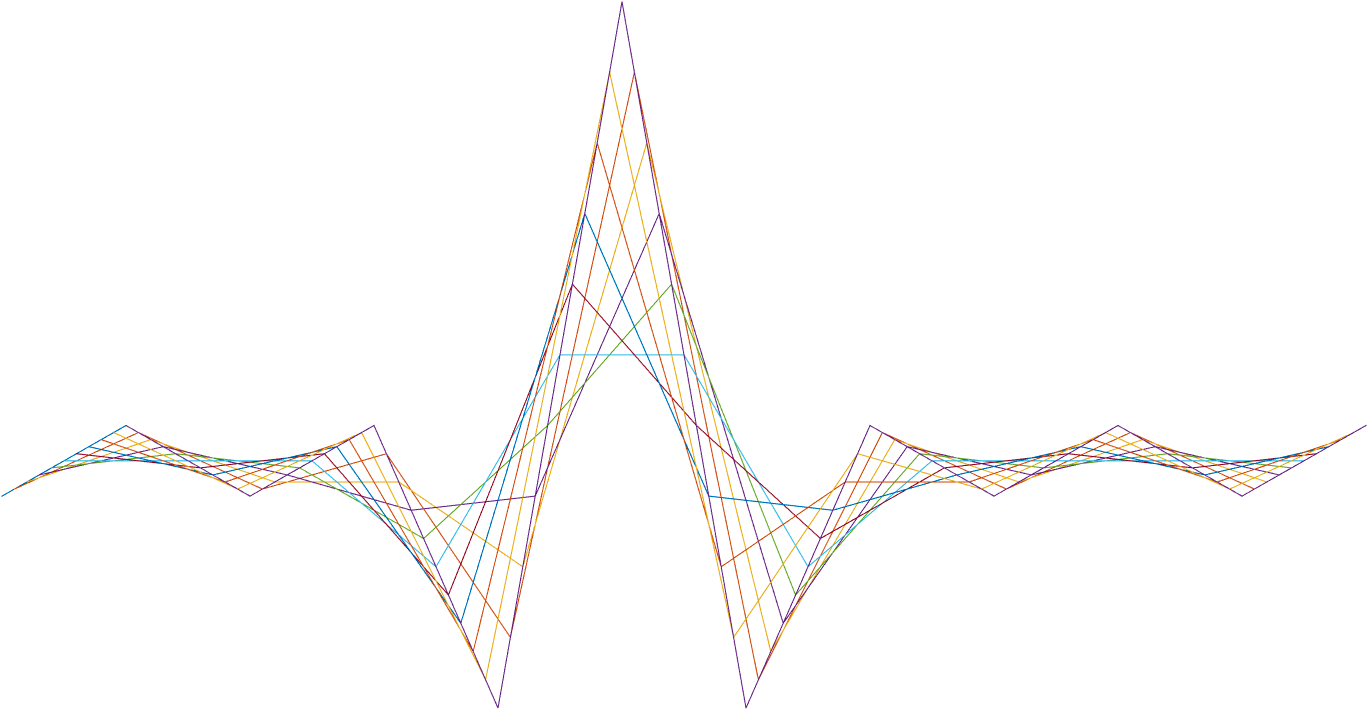
IVCAM2.0 3D Imaging Camera



ASIC A0 JFIL INN / DNN specification

Intel

Top Secret

10 October 2016

Revision 0.5.0

Intel Top Secret

Table : Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matlab Version | Revision Number | Revised by | Description | Revision Date |
|  |  | Omer Sella | Moved to this document instead of the master document. | 04/09/2016 |
| 0.6.37 |  | Omer Sella | Added test plan.  Added description of the layers and features. | 22/09/2016 |
| 0.7.38 |  | Omer Sella | Removed obsolete NN image. | 25/09/2016 |
| 0.7.39 |  | Ohad Menashe | Add activation function | 26/09/2016 |
| 0.7.39 |  | Omer Sella | Following a review with Miri:   1. Change the activation function from LUT based to logic and arithmetic based. 2. Added the normalization factor before and after the NN (multipolication by fp18 number on entry and multiplication by its inverse on exit). 3. Added registers. 4. Need to complete the TP according to the new registers. | 05/10/2016 |
| 0.7.40 |  | Omer Sella | Added zero depth handling – pixels that are set to 0 depth by the nnd must receive 0 confidence setting as well. |  |
| 0.7.42 |  | Omer Sella | Added an explanation on how the activation function is implemented.  Updated register names. |  |

Contents

[1 Introduction 3](#_Toc462293724)

[2 Interfaces 6](#_Toc462293725)

[2.1.1 Input 6](#_Toc462293726)

[2.1.2 Output 6](#_Toc462293727)

[3 Detailed description 7](#_Toc462293729)

[4 Registers 8](#_Toc462293730)

[5 Test Plan 9](#_Toc462293731)

[5.1 General randomization 9](#_Toc462293732)

[5.2 Constraints 9](#_Toc462293733)

[6 10](#_Toc462293734)

Tables

[Table 1: Revision history 1](#_Toc462293735)

[Table 3: Registers 8](#_Toc462293736)

Introduction

The JDNN is a neural network of an Input layer, 3 hidden layers and an output layer. Each layer is a linear transformation of the values (weighted sum) going through a LUT.

The JDNN is implemented in 18-bit half-precision floating-point.

* 1 bit for sign.
* 5 bits for exponent.
* 12 bits for the mantissa.

Except for the final computation which is done at single precision.

The inputs to the NN are referred to as features, and they are prepared by the feature extractor block.

Figure : Block diagram for the NND



Figure 2: Block diagram for the NNI



Interfaces

Input

* 16b depth
* 12b ir
* 4b Confidence
* flags – 4bit (to DEV/NULL)
  + Usage: TBD
* For the NND (Depth network):
  + 7\*16b depth – 7 pixels of different filtered (convolution) depth.
  + 1\*16b depth – 1 pixels of filtered (convolution) confidence.
  + 2\*16b depth – 2 pixels of different filtered depth inputs via bilateral filters (pass through – delayed to arrive in synchronization with the rest of the data path).
  + 1\*16b depth – 1 pixel coming out of Edge2 filter.
  + 9\*16b depth – sorted pixels in sliding window of 3\*3.
* For the NNI (IR network):
  + 2\*12b IR – 2 pixels of different filtered IR.
  + 9\*12b IR – sorted pixels in sliding window of 3\*3.

Output

* For the NND (Depth network):
  + 1X16b depth
  + 1X4bit confidence
* For the NNI (IR network):
  + 1X12 bit IR

Detailed description

In essence, the NN (both for the IR and the depth case) is made of multiplications and additions using fp18, as well as passing the result through an activation function (implemented using a LUT) between layers.

We refer to a module performing multiplication of all inputs by a weights vector, followed by a bias adding as a”neuron”.

Each layer is fully connected to the next one.

NND:

Normalization – before entering the NN block, all features are multiplied by a normalization factor, which is an fp18 number. Expected to be 1/1024 , 1/512, ½ but possibly other. This value is given in the register (auto generated) regsJFILinvZnorm

1. Layer 1 (input layer)
   1. 10 neurons, each composed of 22 inputs from the feature extractor, each performs 22 **fp18** multiplications followed by adding the result to a bias Each neuron has its own weights and its own bias.
   2. The result of each neuron of layer 1 is passed through a LUT and the result passed to all neurons of Layer 2.
2. Layer 2 (hidden layer)
   1. 5 neurons, each composed of 10 inputs (corresponding to the 10 neurons in layer 1), each performs 10 **fp18** multiplications followed by adding the result to a bias Each neuron has its own weights and its own bias.
   2. The result of each neuron of layer 2 is passed through a LUT and the result passed to all neurons of layer 3.
3. Layer 3 (output layer)
   1. 4 neurons each composed of 5 inputs (corresponding to the 5 neurons in layer 2), each performs 5 **fp18** multiplications followed by adding the result.
   2. The results from all 4 neurons is then summed, along with a bias, resulting in a 16 bit depth pixel.
4. Output:
   1. The outputs from all neurons of the final layer are gathered, along with a bias and multiplied **in single precision** to produce the final output.

Normalization – before continuing to the pipe, the values coming out of the NN must be normalized back to depth and confidence values, so they are multiplied by the inverse of regsJFILinvZnorm which is auto generated and stored as regsGNRLzNorm . NNI:

The IR network is a simpler version of the depth network, with layers:

1. Layer 1 – 5 neurons.
2. Layer 2 – 3 neurons.
3. Output

Activation function

This block implements the activation function for the neural network architecture. It receives an fp18 and applies a piecewise linear function to it, returning an fp18 number, based on comparator logic between fp18 numbers and fp18 multiplication and addition. As an example for the kind of piecewise linear function we wish to implement see the following graph:



Figure :Activation funtion

### Activiation function implementation for the dnn

This section elaborates on how to implement the kind of piecwise linear function that we want:

1. We list hard coded 7 points in STRICTLY INCREASING order:

Since these are STRICTLY INCREASING values, given an input x, x must satisfy precisely one of the following (note when the inequality is strict and when is it weak):

1. There exists an i such that: x lies between a pair x\_i < x <= x\_(i+1) where x\_i are taken from . In this we set the output to be the following arithmetic in fp18:
2. x < -1 in which case we set the output to be -1.
3. 1 <= x in which case we set the output to be 1.

The implementation of the activation function uses logic and arithmetic that represents a continuous, piecewise linear function.

The registers regs.JFIL.dnnActFuncSlope\* are FW generated from the seven registers

### Activiation function implementation for the inn

Activation function for the IR nn is implemented similarly to the depth nn, using different registers.

## Zero depth handling

Since the nnd may output a pixel with zero depth, special handling must be placed such that pixels with zero depth receive zero confidence, as in the rest of the pipe.

Registers

Table : Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Size** | **Default** | **Range** | **Special values/ description** |
| **GNRL** |  |  |  |  |
|  |  |  |  |  |
| **JFIL** |  |  |  |  |
| regsJFILdnnBypass | 1 | 0 | 0/1 |  |
| regsJFILinnBypass | 1 | 0 | 0/1 |  |
| regsJFILnnEntryMultiplier | 18 bits |  | FP18 number, non zero. | Intended to normalize the incoming features. |
| regsJFILTact1B0 |  |  |  |  |
| regsJFILTact1B1 |  |  |  |  |
| regsJFILTact1B2 |  |  |  |  |
| regsJFILTact1B3 |  |  |  |  |
| regsJFILTact1B4 |  |  |  |  |
| regsJFILTact1B5 |  |  |  |  |
| regsJFILTact1B6 |  |  |  |  |
| regsJFILTact1B7 |  |  |  |  |
|  |  |  |  |  |
| regsJFILTact1A1 |  |  |  |  |
| regsJFILTact1A2 |  |  |  |  |
| regsJFILTact1A3 |  |  |  |  |
| regsJFILTact1A4 |  |  |  |  |
| regsJFILTact1A5 |  |  |  |  |
| regsJFILTact1A6 |  |  |  |  |
|  |  |  |  |  |
| regsJFILact1X0 |  |  |  |  |
| regsJFILact1X1 |  |  |  |  |
| regsJFILact1X2 |  |  |  |  |
| regsJFILact1X3 |  |  |  |  |
| regsJFILact1X4 |  |  |  |  |
| regsJFILact1X5 |  |  |  |  |
| regsJFILact1X6 |  |  |  |  |
| regsJFILact1X7 |  |  |  |  |
| regsJFILTact2B0 |  |  |  |  |
| regsJFILTact2B1 |  |  |  |  |
| regsJFILTact2B2 |  |  |  |  |
| regsJFILTact2B3 |  |  |  |  |
| regsJFILTact2B4 |  |  |  |  |
| regsJFILTact2B5 |  |  |  |  |
| regsJFILTact2B6 |  |  |  |  |
| regsJFILTact2B7 |  |  |  |  |
|  |  |  |  |  |
| regsJFILTact2A1 |  |  |  |  |
| regsJFILTact2A2 |  |  |  |  |
| regsJFILTact2A3 |  |  |  |  |
| regsJFILTact2A4 |  |  |  |  |
| regsJFILTact2A5 |  |  |  |  |
| regsJFILTact2A6 |  |  |  |  |
|  |  |  |  |  |
| regsJFILact2X0 |  |  |  |  |
| regsJFILact2X1 |  |  |  |  |
| regsJFILact2X2 |  |  |  |  |
| regsJFILact2X3 |  |  |  |  |
| regsJFILact2X4 |  |  |  |  |
| regsJFILact2X5 |  |  |  |  |
| regsJFILact2X6 |  |  |  |  |
| regsJFILact2X7 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Test Plan

## General randomization

Table 4 describes the general test randomization distribution.

Table 4: General randomization distribution

|  |  |  |
| --- | --- | --- |
| **Name** | **Special values/ description** | **Distribution** |
| **JFIL** |  |  |
| regsJFILdnnBypass | 0 | 1% |
| 1 | 99% |
| regsJFILinnBypass | 0 | 1% |
| 1 | 99% |