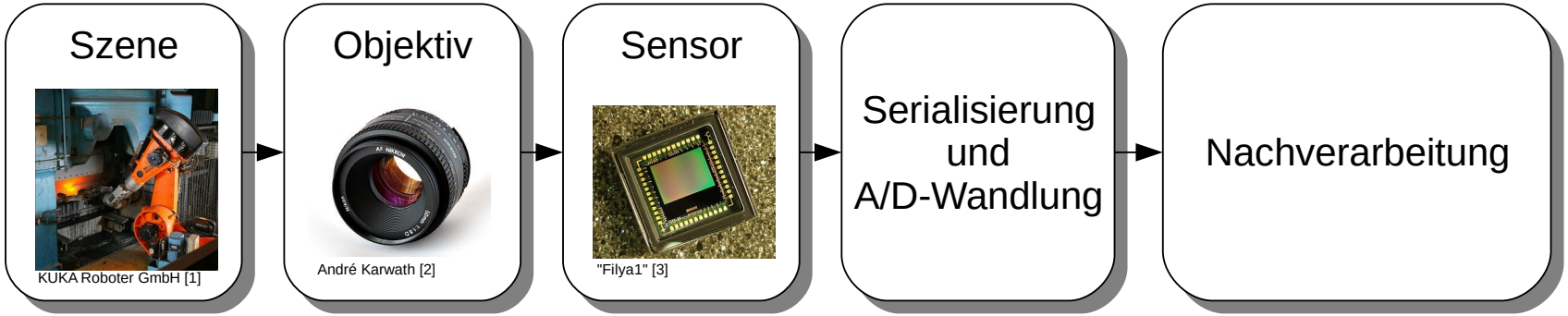
Entwurf eines Generators zur Erzeugung von Hard- und Software-Beschreibungen für Bildverarbeitungspipelines

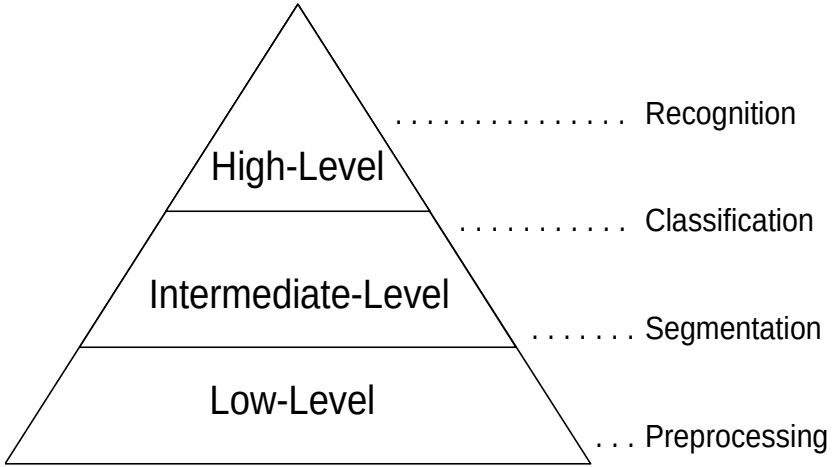
Verteidigungsvortrag zur Studienarbeit

Ludger Irsig

Digitale Kameras allgemein



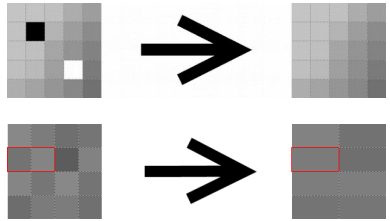
Machine Vision

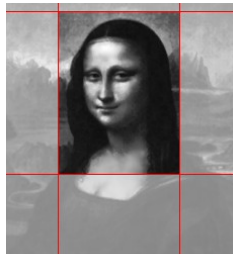


nach Donald G. Bailey: *Design for Embedded Image Processing on FPGAs* [4]

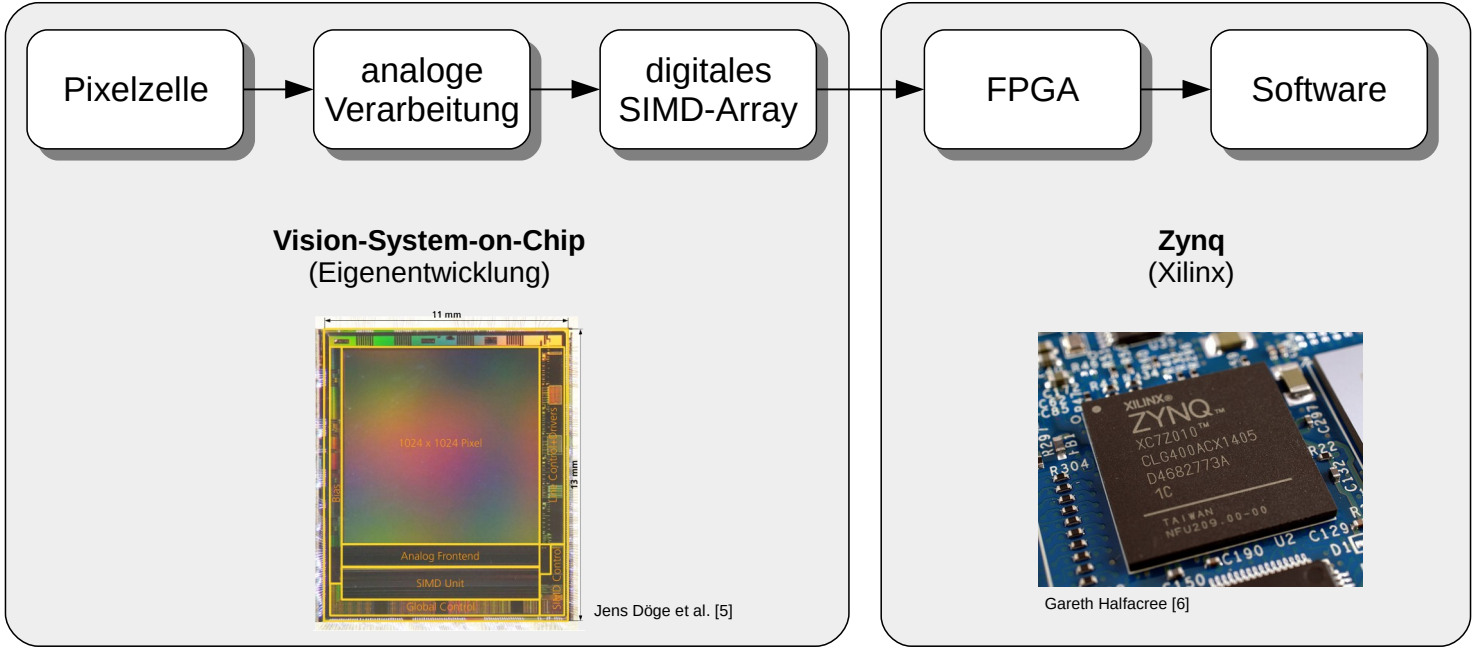
Bsp. für Low-Level:

- Defektpixelkorr.
- Binning
- Region of Interest
- Kontrastanpassung





Machine Vision mittels VSoC am Fraunhofer IIS



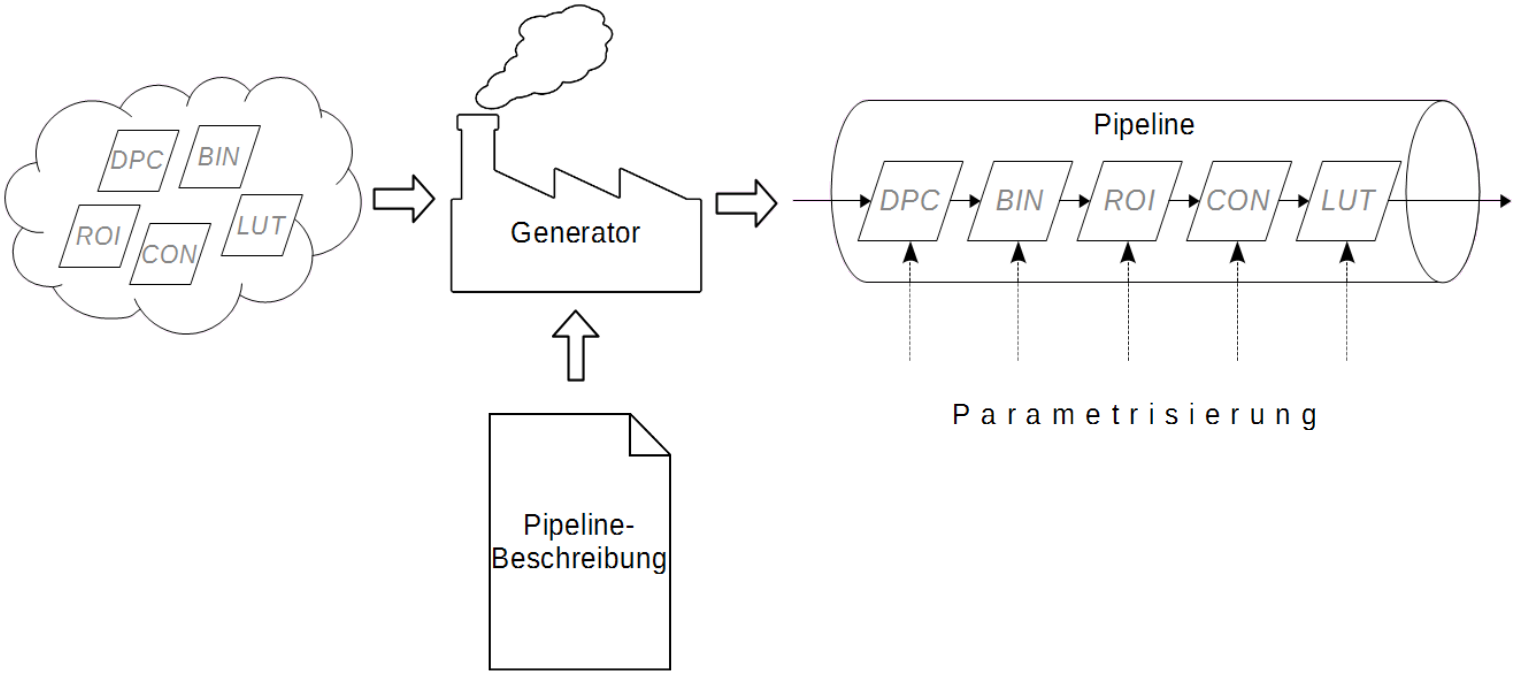
Motivation der Studienarbeit

- unterschiedliche Probleme erfordern unterschiedliche Verarbeitungsprozesse
- lassen sich in wiederkehrende Elemente zerlegen

- Problem: händisches Zusammenfügen ist zeitaufwändig und fehleranfällig; häufig Iterationen nötig

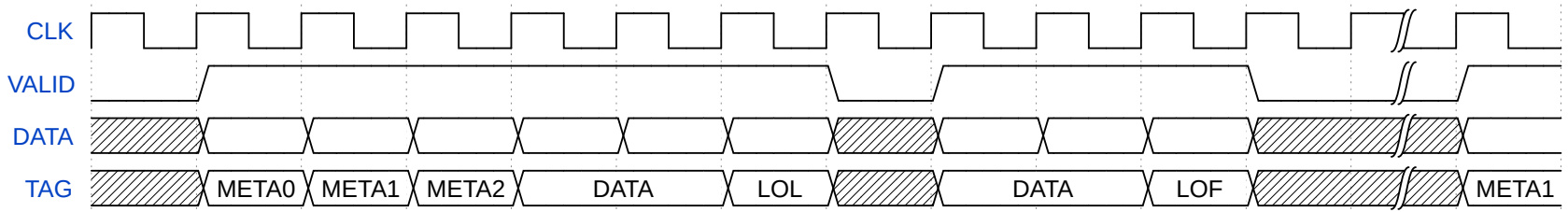
- Lösung: Generatorwerkzeug + Bibliothek von Blöcken

Was ist zu tun?



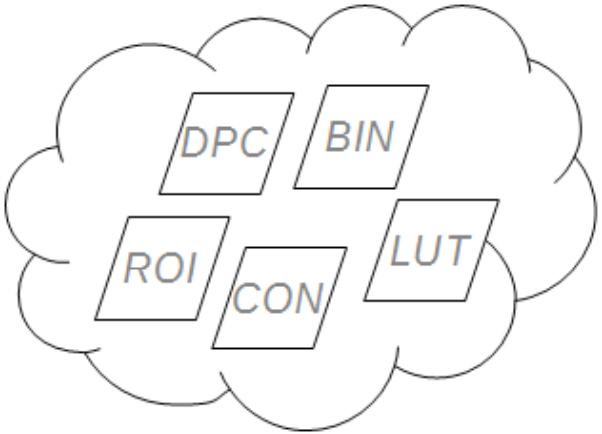
Pipelinedatenformat

- Pixeldatenstrom, mit Pausen
- Metadaten als Header (3 Takte)
- ≥16 Bit Daten, 3 Bit Tag, 1 Bit Valid



Pipelineelemente

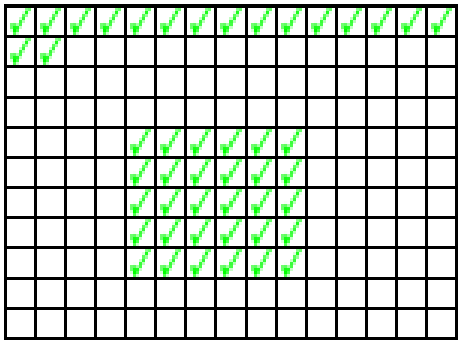
- Blöcke = VHDL-Entities
- erweitert um Annotationen
  - Gruppen von Signalen
  - Parameter
- Implementierung:
  - Stream Processing
  - generisch, Wiederverwendung



Parametrierung

- Bsp.: Region of Interest
  - Parameteränderung stört Datenstrom
- Zwischenpuffer für Parameter
- automatisch anlegen

- x_low = 4
- x_high = 9
- y_low = 4
- y_high = 8



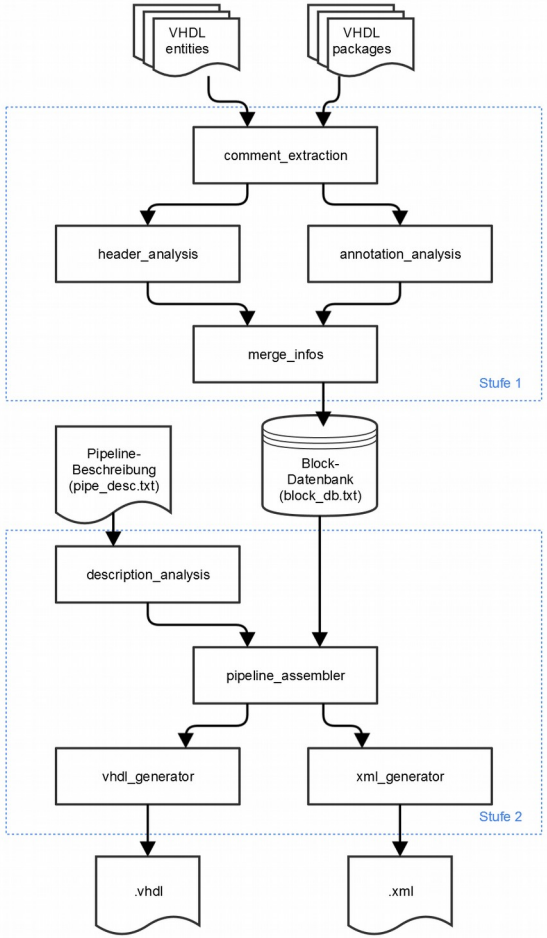
Pipelinebeschreibung

- eigenes Format
- führt Elemente auf
- gibt Verbindungen an
- legt Generics fest

```
1
2  GLOBAL
3
4    NAME mypipe
5
6    CLOCK sysclk
7    RESET sysrst
8
9    CBUS_ID_LIST range(0x10,0x1f)
10
11   PORT stream_in
12   PORT stream_out
13
14   GENERIC G_STREAM_BITS 32
15   GENERIC G_MAX_COMP_PATHS 4
16   GENERIC G_MAX_COMP_BITS 12
17
18 END GLOBAL
19
20 ------------------------------
21
22 ENTITY dpc_i IS TYPE dpc
23   CONNECT sysclk       sysclk
24   CONNECT sysrst       sysrst
25   CONNECT dstream_in   stream_in
26 END ENTITY
27
28 ENTITY binning_i IS TYPE binning
29   CONNECT sysclk       sysclk
30   CONNECT sysrst       sysrst
31   CONNECT dstream_in   dpc_i_dstream_out
32 END ENTITY
33
```

Was tut der Generator? (I)

- zweistufig:
  - analysiere Elemente und ihre Schnittstellen
  - setze Pipeline zusammen



Was tut der Generator? (II)

- testet Kompatibilität der Schnittstellen
- Bsp.:

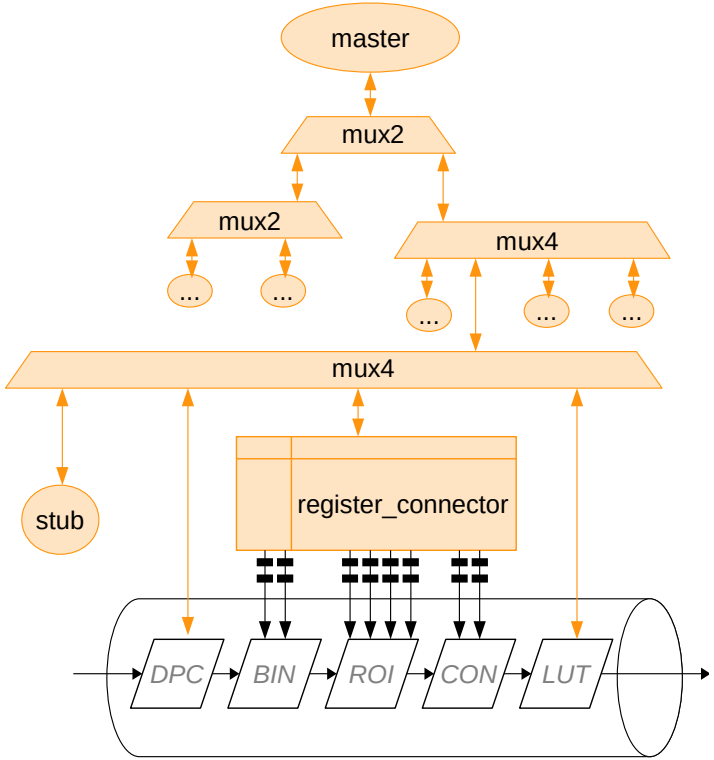
```
std_logic

std_logic_vector ( G_STREAM_BITS - 1  downto  0 )

std_logic_vector ( min(16, C_META_SIZE) - 1  downto  0 )
```

Was tut der Generator? (III)

- Verbinden der Parameter-Schnittstellen
  - CBus
    - ein Master
    - bis zu 32 Komponenten
    - dezentrale Adressierung durch eindeutige IDs
- Adressraum aufteilen



Software-Schnittstelle

- C++-Bibliothek, Abstraktionsschicht
- vorher: Zugriff über Komponenten-ID und Adresse
  - z.B. `write ( 0x10, 0x0001, 0x00420023 )`
- jetzt: Zugriff über Namen des Elements und des Parameters
  - z.B. `write ( "roi_i", "x_low", 3.14159 )`
  - behält auch nach Änderung Gültigkeit!
- Synchronisation

Nutzen des Generatorwerkzeugs

- massiv verkürzte Schreibweise (Faktor 10)
- ~~keine~~ weniger Copy-Paste-Fehler
- reproduzierbar, Versionskontrolle
- Einbindung in automatisierten Workflow möglich

```
GLOBAL

  NAME mypipe

  CLOCK sysclk
  RESET sysrst

  CBUS_ID_LIST range(0x10,0x1f)

  PORT stream_in
  PORT stream_out

  GENERIC G_STREAM_BITS 32
  GENERIC G_MAX_COMP_PATHS 4
  GENERIC G_MAX_COMP_BITS 12

END GLOBAL

------------------------------

ENTITY dpc_i IS TYPE dpc
  CONNECT sysclk ···· sysclk
  CONNECT sysrst ···· sysrst
  CONNECT dstream_in  stream_in
END ENTITY

ENTITY binning_i IS TYPE binning
  CONNECT sysclk ···· sysclk
  CONNECT sysrst ···· sysrst
  CONNECT dstream_in  dpc_i_dstream_out
END ENTITY

ENTITY roi_i IS TYPE roi
  CONNECT sysclk ···· sysclk
  CONNECT sysrst ···· sysrst
  CONNECT dstream_in  binning_i_dstream_out
END ENTITY

ENTITY contrast_i IS TYPE contrast
  CONNECT sysclk ···· sysclk
  CONNECT sysrst ···· sysrst
  CONNECT dstream_in  roi_i_dstream_out
END ENTITY

ENTITY lut_i IS TYPE lut
  CONNECT sysclk ···· sysclk
  CONNECT sysrst ···· sysrst
  CONNECT dstream_in  contrast_i_dstream_out
  CONNECT dstream_out stream_out
END ENTITY

```

Vielen Dank!

Insbesondere an die

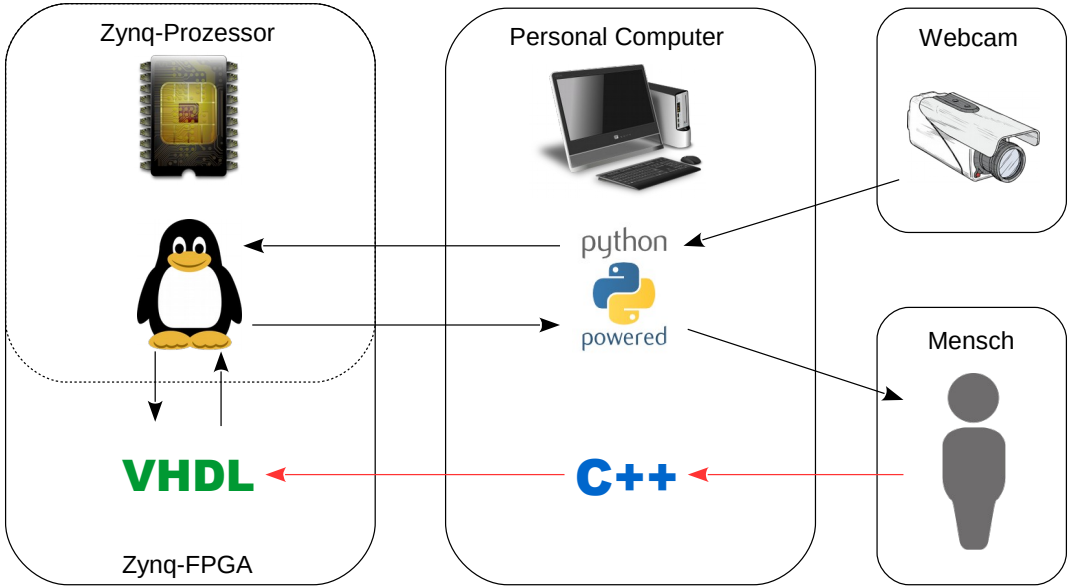
Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

der TU Dresden unter Prof. Spallek

und die

Arbeitsgruppe Optische Sensorik des Fraunhofer IIS/EAS

Demonstration



unter Nutzung von openclipart.org [7]

Backup: Parsen der VHDL-Grammatik

- EBNF:
  design_file ::= design_unit { design_unit }
  design_unit ::= context_clause library_unit
  library_unit ::= primary_unit | secondary_unit
- Pyparsing:

```
library_unit = (primary_unit | secondary_unit)
design_unit = (Group(context_clause + library_unit))("design_unit")
design_file = (Group(OneOrMore(design_unit)))("design_file")
```

- Ergebnis in XML:

```
<parseResults>
  <design_file>
    <design_unit>
      <context_clause>
        ...
      </context_clause>
      <entity_declaration>
        ...
      </entity_declaration>
    </design_unit>
```

Backup: Annotiertes Pipelineelement

```
48  entity roi is
49      generic (
50          G_STREAM_BITS       : positive
51      );
52      port (
53          -- global signals
54          sysclk          : in std_logic;
55          sysrst          : in std_logic;
56
57          -- stream input
58          --$ BUS dstream_in
59          in_stream_data  : in std_logic_vector(G_STREAM_BITS-1 downto 0);     --$ IS d
60          in_stream_tag   : in std_logic_vector(C_PIPE_TAG_BITS-1 downto 0);  --$ IS t
61          in_stream_valid : in std_logic;                                     --$ IS v
62          --$ END BUS
63
64          -- stream output
65          --$ BUS dstream_out
66          out_stream_data  : out std_logic_vector(G_STREAM_BITS-1 downto 0);     --$ IS d
67          out_stream_tag   : out std_logic_vector(C_PIPE_TAG_BITS-1 downto 0);  --$ IS t
68          out_stream_valid : out std_logic;                                     --$ IS v
69          --$ END BUS
70
71          -- control
72          in_frame : out std_logic;
73          --$ CREGS WRITE
74          x_low    : in std_logic_vector(15 downto 0); --$ SYNC LOCK in_frame FORMAT uint(16) DEFAULT     0 DESCRIPTION first column
75          x_high   : in std_logic_vector(15 downto 0); --$ SYNC LOCK in_frame FORMAT uint(16) DEFAULT 65535 DESCRIPTION last column
76          y_low    : in std_logic_vector(15 downto 0); --$ SYNC LOCK in_frame FORMAT uint(16) DEFAULT     0 DESCRIPTION first row
77          y_high   : in std_logic_vector(15 downto 0)  --$ SYNC LOCK in_frame FORMAT uint(16) DEFAULT 65535 DESCRIPTION last row
78          --$ END CREGS
79      );
80  end entity;
```

Bildquellen

[1] KUKA Roboter GmbH, https://commons.wikimedia.org/wiki/File:Automation_of_foundry_with_robot.jpg (gemeinfrei)

[2] André Karwath, https://commons.wikimedia.org/wiki/File:Lens_Nikkor_50mm.jpg (Creative Commons)

[3] "Filya1", https://commons.wikimedia.org/wiki/File:Matrixw.jpg (Creative Commons)

[4] nach Bailey, Donald G.: Design for Embedded Image Processing on FPGAs. John Wiley & Sons, 2011

[5] Döge, Jens; Hoppe, Christoph; Reichel, Peter; Peter, Nico: A 1 Megapixel HDR Image Sensor SoC with Highly Parallel Mixed-Signal Processing. In: 2015 International Image Sensor Workshop (IISW), 2015

[6] Gareth Halfacree, https://www.flickr.com/photos/120586634@N05/15455173526 (Creative Commons)

[7] https://openclipart.org/ (gemeinfrei)