Design of Embedded Hardware and Firmware

Edge Detection

Andrea Guerrieri

HES-SO//Genève

andrea.guerrieri@hesge.ch

Problem specifications

We need real-time edge-detection on a live video stream

Problem specifications

How to approach the problem??

Problem specifications

- Real-time?
- Edge-detection?
- Video stream?

Real-Time

Definition:

"A system where the response is time-predictable in respect with a determined deadline"

Edge-Detection

Definition:

“An image filter, with the aim to extract boundaries of objects in images”

Video Stream

Definition:

"A video flow, where the data are continuously delivered from the source to the sink"

Problem specifications

Implementation??

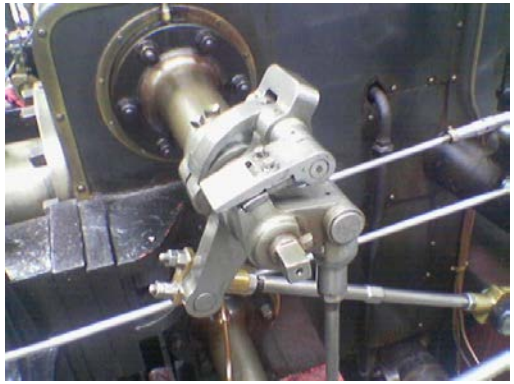
Solution Space

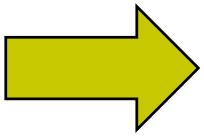
We need to add some constraints!

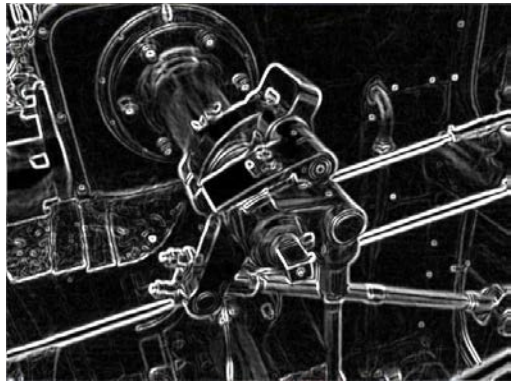
Problem specifications

Edge detection

We need to adapt to our problem







Edge detection

1. RGB to Grayscale

2. Sobel filter

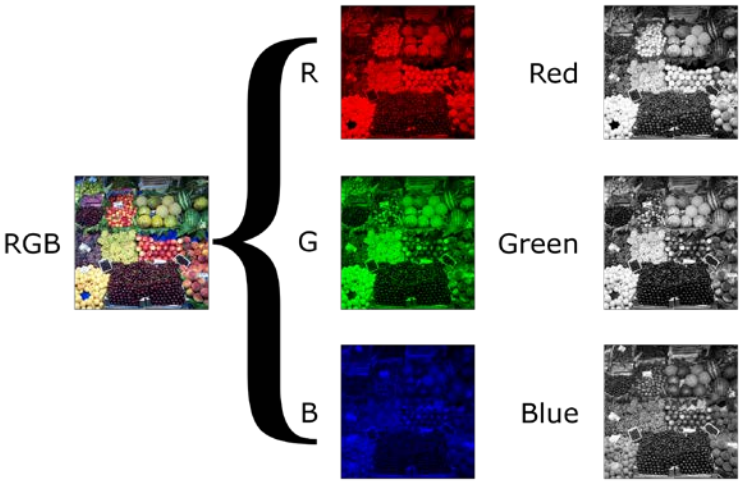
3. Threshold

RGB to Grayscale

RGB

Greyscale

Each pixel is composed by RED GREEN BLUE samples



Each pixel is a single sample representing only the intensity of light

How to transform it?

RGB to Grayscale

$$grayscale = \frac{Red + Green + Blue}{3}$$

RGB to Grayscale

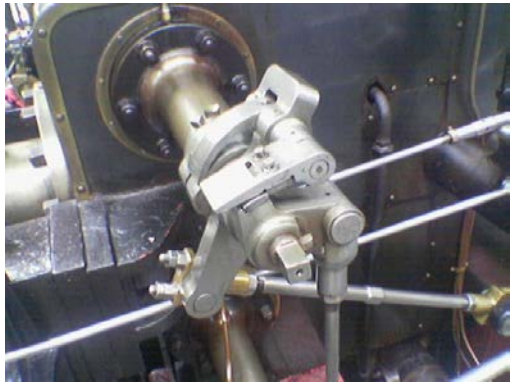
$$grayscale = \frac{Red + Green + Blue}{3}$$

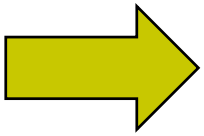
Well…is a bit more complex than this

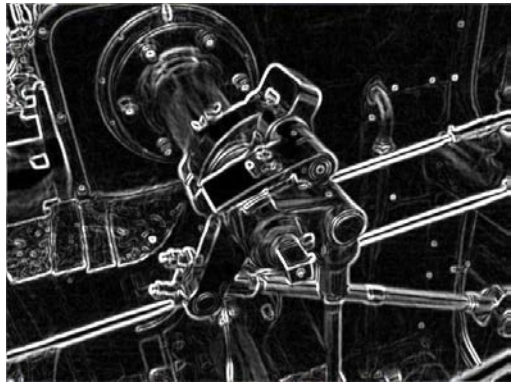
Depending on the standard (ITU-R BT.709/ITU-R BT.2100),the luma component has different coefficient. In our case ITU−R Rec.601

$$grayscale = 0.3 \cdot Red + 0.59 \cdot Green + 0.11 \cdot Blue$$

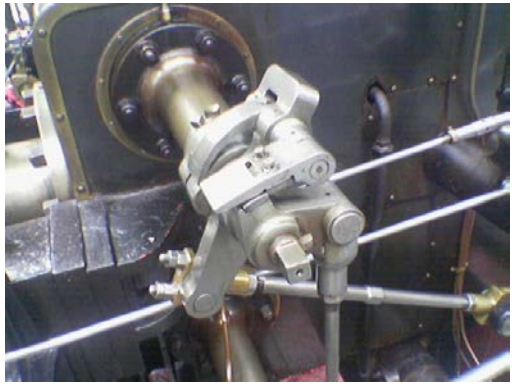
Sobel filter

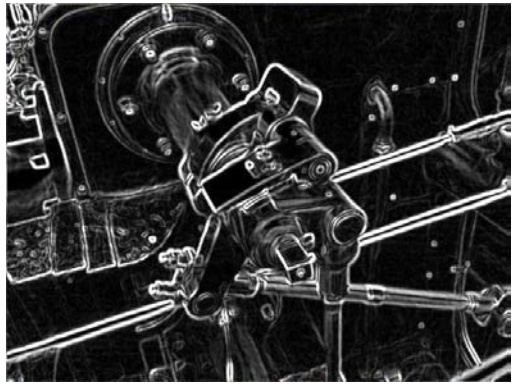






Sobel filter





Computing the gradient! For each pixel of the image, the result of the Sobel–Feldman operator represent the gradient vector or the norm of this vector.

Sobel filter

The gradient of an image is a vector of its partial derivatives

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix}$$

$$where\ \frac{\partial f}{\partial x}\ is\ the\ gradient\ in\ x$$

$$where\ \frac{\partial f}{\partial x}\ is\ the\ gradient\ in\ y$$

The gradient direction

$$\theta = \tan^{-1} \begin{bmatrix} G_x \\ G_y \end{bmatrix}$$

The gradient magnitude

$$M = \sqrt{G_x^2 + G_y{}^2}$$

Sobel filter

The derivative can be approximated using the finite differences. Using the central difference, the computation is obtained with the convolution of the image for 1-dimensional filter

$$\frac{\partial f}{\partial x} = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * I$$

Sobel filter

The Sobel filter uses two 3x3 kernels to calculate the approximations of the derivatives, one for the horizontal and one for the vertical

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & +1 \\ 0 & 0 & 0 \\ +1 & 2 & +1 \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = G_x = S_x * I \qquad \frac{\partial f}{\partial x} = G_y = S_y * I$$

Sobel filter

The Sobel filter uses two 3x3 kernels to calculate the approximations of the derivatives, one for the horizontal and one for the vertical

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & +1 \\ 0 & 0 & 0 \\ +1 & 2 & +1 \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = G_x = S_x * I \qquad \frac{\partial f}{\partial x} = G_y = S_y * I$$

*** Convolution??**

Sobel filter

Convolution is a mathematical operation which output is the shape relation between the functions arguments

Definition

$$f * g(t) \triangleq \int_{-\infty}^{+\infty} f(\tau) \cdot g\,(t-\tau) d\tau$$

Sobel filter

Convolution is a mathematical operation which output is the shape relation between the functions arguments

Definition

$$f * g(t) \triangleq \int_{-\infty}^{+\infty} f(\tau) \cdot g\,(t-\tau) d\tau$$

Discrete approximation

$$f * g[n] \triangleq \sum_{m=-\infty}^{m=+\infty} f[m] \cdot g[n-m]$$

Sobel filter

The gradient computation in discrete domain (Sobel)

$$\frac{\partial f}{\partial x} = G_x \rightarrow I[i,j] = \sum_{m=0}^{m=+2} \sum_{n=0}^{n=+2} I[n,m] \cdot S_x[n,m]$$

$$\frac{\partial f}{\partial y} = G_y \rightarrow I[i,j] = \sum_{m=0}^{m=+2} \sum_{n=0}^{n=+2} I[n,m] \cdot S_y[n,m]$$

Threshold

Apply a high-pass filter to the output results

$$\mathrm{I} = \begin{cases} 0, & I < Threshold \\ I & I \geq Threshold \end{cases}$$

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Sobel filter

-1	0	1
-2	0	2
-1	0	1

Sobel X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X								X
X								X
X								X
X								X
X								X
X								X
X								X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0							X
X								X
X								X
X								X
X								X
X								X
X								X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

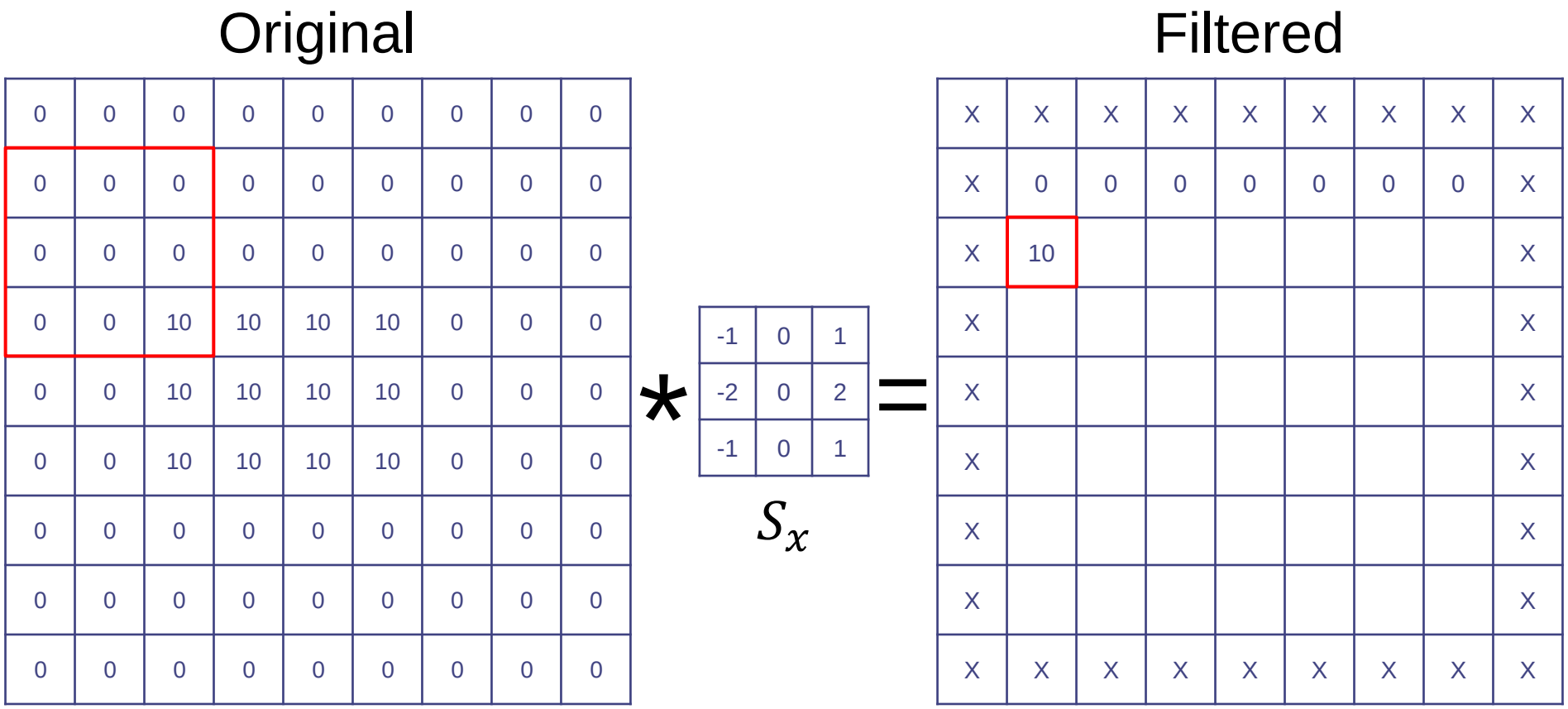
S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0						X
X								X
X								X
X								X
X								X
X								X
X								X
X	X	X	X	X	X	X	X	X

Sobel filter



Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	10						X
X								X
X								X
X								X
X								X
X								X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	10	0	0	-10	-10	0	X
X	30							X
X								X
X								X
X								X
X								X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	10	0	0	-10	-10	0	X
X	30	30	0	0	-30	-30	0	X
X	40	40	0	0	-40	-40	0	X
X	30	30	0	0	-30	-30	0	X
X	10	10	0	0	-10	-10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	0	1
-2	0	2
-1	0	1

S_x

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	10	0	0	-10	-10	0	X
X	30	30	0	0	-30	-30	0	X
X	40	40	0	0	-40	-40	0	X
X	30	30	0	0	-30	-30	0	X
X	10	10	0	0	-10	-10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$*$

-1	-2	-1
0	0	0
1	2	1

S_y

$=$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	30	40	40	30	10	0	X
X	10	30	40	40	30	10	0	X
X	0	0	0	0	0	0	0	X
X	-10	-30	-40	-40	-30	-10	0	X
X	-10	-30	-40	-40	-30	-10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

-1	0	1
-2	0	2
-1	0	1

S_x

-1	-2	-1
0	0	0
1	2	1

S_y

Sobel filter

S_x

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	10	0	0	-10	-10	0	X
X	30	30	0	0	-30	-30	0	X
X	40	40	0	0	-40	-40	0	X
X	30	30	0	0	-30	-30	0	X
X	10	10	0	0	-10	-10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

S_y

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	10	30	40	40	30	10	0	X
X	10	30	40	40	30	10	0	X
X	0	0	0	0	0	0	0	X
X	-10	-30	-40	-40	-30	-10	0	X
X	-10	-30	-40	-40	-30	-10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$$\sqrt{S_x^2 + S_y^2}$$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	14	32	40	40	32	14	0	X
X	32	42	40	40	42	32	0	X
X	40	40	0	0	40	40	0	X
X	32	42	40	40	42	32	0	X
X	14	32	40	40	32	14	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$$\sqrt{S_x^2 + S_y^2}$$

$$|S_x| + |S_y|$$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	20	40	40	40	40	20	0	X
X	40	60	40	40	60	40	0	X
X	40	40	0	0	40	40	0	X
X	40	60	40	40	60	40	0	X
X	20	40	40	40	40	20	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$$|S_x| + |S_y|$$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	0	0	0	0	0	0	0	X
X	0	60	0	0	60	0	0	X
X	0	0	0	0	0	0	0	X
X	0	60	0	0	60	0	0	X
X	0	0	0	0	0	0	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Threshold: 50

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$$|S_x| + |S_y|$$

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	0	40	40	40	40	0	0	X
X	40	60	40	40	60	40	0	X
X	40	40	0	0	40	40	0	X
X	40	60	40	40	60	40	0	X
X	0	40	40	40	40	0	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Threshold: 30

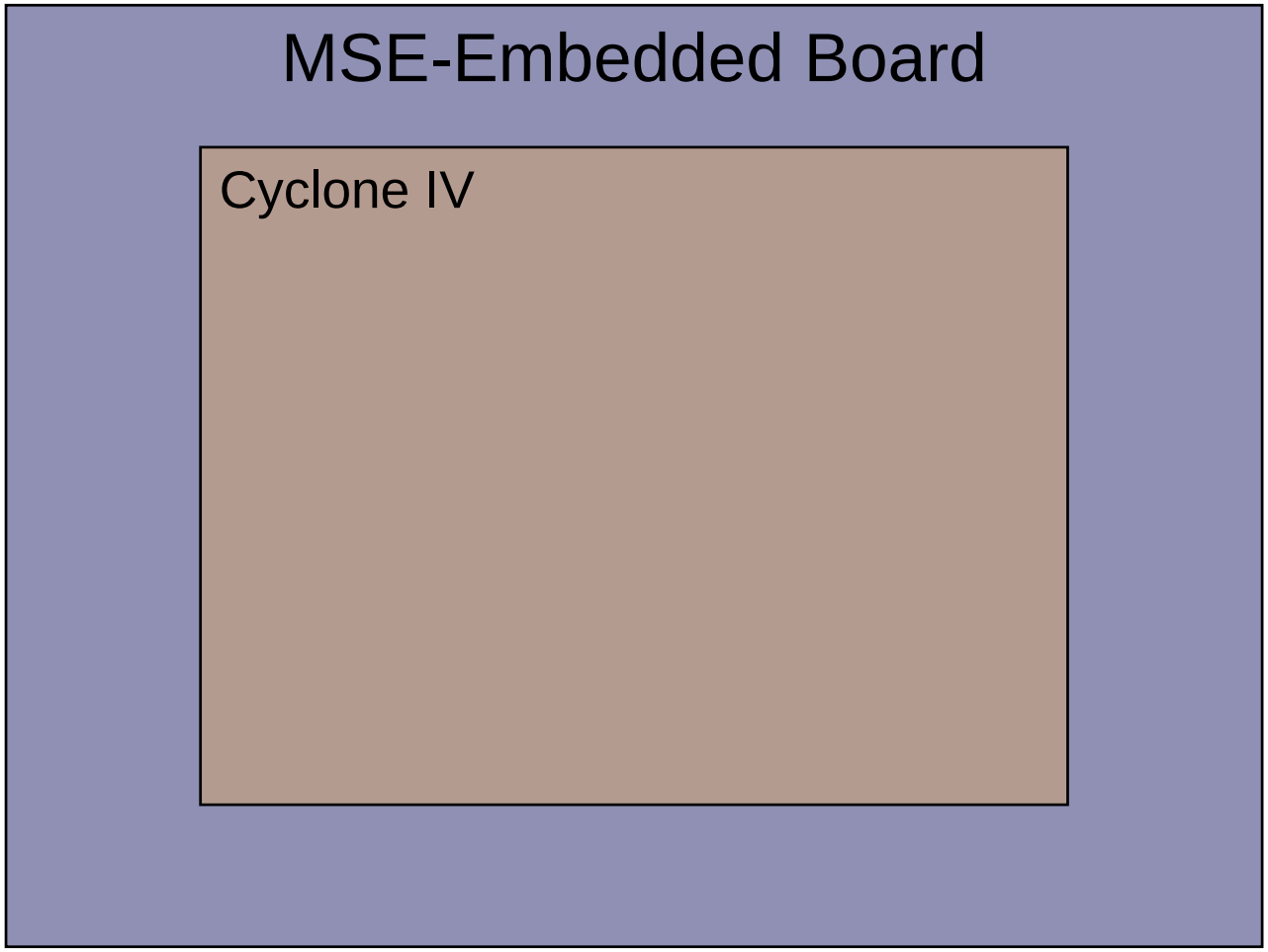
Problem specifications

How put into an embedded target?

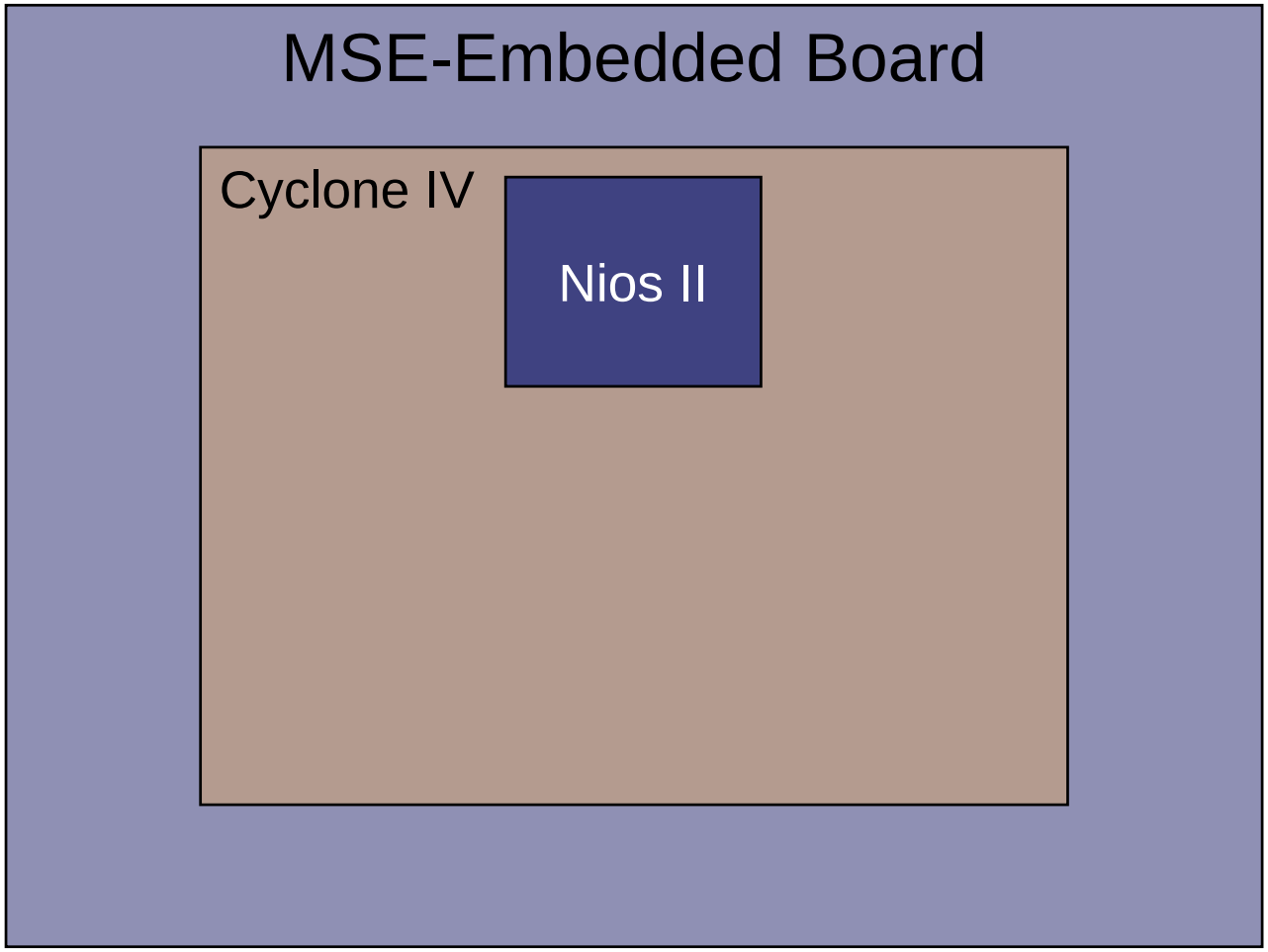
Problem specifications

MSE-Embedded Board

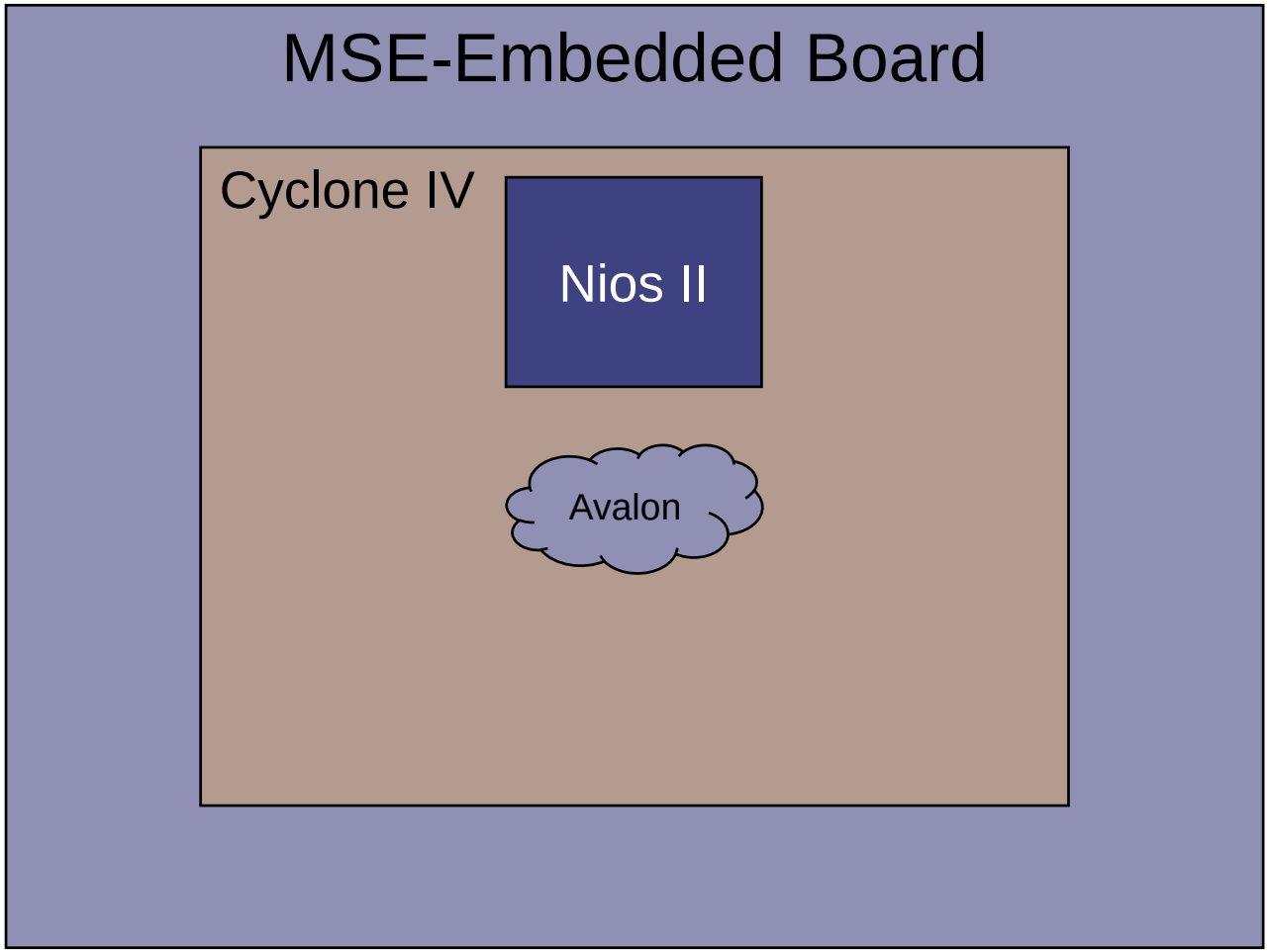
Problem specifications



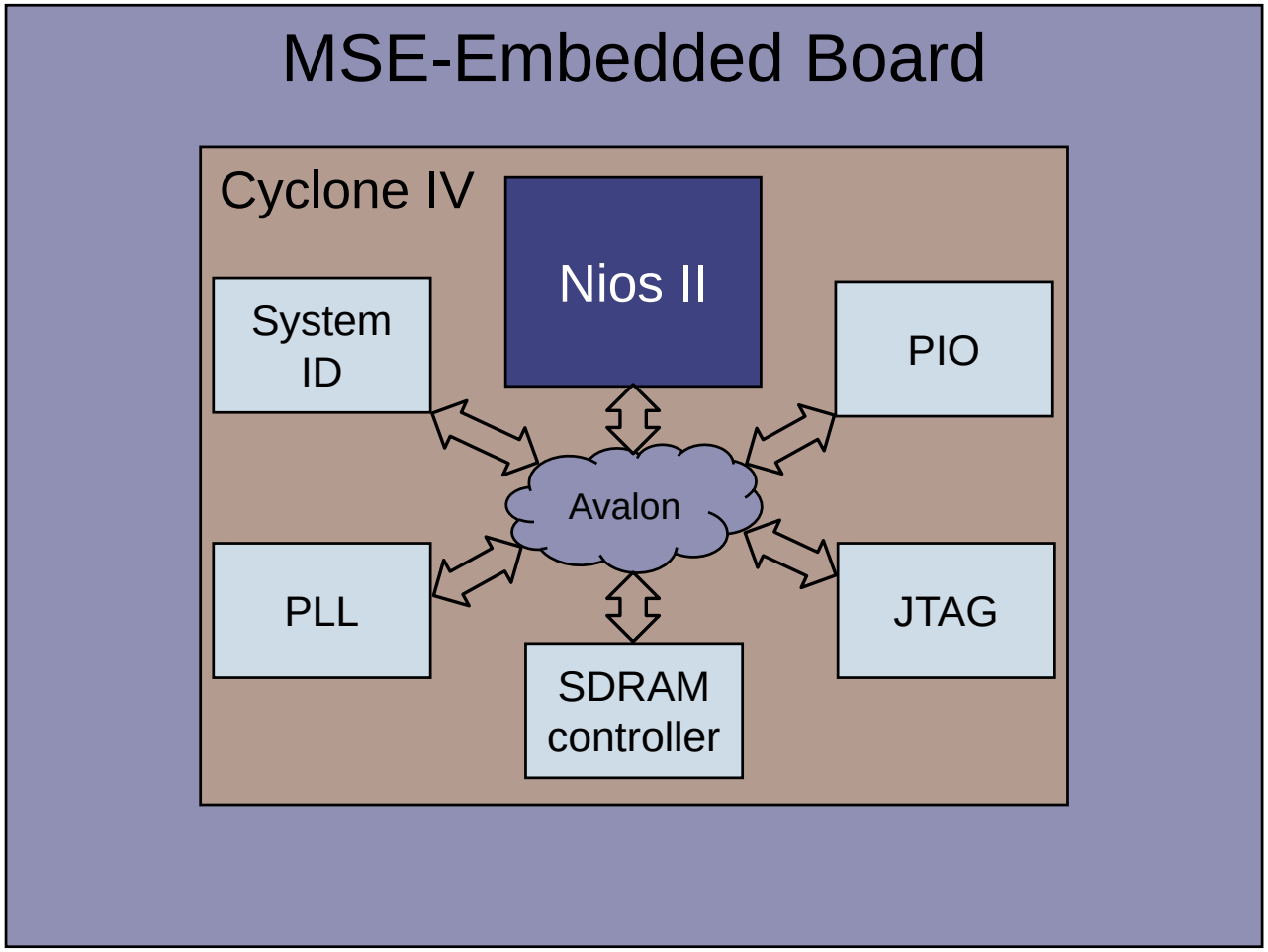
Problem specifications



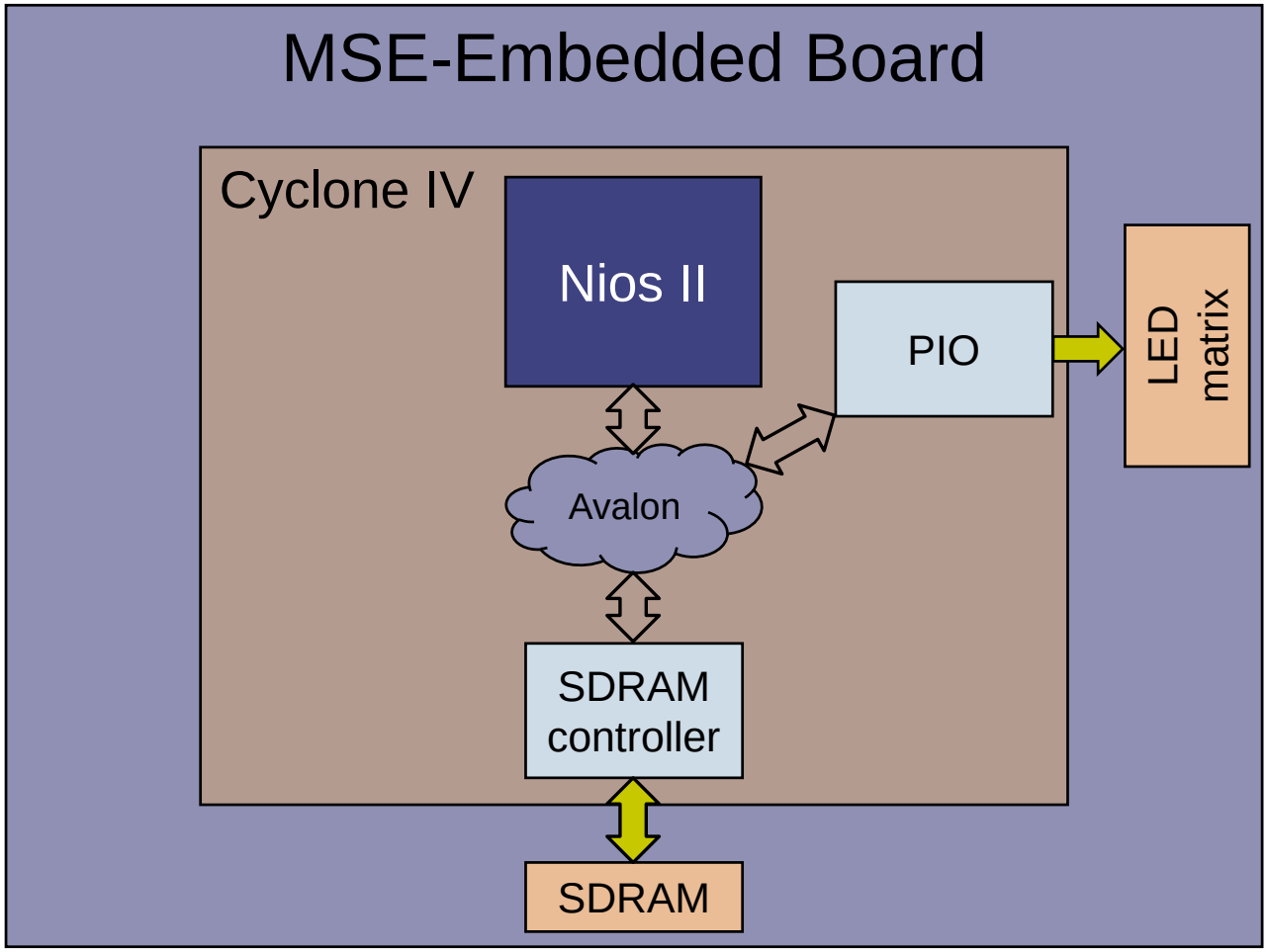
Problem specifications



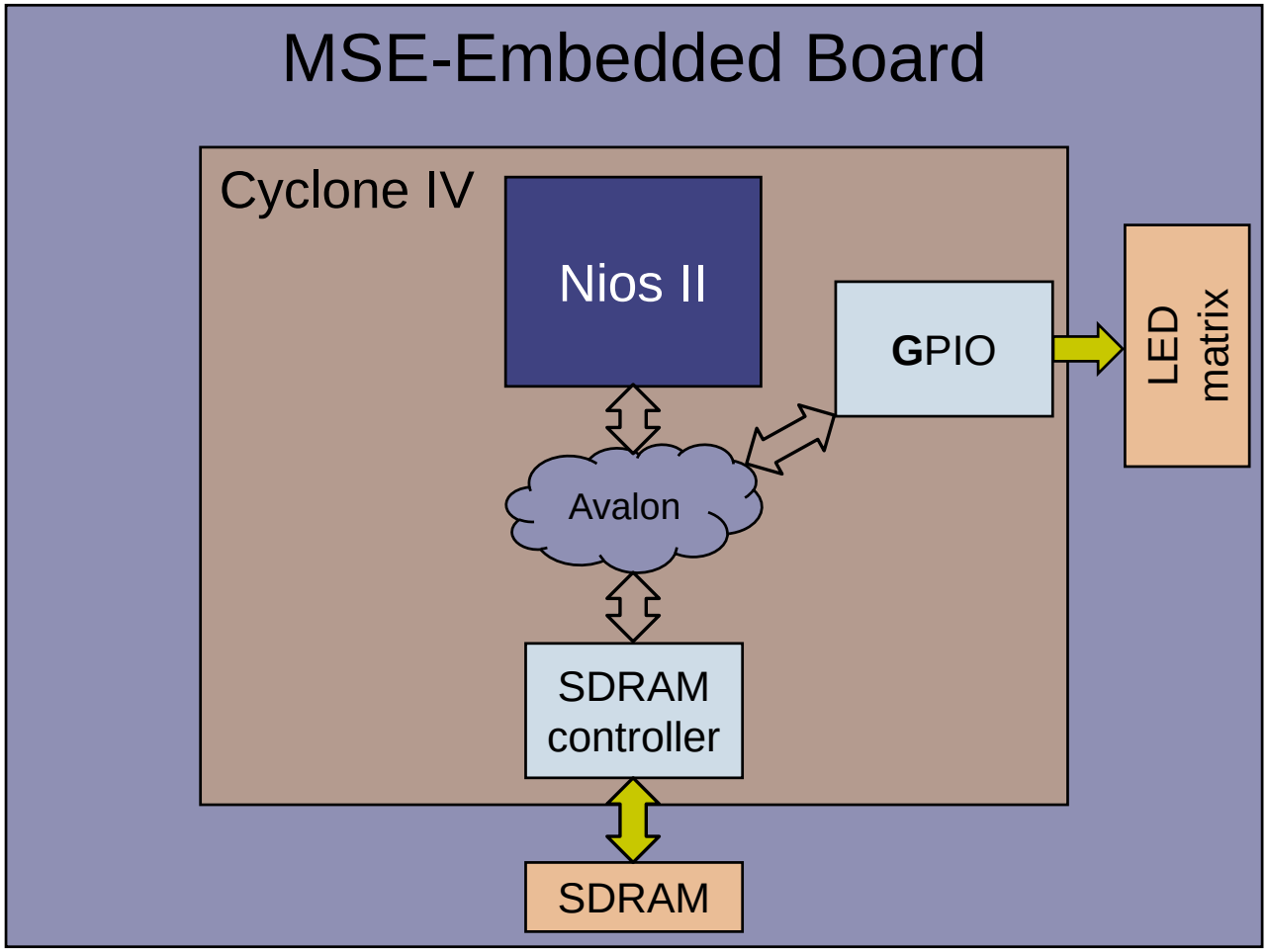
Problem specifications



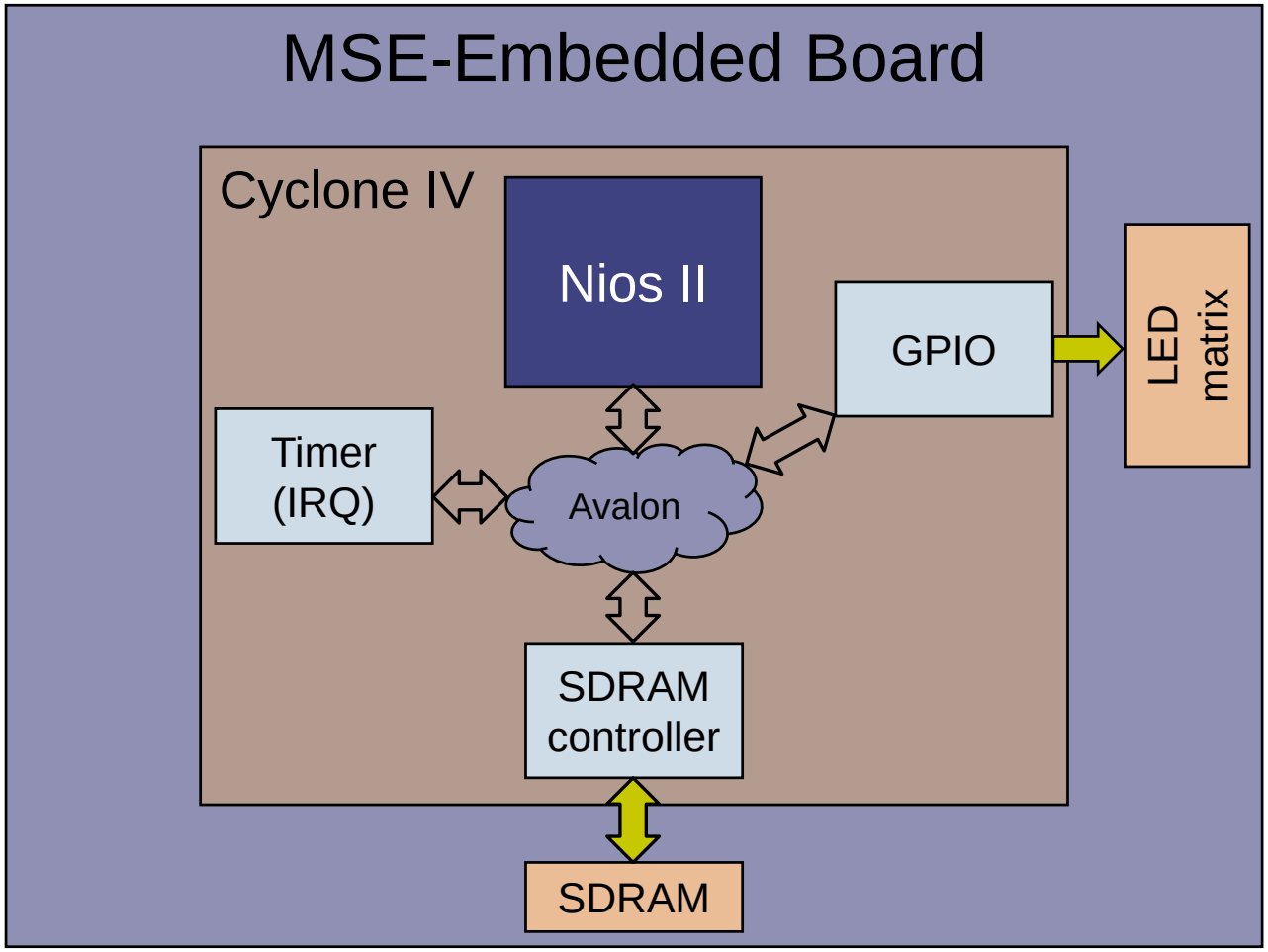
Problem specifications



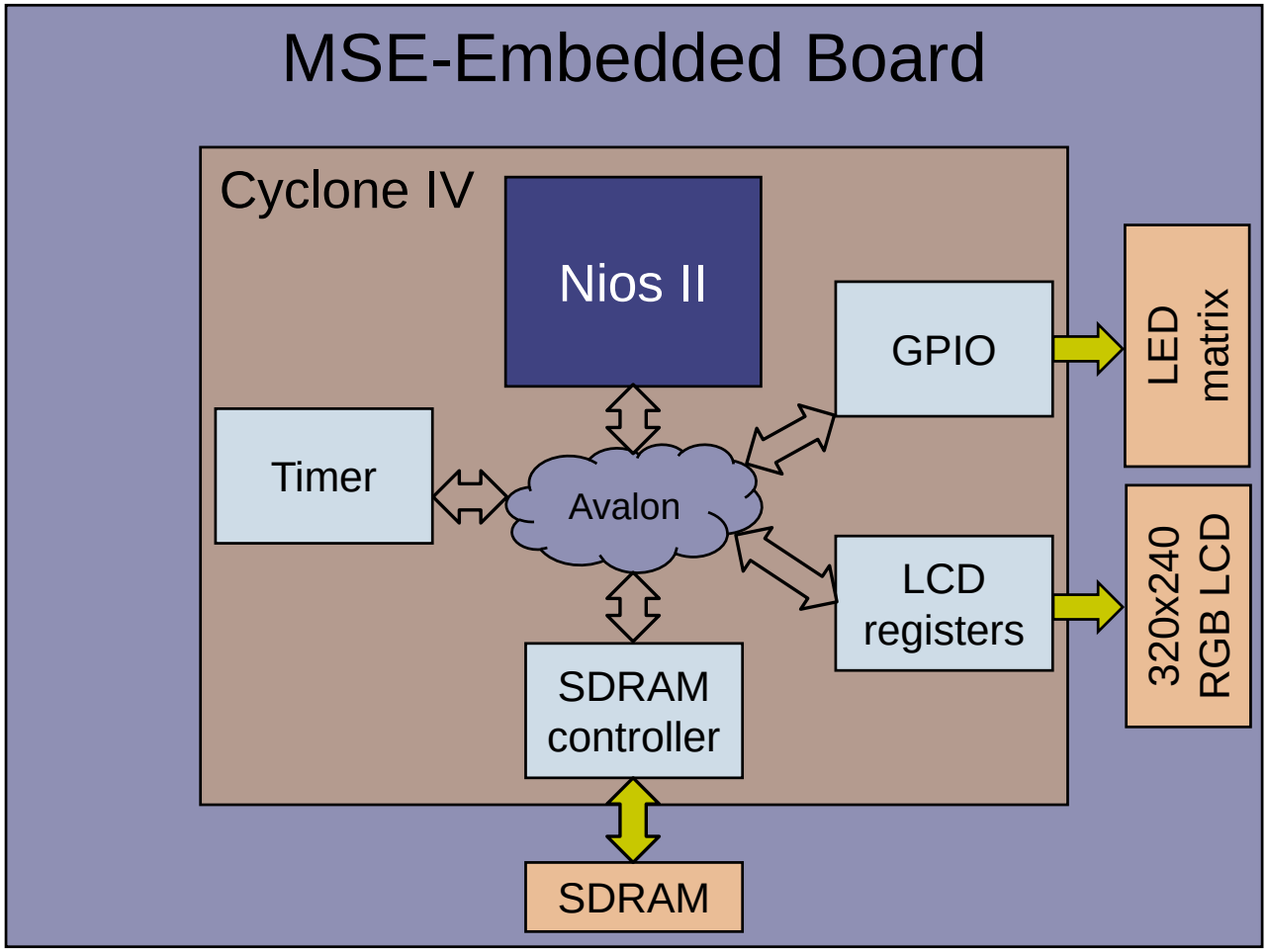
Problem specifications



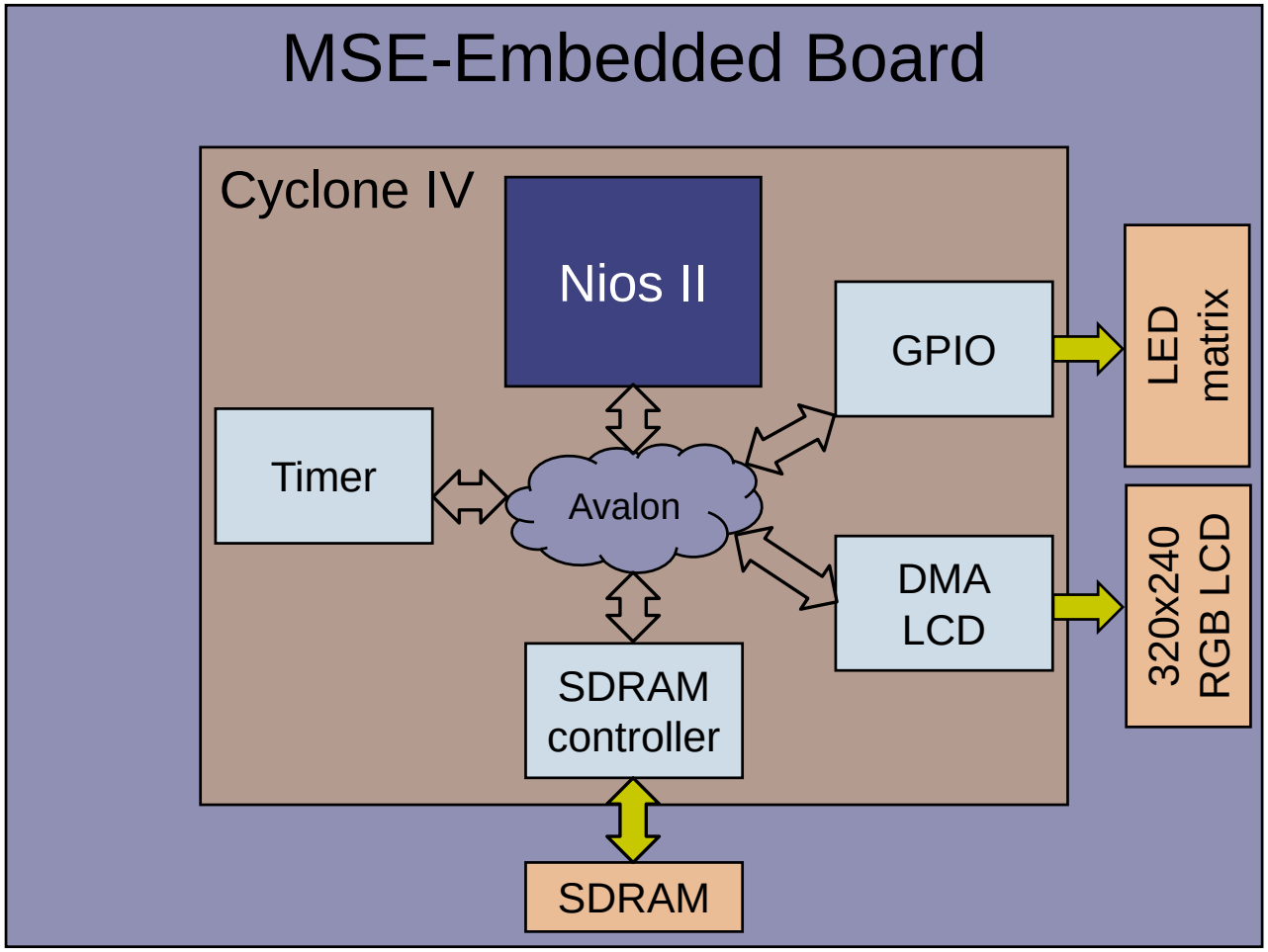
Problem specifications



Problem specifications

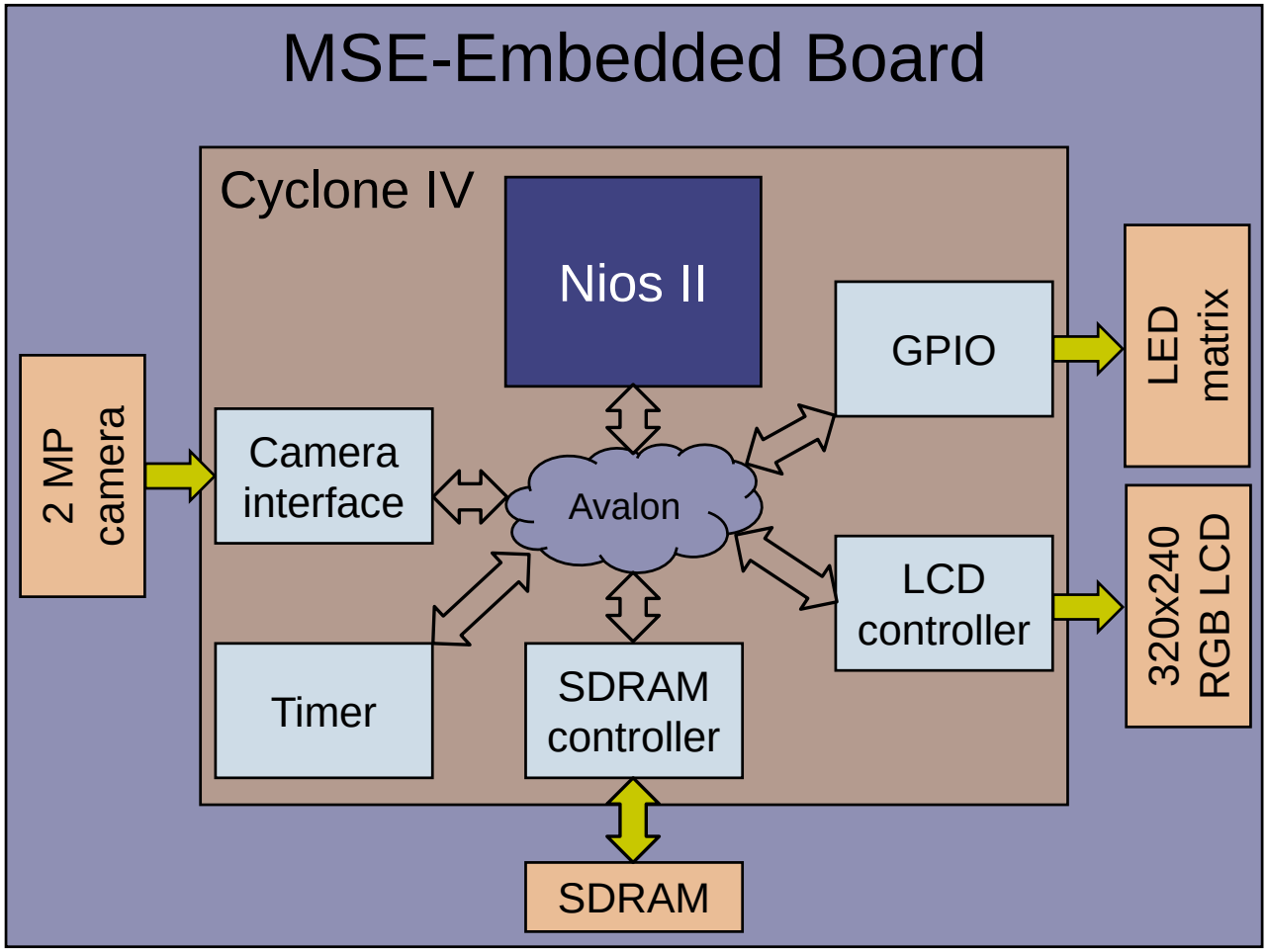


Problem specifications

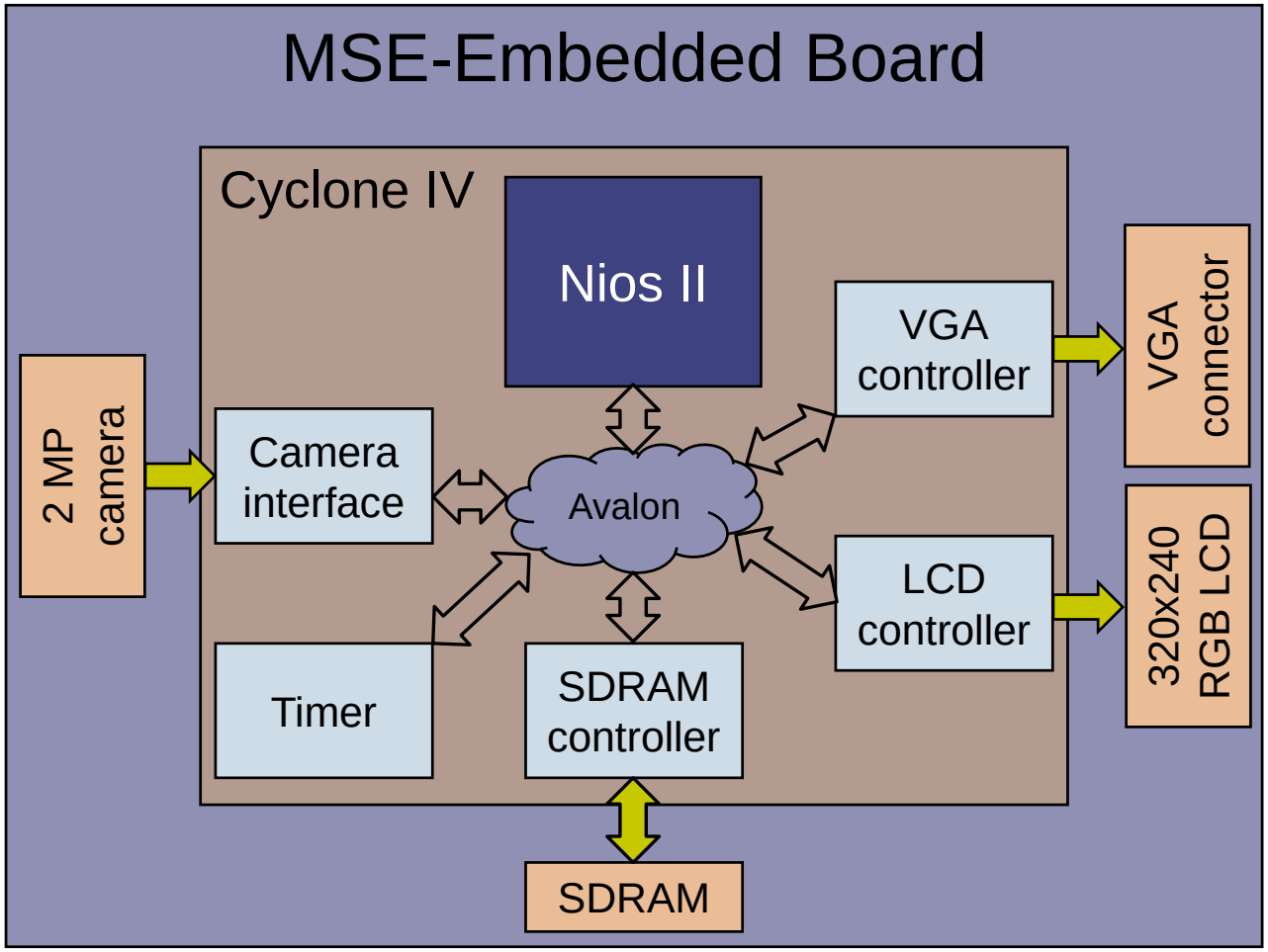


(First part)

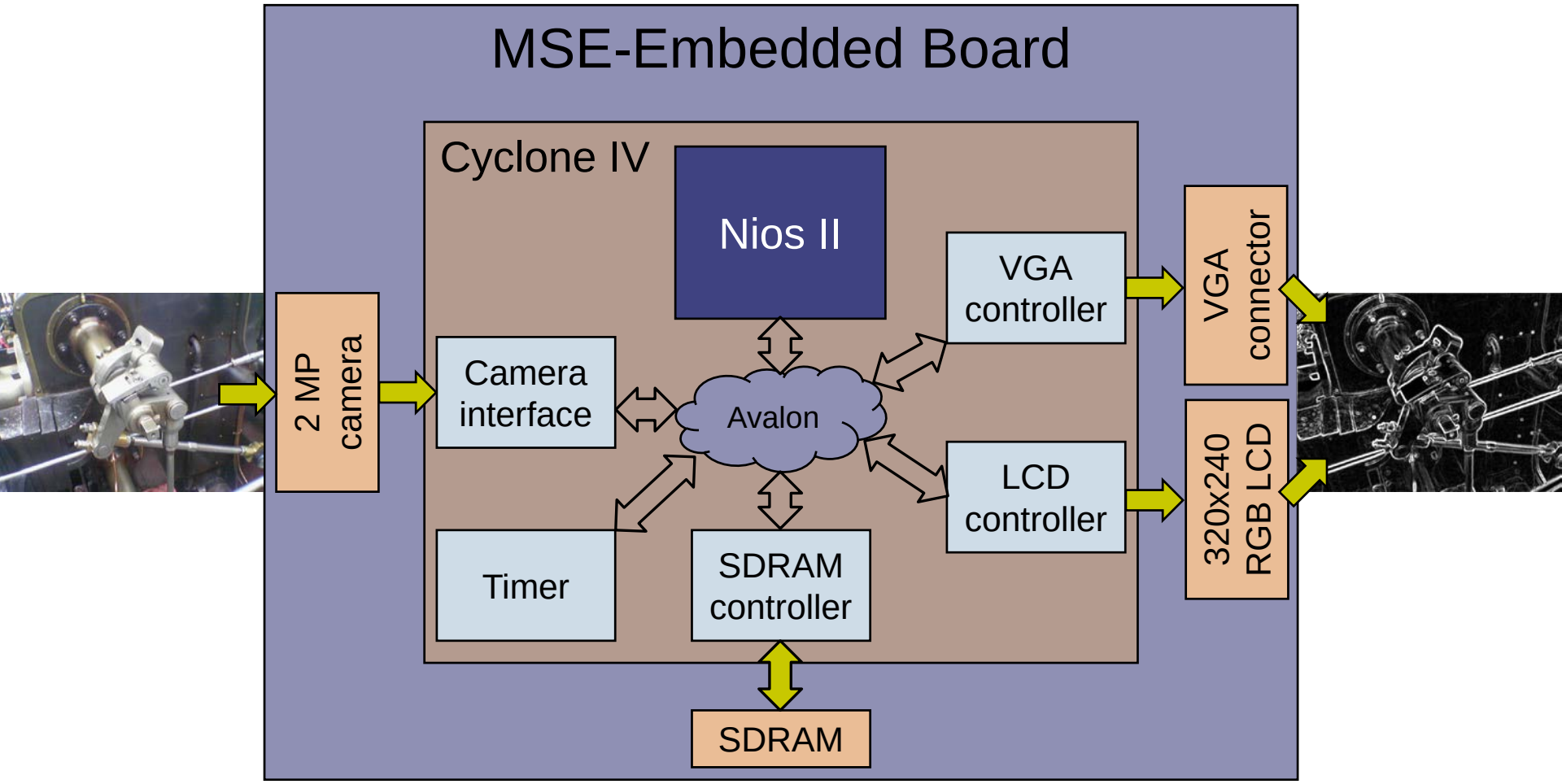
Problem specifications



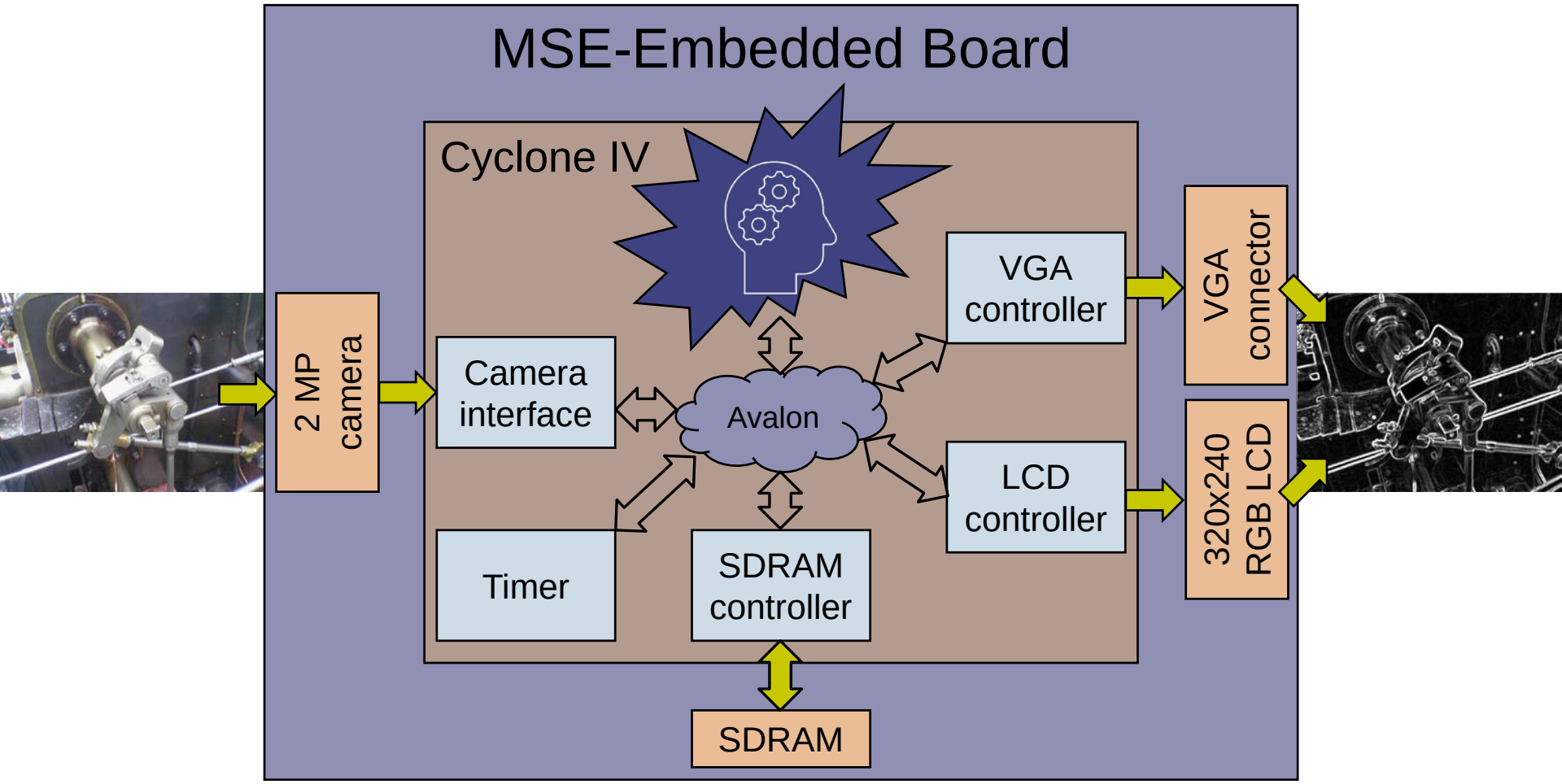
Problem specifications



Problem specifications

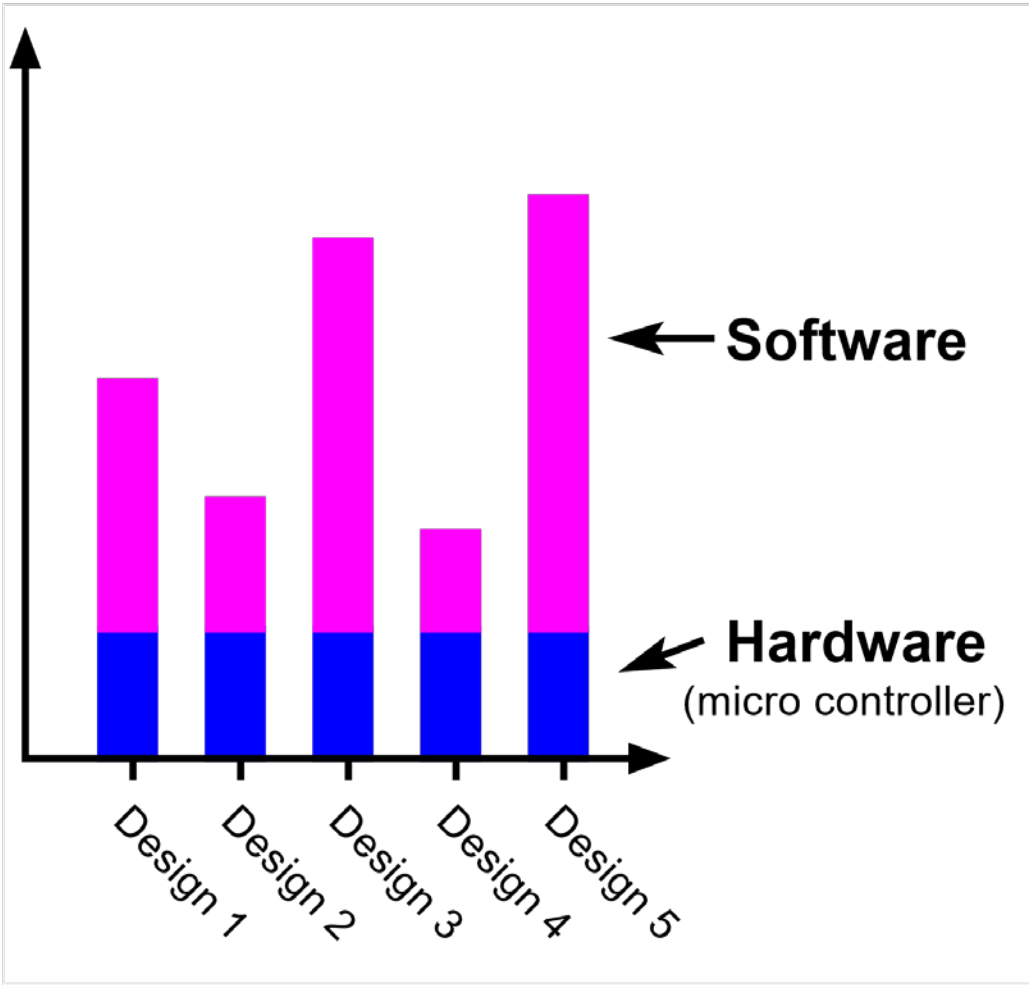


Problem specifications



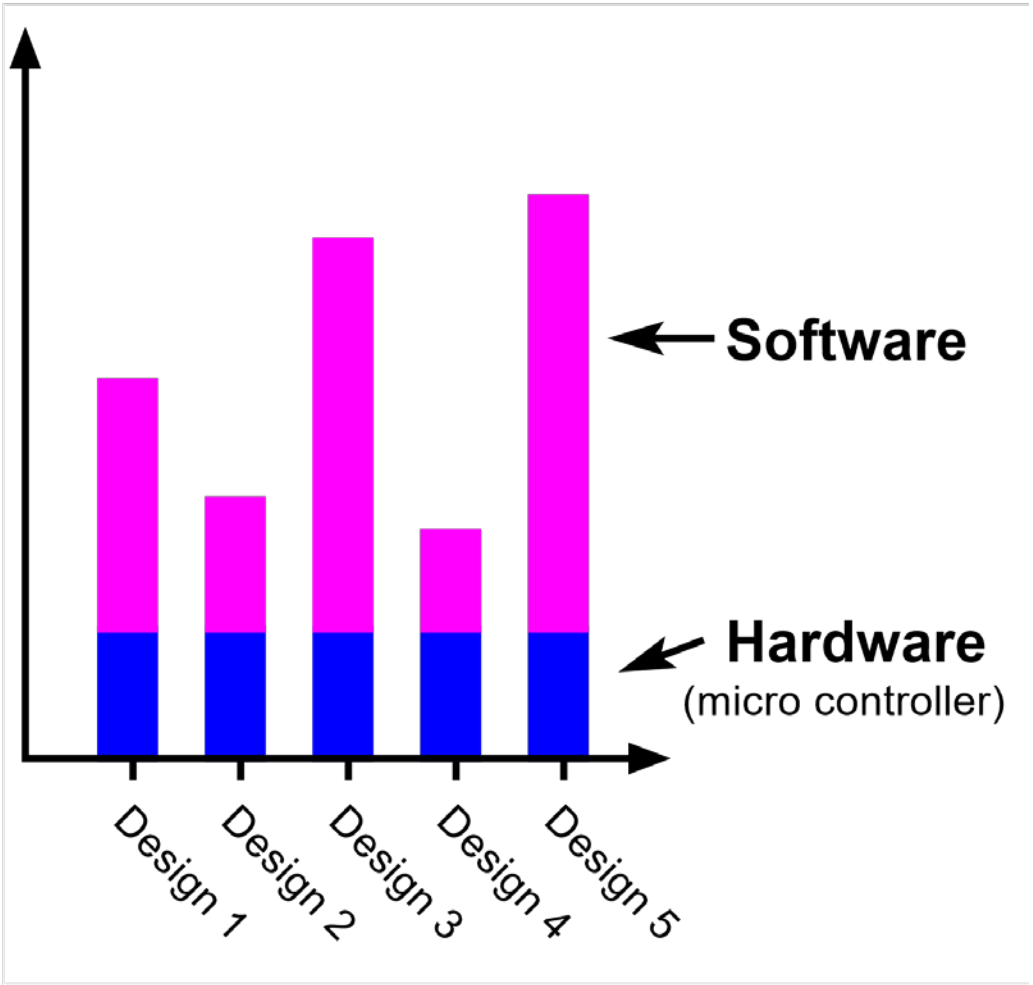
Hardware-software partitioning

- Traditional approach:
  - Constant HW
  - SW adapts



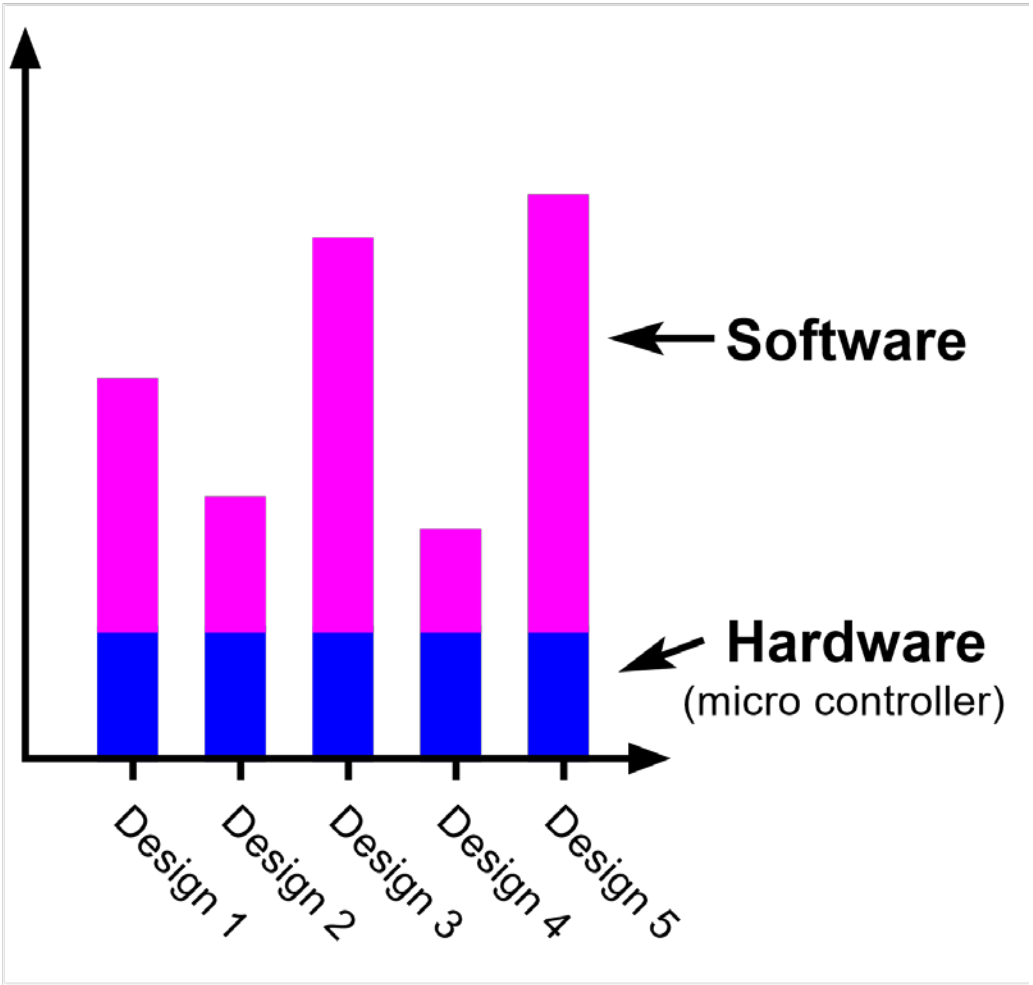
Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts



Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts
    - (Limited by HW)

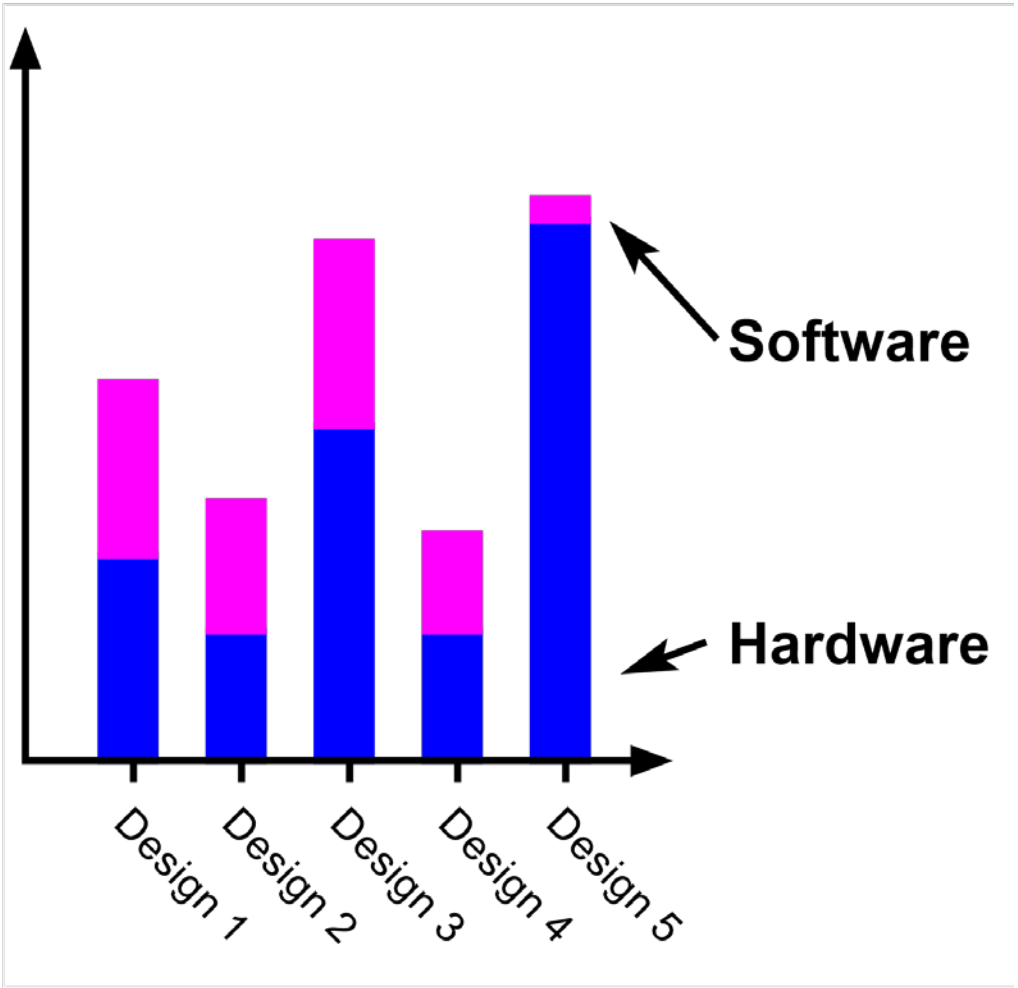


Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts
    - (Limited by HW)

vs.

- Modern approach:
  - HW+SW adapt

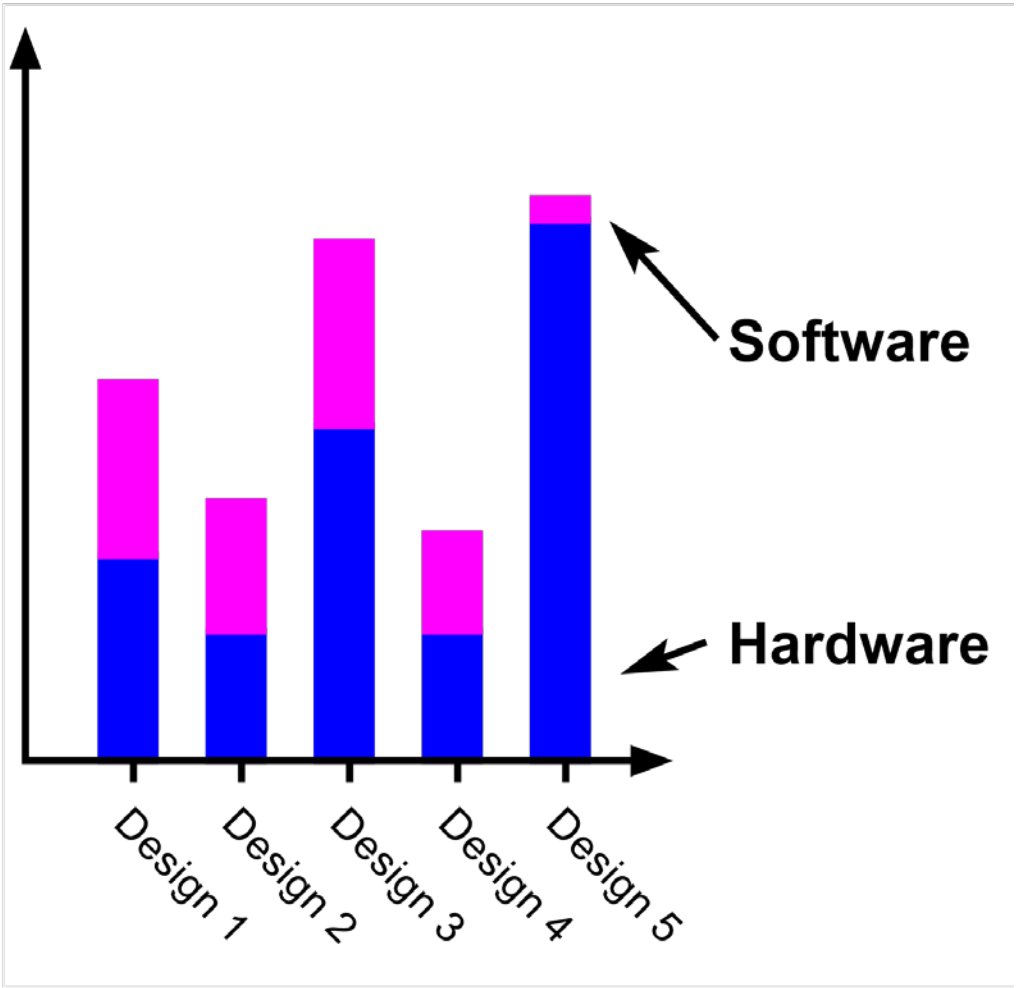


Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts
    - (Limited by HW)

vs.

- Modern approach:
  - HW+SW adapt
    - (Flexible HW: FPGA)

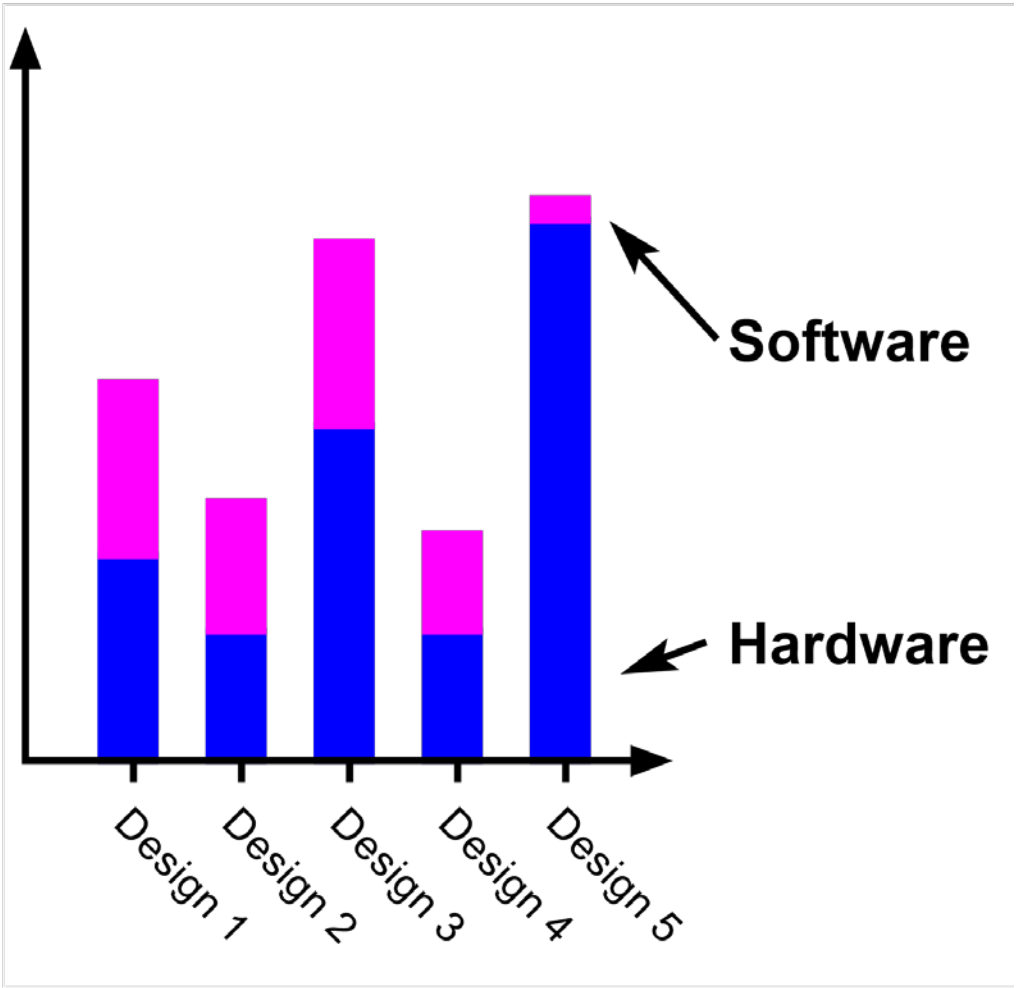


Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts
    - (Limited by HW)

vs.

- Modern approach:
  - HW+SW adapt
    - (Flexible HW: FPGA)
  - HW+SW codesign

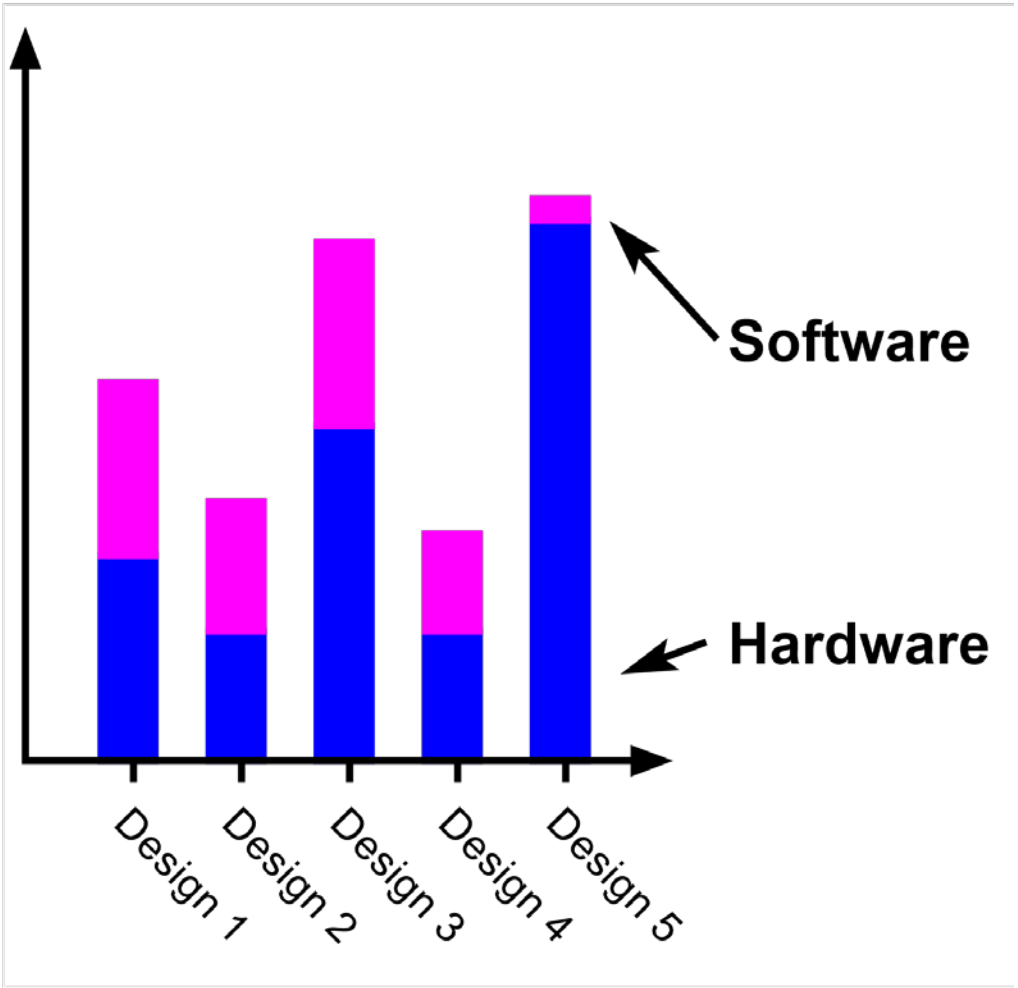


Hardware-software partitioning

- Traditional approach:
  - Constant HW
    - (Moore's law)
  - SW adapts
    - (Limited by HW)

vs.

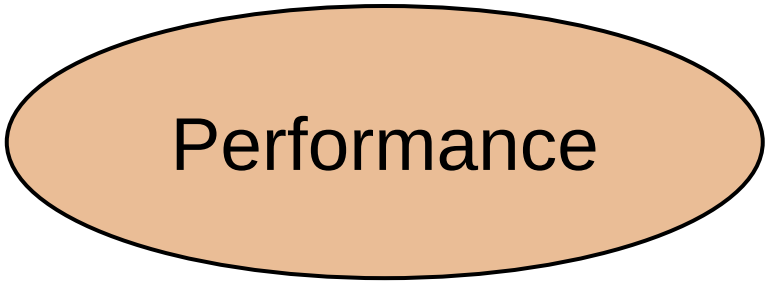
- Modern approach:
  - HW+SW adapt
    - (Flexible HW: FPGA)
  - **HW+SW codesign**
    - (Tasks partitioning)



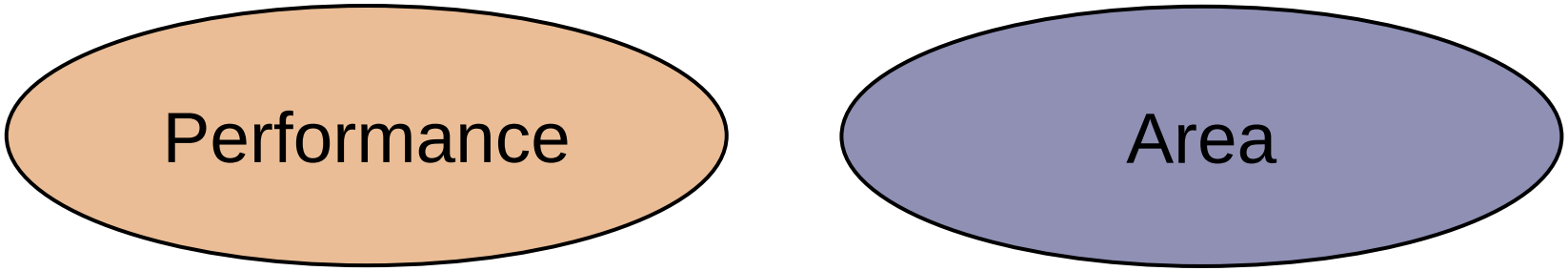
Hardware-software codesign

How to partition the system?

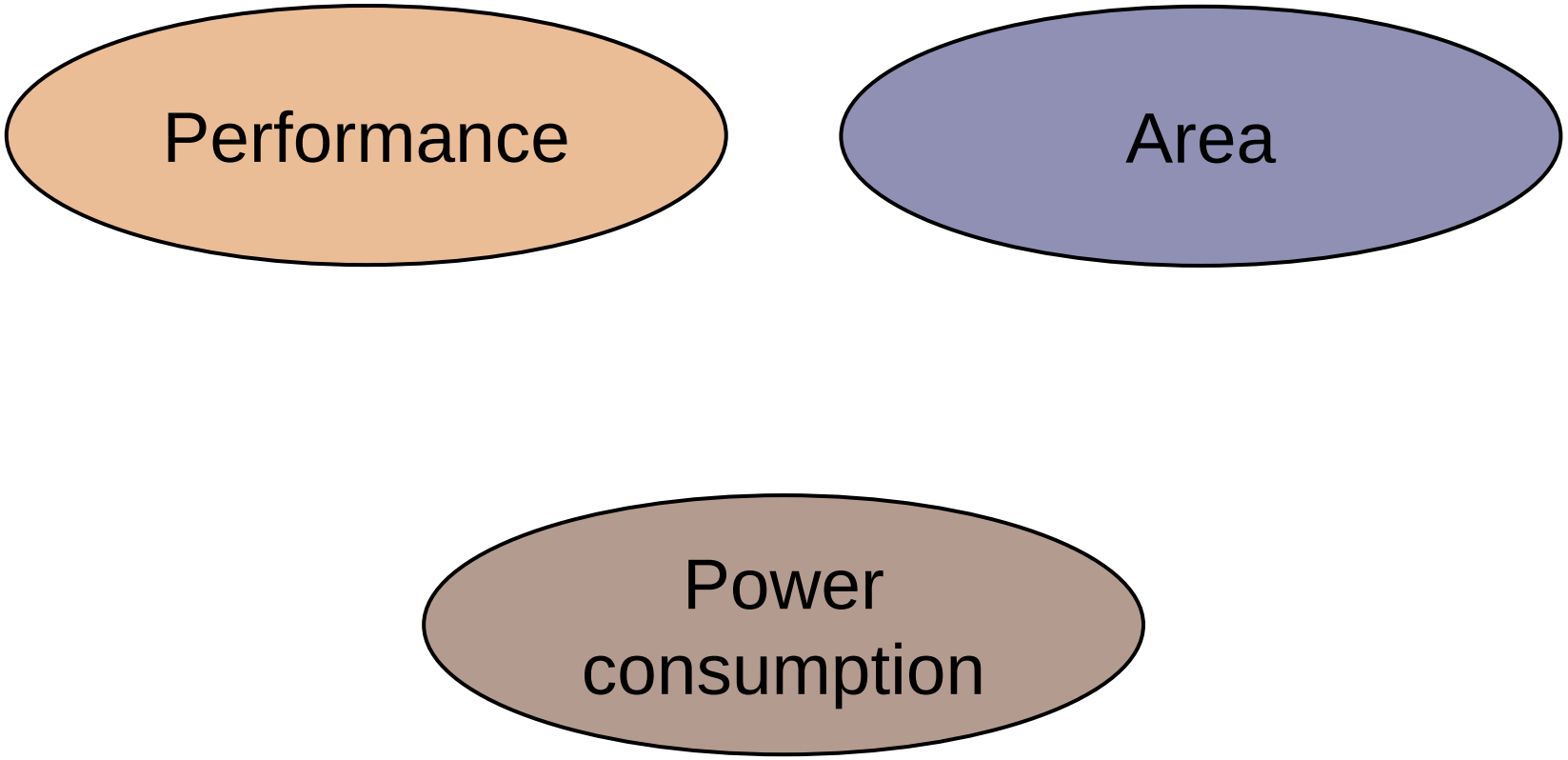
Hardware-software codesign



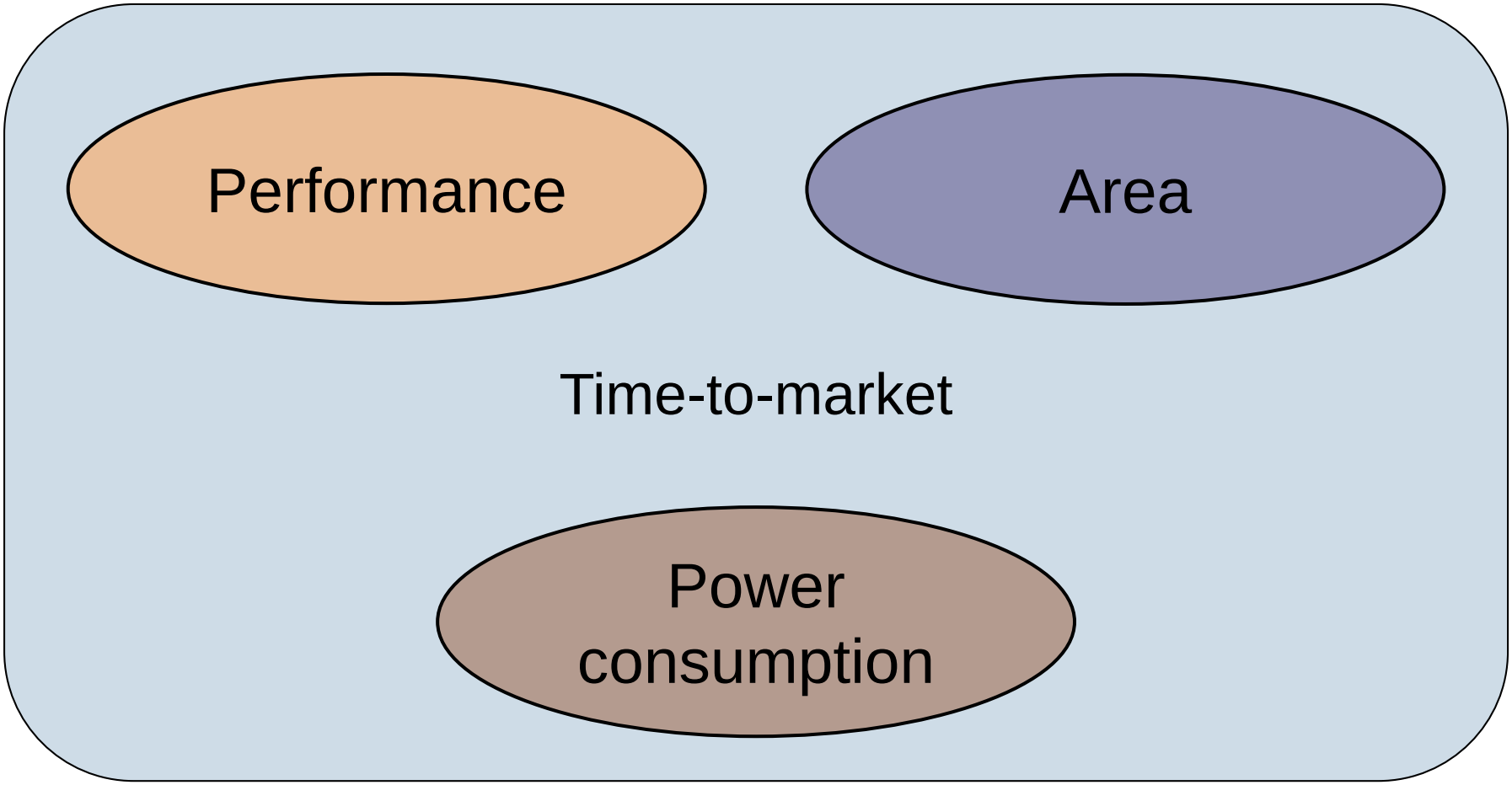
Hardware-software codesign



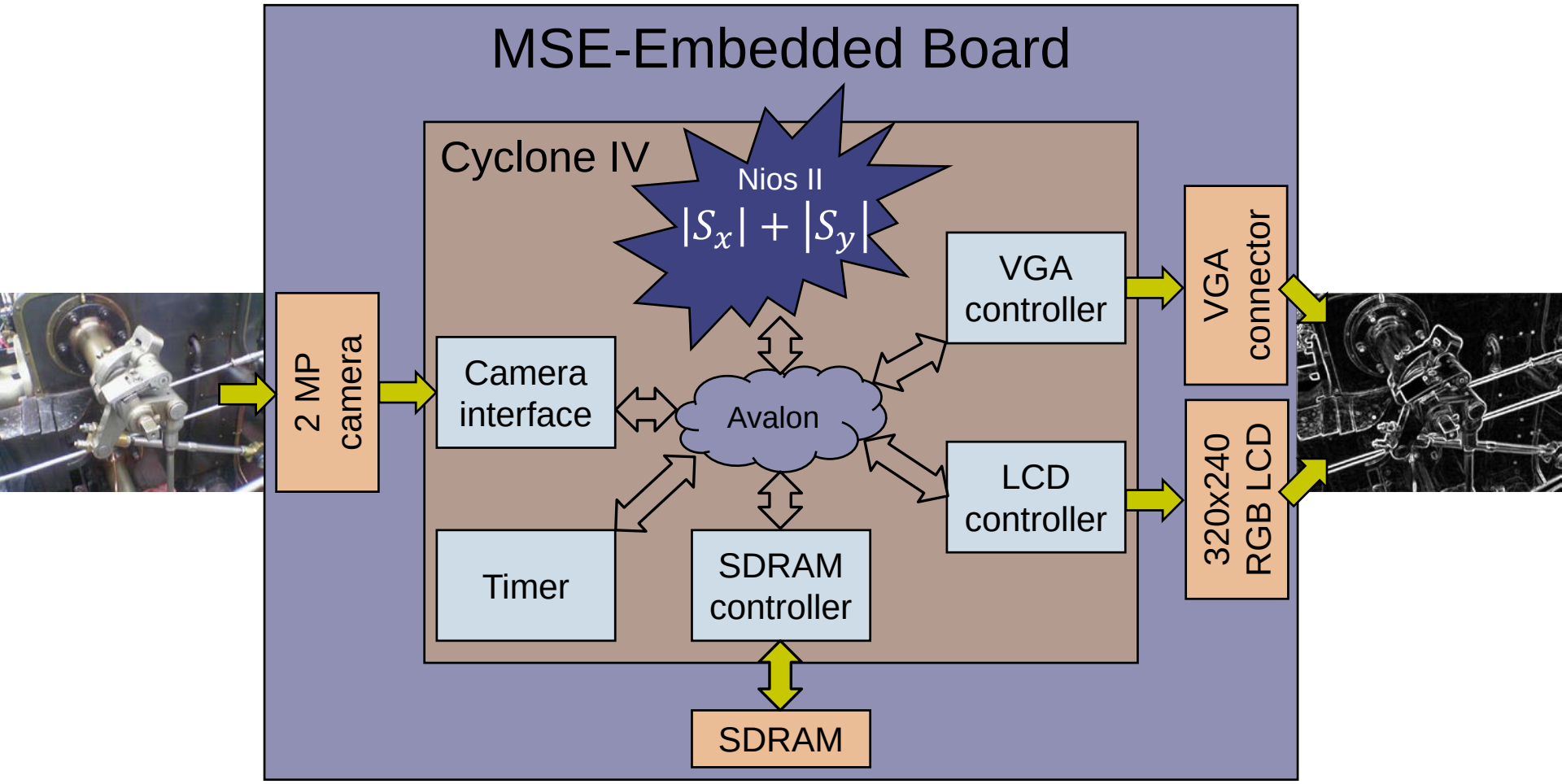
Hardware-software codesign



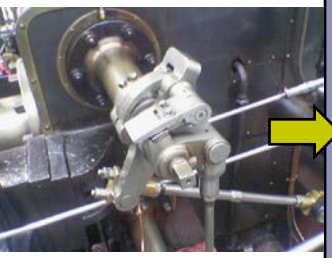
Hardware-software codesign



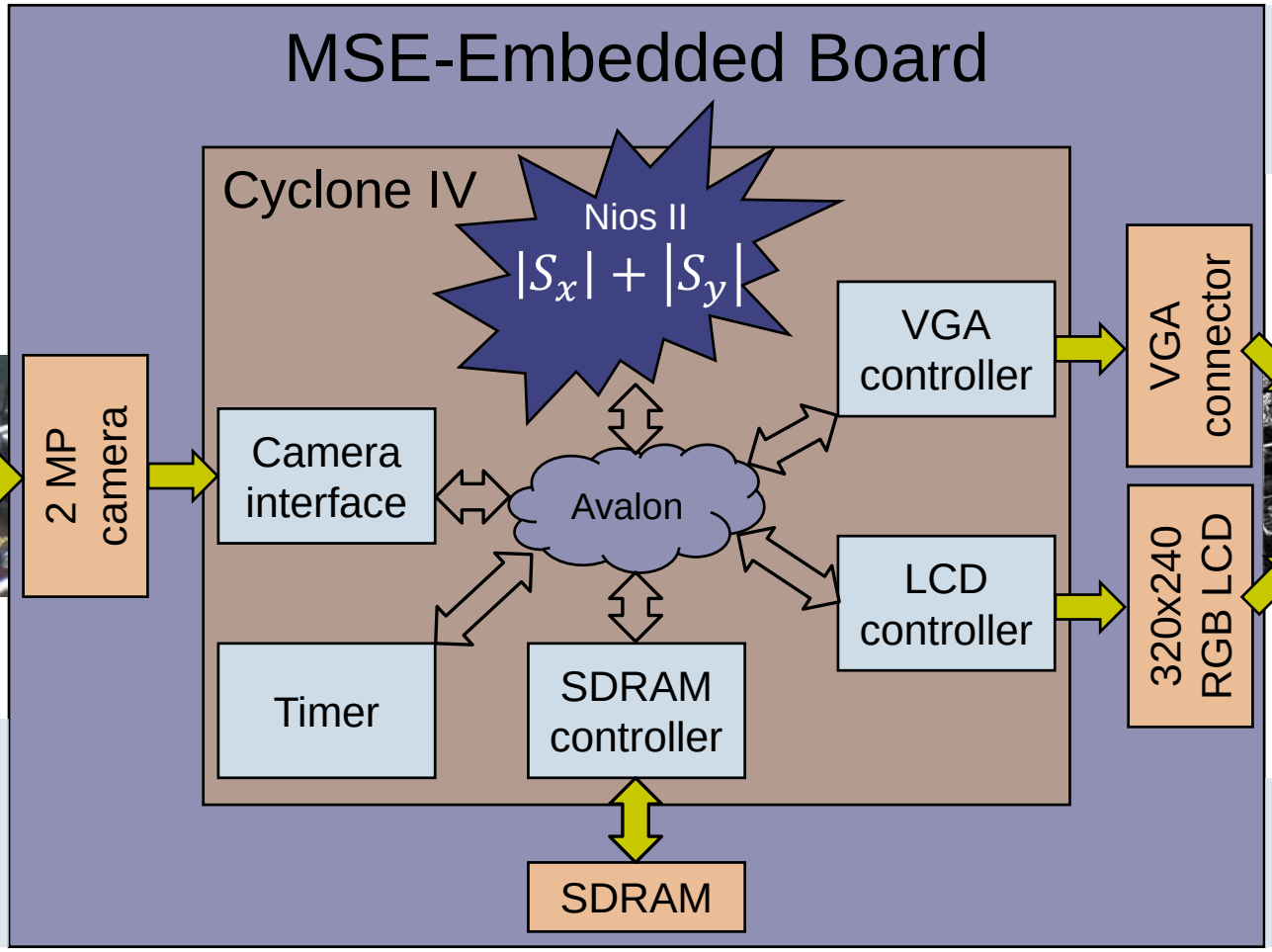
Problem specifications



Problem specifications

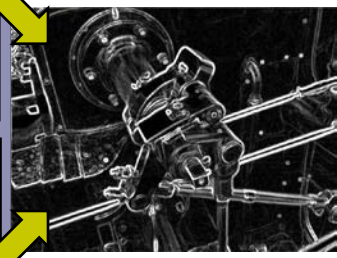


Camera:
- **RGB565**
- **512x384**
- **5-15 fps**



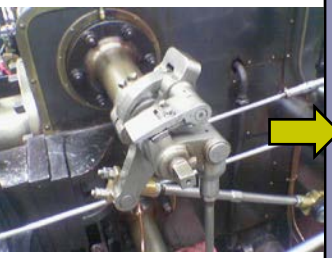
VGA:
- **RGB10-10-10**
- **1024x768**

?? fps



LCD:
- **RGB565**
- **240x320**

Some numbers



2 MP camera

MSE-Embedded Board

Cyclone IV

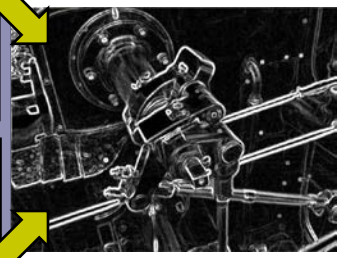
Nios II

$|S_x| + |S_y|$

- 512 pixels x 384 lines
- 15 fps
- 50 MHz

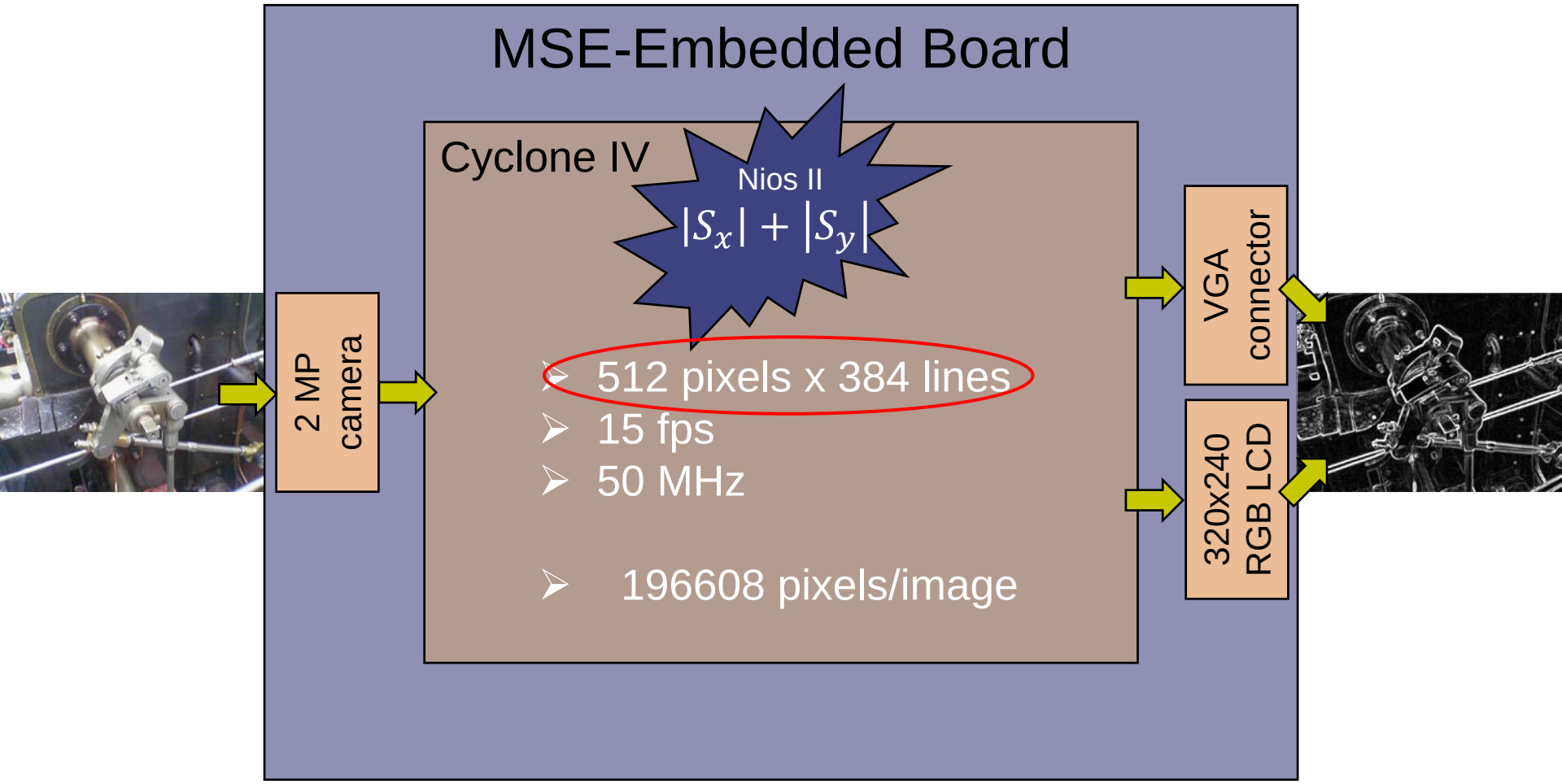
VGA connector

320x240 RGB LCD

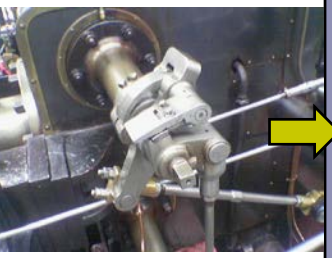


Camera:
- **RGB565**
- **512x384**
- **5-15 fps**

Some numbers



Some numbers



2 MP camera

MSE-Embedded Board

Cyclone IV

Nios II

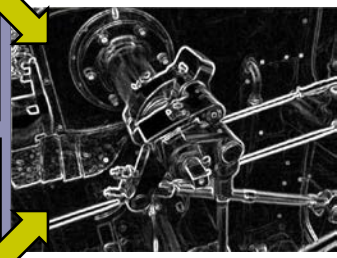
$|S_x| + |S_y|$

- 512 pixels x 384 lines
- 15 fps
- 50 MHz

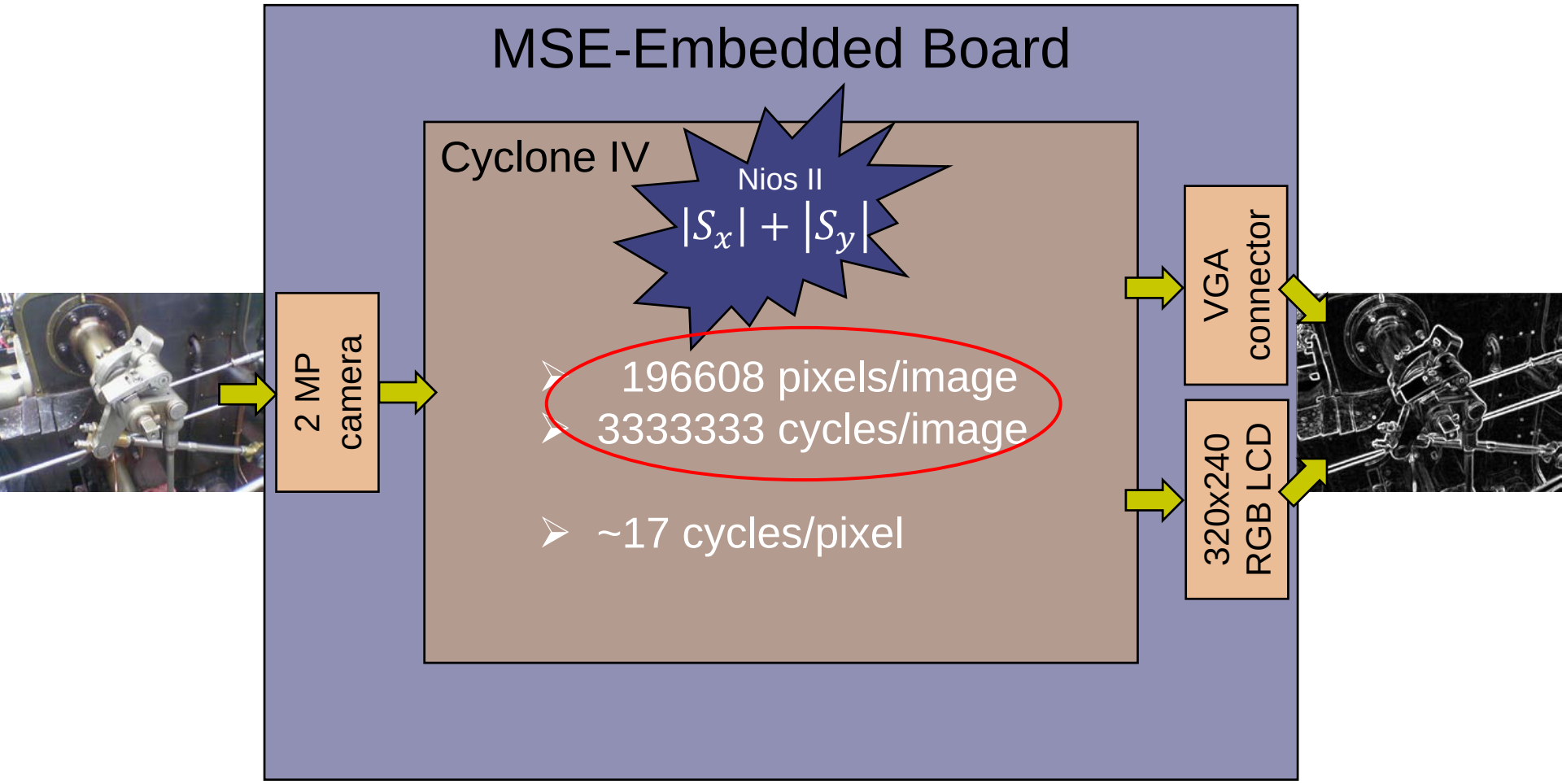
- 196608 pixels/image
- 3333333 cycles/image

VGA connector

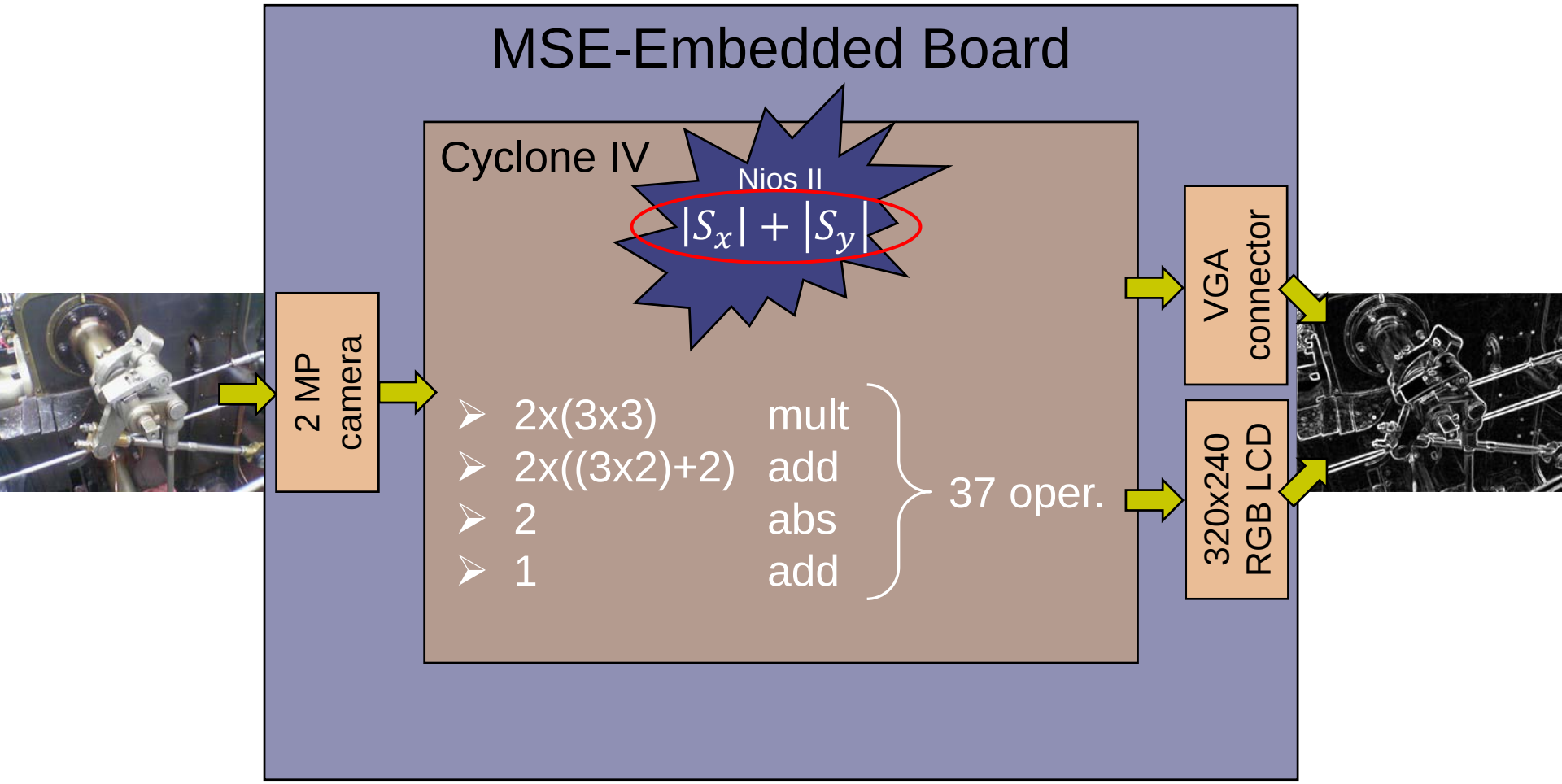
320x240 RGB LCD



Some numbers



Some numbers



Initial implementation

Function	Cycles/image	Cycles/pixel
conv_grayscale	127336549	~648
sobel_x	729624798	~3711
sobel_y	729677739	~3711
sobel_threshold	102849150	~523
TOTAL	**1689488236**	**8593**

Initial implementation

Function	Cycles/image	Cycles/pixel
conv_grayscale	127336549	~648
sobel_x	729624798	~3711
sobel_y	729677739	~3711
sobel_threshold	102849150	~523
TOTAL	**1689488236**	**8593**

Camera:
- **RGB565**
- **512x384**
- **5-15 fps**

vs.

17 cycles/pixel

(~505x improvement)

Problem specifications

We need real-time edge-detection on a live video stream

Backup

Initial implementation

Function	Cycles/image	Cycles/pixel
conv_grayscale	50845746	~259
sobel_x	161582796	~822
sobel_y	161976127	~824
sobel_threshold	32185548	~164
TOTAL	**406590217**	**2068**

Initial implementation

Function	Cycles/image	Cycles/pixel
conv_grayscale	50845746	~259
sobel_x	161582796	~822
sobel_y	161976127	~824
sobel_threshold	32185548	~164
TOTAL	**406590217**	**2068**

Camera:
- **RGB565**
- **512x384**
- **5-15 fps**

vs.

17 cycles/pixel

(~122x improvement)

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

Simple X

=

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	-10	-10	0	0	10	10	0	X
X	-20	-20	0	0	20	20	0	X
X	-30	-30	0	0	30	30	0	X
X	-20	-20	0	0	20	20	0	X
X	-10	-10	0	0	10	10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X

Sobel filter

Original

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

*

1	1	1
0	0	0
-1	-1	-1

Simple Y

=

Filtered

X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	X
X	-10	-20	-30	-30	-20	-10	0	X
X	-10	-20	-30	-30	-20	-10	0	X
X	0	0	0	0	0	0	0	X
X	10	20	30	30	20	10	0	X
X	10	20	30	30	20	10	0	X
X	0	0	0	0	0	0	0	X
X	X	X	X	X	X	X	X	X