESTsmoothD for each of the filters , where the kernel value is given by:

kerX[0] = RegsDESTsmoothX

kerX[1] = 64-2\*RegsDESTsmoothX

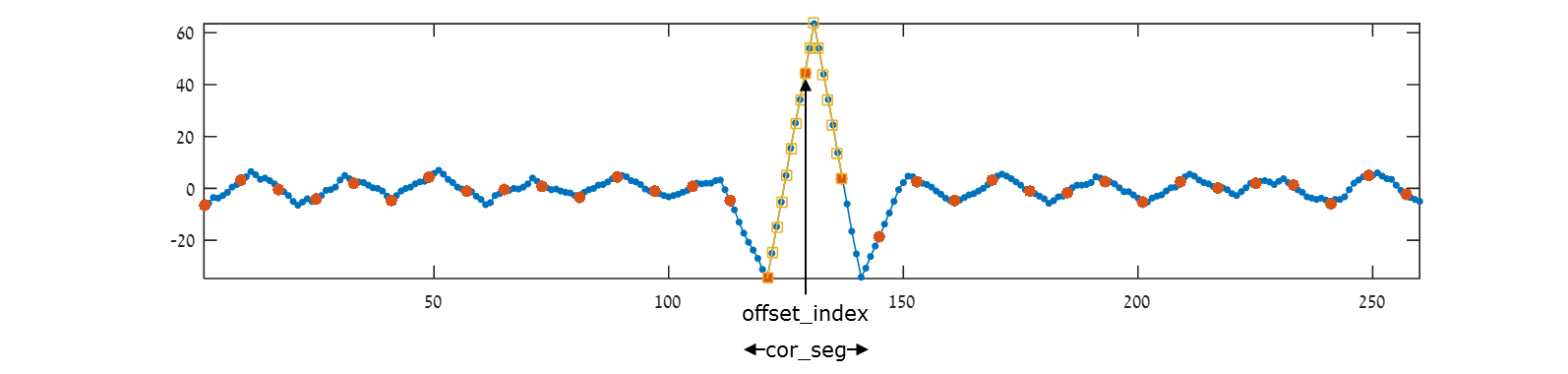
kerX[2] = RegsDESTsmoothX



**Figure 2: smoothing filter implementation using 4 filters, each size of 3**

Once smoothed, the actual maximum location and value are located. The maximum value is sent for confidence calculation, while the maximum location along with two adjacent samples is kept for processing.

### Peak location extraction



**Figure 3:cor\_seg in full correlation**

Given 3 samples ( , the maximum value) with the corresponding index points , the subsample maxima index is located by fitting a quadratic function, and finding its peak by solving the below equations:

Since:

We can write

In matrix notation:

And the solution:

Where the maxima index is

And the final peak location is:

Note that:

* In case of flat correlation, value is invalid and the depth value should be set to zero. Additionally, the confidence value should be set to the predefined value of zero.
* since and then when then (but not the other way around)

## Index to depth

Index to round trip distance

The result received from the previous block represents the index location (in sub sample) of the peak location. The temporal distance between two samples corresponds to the sampling frequency. Given the speed of light in air, the distance traveled from the projector to the object and to the detector is given by

Where is the distance traveled within a single sample, represent the speed of light in air roughly equals to , and is the sampling frequency.

As may change for each pixel from 1000MHz, 500MHZ, and 250MHz. value is calculated for each of the 3 states and saved in 3 separate registers (RegsDESTtxFRQpdXX). The value is then selected in runtime according the appropriate transmission mode.

The terms , are offsets in mm to cause by the RX and TX paths accordingly:

- Due to the system’s TX power envelope, the laser output power continuously changing. Each output power value creates a different propagation delay in the laser driver / diode. The corresponding affect in mm is stored in a 129x32b LUT generated in calibration time.  
 - Due to the system’s intensity dependent propagation delay, an offset location is subtracted. This offset is stored in a 129x32b LUT generated in calibration time. Since the IR is 12bit, the remaining 4 bit are interpolated using linear interpolation from the two nearest values extracted from the table.

– is another offset calculated at calibration.

Furthermore, 3 float registers called txPWRpdScale\_XXX are in to scale the LUT output according to the tx frequency. We choose the register we want according to the frequency (done with pipe flags), and then each lut output is multiplied by the chosen reg. the same is done with the rx LUT (rxPWRpdScale\_XXX).

* Note that txrx\_mode=3 is reserved for mixed frequency pixel. These pixels should be ignored and set as invalid pixels (depth and confidence are zero)
* In case of rangefinder mode, control is changed to the lsb of the x location. In this mode y is set to constant 0, while x changes from 0 to 1 indicating the high power or low power sequence.

Temperature model

The core temperature greatly affect the measured depth. This model can be learned, and a linear transformation is applied as function of temperature.

Removing ambiguity

Due to the nature of the cyclic code transmission, any echo received with a time delay greater than the code length, will result in ambiguity / aliasing. This ambiguity may result due to the additional system propagation delay, causing echoes received within the time frame to appear after time greater than the code length.

In order to solve this distance, the round trip distance must be folded after decreasing the propagation delay by a factor if ambiguityRTD (RTD=RoundTripDistance)

Where:

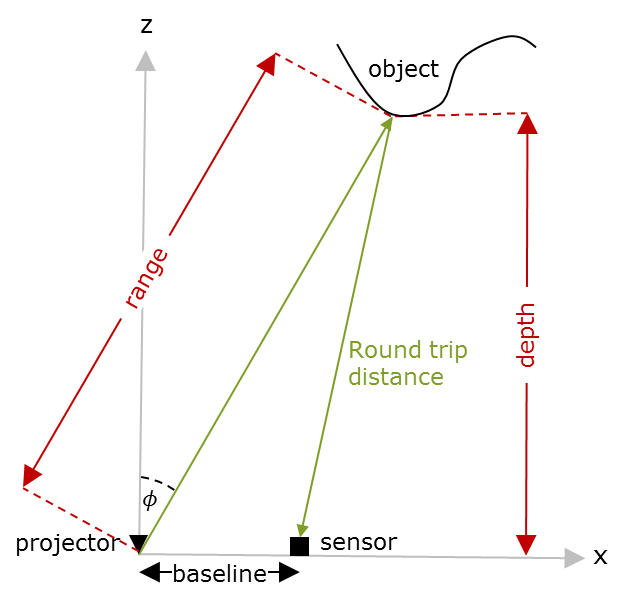
may change for each pixel from 1000MHz, 500MHZ, and 250MHz. this value is calculated for each of the 3 states and saved in 3 separate registers. The value is then selected in runtime according the appropriate transmission mode.

Ambiguity filter

It is possible that the system will receive echoes from ranges further away from the ambiguity range. These echoes will appear on a close range due to aliasing and will create depth artifacts.  
In order to remove them, this filter adds the ambiguity range to values suspected as aliased echoes.

The filtration mechanism uses two thresholds: if the suspected pixel’s RTD is below the preset value of RegsDESTambiguityMinRTD (i.e. the range is small) and the suspected pixel’s confidence is below a preset value of RegsDestambiguityMinconf (i.e. small PSNR), add the ambiguity round trip distance to the current RTD value.

Round trip distance to range



**Figure 4: round trip distance to range**

Since the sensor and projector are not co-centered and some known baseline L exists, the range from the projector focal point is calculated using:

, where is the angle along the baseline axis L is ( is given at ‎3.2.6)

Range to depth

The output from the peak estimation block refers to the distance from the (theoretical) focal point of the projector. This means that our data is given in spherical coordinate space, and should be translated to Cartesian coordinate space. The Z value represents the distance from da hypothetical orthographic camera in which the camera plane is perpendicular to the projector focal axis.

The angular data can be easily calculated from the known field of view and the output resolution.

Given the angles and the range, the Cartesian coordinates for each point in the image are calculated as:

Since this data is redundant, only the Z coordinate is calculated in the ASIC and (x,y) coordinates are calculated in firmware by multiplication in the normalization factor:

(not calculated in ASIC).

Depth ad range (unproject bypass)

In case the flag RegsDESTdepthAsRange is high, the range data is save in the depth output (after zNorm factoring):

depth=uint16 (max(1,range\*regs.GNRL.zNorm));

Low level Z (depth) calculation

#### Trigonometric functions calculations

The pixel spherical angles are derived from the known FOV and resolution:

Where and are the horizontal vertical FOV, and and are the horizontal and vertical keystone (offset from center). Note that,, , , , can change between configurations to any number (crop/zoom/ROI mode).

Since only the sin of are used, and the creation of the angles involves function, the LUT of and are used:

As the input data arrives as pixel location (xi,yi), it can be easily converted into the tangent of the angle by simple linear transformation (registers , , ). In order to acces the 2048 LUT, these registers encapsulates a factor that transforms xi& yi to a number in the range of [-2047 2047]. Accessing the LUT with the absolute of this values yields the output angle.

#### LUT implementation

This section elaborates the implementation of a float to float LUT with linear interpolation.

Given N sampling point (i.e. LUT data) lut, with the LUT frequency fs (inverse of the distance between each of the LUT sampling points), the interpolated result of input data x is given as:

ii = x \* fs; % index of the input x

i0 = max(floor(ii),0); % the index round down (cannot be negative)

i1 = min(i0+1,length(lut)-1); % the index rounded up (cannot exceed lut size-1)

y0 = lut(i0+1); % LUT value of index i0

y1 = lut(i1+1); % LUT value of index i1

val=(y1-y0).\*(ii-i0)+y0; %weighted mean of the two points (linear interp)

As can be seen in the figure below, the values between two data point is linearly interpolated, and the error peaks in the midpoint between the lut sample data.



**Figure 5: float to float LUT**

#### Range calculation

To reduce floating point calculations, in addition to the baseline, the power of 2 is saved in a separate register.

#### Data output format

The depth data is transformed to 16 bit with a configurable accuracy loss:

Since value of 0 is marked as undefined depth, this value saturated with 1 in order to prevent illegal state.

sets the numerical accuracy and the maximum range according to the table 3

|  |  |  |
| --- | --- | --- |
| **value** | **accuracy** | **Max distance [M]** |
| 1 | 16.0 | 64 |
| 2 | 15.1 | 32 |
| 4 | 14.2 | 16 |
| 8 | 13.3 | 8 |

**Table 3: zNorm values**

## max\_val calculation

max\_val is generated from the peak value found in the peak locator section. Using pre calculated fixed point registers the peak value is calculated as:

Where take the 6 LSB of the above unsigned result (negative values are converted to zero).

## Alternative IR

Due to the low quality of IR, an alternate IR image base on the peak correlation values can replace the slow path original IR image.

Since the peak value is 22b some transformation must be made. The value is multiplied by RegsAltIrMul, and then shift by RegsAltIrDivExp bits. The value is then biased by RegsAltIrBias and saturated by 4095.

## Confidence level calculation

Input

* Duty\_cycle – 4b
* Psnr – 6b
* Max\_val - 6b
* Ir – 12b

Output

* 1X4 bit bus, conf.

Detailed description

The purpose of this block is to calculate a 4 bit confidence level out of the various inputs, by diffusing the data using a 2 layers neural network mechanism. One of the main components in the neural network is the activation function. This function applies a non-linear transformation on the input, using two parameters.

In the special case of invalid depth, the confidence value is not calculated and is set to zero.



**Figure 5: Confidence block diagram**

The block essentially implements two layers of multiplication by corresponding weights, whilst considering the dynamic range of each input.

Initially, each input is scaled (bit shift) such that the values are in the range of [0-63]:

Additionally, The 8 bit weights are extracted from their designated registers:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

The weights of each set is multiplied by the scaled input, and summed to form a 16bit weighted input (8bit per weight x 6 bit input x 4 inputs). The weighted input is then passed through the signed activation function.

With all the calculation preformed using fixed point signed operations.

Finally, the results is weighted summed using q, and the unsigned activation function is applied.

|  |  |
| --- | --- |
|  |  |

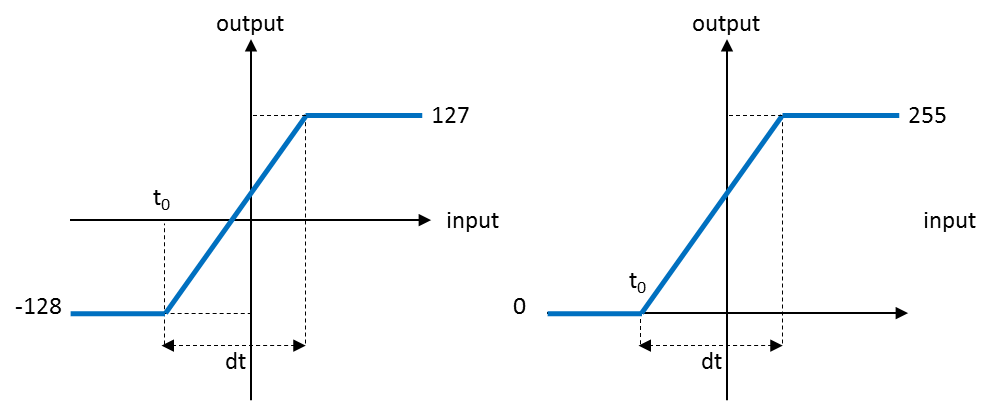
Finally, the 4 MSBs of the output are set as the final confidence value.

**Important Note:**

* This block should never output confidence value = 0, as this value is reserved for a no-scan pixel.

#### Activation function

The activation function applies a non-linear transformation on a 16bit input value, using a 32 bit register, and transforms it to a 8 bit value, with two version: signed and unsigned. The data in the register is divided into (register[32-16]) and (register[15-0]). Input Is linearly interpolated and truncated according to the input type (signed/unsigned, see graphs below).

  
Figure : Register based LUT

corrDebug

one can set a register named RegsDCORcorrDebug to 1 if he wants to output the corrOffset and one chosen corrSegmant (out of 33, chosen by the register RegsDCORcorrDebugSegment). If so, the output is as follows:

* The 8b corrOffset is out through the IR line.
* The 22b corrSegment is out like so:

chosenCorrSegment = min(squeeze(corrSegment(regs.DCOR.corrDebugSegment,:,:)),2^20-1); %output with overflow!

zImgRAW = uint16(mod(chosenCorrSegment,2^16)); %16b LSB

cImgRAW = uint8(idivide(chosenCorrSegment,2^16, 'floor') ); %4b after 16b LSB

so when we add the bits of the conf channel after the bits of the z channel we get the chosen corrSegmet (up to overflow).

Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Size** | **Default** | **Range** | **Special values/ description** |
| **General** |  |  |  |  |
| RegsGNRLzMaxSubMMExp | 3 | 3 | [0-3] |  |
| **DEST** |  |  |  |  |
| RegsDESTbypass | 1 | 0 | 0, 1 | Bypass mode |
| RegsDESTsmoothA | 5 | 21 | [0-21] | Correlation smoothing configuration  SL = RegsDESTsmooth  ML = 64-2\* RegsDESTsmooth  K = [SL ML SL]  K = [ 0 64 0]  K = [21 22 21] |
| RegsDESTsmoothB |
| RegsDESTsmoothC |
| RegsDESTsmoothD |
| RegsDESTmaxvalDiv | 13 | 0 |  | autogen |
| RegsDESTmaxvalSub | 8 | 0 |  | autogen |
| RegsDESTp2axa | 32 | 0 |  | autogen |
| RegsDESTp2aya |
| RegsDESTp2axb |
| RegsDESTp2ayb |
| RegsDESTtxFRQpd\_00 | 32 | 0 | [0.0-128.0] | float |
| RegsDESTtxFRQpd\_01 | float |
| RegsDESTtxFRQpd\_02 | float |
| RegsDESTsampleDist\_00 | 32 | 18.73141 | {  18.73141, 37.46282, 74.92564,  149.85128, 299.70256  } | [mm/sample]@1GHz tx |
| RegsDESTsampleDist\_01 | 37.46282 | [mm/sample]@ 500MHz tx |
| RegsDESTsampleDist\_02 | 74.92564 | [mm/sample]@250MHz tx |
| RegsDESTambiguityRTD\_00 | 32 | 0 |  | autogen |
| RegsDESTambiguityRTD\_01 |
| RegsDESTambiguityRTD\_02 |
| RegsDESTambiguityMinRTD | 32 | 600.0 |  | float |
| RegsDestambiguityMinconf | 4 | 3 | [1-14] |  |
| RegsDESTbaseline | 32 | 30.0000 |  | [mm], Static calibration |
| RegsDESTbaseline2 | 32 | 0 |  | autogen |
| DESTconfw1 | 32 | 0x00000100 | [0-255]x4 | 4x8b weights {w1,w2,w3,w4} |
| DESTconfw2 | 32 | 0x00000000 | [0-255]x4 | 4x8b weights {w1,w2,w3,w4} |
| DESTconfq | 16 | 0x0100 | [0-255]x2 | 2x8b weights {q1,q2} |
| DESTconfact1 | 32 | 0xFF8000FF | [0-65535]x2 | Activation function t0,dt |
| DESTconfact2 | 32 | 0x0000003F | [0-65535]x2 | Activation function t0,dt |
| DESTtxPWRpd\_00  …  DESTtxPWRpd\_64 | 32 | 0 | [0.0-10000] |  |
| DESTrxPWRpd\_00  …  DESTrxPWRpd\_64 | 32 | 0 | [0.0-10000] |  |
| DESTtxFRQpdScale\_000  DESTtxFRQpdScale\_001  DESTtxFRQpdScale\_002 | 32 | 1.0  2.0  4.0 | [0.0:10.0] |  |
| DESTrxPWRpdScale\_000  DESTrxPWRpdScale\_002  DESTrxPWRpdScale\_002 | 32 | 1.0  2.0  4.0 | [0.0:10.0] |  |

**Table 7: Registers**

Test plan

|  |  |  |
| --- | --- | --- |
| **Name** | **value** | **distribution** |
| **General** |  |  |
| RegsGNRLzMaxSubMMExp |  | `100% |
| **DEST** |  |  |
| RegsDESTsmoothA | [0-21] | 100% |
| RegsDESTsmoothB | [0-21] | 100% |
| RegsDESTsmoothC | [0-21] | 100% |
| RegsDESTsmoothD | [0-21] | 100% |
| RegsDESTmaxvalDiv | autogen |  |
| RegsDESTmaxvalSub | autogen |  |
| RegsDESTp2axa | autogen |  |
| RegsDESTp2aya | autogen |  |
| RegsDESTp2axb | autogen |  |
| RegsDESTp2ayb | autogen |  |
| RegsDESTtxFRQpd00 | [0-106.74] | 100% |
| RegsDESTtxFRQpd01 | [0-106.74] | 100% |
| RegsDESTtxFRQpd10 | [0-106.74] | 100% |
| RegsDESTsampleDist00 | {  18.73141, 37.46282, 74.92564,  149.85128, 299.70256  } | 100% |
| RegsDESTsampleDist01 | 100% |
| RegsDESTsampleDist10 | 100% |
| RegsDESTambiguityRTD00 | autogen |  |
| RegsDESTambiguityRTD01 | autogen |  |
| RegsDESTambiguityRTD10 | autogen |  |
| RegsDESTambiguityMinRTD | [0-1000.0] | 100% |
| RegsDestambiguityMinconf | [1-14] | 100% |
| RegsDESTbaseline | [0-60.0] | 100% |
| RegsDESTbaseline2 | autogen |  |
| DESTconfw1 | [0x00000000 -  0xFFFFFFFF] | 100% |
| DESTconfw2 | 100% |
| DESTconfq | [0x0000-0xFFFF] | 100% |
| DESTconfact1 |  | 100% |
| DESTconfact2 |  | 100% |

**Table 8: Test plan**