Efficient Pattern Recognition Algorithm Including a Fast Retina Keypoint FPGA Implementation

Lester Kalms
Technische Universität
Dresden, Germany
lester.kalms@tu-dresden.de

Maximilian Hajduk
Technische Universität
Dresden, Germany
maximilian.hajduk@tu-dresden.de

Diana Göhringer
Technische Universität
Dresden, Germany
diana.goehringer@tu-dresden.de

Lester Kalms
Barcelona, 09. September 2019

Content

Motivation

What is pattern recognition?

- Feature detector: finds points of interest in images (e.g. corners, edges or blobs)
- Feature descriptor: describes features to be comparable with others

Why pattern recognition?

- Still algorithms make use of pattern recognition (e.g. object tracking or SLAM)
- Most neural networks need a high amount of input data

Overview

First full FPGA implementation of the FREAK descriptor

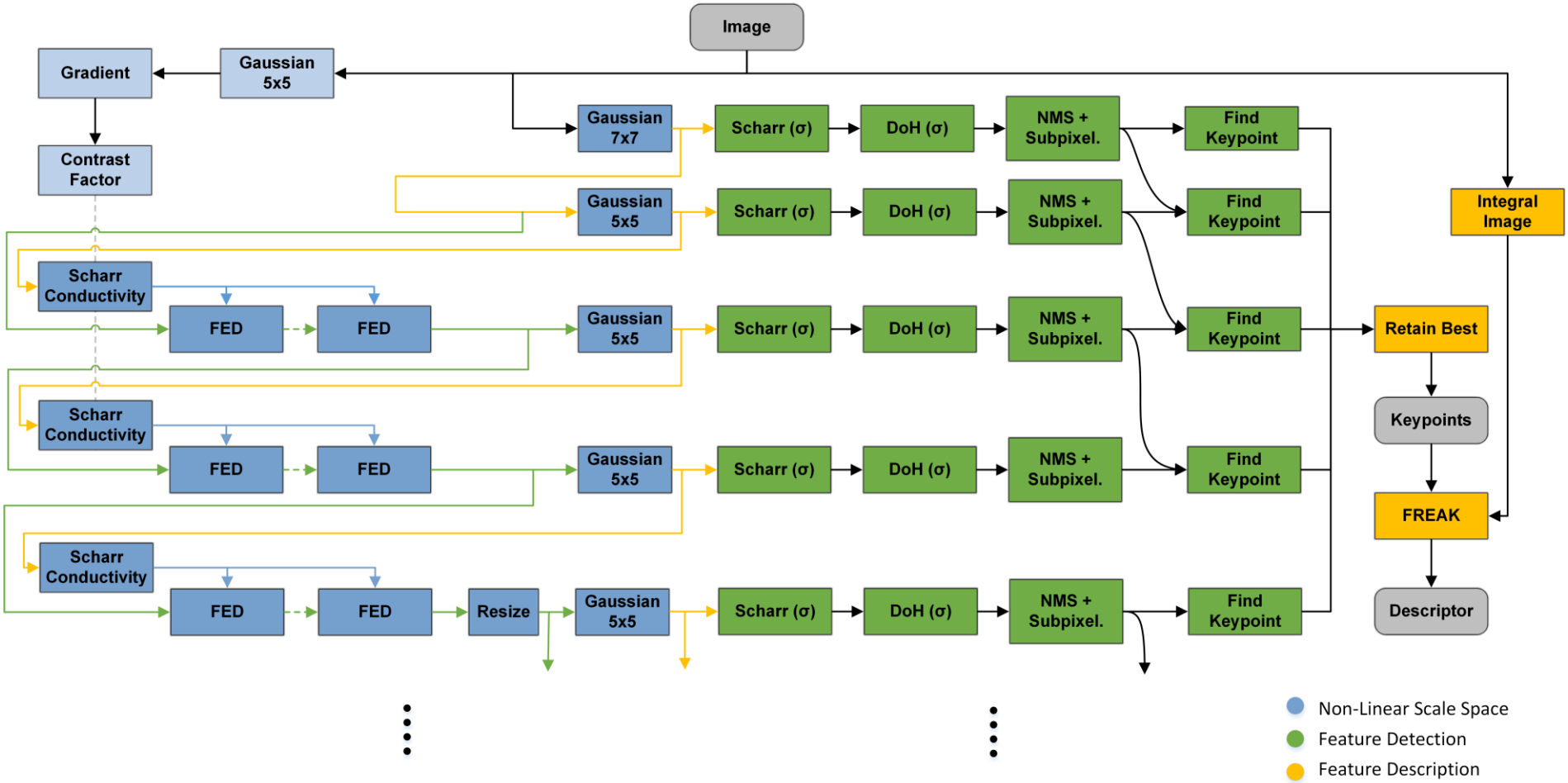
- Related work does not contain orientation computation
- Contains optimized pattern generator (partially pre-computed)
- Frequency optimized integral image with parallel prefix sum

Improved pattern recognition algorithm

- Compared to different combinations of AKAZE, ORB, BRISK and FREAK
- Improved and combined AKAZE detector & FREAK descriptor
- Added a Retain Best function

SW Implementation

AKAZE feature detection & FREAK feature description



HW Implementation

Integral Image Function

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Input Image

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Output Image

SW: $I_{x,y} = i_{x,y} + I_{\mathrm{x-1,y}} + I_{\mathrm{x,y-1}} - I_{\mathrm{x-1,y-1}}$

HW: $s = s + i_{x,y}$

$a = [(y > 0) \rightarrow (s + buf_x)] \wedge [(y \leq 0) \rightarrow s]$

$I_{x,y} = buf_x = a$

HW Implementation

Integral Image Function

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Input Image

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Output Image

SW: $I_{x,y} = i_{x,y} + I_{\mathrm{x-1,y}} + I_{\mathrm{x,y-1}} - I_{\mathrm{x-1,y-1}}$

HW: $s = s + i_{x,y}$

$a = [(y > 0) \rightarrow (s + buf_x)] \wedge [(y \leq 0) \rightarrow s]$

$I_{x,y} = buf_x = a$

HW Implementation

Integral Image Function

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Input Image

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Output Image

SW: $I_{x,y} = i_{x,y} + I_{\mathrm{x-1,y}} + I_{\mathrm{x,y-1}} - I_{\mathrm{x-1,y-1}}$

HW: $s = s + i_{x,y}$

$a = [(y > 0) \rightarrow (s + buf_x)] \wedge [(y \leq 0) \rightarrow s]$

$I_{x,y} = buf_x = a$

HW Implementation

Integral Image Function

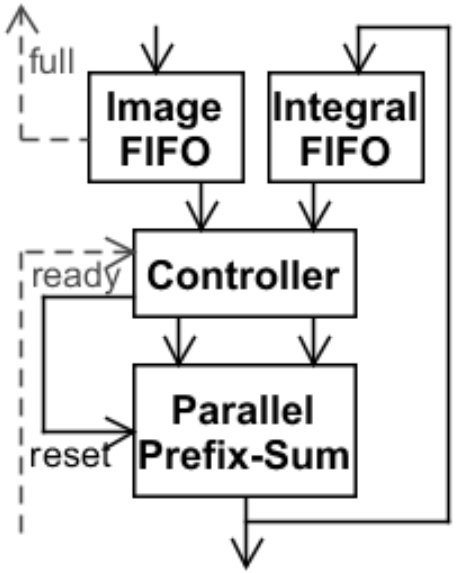
- 4 Pixels each clock cycle
- High frequency

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

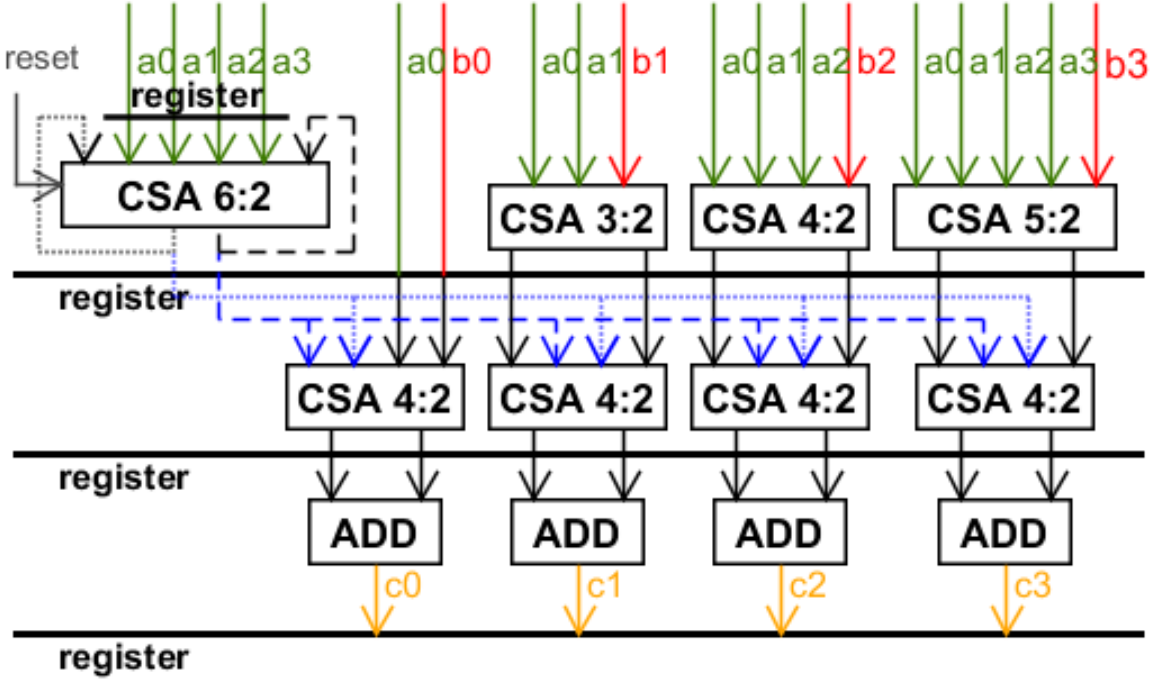
Input Image

1	2	3	4	5	6	7	8
2	4	6	8	10	12	14	16
3	6	9	12	15	18	21	24
4	8	12	16	20	24	28	32

Output Image



CSA = Carry Save Adder

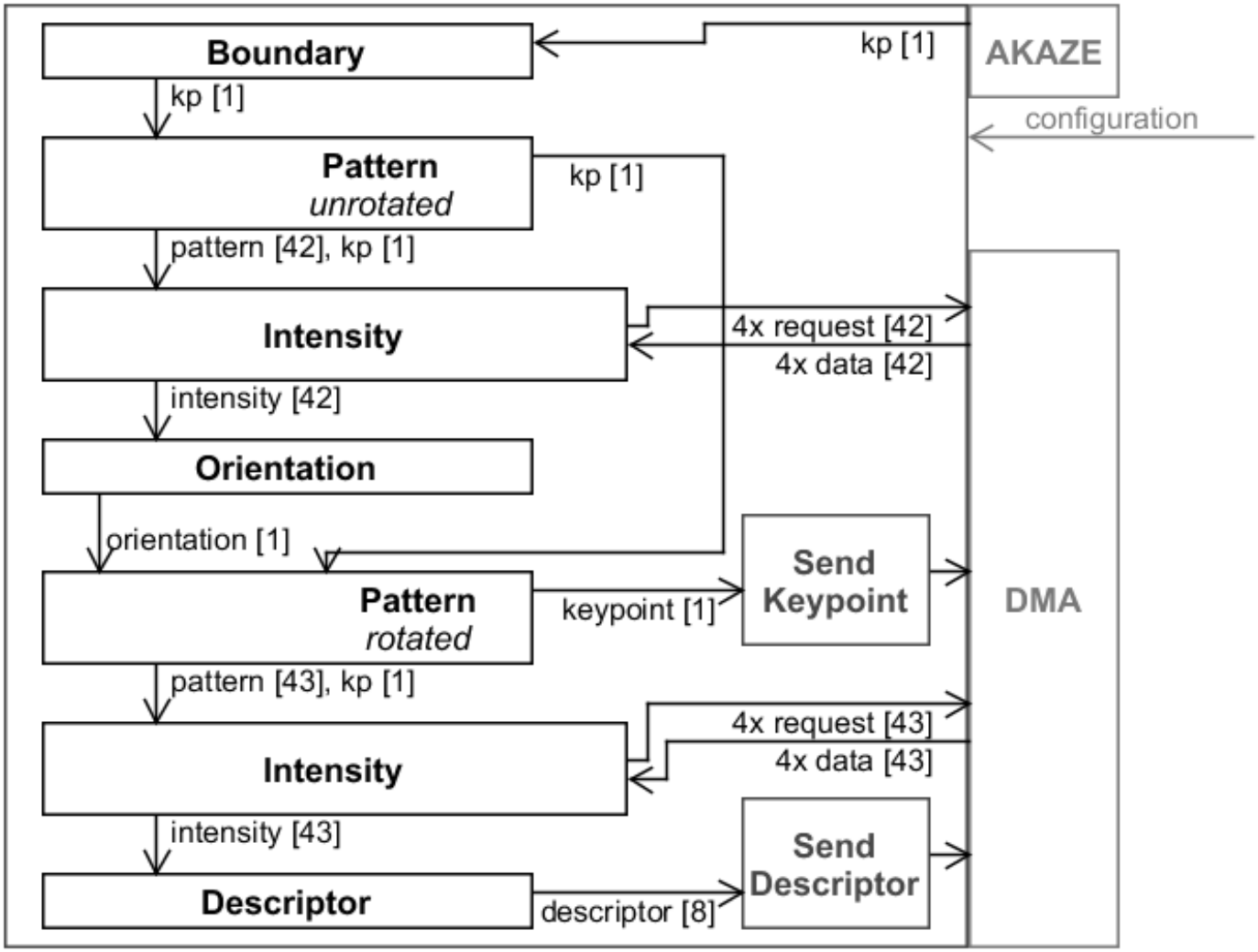


HW Implementation

FREAK Descriptor

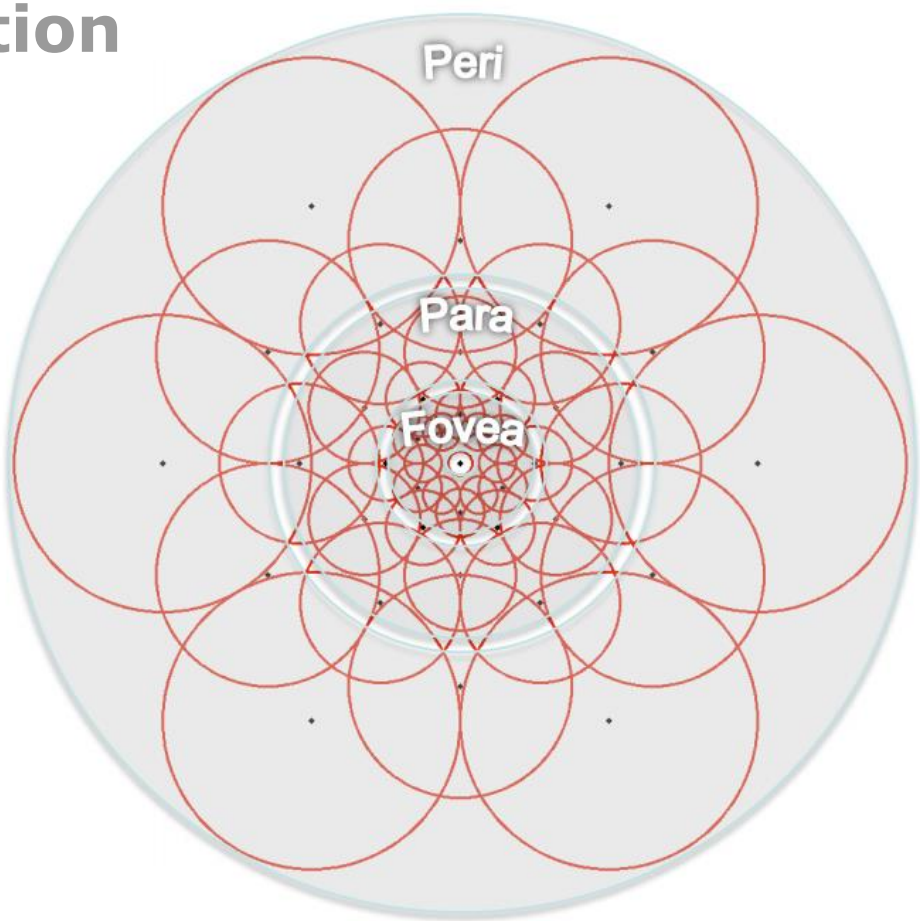
Implementation

- VHDL
- Fixed-Point
- Pipelined



HW Implementation

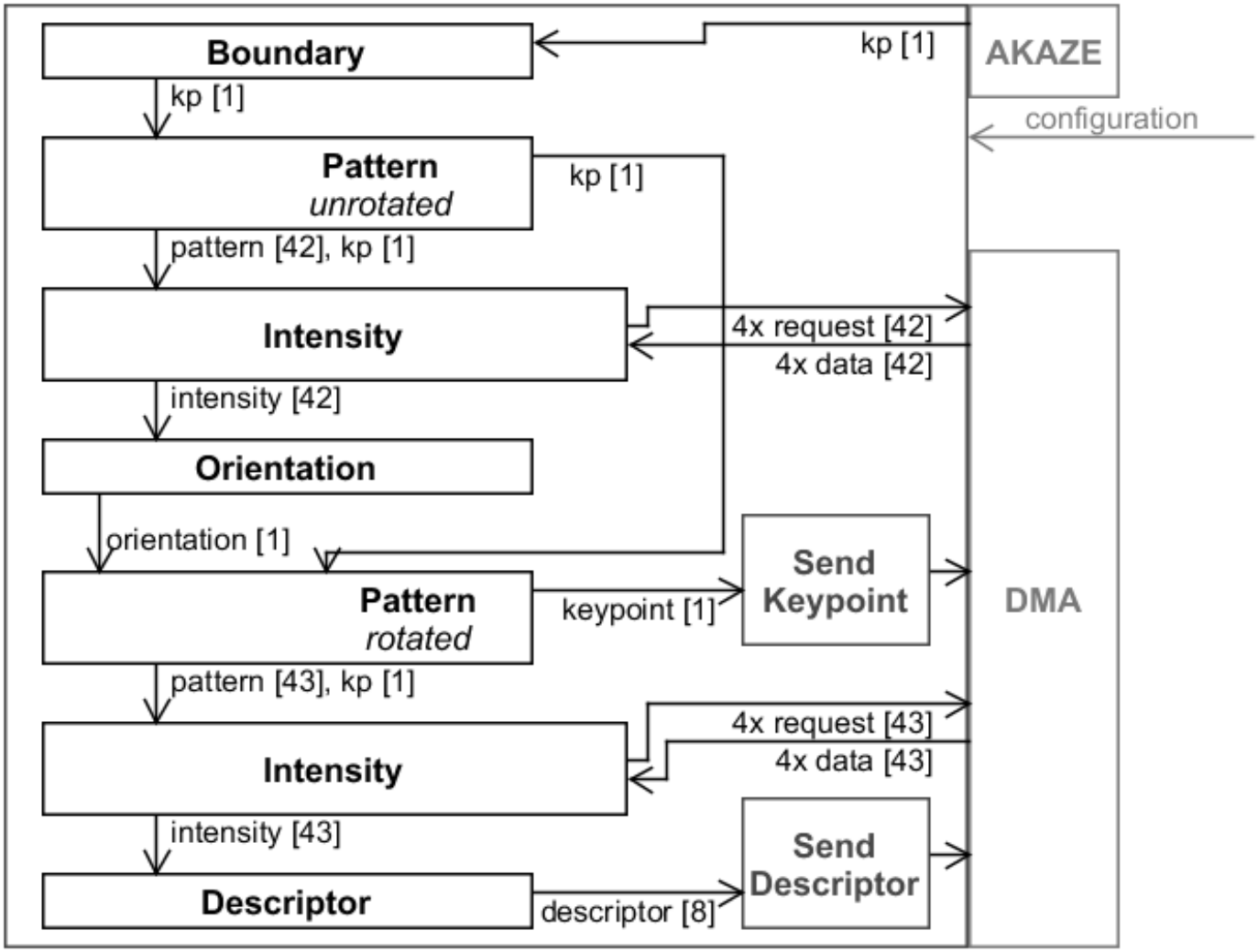
FREAK Descriptor (Pattern)



FREAK sampling pattern similar to the retinal ganglion cells [1]

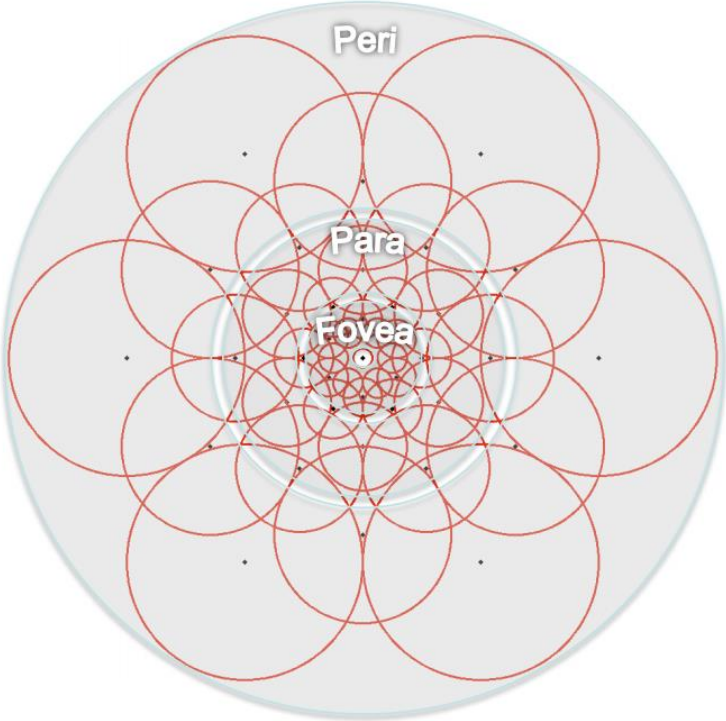
HW Implementation

FREAK Descriptor



HW Implementation

FREAK Descriptor (intensity)



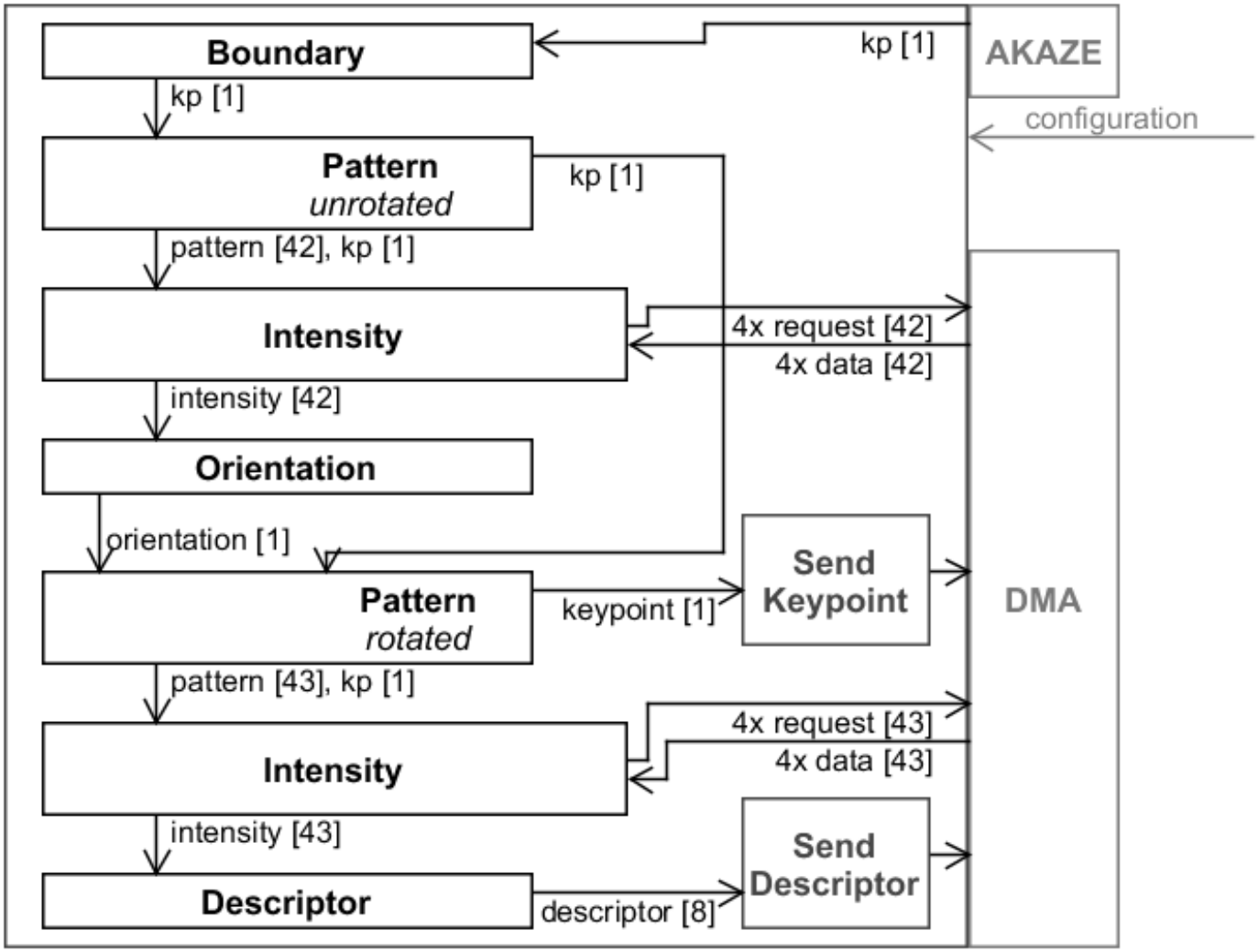
FREAK sampling pattern similar to the retinal ganglion cells [1]

Intensity value calculation

$$i = \frac{I_{y_b,x_r} - I_{y_b,x_l} + I_{y_t,x_l} - I_{y_t,x_r}}{(x_r - x_l) \cdot (y_b - y_t)}$$

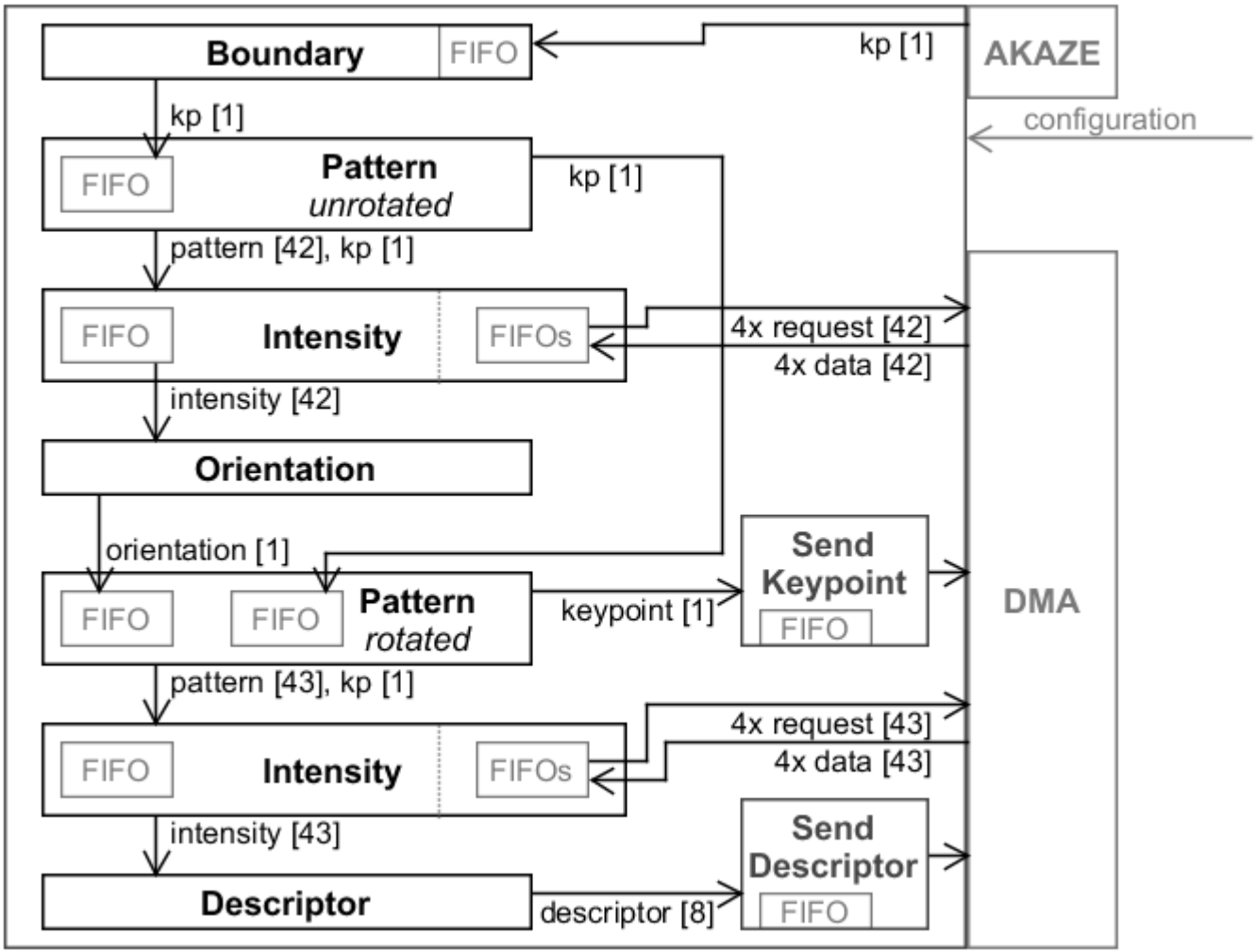
HW Implementation

FREAK Descriptor



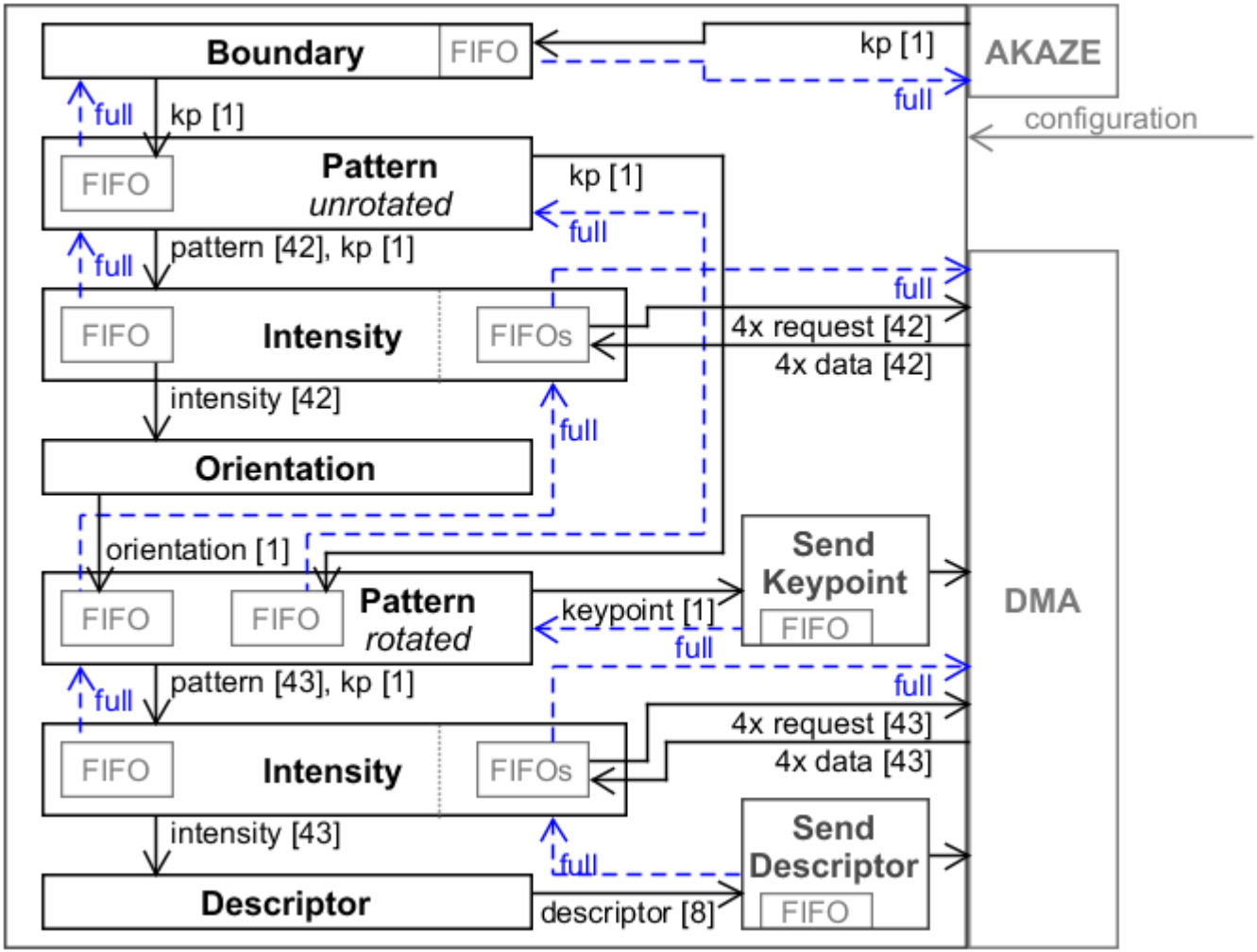
HW Implementation

FREAK Descriptor



HW Implementation

FREAK Descriptor



Evaluation

Software (Repeatability)

Setup

- Core-i7 7700
- Oxford Dataset: 800x640 - 1000x700
- Quality: luminance, blur & compression
- Position: Scale, rotation & viewpoint

	Position			Quality			Time
detect + describe	match ratio	avg. inlier ratio	inlier ratio	match ratio	avg. inlier ratio	inlier ratio	time (ms)
akaze + akaze	21.6	63.3	85.4	42.5	**82.2**	85.0	67.0
akaze + freak	**23.1**	57.2	84.7	**46.8**	83.2	86.2	70.3
akaze + brisk	19.4	59.4	**87.8**	46.6	**85.3**	88.0	77.0
akaze + orb	11.5	37.8	86.2	37.8	84.4	**88.5**	68.4
brisk + brisk	17.3	**66.1**	85.1	26.4	70.6	87.7	29.9
brisk + freak	**21.7**	58.7	81.4	31.6	64.0	82.5	29.5
brisk + orb	11.2	49.8	82.0	26.5	67.7	86.3	29.9
orb + orb	15.0	61.2	82.2	46.6	75.9	84.8	**16.0**
orb + freak	19.3	65.8	84.9	**48.2**	79.3	87.8	**17.7**
orb + brisk	16.9	**66.8**	**85.8**	44.9	81.3	**89.5**	25.3
proposed	**23.1**	**70.2**	**87.9**	**60.9**	**87.2**	**92.8**	**24.9**

Evaluation

Hardware (Timing)

Setup

- ZedBoard (Zynq 7020) & Vivado 2018.2
- Vhdl, Fixed-point data types, 2048 key-points per frame

Frequency / FPS (detector)

- 185 MHz (synthesis)
- Simulation: for 150 MHz -> 850 fps (5.85 Gbit/s bandwidth)
- Real system: for 167 MHz -> 73.4 fps (including DMAs)

Frequency (Integral)

- 204 MHz (synthesis)
- 3ns for parallel prefix sum

Evaluation

Hardware (Resources)

Module	FF	LUT	LUTRAM	BRAM	DSP
Parallel sum	596	448	0	0	0
Controller	209	51	0	0	0
Integral (%)	**(0.75) 800**	**(0.97) 515**	**(0.01) 2**	**(1.79) 2.5**	**(0.00) 0**
Boundary	334	131	33	1	0
Pattern unrotated	311	12	0	2	1
Pattern rotated	1673	1570	104	4.5	3
Intensity	1183	1719	24	4.5	5
Orientation	1292	1292	1	0	6
Descriptor	2414	1837	0	0	0
Send keypoint	130	59	0	1.5	0
Send descriptor	187	67	0	1.5	0
Descriptor (%)	**(7.63) 8115**	**(17.17) 9133**	**(1.06) 184**	**(13.93) 19.5**	**(9.09) 20**

Evaluation

HW/SW (Comparison)

Algorithm	Inliers Ratio	Avg. Inliers Ratio (arithmetic mean)
Proposed SW	91.4	78.7
Proposed HW	90.9	77.0
ORB + FREAK	88.4	72.5
ORB + BRISK	88.0	74.0

Conclusion

Summary

- Proposed implementation of AKAZE detector + FREAK descriptor
- Improved algorithm, parameters and added Retain Best function
- HW implementation of FREAK detector + Integral image
- Achieved good computation time and repeatability in comparison to others
- Achieved a low resource consumption

Outlook

- Combine AKAZE detector with FREAK descriptor in HW
- Integrate it into computer vision algorithm

Acknowledgment

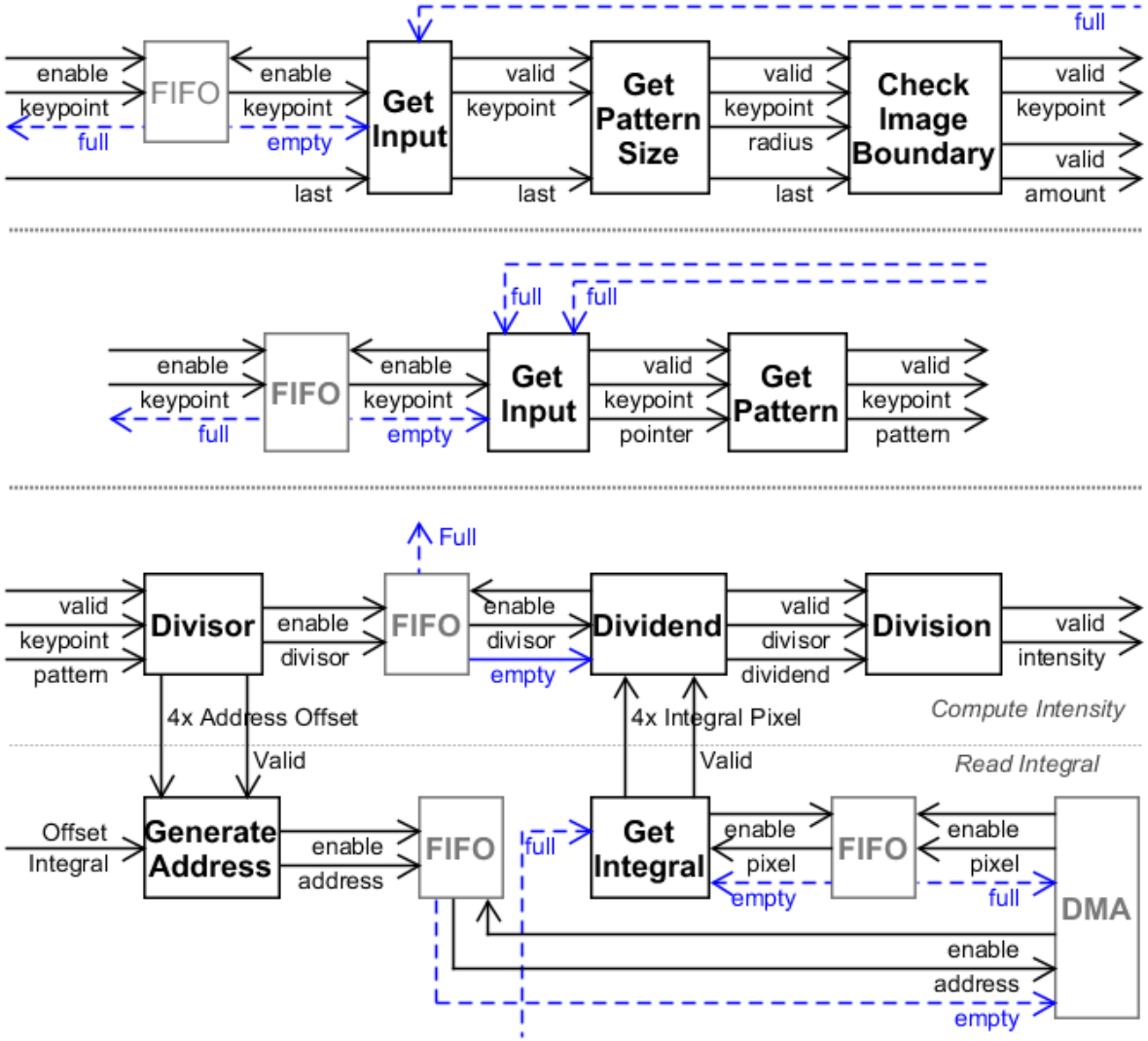
This work has been funded by the German Federal Ministry of Education and Research BMBF as part of the PARIS project under grant agreement number 16ES0657.

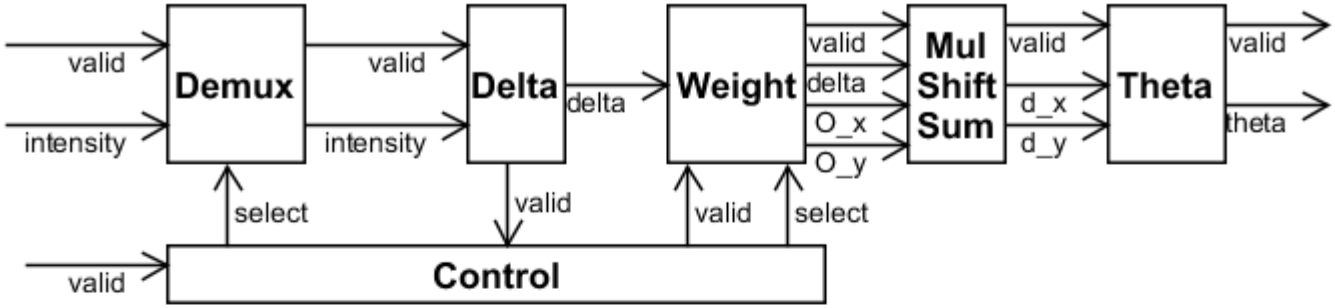
References

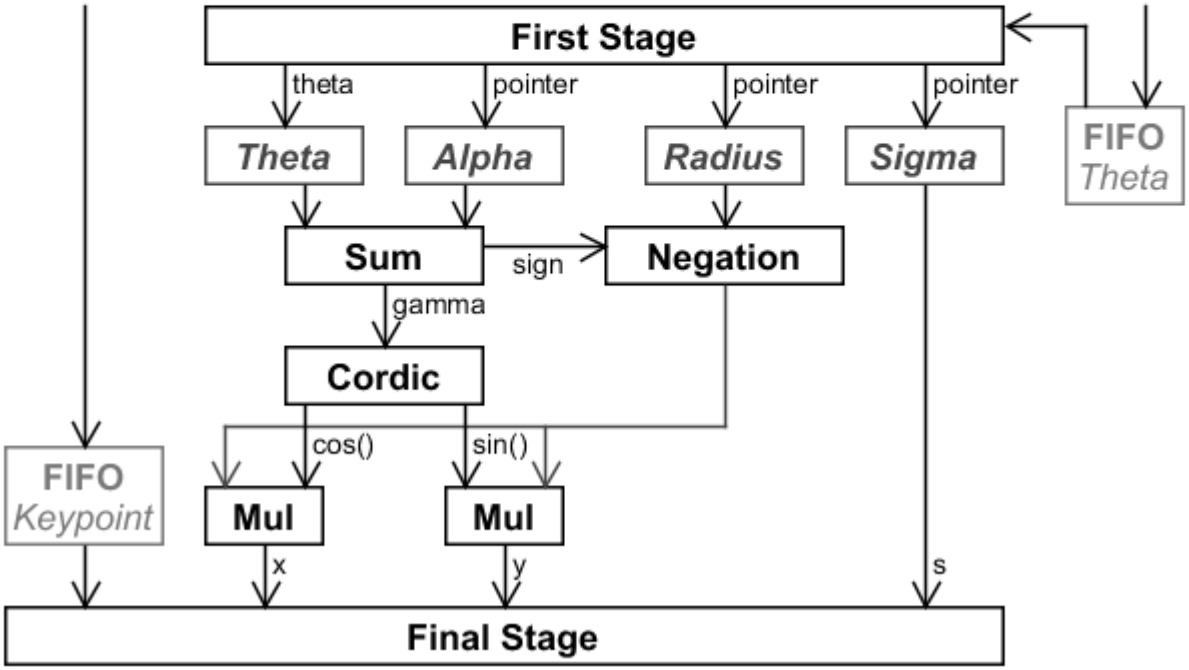
[1] A. Alahi, R. Ortiz, and P. Vandergheynst. *Freak: Fast retina keypoint*. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 510–517, Jun 2012.

Thank you for your attention!

Any questions?


full
enable
keypoint
full
FIFO
enable
keypoint
empty
Get Input
valid
keypoint
last
Get Pattern Size
valid
keypoint
radius
last
Check Image Boundary
valid
keypoint
valid
amount
full
full
enable
keypoint
full
FIFO
enable
keypoint
empty
Get Input
valid
keypoint
pointer
Get Pattern
valid
keypoint
pattern
Full
valid
keypoint
pattern
Divisor
enable
divisor
FIFO
enable
divisor
empty
Dividend
valid
divisor
dividend
Division
valid
intensity
4x Address Offset
Valid
4x Integral Pixel
Valid
Compute Intensity
Read Integral
Offset
Integral
Generate Address
enable
address
FIFO
full
Get Integral
enable
pixel
empty
FIFO
enable
pixel
full
DMA
enable
address
empty


valid
intensity
Demux
valid
intensity
Delta
delta
Weight
valid
delta
O_x
O_y
Mul
Shift
Sum
valid
d_x
d_y
Theta
valid
theta
select
valid
valid
select
valid
Control


First Stage
theta
pointer
pointer
pointer
Theta
Alpha
Radius
Sigma
FIFO
Theta
Sum
sign
Negation
gamma
Cordic
cos()
sin()
FIFO
Keypoint
Mul
Mul
x
y
s
Final Stage