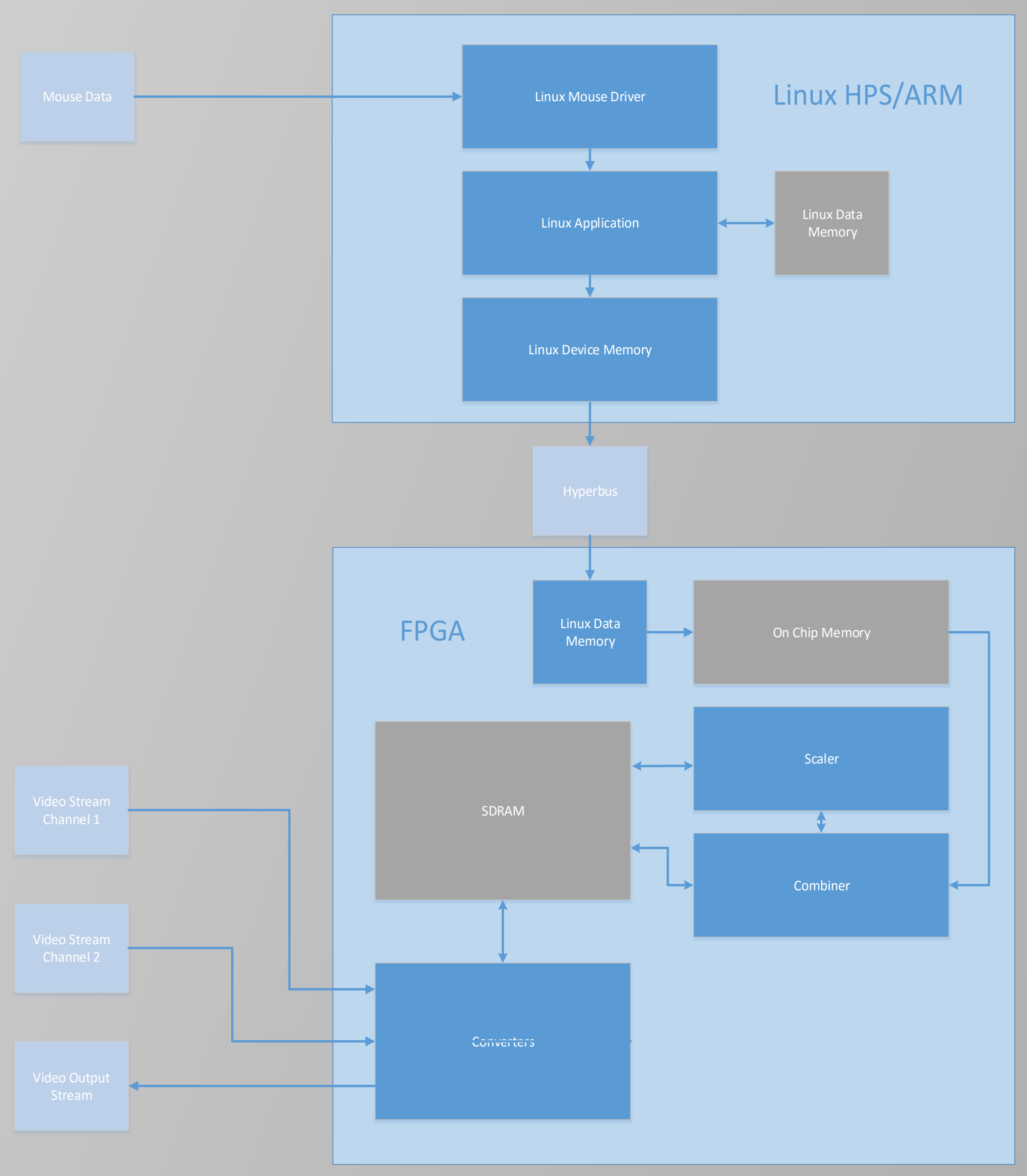
HIGH SECURITY GRAPHICS DATA MANAGER

Andre Mastenbroek, Patrick Neger, Peter Verschuure, Wim van Vugt
Electrical Engineering, Hogeschool Rotterdam University of Applied Sciences
G.J. de Jonghweg 4-6, 3015 GG Rotterdam

For the *Altera Innovate Europe Design Contest 2016*, a secure, hardware based, graphics data manager is designed. The manager allows for multiple video input streams, which are combined together and sent to a single display output port. Within this configuration, the user is able to modify the size and location of each individual stream on the final output, without being able to access the streamed data itself.



Functionality

The data manager separates the OS from the FPGA. The OS controls the position of the window(s) on the output frame. The FPGA processes al graphic data and builds the output frame based on information from the OS. The separation between OS and FPGA improves the security of the graphic data. The user nor OS have access to the data, only the FPGA is able to read the graphic data.

The graphics data manager has multiple functions. These functions define the purpose of the system. The system is capable of the following operations:

- Merge multiple video inputs to one video output.
- The user is able to scale and move the windows of the video inputs on the video output.
- The video data is stored unreachable by the user and the OS.

Data Conversions

The video data passes through multiple conversions. Each conversion is required to receive, store or transmit the data. The video data input has a Y422 PAL format. Before the data is stored in memory the Y422 is converted to 32 bit RGBA. The 32 bit RGBA is converted to 24 bit RGB when the data is written to the VGA output.

OS Linux

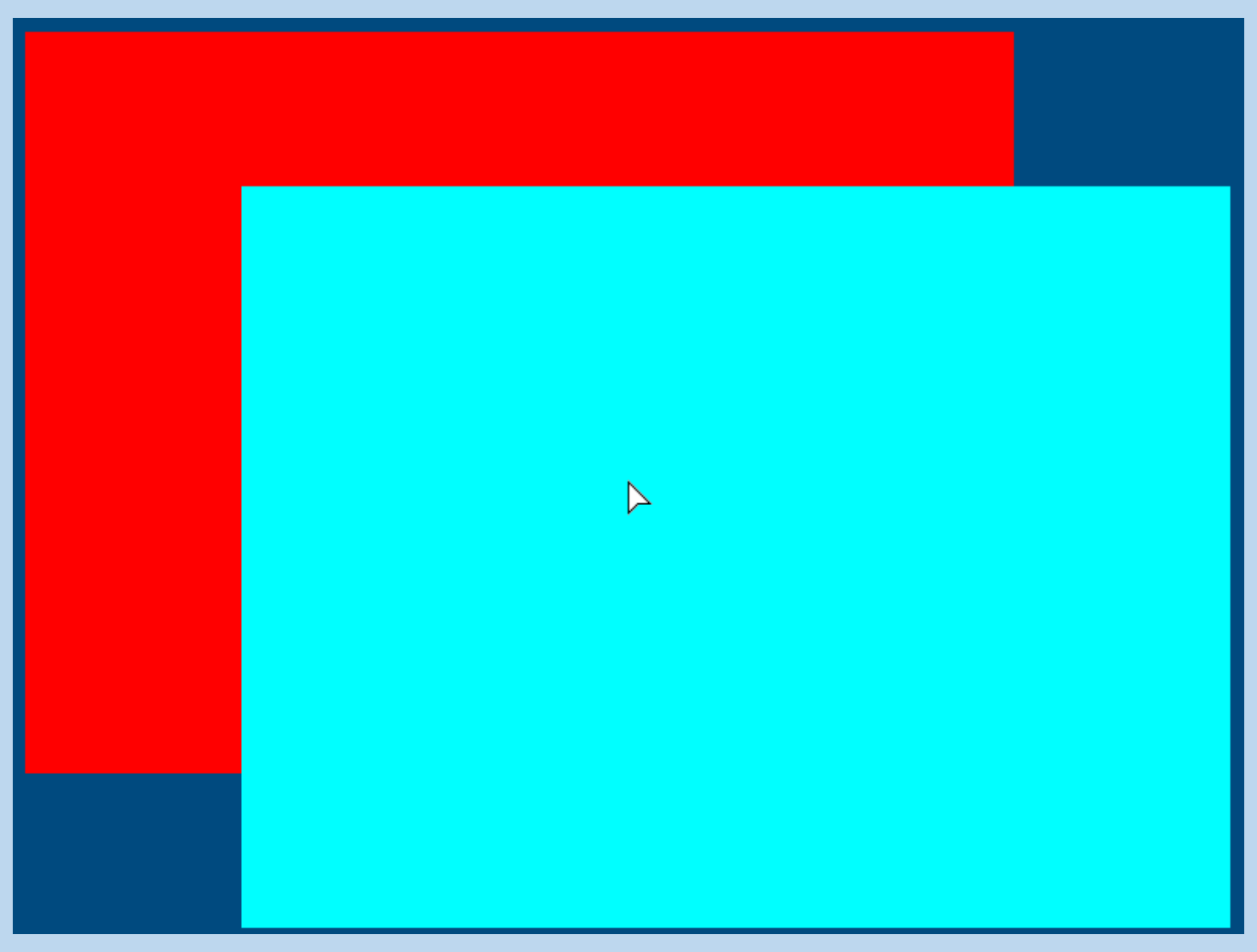
On top of the Hard Processor System of the DE1-SoC runs an LXDE distribution provided by Altera. This application handles the positioning and resizing of the windows and the movement of the mouse. The data is sent to the on-chip-memory on the FPGA. When a window is moved or resized or the mouse is moved, the application will send the updated data to the FPGA. The behavior of the application is defined by a state machine.

Type	Name	Description
Event	Mouse moved	New location of the mouse saved
	Window moved or resized	New window location and/or dimensions saved
Action	Update mouse	Sent new mouse data to the FPGA
	Update window	Sent new window data to the FPGA
State	Initial	Start up
	Mouse moving	Mouse movement detected
	Window moving	Windows movement detected
	Stable	Idle

Window manager

The window manager determines for each pixel individual the layer with the highest priority. When the top layer is known the RGB value for the related pixel and layer is requested. The RGB value is stored in the SDRAM as the final frame. With this method both video inputs and mouse are merged into one frame. The algorithm starts in the upper left corner and determines the final frame row for row. The mouse has always the highest priority possible and is always visible.

Layer	Priority	
Mouse	1	↖
Input Frame 1 (Cyan)	2	🟦
Input Frame 2 (Red)	3	🟥
Background	4	🟦



Scaling algorithm

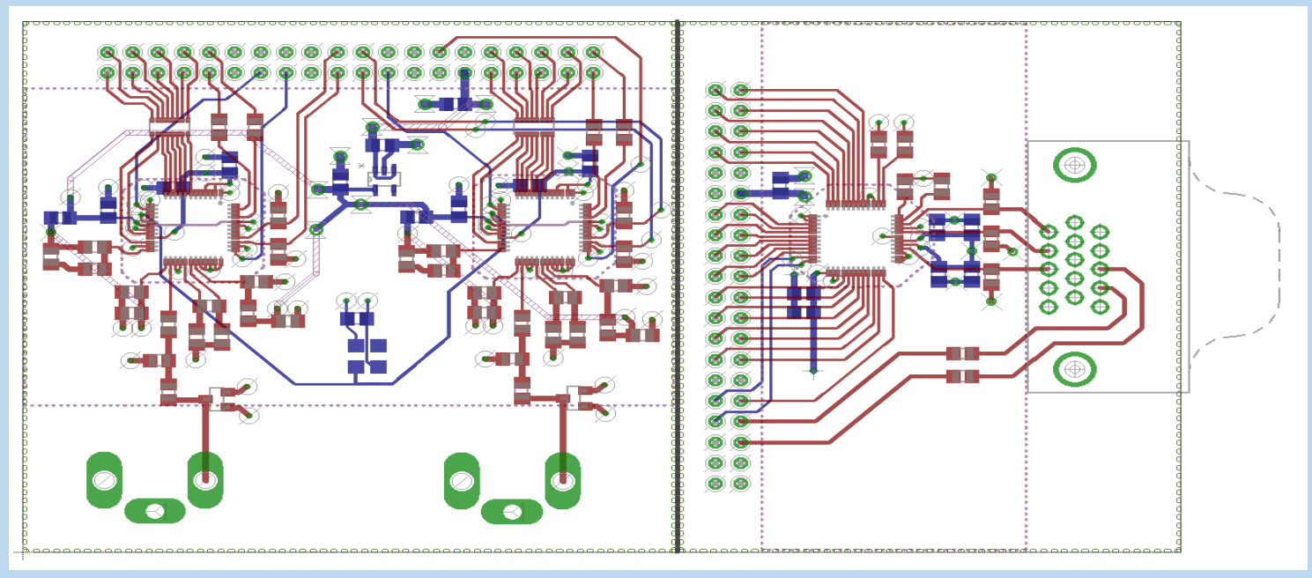
When the user resizes a window in the OS, the video data on the output must be scaled to the new window dimensions. The new RGB value for each pixel within the new dimensions is calculated. To accomplish this task, several possible algorithms can be used, each with varying results:

Interpolation algorithm	Size array	Weighted arithmetic mean
Nearest Neighbour/None	1×1	$h(x,y) = \begin{cases} 1, & \lvert x\rvert \le 0{,}5 \wedge \lvert y\rvert \le 0{,}5 \\ 0, & \lvert x\rvert > 0{,}5 \wedge \lvert y\rvert > 0{,}5 \end{cases}$
Bilinear/Linear	3×3	$h(x,y) = \begin{cases} (1-\lvert x\rvert)(1-\lvert y\rvert), & \lvert x\rvert \le 1 \wedge \lvert y\rvert \le 1 \\ 0, & \lvert x\rvert > 1 \wedge \lvert y\rvert > 1 \end{cases}$
Bicubic/Cubic	5×5	$h(x) = \begin{cases} 1 - \frac{5}{2}\lvert x\rvert^2 + \frac{3}{2}\lvert x\rvert^3, & \lvert x\rvert \le 1 \\ 2 - 4\lvert x\rvert + \frac{5}{2}\lvert x\rvert^2 - \frac{1}{2}\lvert x\rvert^3, & 1 < \lvert x\rvert < 2 \\ 0, & \lvert x\rvert \ge 2 \end{cases}$

Excluding any more complex algorithms, from the above list, the Bicubic Interpolation gives the best result for both enlargement and downsizing of a frame. Due to speed and the ability to work with dynamically sized, moving frames, the graphics data manager employs the Bilinear Interpolation. Using this algorithm results in a good trade-off between quality, speed and memory usage.

Hardware extension

The system is designed to manage at least two video inputs. An external PCB is designed to add two additional video inputs and one VGA output. The PCB is divided into two parts. Both parts are designed with a 40-pins connector which will be placed in both GPIO slots on the DE1-SoC.



Results

We have designed and developed a system capable of processing video streams into one final output frame. The data manager is able to convert and scale the video before building the final frame. The separation between OS and FPGA guarantees the security of the video data while the user can still determine the locations of the windows.

The scaling algorithm is well implemented however, when a frame is being scaled the SDRAM is accessed too much and delays the system. In theory by applying a ring-buffer method in the SDRAM an acceptable FPS is achieved.