RECRLab
Reconfigurable Computing Research Laboratory

A Scalable Pipelined Architecture for Biomimetic Vision Sensors

DANIEL LLAMOCCA AND BRIAN K. DEAN

Electrical and Computer Engineering Department,

Oakland University

September, 3rd, 2015

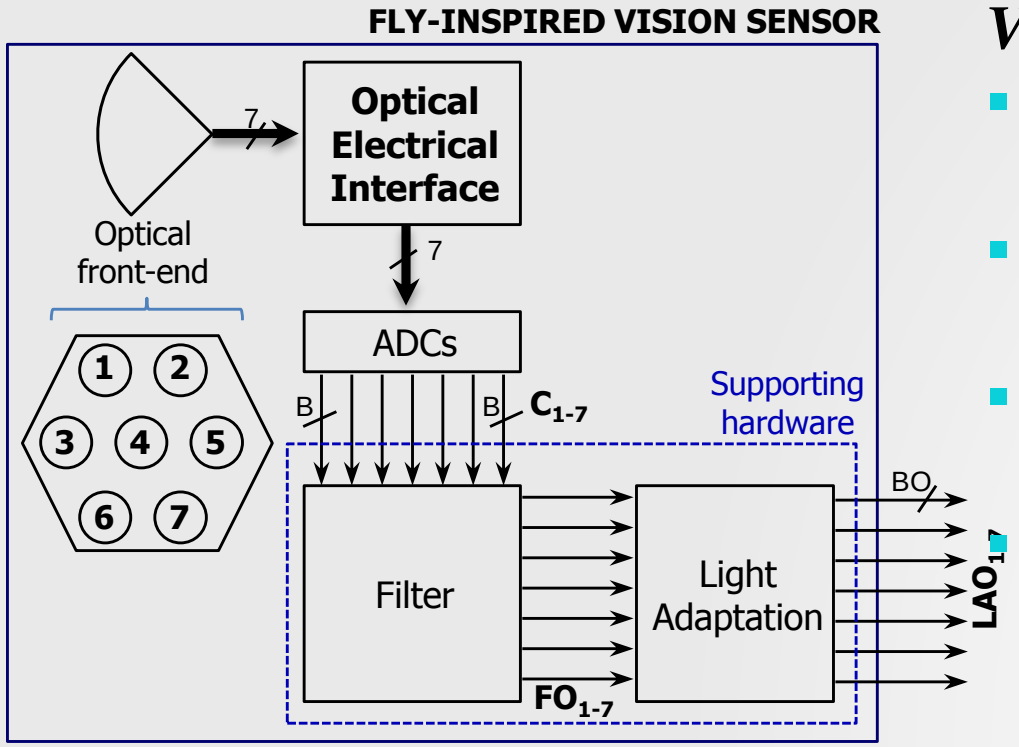
OAKLAND UNIVERSITY

Outline

- Motivation
- Contributions
- Methodology
- Results
- Conclusions

Motivation

- ***Fly-inspired vision algorithms can outperform traditional image processing algorithms in motion detection and in-flight obstacle tracking and interception.***
  - *Applications include high-speed target tracking for unmanned aerial and ground vehicles, structural monitoring.*
- ***Dedicated hardware implementations are desired when the large amounts of data are to be processed in parallel.***

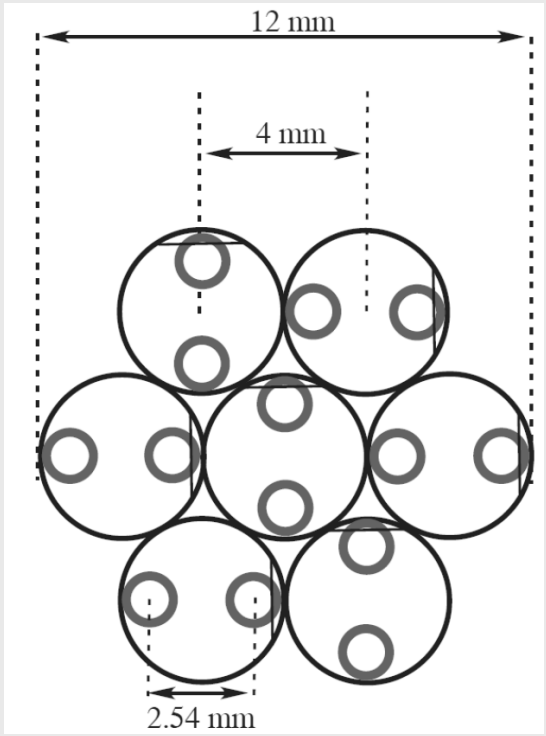


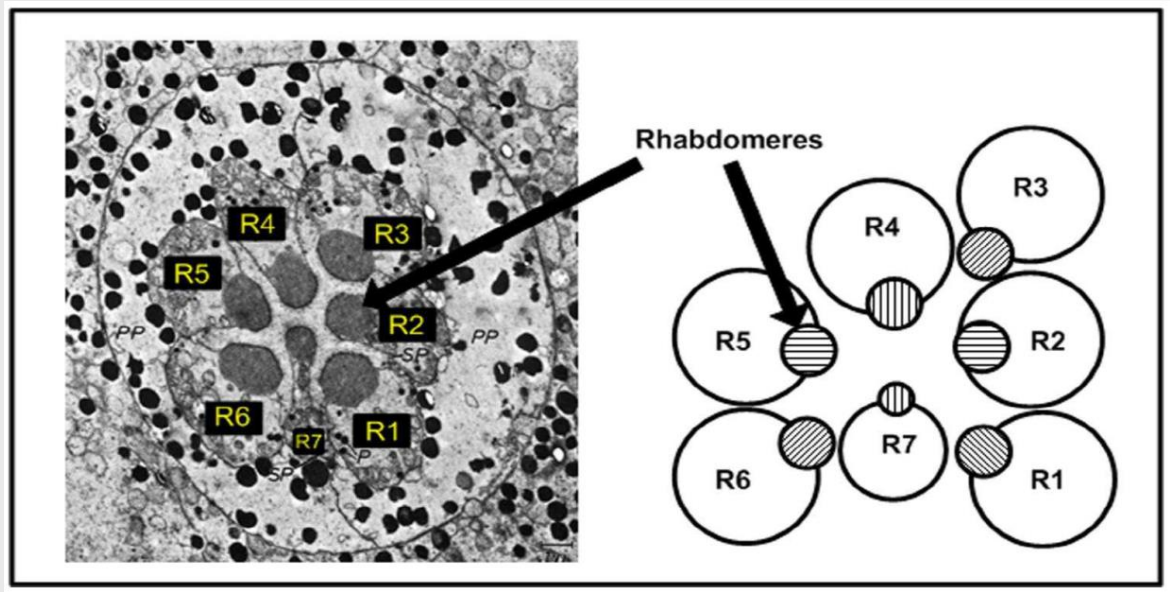
***Vision Sensor**:*

- *Optical front-end*
- *Optical electrical interface*
- *Analog-to-digital converters*
- *Supporting **digital hardware for filtering and light adaptation**.*

Motivation

- ***Vision Sensor: Optical front-end***
  - *Plano-convex lens (12 mm diameter and 12 mm focal length) placed above seven photodiodes arranged in a hexagonal pattern*
  - *Hexagonal pattern approximates fly optical arrangement.*
  - *Photoreceptor response: overlapping Gaussians*



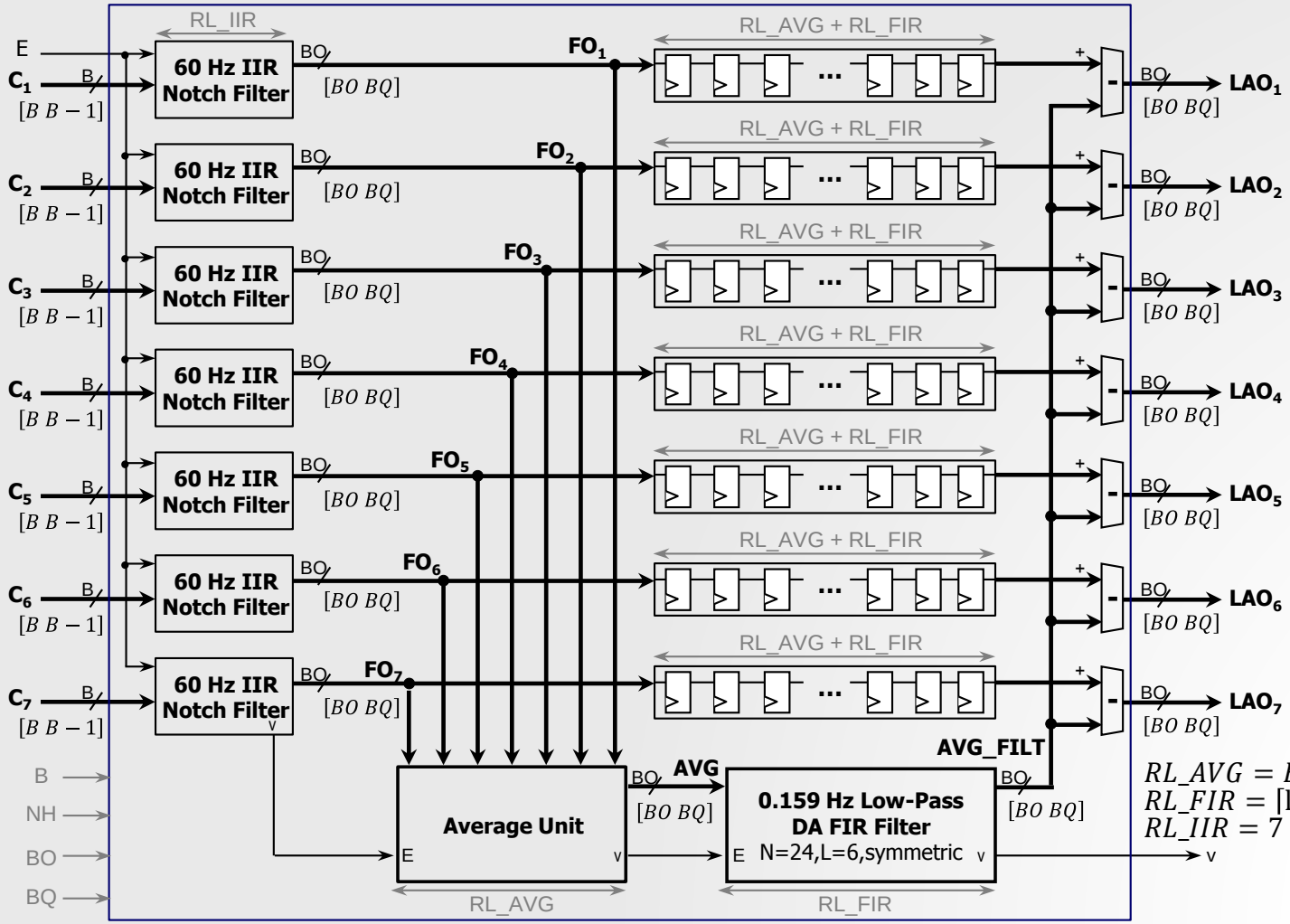


Contributions

- ***Fully pipelined and Scalable Hardware Implementation for the Biomimetic Sensor***
  - *The fully-customizable fixed-point architecture allows users to quickly modify design parameters (# of input bits, output format, # of bits per of iterations, # of bits of filters' coefficients).*
  - *Fully-pipelined architecture is achieved by unrolling the IIR filter architecture.*
- ***Generic VHDL code validated on an FPGA***
  - *The fully-parameterized RTL VHDL code is not tied to a particular device or vendor.*
- ***Design Space Exploration***
  - *The fully-parameterized VHDL code allows us to create a set of different hardware profiles by varying the design parameters. We can then explore trade-offs among design parameters, accuracy, resources, and execution time.*

Methodology

- ***Block Diagram:*** *Data path uses fixed-point representation:*
  - *Input: [B B-1], Output/Intermediate Signals [BO BQ]*
  - *Design Parameters: B, BO, BQ, NH (# of bits per filters' coefficients)*



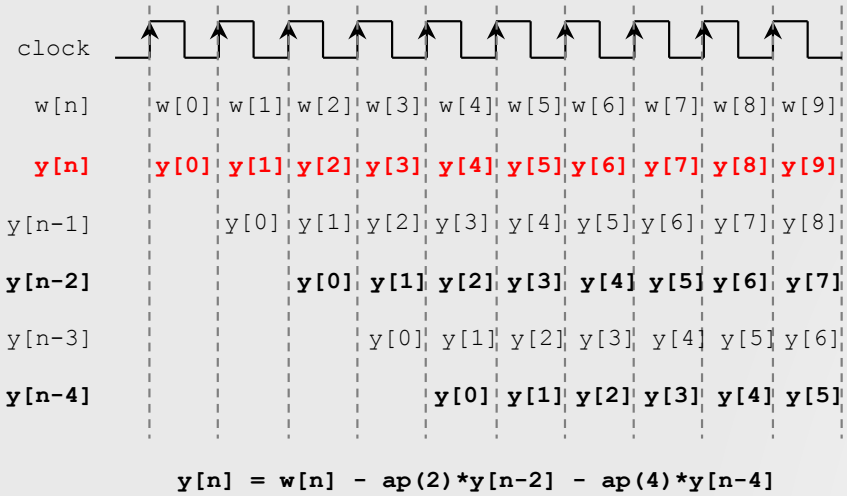
Architecture:
7 IIR filters,
1 average unit,
1 FIR filter,
7 subtractors,
7 chain registers (synchronization registers that allow for pipelining)

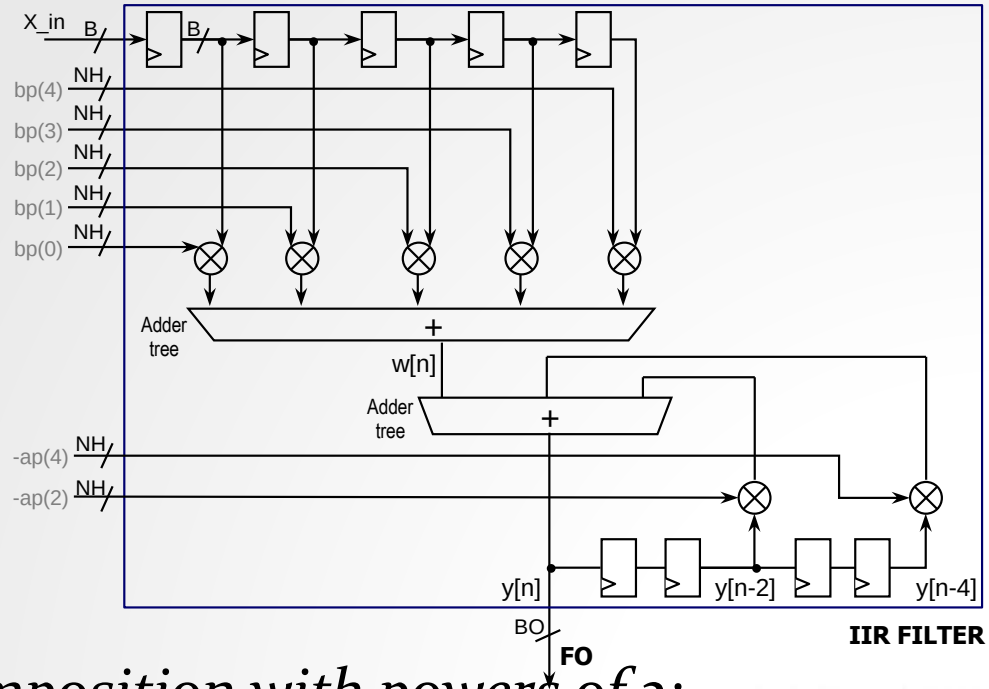
$$RL\_AVG = BO + 2\lceil \log_2 N \rceil$$
$$RL\_FIR = \lceil \log_2 BO + 1 \rceil + \lceil \log_2 N/2L \rceil + 2$$
$$RL\_IIR = 7$$

Methodology

- ***IIR Filter (60 Hz Notch filter):*** *fs=1 KHz, 2nd order IIR filter*
  - *Direct implementation: data dependencies prevent pipelining*
  - *Look-ahead transformation: The 2nd order IIR filter is turned into a 4th order IIR filter with no data dependencies.*

$$H(z) = \frac{b_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}} \rightarrow HP(z) = \frac{bp_0+bp_1z^{-1}+bp_2z^{-2}+bp_3z^{-3}+bp_4z^{-4}}{1+ap_2z^{-2}+ap_4z^{-4}}$$



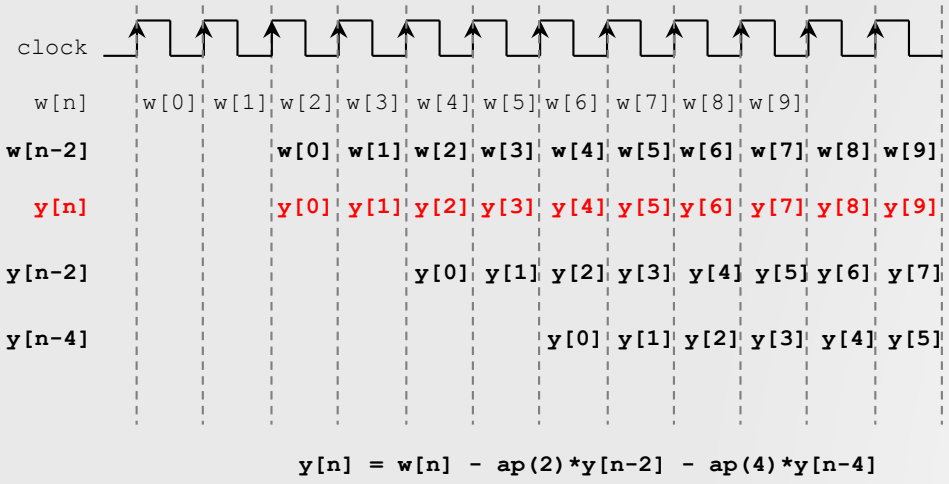


Assumption: The adder tree does not have delay units

- *Scattered look-ahead decomposition with powers of 2: coefficients of the 4th order IIR filter avoid instability.*

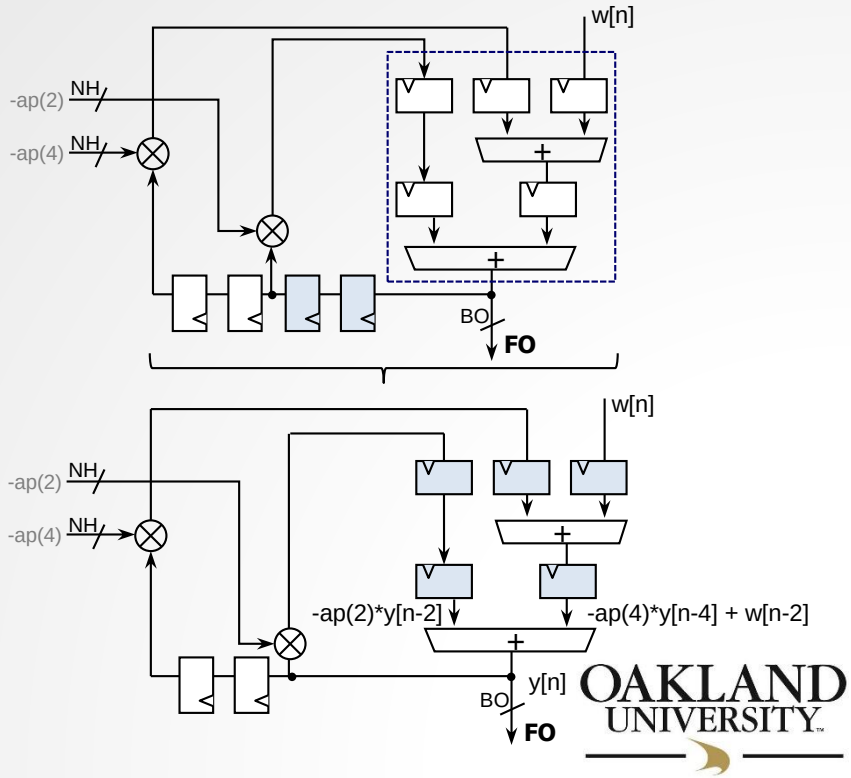
Methodology

- ***IIR Filter (60 Hz Notch filter):*** *fs=1 KHz, 2nd order IIR filter*
  - ***Retiming:*** *An actual adder tree usually includes register levels in order to increase the frequency of operation.*
  - *For example, a 3-input adder tree usually has 2 register levels. The delay breaks the pipeline of the previous figure.*
  - *Retiming is used here to address this issue: the delay units that create y[n-2] are embedded into the two register levels of the adder tree.*



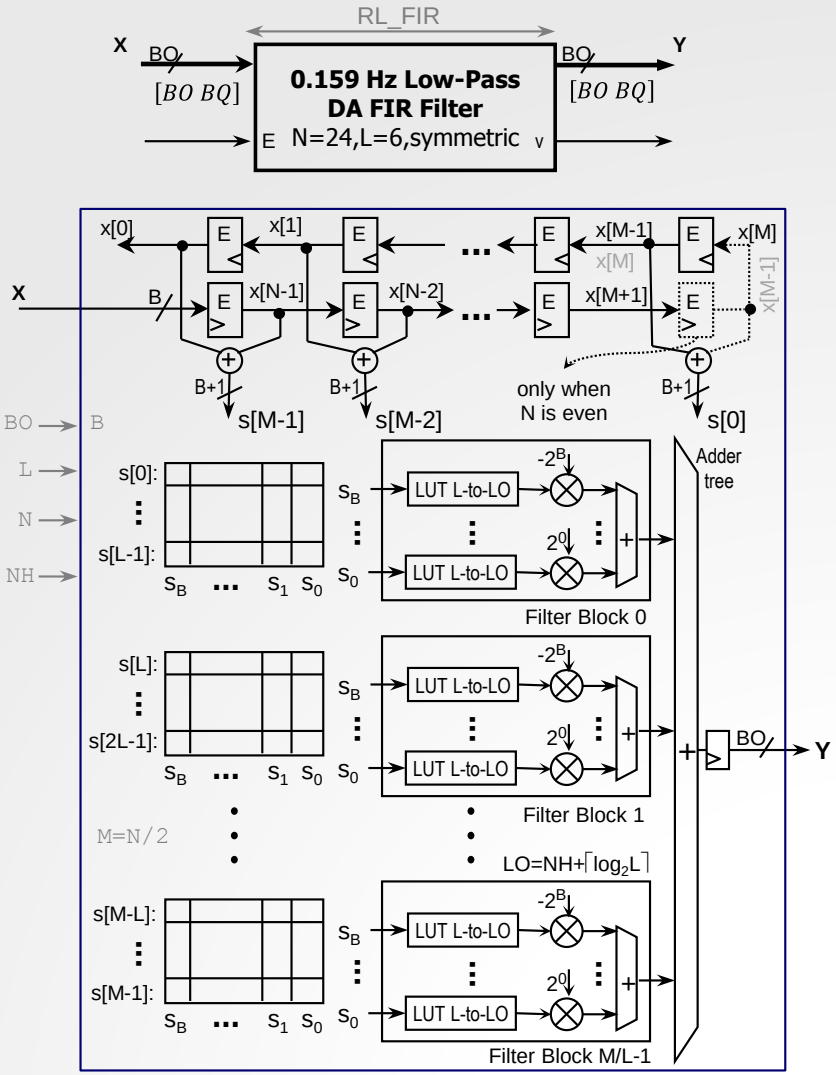
$$y[n] = w[n] - ap(2)*y[n-2] - ap(4)*y[n-4]$$

Assumption: The adder tree has two delay units



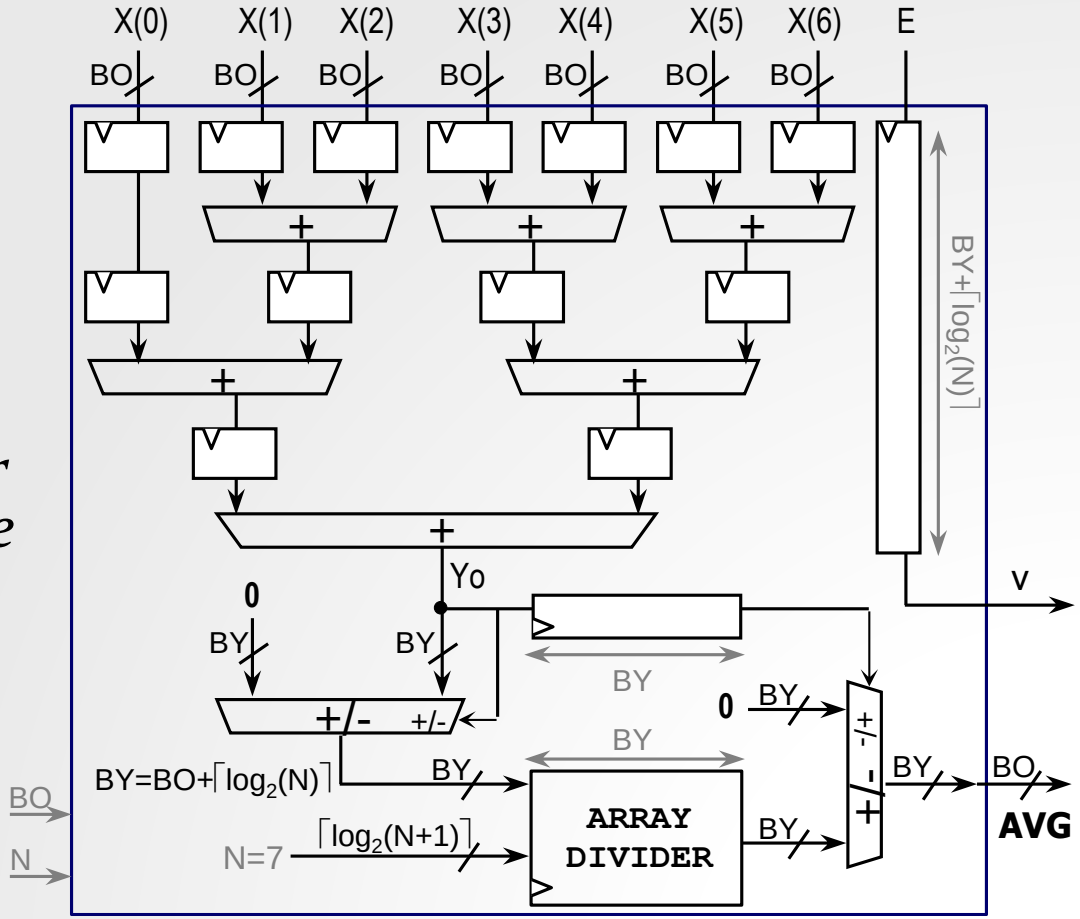
Methodology

- ***FIR Filter with Distributed Arithmetic***
  - *Efficient multiplier-less implementation where coefficients are constant.*
  - *Cut-off frequency: 0.159 Hz.*
  - *Stopband: -41dB*
  - *24-tap symmetric low-pass filter*
  - *Fully pipelined system with I/O delay of RL_FIR.*
  - *LUT input size = 6*
  - *Coefficients format: [NH NH-1]*
  - *Constant coefficients loaded as a text file.*



Methodology

- ***Averaging unit***
  - *The seven outputs FOx (x=1..7) are averaged out by this block. This requires a 7-input pipelined adder tree and an array divider.*
- *Input format: [BO BQ]*
- *Output format: [BO BQ]*
- *I/O delay:* $RL\_AVG = BO + 2\lceil \log_2 N \rceil$
- *The adder tree output requires* $\lceil \log_2 N \rceil$ *extra integer bits, but the divider gets rid of those bits, hence the output of the Average Unit only needs BO bits.*
- *Accuracy can always be increased by incrementing the number of fractional bits the divider generates.*

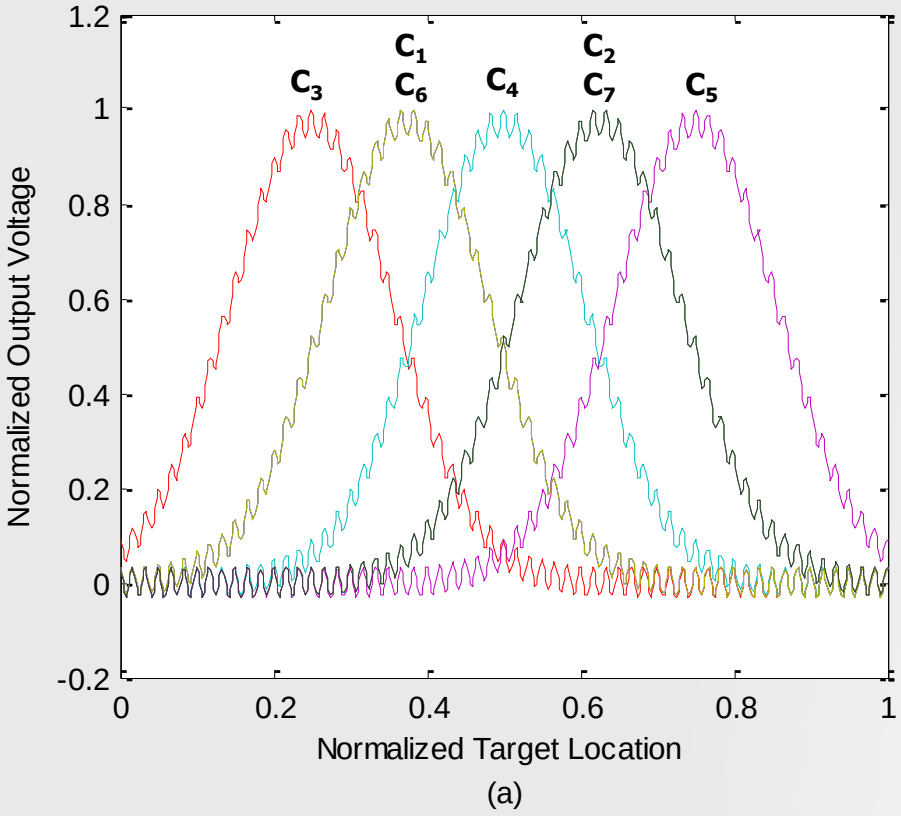


Methodology

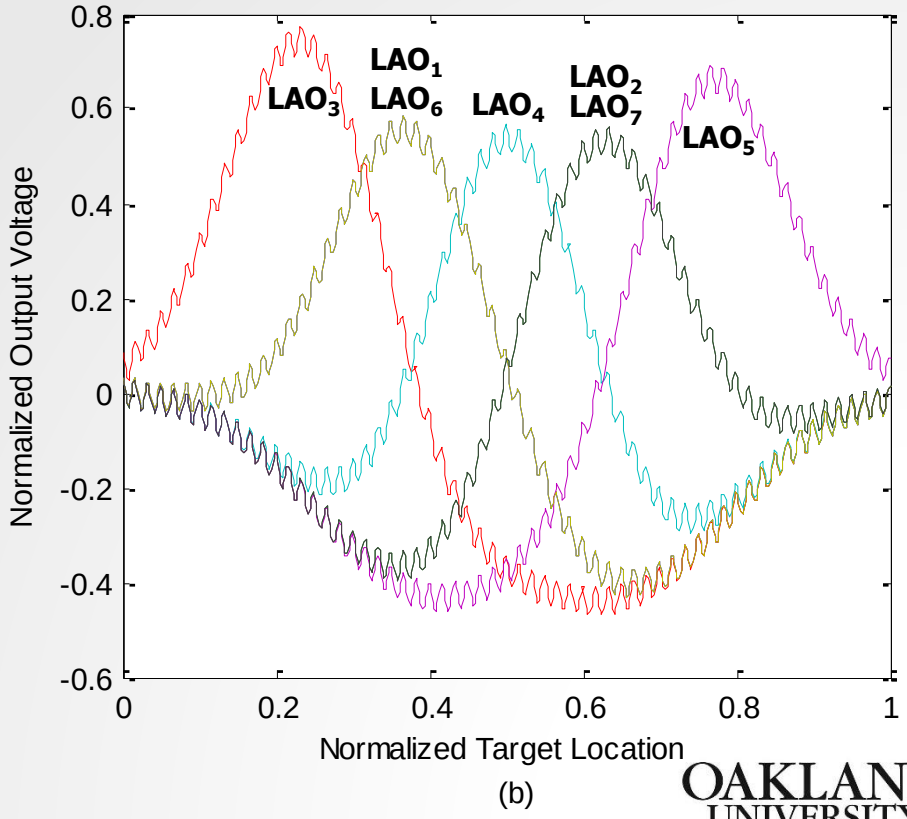
- ***Experimental Setup:***
  - *Input signals: seven overlapping Gaussian-shaped signals (close match to the angular displacement response of the fly's rhabdomers). 500 samples generated per channel, values quantized with 8, 10, and 12 bits per sample.*
  - *Design Space Exploration: Parameters:*
    - *BO=16, NH =10,12,14,16, B=8,10,12, BQ=10,11,12,13,14*
  - *Accuracy measurement: PSNR*
    - *Test 1: FPGA and software (MATLAB) implementation uses the quantized input samples. This allows us to study the effect of the fixed-point architecture on accuracy.*
    - *Test 2: Only FPGA uses the quantized input samples. This allows us to study the effect of input quantization and the fixed-point architecture on accuracy.*
  - *Synthesis of VHDL code: Artix-7 XC7A100T FPGA*

Results

- ***Input/Output Behavior***
  - *Case: B=12, [BO BQ] = [16 14], NH=16. There is not much visual difference if we change the parameters.*
  - *The output signals constitute the output of a primary signal path required for all image processing techniques.*



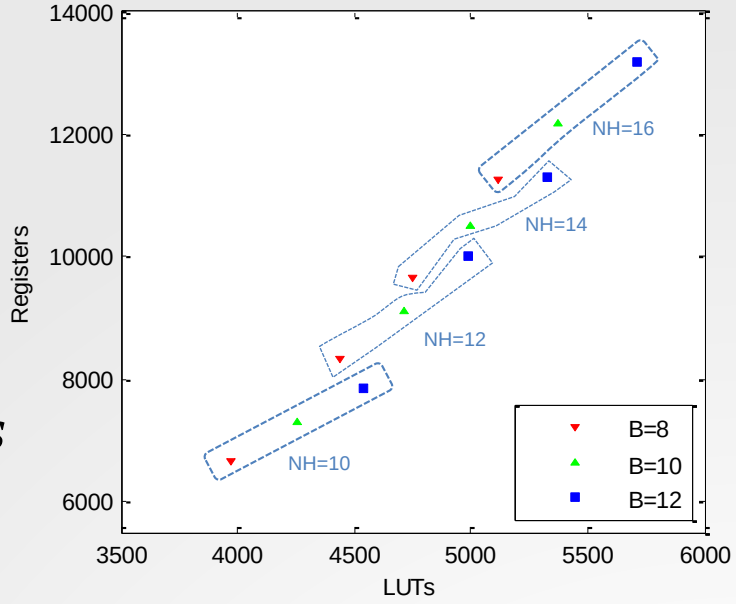
(a)



(b)

Results

- ***Hardware Resource Utilization***
  - *The figure shows resources (in terms of 6-input LUTs and registers) for all the cases where [BO BQ] = [16 14]*
  - *The effect of BQ on resources is negligible and it is not shown.*
  - *For proper comparison, the DSP48E1s blocks were not used.*



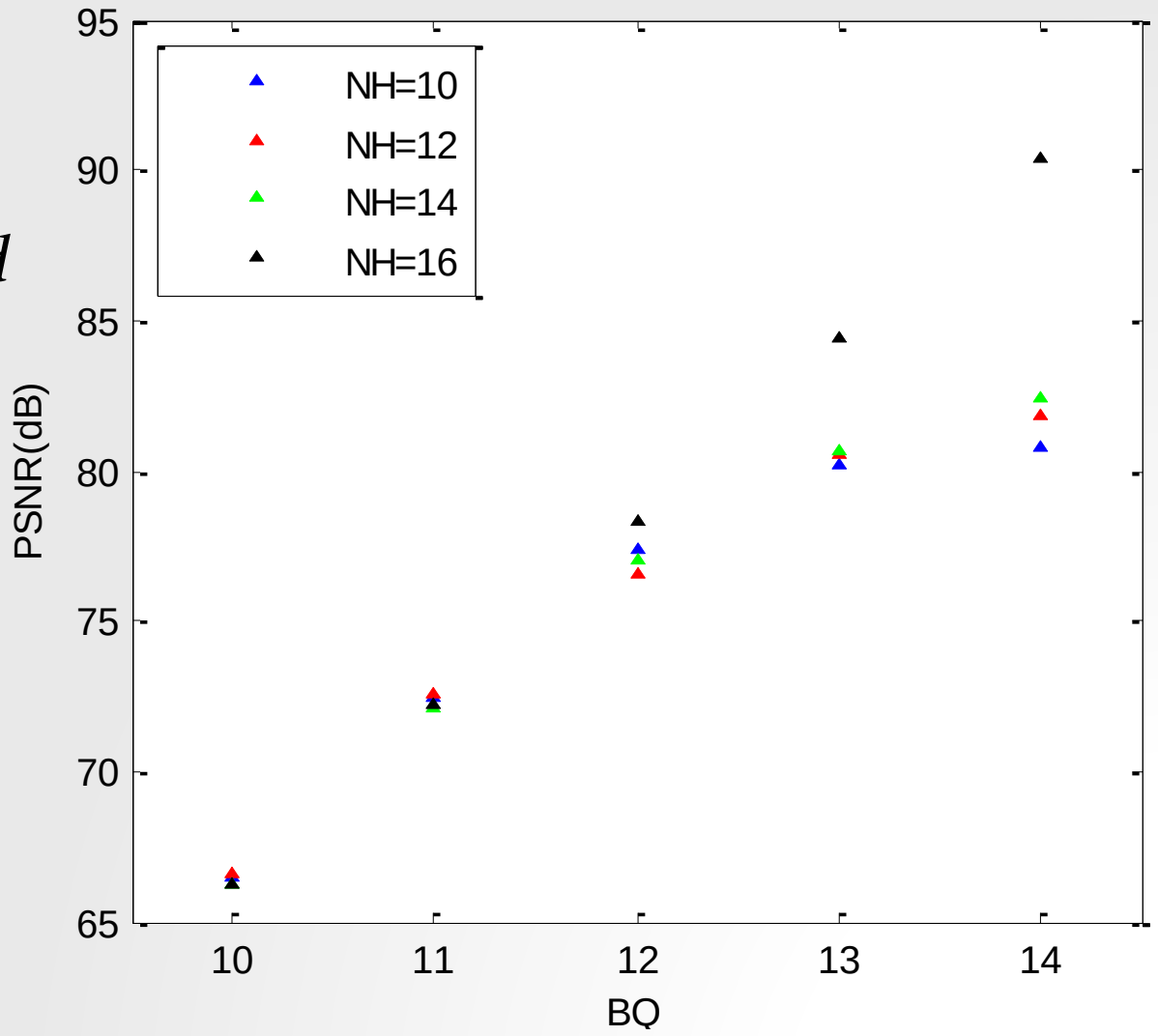
- ***Execution Time***
  - *For BO=16, N=7, L=6, the I/O delay is given by:*

$$RL\_SYS = RL\_IIR + RL\_AVG + RL\_FIR = 36 \text{ cycles}$$

  - *To compute NS samples per channel, we need 36+NS cycles.*
  - *For 100 MHz, the execution time is:* $(36 + NS) \times 10^{-8} seconds$
  - $Throughput = \frac{10^8}{1+{}^{36}/_{NS}}$ *samples per channel/second*

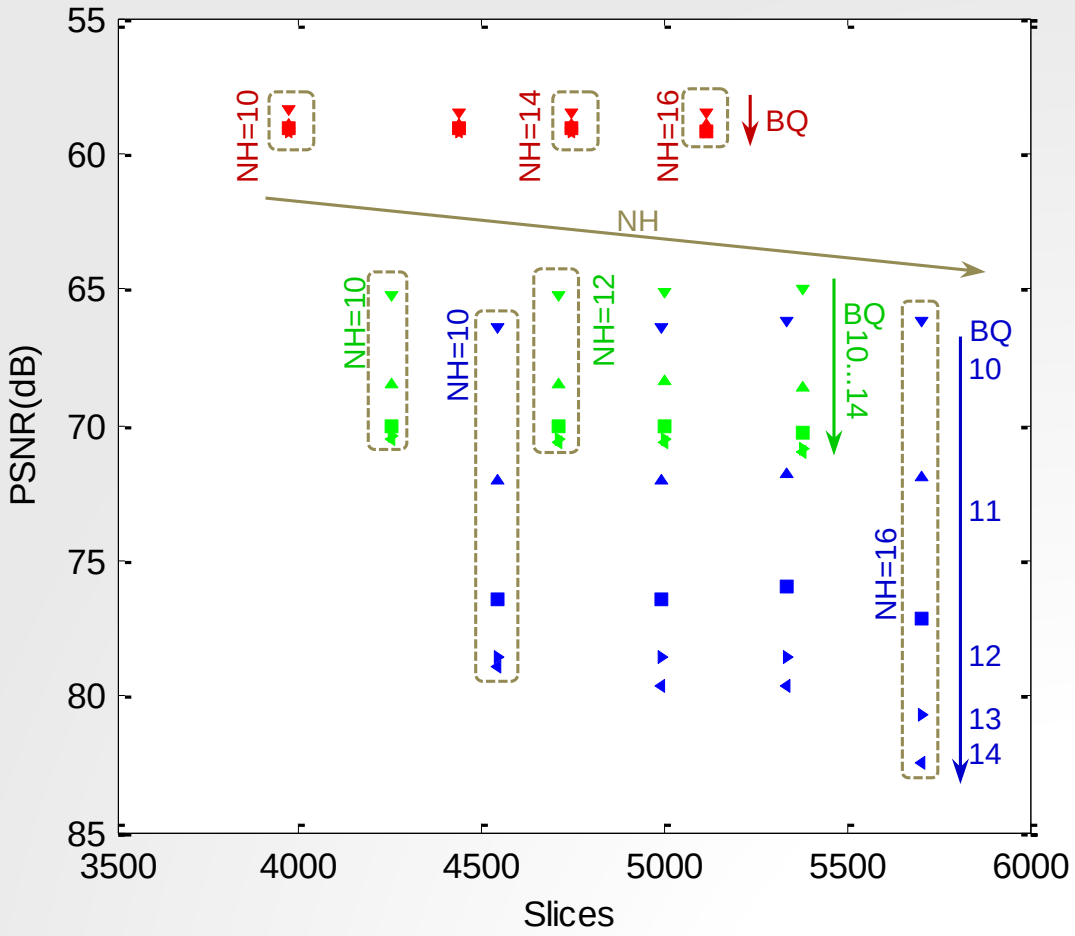
Results

- ***Accuracy (PSNR)***
  - *Test 1: FPGA and software implementations use the quantized input samples.*
  - *Results only shown for B=12, as the effect of input bit-width (B) is negligible. NH and BQ have the strongest effect on accuracy.*



Results

- ***Accuracy (PSNR)***
  - *Test 2: Only FPGA uses the quantized input samples.*
  - *Accuracy-resource trade-offs are indicated.*
  - *Accuracy is heavily affected by B and BQ. NH has some effect.*
  - *Resources are affected by B and NH.*



Results

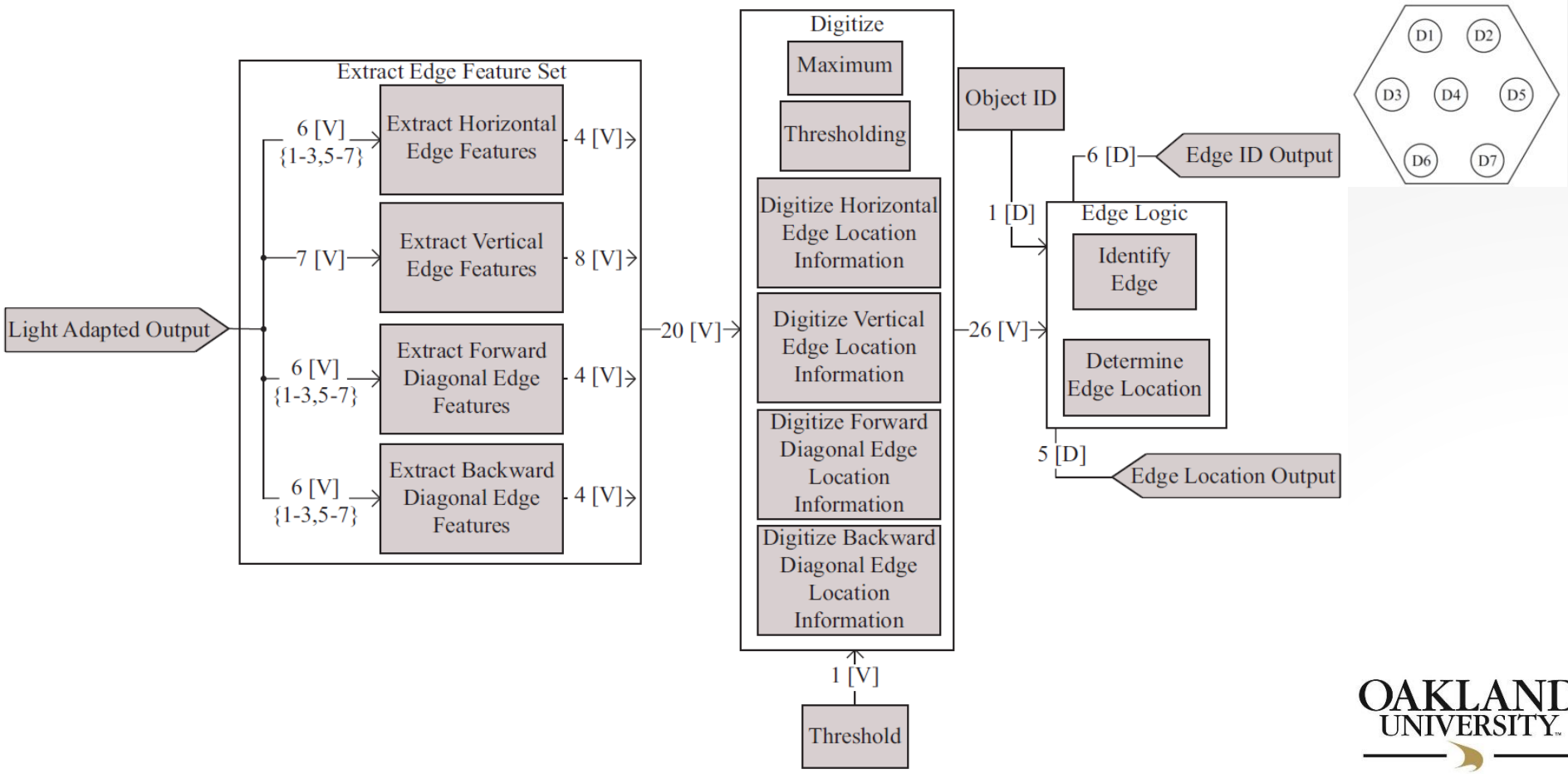
- ***Application: Edge Detection***
  - *Edge feature extraction: Gradients are computed specific to the orientation*

$$\text{HorizontalDifference} = \left|\frac{D1 + D2}{2} - \frac{D6 + D7}{2}\right|$$

$$\text{VerticalDifference} = \left|\frac{D1 + D3 + D6}{2} - \frac{D2 + D5 + D7}{2}\right|$$

$$\text{ForwardSlashDifference} = \left|\frac{D1 + D3}{2} - \frac{D5 + D7}{2}\right|$$

$$\text{BackwardSlashDifference} = \left|\frac{D2 + D5}{2} - \frac{D3 + D6}{2}\right|$$



Conclusions

- *A scalable and fully-pipelined fixed-point architecture was implemented and successfully validated.*
- *This works demonstrates the feasibility of incorporating digital hardware into the design of largely analog compound vision sensors.*
- *The next step is to implement a system with several sensors on an FPGA that can adapt resources at run-time based on user-generated or automatic constraints.*
- *Current work consist on implementing selected image processing algorithms based on the outputs LAOx (x=1..7) generated by the system.*