Praktikum: Entwicklung interaktiver eingebetteter Systeme

Filter-Labs
(Designing video streaming applications in SystemC)

Joachim Falk (falk@cs.fau.de)
Running the Virtual Prototype

- Open the Eclipse workspace “workspace03”
Running the Virtual Prototype

- Open the project “hw”
Virtual Prototype

- Implementation of a SoC for video stream processing
  - Using a VideoSource transmitting video data from a webcam
  - to a custom filter module(s)
    - Connected via FIFO or
    - RTL signal channels
  - finally arriving at the VideoSink used for display.

And create adapters to interface between abstraction levels

SystemC modules

Today we implement filters here

Video source module → channel → Video filter module → channel → Video sink module

Communication channels (signal or FIFO type)
Agenda

- Video Filter Image Colorer
  - Task 1) Port definition for RTL signal implementation
  - Task 2) Port definition for FIFO based implementation
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
- Video Source Color Check Board
- Video Filter Skin Color Detector
- Video Filter Sliding Window
- Video Filter Integral
Task 1) Add RTL support to VideoFilterImageColorer

- Modify VideoFilterImageColorer.hpp file in order to:
  - Define input and output ports called Video_in and Video_out, respectively. The ports should be for RTL level signals carrying sc_dt::sc_uint<32> values.
  - Define the clock input port called "Clock" to connect to a boolean signal sc_core::sc_signal<bool>.
  - HINT: An example of boolean port definition for an input port is given. The reset input port called "Reset" that is connected to a boolean signal sc_core::sc_signal<bool>.

Compile and Run the program using the eclipse

- “Build” menu
- “Run” menu
Task 2) Add FIFO support to VideoFilterImageColorer

- Modify VideoFilterImageColorer.hpp in order to:
  - Define input and output ports called Video_in and Video_out, respectively. The ports should be for SystemC FIFO channels carrying the user defined data type VideoSignal.

Compile and Run the program using the eclipse

- “Build” menu
- “Run” menu
Agenda

- Video Filter Image Colorer

- Video Filter Grayscale Conversion
  - Task 3) Port definition for RTL signal implementation
  - Task 4) Process implementation of grayscale conversion
  - Task 5) FIFO based implementation and port definition

- FIFO->Signal & Signal->FIFO adapters

- Video Source Color Check Board

- Video Filter Skin Color Detector

- Video Filter Sliding Window

- Video Filter Integral
Task 3) VideoFilterGrayscaleConversion RTL support

- In VideoFilterGrayscaleConversion.hpp:
  - Define input and output ports called Video_in and Video_out, respectively. The ports should be for RTL level signals carrying `sc_dt::sc_uint<32>` values.
  - Define the clock input port called "Clock" to connect to a boolean signal `sc_core::sc_signal<bool>`.
  - **HINT:** An example of boolean port definition for an input port is given. The reset input port called "Reset" that is connected to a boolean signal `sc_core::sc_signal<bool>`.

- In VideoFilterGrayscaleConversion.cpp:
  - Name the clock input port
  - Name the Video_in input port
  - Name the Video_out output port
  - **HINT:** The port naming for the Reset port is given as an example
Video Filter Image Grayscale

Task 4) Process implementation of grayscale conversion

- In VideoFilterGrayscaleConversion.hpp:
  - Modify the void pixelThrd() thread in order to implement the gray scale conversion filter.

  
  ```
  convertedPixel.red = convertedPixel.blue = convertedPixel.green =
  0.299*originalPixel.red + 0.587*originalPixel.green +
  0.114*originalPixel.blue
  ```

Compile and Run the program using the eclipse

- “Build” menu

- “Run” menu
Video Filter Image Grayscale

Task 5) VideoFilterGrayscaleConversion FIFO support

- Modify VideoFilterGrayscaleConversion.hpp in order to:
  - Define input and output ports called Video_in and Video_out, respectively. The ports should be for SystemC FIFO channels carrying the user defined data type VideoSignal.

- In VideoFilterGrayscaleConversion.cpp:
  - Switch the void pixelThrd() process to use FIFO channels. Guard your changes with

```c++
#if VIDEOBARABSTRACTION_VIDEOFILTERGRAYSCALECONVERSION == VIDEOBARABSTRACTION_FIFO
// your changes
#else
// the old code
#endif // VIDEOBARABSTRACTION_VIDEOFILTERGRAYSCALECONVERSION == VIDEOBARABSTRACTION_FIFO
```

- Compile and Run the program using the eclipse
  - “Build” menu
  - “Run” menu
Agenda

- Video Filter Image Colorer

- Video Filter Grayscale Conversion

- FIFO->Signal & Signal->FIFO adapters
  - Interlude) VGA Timings
  - Task 6) FIFO to RTL signal adapter
  - Task 7) RTL signal to FIFO adapter

- Video Source Color Check Board

- Video Filter Skin Color Detector

- Video Filter Sliding Window

- Video Filter Integral
VGA Signals

- VGA synchronization signals:

<table>
<thead>
<tr>
<th>Name</th>
<th>In Pixels</th>
<th>Name</th>
<th>In Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGA_HFRONT_PORCH</td>
<td>16</td>
<td>VGA_VFRONT_PORCH</td>
<td>10</td>
</tr>
<tr>
<td>VGA_HBACK_PORCH</td>
<td>48</td>
<td>VGA_VBACK_PORCH</td>
<td>33</td>
</tr>
<tr>
<td>VGA_HVISIBLE</td>
<td>640</td>
<td>VGA_VVISIBLE</td>
<td>480</td>
</tr>
<tr>
<td>VGA_HSYNC</td>
<td>96</td>
<td>VGA_VSYNC</td>
<td>2</td>
</tr>
</tbody>
</table>
Model of I/O-Bar
Pixel Signals

- FPGA operates at a higher frequency than the camera
  - Only every 3rd or 4th clock cycle arrives a new pixel
  - A rising edge in the pixel clk signalizes a new pixel
Pixel Signals

- Signals in the VideoBar
  - Hsync: indicates a new line
  - Vsync: indicates a new frame
  - Visible: indicates the visible area
Agenda

- Video Filter Image Colorer
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
  - Interlude) VGA Timings
  - Task 6) FIFO to RTL signal adapter
  - Task 7) RTL signal to FIFO adapter
- Video Source Color Check Board
- Video Filter Skin Color Detector
- Video Filter Sliding Window
- Video Filter Integral
FIFO to RTL Signal Adapter

- Implement a FIFO to RTL signal adapter that
  - has a `sc_core::sc_fifo` as input and
  - has a `sc_core::sc_signal` as output

To Compile and Run the filter chain use the eclipse

- “Build” menu
- “Run” menu
FIFO to RTL Signal Adapter

- Task 6) FIFO to RTL signal adapter
  - In VideoAdapterFIFOToRTL.hpp:
    - Add required methods and definitions
  - In VideoAdapterFIFOToRTL.cpp:
    - Register a SC_THREAD process in the constructor to do the work
    - **Hint:** Within the registered SC_THREAD process:
      - **Hint:** For each value read from the sc_fifo, write
      - **Hint:** one value with pixelClk true to the sc_signal and
      - **Hint:** one value with pixelClk false.

```
tmp <- Video_in
wait(<correct time to correspond to vga timings>)
tmp.pixelClk = 1;
Video_out <- tmp
wait(<correct time to correspond to vga timings>)
tmp.pixelClk = 0;
Video_out <- tmp
```

- Compile, build and run the project as shown in the previous slide
RTL Signal to FIFO Adapter

- Implement a RTL to FIFO signal adapter that
  - has a `sc_core::sc_signal` as input and
  - has a `sc_core::sc_fifo` as output

To Compile and Run the filter chain use the eclipse
- “Build” menu
- “Run” menu
Task 7) RTL signal to FIFO adapter

- In VideoAdapterRTLToFIFO.hpp:
  - Add required methods and definitions

- In VideoAdapterRTLToFIFO.cpp:
  - Register a `SC_THREAD` process in the constructor to do the work
  - **Hint:** Within the registered `SC_THREAD` process:
    - **Hint:** For each rising edge for `pixelClk` write a value to the `sc_fifo`

```cpp
wait(<for rising edge of Video_in.pixelClk>)
tmp <- Video_in
Video_output <- tmp
```

Compile, build and run the project as shown in the previous slide
Agenda

- Video Filter Image Colorer
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
- Video Source Color Check Board
  - Task 8) Process implementation for FIFO and RTL case (Synthesize and test on the SPARTAN3 board)
- Video Filter Skin Color Detector
- Video Filter Sliding Window
- Video Filter Integral
Check Board Pattern Generation

- Implement a color check board generator video source
  - The **red** color of the pixels appear in boxes with size of 16 pixels, in a check board pattern
  - The **green** color of the pixels appear in boxes with size of 32 pixels, in a check board pattern
  - The **blue** color of the pixels appear in boxes with size of 64 pixels, in a check board pattern
Check Board Pattern Generation

Task 8) Process implementation for FIFO and RTL case

- In VideoSourceColorCheckBoard.hpp:
  - Add required methods and definitions

- In VideoSourceColorCheckBoard.cpp:
  - In the constructor: Register a SC_THREAD process for the FIFO case
  - In the constructor: Register a SC_CTHREAD process for the RTL case
  - Hint: Do not forget to specify the reset signal for the SC_CTHREAD process!
  - Implement the registered process to generate the color check board
  - Hint: Use the following for differences between FIFO and RTL case

```c++
#if VIDEOBARABSTRACTION_VIDEOSOURCECOLORCHECKBOARD == VIDEOBARABSTRACTION_FIFO
  // Code for FIFO case
#else
  // Code for RTL case
#endif
```
Check Board Pattern Generation

- Compile and Run the FIFO case using the eclipse
  - “Build” menu
  - “Run” menu

- Compile and Run the RTL case using the eclipse
  - “Build” menu
  - “Run” menu
Agenda

- Video Filter Image Colorer
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
- Video Source Color Check Board
- Video Filter Skin Color Detector
  - Task 9) Process registration for FIFO and RTL case
  - Interlude) Simple Skin Color Detection
  - Task 10) Process implementation for FIFO and RTL case (Synthesize and test on the XUPv2 board)
- Video Filter Sliding Window
- Video Filter Integral
Video Filter Skin Color Detector

- **Task 9**) Process registration for FIFO and RTL case
  - In VideoFilterSkinColorDetector.hpp:
    - Add required methods and definitions
  - In VideoFilterSkinColorDetector.cpp:
    - In the constructor: Register a **SC_THREAD** process for the FIFO case
    - In the constructor: Register a **SC_CTHREAD** process for the RTL case
    - **Hint:** Do not forget to specify the reset signal for the **SC_CTHREAD** process!
    - Implement the registered process to generate the color check board
    - **Hint:** Use the following for differences between FIFO and RTL case

```
#if VIDEOBARABSTRACT_ACTION_VIDEOFILTERSKINCOLORDETECTOR == VIDEOBARABSTRACT_ACTION_FIFO
// Code for FIFO case
#else // VIDEOBARABSTRACT_ACTION_VIDEOFILTERSKINCOLORDETECTOR != VIDEOBARABSTRACT_ACTION_FIFO
// Code for RTL case
#endif // VIDEOBARABSTRACT_ACTION_VIDEOFILTERSKINCOLORDETECTOR != VIDEOBARABSTRACT_ACTION_FIFO
```
Simple Skin Color Detection

- Skin color detection via threshold logic
  \[
  \{ (R, G, B) \mid r_{\text{min}} < R < r_{\text{max}} \land g_{\text{min}} < G < g_{\text{max}} \land b_{\text{min}} < B < b_{\text{max}} \}\]

- A slightly more complex algorithm proposed by [VSA03]
  \[
  (R, G, B) \mid R > 95 \land G > 40 \land B > 20 \land \\
  \text{max}\{R, G, B\} - \text{min}\{R, G, B\} > 15 \land \\
  |R - G| > 15 \land R > G \land R > B
  \]

Video Filter Skin Color Detector

- **Task 10) Process implementation for FIFO and RTL case**
  - In `VideoFilterSkinColorDetector.cpp`:
    - Implement the registered process for skin color detection
    - **Hint:** Use the following for differences between FIFO and RTL case
      ```
      #if VIDEOBARABSTRACTION_VIDEOFILTERSKINCOLORDETECTOR == VIDEOBARABSTRACTION_FIFO
      // Code for FIFO case
      #else // VIDEOBARABSTRACTION_VIDEOFILTERSKINCOLORDETECTOR != VIDEOBARABSTRACTION_FIFO
      // Code for RTL case
      #endif // VIDEOBARABSTRACTION_VIDEOFILTERSKINCOLORDETECTOR != VIDEOBARABSTRACTION_FIFO
      ```
    - Experiment with the methods in the previous slide to find an adequate method to detect skin color

- **Compile and Run the FIFO/RTL case using the eclipse**
  - “Build” menu
    - Run Configurations…
  - “Run” menu
    - Organize Favourites…
Agenda

- Video Filter Image Colorer
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
- Video Source Color Check Board
- Video Filter Skin Color Detector

- Video Filter Sliding Window
  - Interlude) Edge Detection with the Sobel Filter
  - Task 11) Implement edge detection by combining
    - horizontal Sobel filter and
    - Vertical Sobel filter results

- Video Filter Integral
# Edge Detection with the Sobel Filter

Horizontal edges

$H_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Vertical edges

$H_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

Combined result of both sobel filters

Friedrich-Alexander-Universität Erlangen-Nürnberg
Andreas Weichslgartner and Joachim Falk
Task 11) Implement edge detection by combining two Sobel filters

- In VideoFilterSlidingWindow.cpp:
  - Modify the pixelThrd to calculate the two Sobel Filters and return their results in the red and green pixel values
  - **Hint:** The Sobel filter needs a grayscale image as input!

Compile and Run the FIFO/RTL case using the eclipse

- “Build” menu
- “Run” menu
Agenda

- Video Filter Image Colorer
- Video Filter Grayscale Conversion
- FIFO->Signal & Signal->FIFO adapters
- Video Source Color Check Board
- Video Filter Skin Color Detector
- Video Filter Sliding Window

- Video Filter Integral
  - **Task 12)** Port definition for FIFO and RTL case
  - **Task 13)** Process registration for FIFO and RTL case
  - Interlude) Integral Image
  - **Task 14)** Process implementation for FIFO and RTL case
    (Synthesize and test on the XUPv2 board)
Video Filter Integral

➢ Task 12) VideoFilterIntegral port definitions
  ▪ Modify VideoFilterIntegral.hpp in order to add RTL support:
    - Define input and output ports called Video_in and Video_out, respectively. The ports should be for RTL level signals carrying sc_dt::sc_uint<32> values.
    - Define the clock input port called "Clock" to connect to a boolean signal sc_core::sc_signal<bool>.
    - Define the reset input port called “Reset" to connect to a boolean signal sc_core::sc_signal<bool>.
  ▪ Modify VideoFilterIntegral.hpp to add FIFO support:
    - Define input and output ports called Video_in and Video_out, respectively. The ports should be for SystemC FIFO channels carrying the user defined data type VideoSignal.
    - Hint: Use the following for differences between FIFO and RTL case

```cpp
#if VIDEOBARABSTRACTION_VIDEOFILTERINTEGRAL == VIDEOBARABSTRACTION_FIFO
// Code for FIFO case
#else // VIDEOBARABSTRACTION_VIDEOFILTERINTEGRAL != VIDEOBARABSTRACTION_FIFO
// Code for RTL case
#endif // VIDEOBARABSTRACTION_VIDEOFILTERINTEGRAL != VIDEOBARABSTRACTION_FIFO
```
Task 13) Process registration for FIFO and RTL case

- In VideoFilterIntegral.hpp:
  - Add required methods and definitions

- In VideoFilterIntegral.cpp:
  - In the constructor: Register a `SC_THREAD` process for the FIFO case
  - In the constructor: Register a `SC_CTHREAD` process for the RTL case
  - **Hint:** Do not forget to specify the reset signal for the `SC_CTHREAD` process!
Integral Image

- Also referred to as “summed area table”

- A pixel \( II(x',y',t) \) in the integral image contains
  - the sum of all the pixel values in \( I(x,y,t) \) which above and to the left of \((x', y')\)

- This means:

\[
II(x',y',t) = \sum_{y=0}^{y'} \sum_{x=0}^{x'} I(x,y,t)
\]
Calculating the Integral Image

- It is not necessary to calculate $\text{II}(x', y', t) = \sum_{y=0}^{y'} \sum_{x=0}^{x'} l(x, y, t)$ for each single pixel.
- Instead, the calculation can be performed iteratively:

\[
\begin{align*}
    s(x', y', t) &= s(x', y'-1, t) + l(x', y', t) \\
    \text{II}(x', y', t) &= \text{II}(x'-1, y', t) + s(x', y', t)
\end{align*}
\]

$\text{II}(x-1,y,t)$ is the value of the Integral Image left to $(x,y)$

$s(x,y-1,t)$ contains the sum of pixel values of column $x$ (up to row $y-1$)
Calculating the Integral Image

- It is not necessary to calculate $ll(x', y', t) = \sum_{y=0}^{y'} \sum_{x=0}^{x'} I(x, y, t)$ for each single pixel.
- Instead, the calculation can be performed iteratively:

$$s(x', y', t) = s(x', y'-1, t) + l(x', y', t)$$
$$ll(x', y', t) = ll(x'-1, y', t) + s(x', y', t)$$

$I(x, y, t)$ contains the sum of pixel values of column $x$ (up to row $y-1$).
$s(x, y-1, t)$ contains the sum of pixel values of column $x$ (up to row $y-1$).

$s(x', y', t)$ is the value of the Integral Image left to $(x, y)$.
Video Filter Integral

- Task 14) Process implementation for FIFO and RTL case
  - In *VideoFilterIntegral.hpp*:
    - Specify an array for the storage of the sum of pixel values of column $x$ (up to row $y-1$)
    - **Note:** In hardware the array will be implemented as a synchronous RAM. Hence, each read or write of an array element will cost one cycle.
  - In *VideoFilterIntegral.cpp*:
    - Implement the registered process to calculate the integral image
    - Store the integral sum lower 8 bits in the green pixel value and the integral sum upper 8 bits in the blue pixel value.
    - **Hint:** To ease debugging, you might want to use the remaining red pixel value to denote where skin color was detected.
    - **Note:** For each processed pixel, you will need to update, that is read and write, one element of the storage array for the sum of pixel values of column $x$ (up to row $y-1$). Hence, you will require at least two cycles per pixel. Therefore, parameterize your while(true) {...} loop to handle data each second cycle.
    - **Hint:** Look into *VideoFilterSlidingWindow::pixelThrd()* for inspiration of this two cycle processing time challenge!
Video Filter Integral

- Compile and Run the FIFO/RTL case using the eclipse
  - “Build” menu
  - “Run” menu