# User Guide: Frame Merging Output Peripheral

Opal Densmore, Kei-Ming Kwong, Wahid Rahman

February 5th 2014

Version for EDK 14.2

## 1 Goal

Develop a frame merging IP which merges two input frames by overlaying one input frame on top of another. The output composite frame is stored in memory, where another IP core will output the frame to a monitor.

## 2 Overview

This project provides a custom solution to merge two input frames and output the result to a screen. Two components are required: a method to merge frames, and a method to output the result to a screen. In this project, a frame merging algorithm has been implemented in software, and the result is displayed to HDMI monitor using an existing IP core [1]. Ongoing work is being done to implement the frame merging algorithm as a separate hardware IP core: refer to Appendix A for more details.

The sample project takes two initialized input frames and combines them to generate the final output frame at 640x480 pixel resolution and RGB888 colour space. The following is the terminology in referring to the different frames:

- Draw Frame: an input frame which is overlayed on the top
- Video Frame: an input frame which is under the draw frame
- Composite Frame: an output frame with the output of the merging

The frame merging functionality is realized as a software algorithm. The API is provided in C. This API provides the merging functionality which is executed by the MicroBlaze processor. The sample project provided only supports a specific colour space and resolution; these are customized through the HDL or the C API.

# 3 Directory Structure

The following directory structure indicates important files and folders which are of key interest to a developer using the example project provided.

- project\
  - system.xmp
  - data\
    - system.ucf → Pin connection mapping of the blocks
  - pcores\
    - hdmi_out_v1_00_a [1]
      - user_logic.v → Interface logic between AXI BUS and custom HDL
      - hdmi_core.v → Custom HDL to output HDMI in a configuration
  - workspace\
    - sample_0\src
      - frame_merge.c → C API for frame merging functionality (software or hardware)
      - frame_merge.h → Header file for C API
      - main.c → Sample code to utilize API functions
      - lscript.ld → Linker script to identify different section to address mappings
    - sample_bsp_0\ → Board support headers and src files

# 4 Setup

1. Start XPS and select "Open Project." Browse to the system.xmp within the project folder.
   a. Project → Project Options → Advanced Options → Project Peripheral Search path Ensure that this points to the your local installation of the Atlys_AXI_BSB_support files.
2. Generate Bitstream and wait for the design to compile. This will take several minutes.
3. Export Design and launch Xilinx Software Development Kit.
   a. Select "Include bitstream and BMM file" option.
   b. Use project\workspace as the workspace folder.
4. In the workspace, sample_0\ and sample_bsp_0\ are provided as an example.
5. Create a new Xilinx C Project and select "Empty Application". It is advisable to recreate your own BSP project instead of reusing the sample_bsp_0 BSP.
6. Copy the library files from sample_0\src including the linker script "lscript.ld."

a. Build Project → Rebuild sample_bsp_0 project first if it is being used.

7. Compile the project and Program the FPGA using the .elf file you just compiled.
8. If you have the UART connected, you should see status checks being printed.
9. On the Monitor, the composite frame (shown below) will be on the monitor. By entering the characters '0', '1', or '2' on the UART terminal, the Draw, Video, or Composite Frames will be shown respectively.

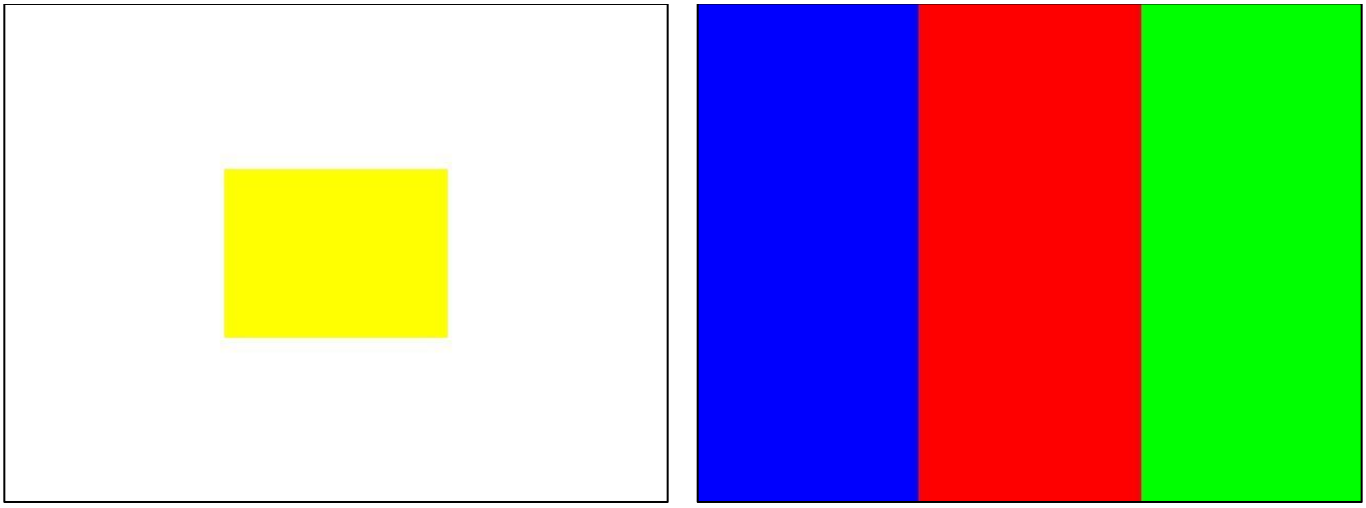

**Figure 1:** Draw Frame (left), Video Frame(right). Black border shows the frame boundaries, for illustrative purposes.

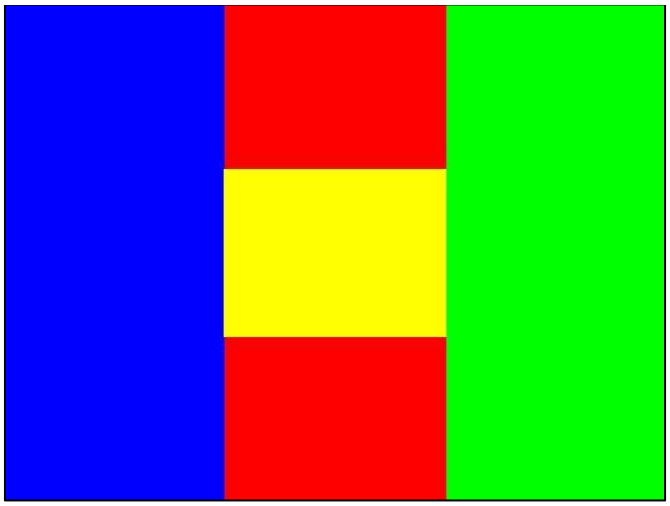

**Figure 2:** Composite Frame (Output)

# 5 Diagrams

The following diagrams provide a more in-depth view of the IP in terms of software algorithm. For hardware system diagrams, refer to Appendix A.

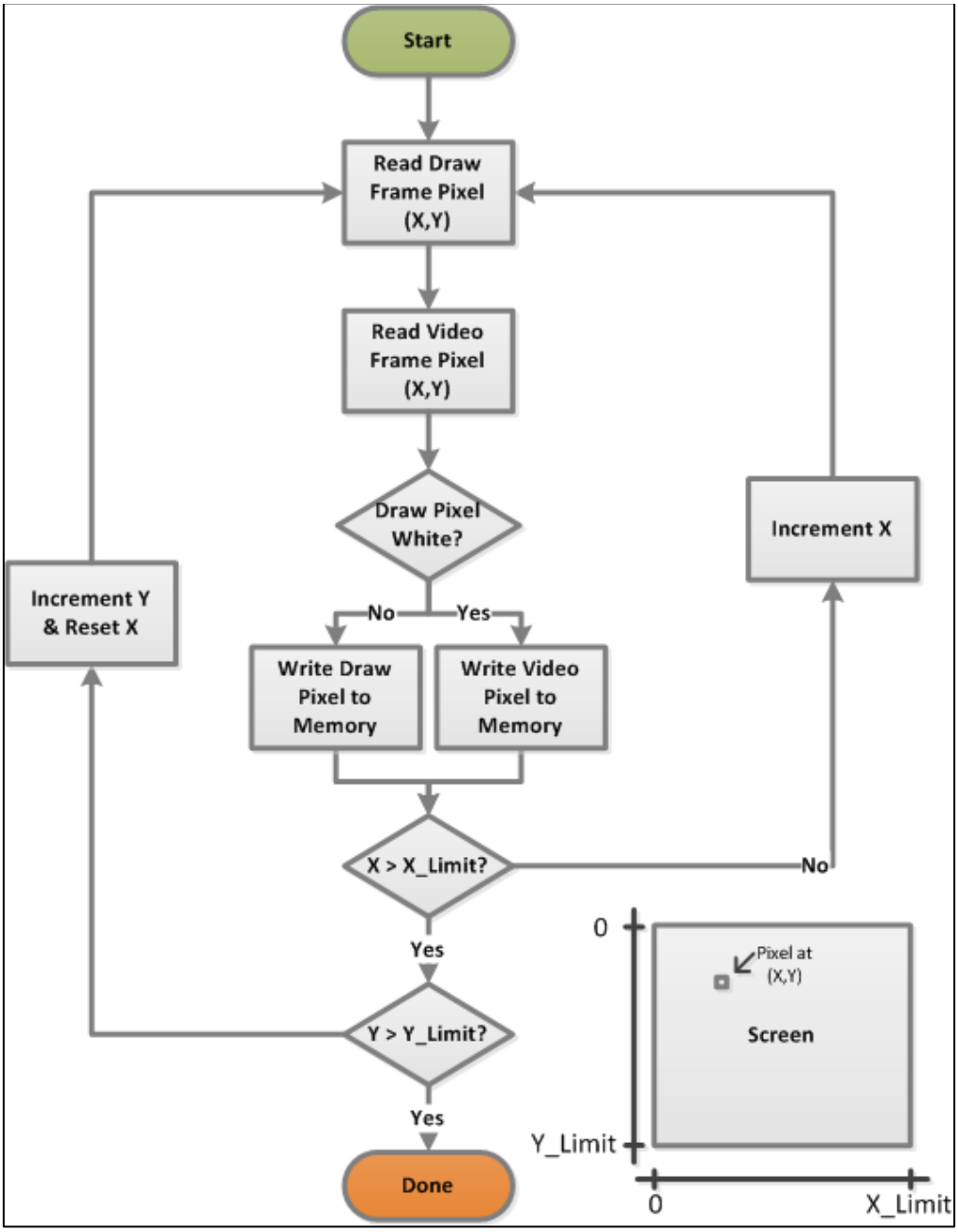


**Figure 3:** Software algorithm for frame merging.

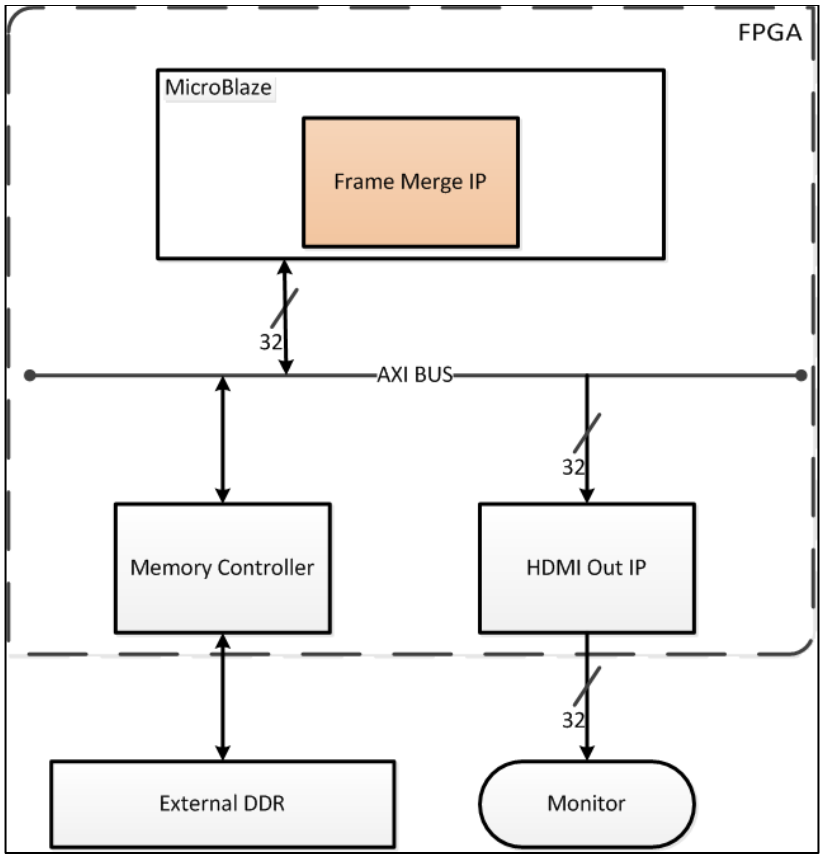


**Figure 4:** System architecture for Frame Merge IP.

# 6 Frame Merge API Usage

**Software API**:

The following API merges two static frames into an output frame. The two input frames are located at specified addresses in external memory. The resulting frame is stored in a specified location in memory as well.

```
void frame_merge_sw(u32* ddr_Addr, u32 Draw_Offset, u32 Video_Offset, u32 Comp_Offset, u32 hres, u32 vres);
```

**Parameters:**

`ddr_Addr`: Pointer to base memory address in DDR3.
`Draw_Offset`: Address offset for Draw Frame from base memory address
`Video_Offset`: Address offset for Video Frame from base memory address.
`Comp_Offset`: Address offset for Composite Frame from base memory address.
`hres`: Horizontal pixel resolution of frame.
`vres`: Vertical pixel resolution of frame.

**Software Usage Example:**

```
// Address Setup, offset is calculated based on resolution and pixel byte size
#define DF_OFFSET   0x0
#define VF_OFFSET   0x400000
#define CF_OFFSET   0x800000
#define HRES 640
#define VRES 480
Frame_Merge Merge_IP

// Define base DDR address and initialize
volatile u32 *ddr_addr = (volatile u32 *) XPAR_S6DDR_0_S0_AXI_BASEADDR;

// run merge
frame_merge_sw(ddr_Addr, DF_OFFSET, VF_OFFSET, CF_OFFSET, HRES,VRES);
```

# 7 Setting parameters

There are areas that need to be modified to change resolution or colour space settings: the HDMI Out IP core, and the C Program.

1. HDMI Out IP Core specifications → Click on the block in XPS to modify.
    a. HDMI HRES refers to the horizontal resolution. For example, 1280 refers to a resolution of 1280x720.
    b. HDMI NUM BYTES PER PIXEL refers to the colour format. 2 refers to RGB565 and 4 refers to RGB888x.
    c. Note that 640x480 and 800x600 need an input clock of 25MHz, while the 1280x720 resolution needs an input clock of 75MHz. The two projects have the correct clock setup. If you change the parameters, you may also need to change the software. Note that line stride is specified in pixels, so line stride may not need to change if only the colour format is different.
2. C program → Pass in valid values into software API call.
    a. HRES refers to the horizontal resolution
    b. VRES refers to the vertical resolution

# 8 Limitations

The following are limitations to the IP project:

- White is used as a reference, a white object will not be overlayed on the video frame.
- Since the HDMI Out IP core lacks support for reset, the reset button will not reset the project.

# 9 Reference Material

The following are reference links for the components used in the IP

- cvPaint - Simple drawing using camera (http://farshid.ws/projects.php?id=122)
- User Guide: HDMI Output peripheral (https://github.com/molohov/gpu2/tree/master/ip)

# 10 Contact

Refer to https://github.com/kming/ece532/tree/master/IP_Project for the most up to date version of the source code/documentation/example project

# 11 References

[1] A. Klimovic, B. Long, V. Zhang, “User Guide: HDMI output peripheral”

# 12 Appendix A - Hardware Frame Merge IP Documentation

## 12.0 Note

The follow sections are meant to address ongoing work on a separate hardware core for merging: frame_merge. As it is a work in progress, there are issues that are being worked on: Refer to 12.7 Limitations for outstanding issues. The IP core and API interface are subject to change with the continued development of the IP core.

## 12.1 Overview

The frame_merge IP is the hardware implementation of the frame merging algorithm. Provided along with the IP is an API in C which allows the user to interact with the IP core. Users can freely switch between the software or hardware algorithm with a slight change in the software. The sample project included contains the source file for the hardware core as well.

## 12.2 Directory Structure

The following directory structure indicates important files and folders which are of key interest to a developer using the example project provided.

- pcores\
    - frame_merge_v1_01_a
        - user_logic.v → Interface logic between AXI BUS and custom HDL
        - frame_merge_core.v → Custom HDL to merge frames
    - hdmi_out_v1_00_a
        - user_logic.v → Interface logic between AXI BUS and custom HDL
        - hdmi_core.v → Custom HDL to output HDMI in a configuration

## 12.3 Diagrams

The following diagrams provide a more in-depth view of the hardware IP.

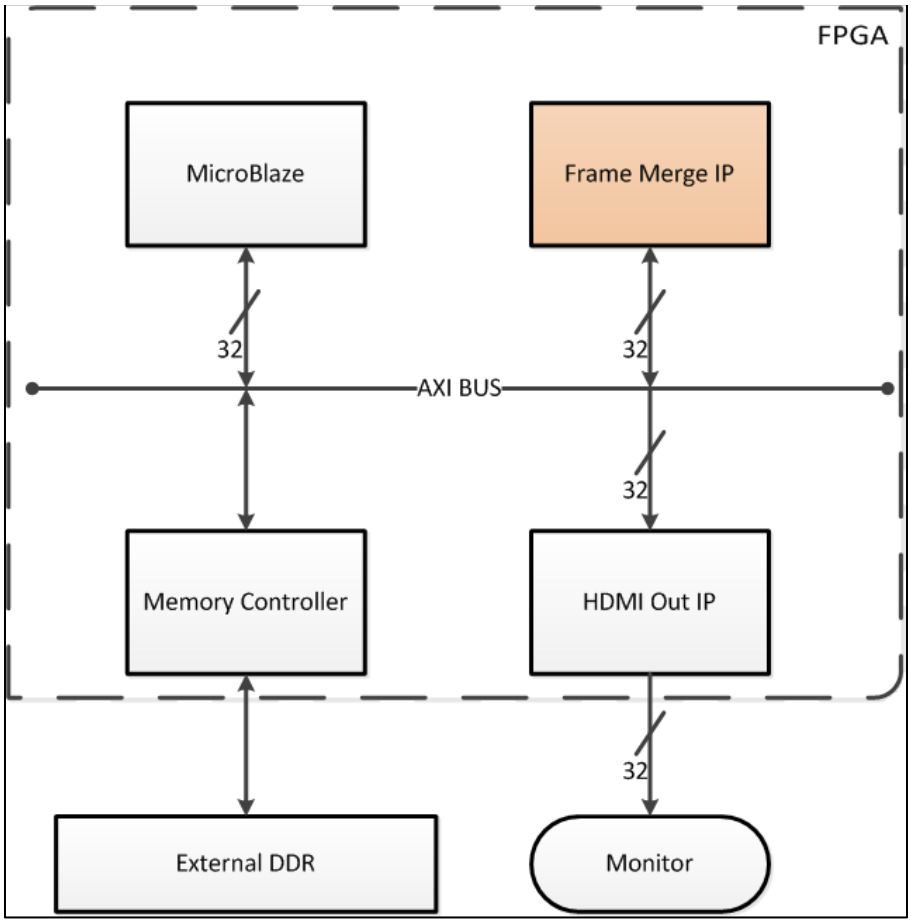


**Figure 12.1** shows the system architecture, highlighting the relationship between the different blocks and the frame merge IP

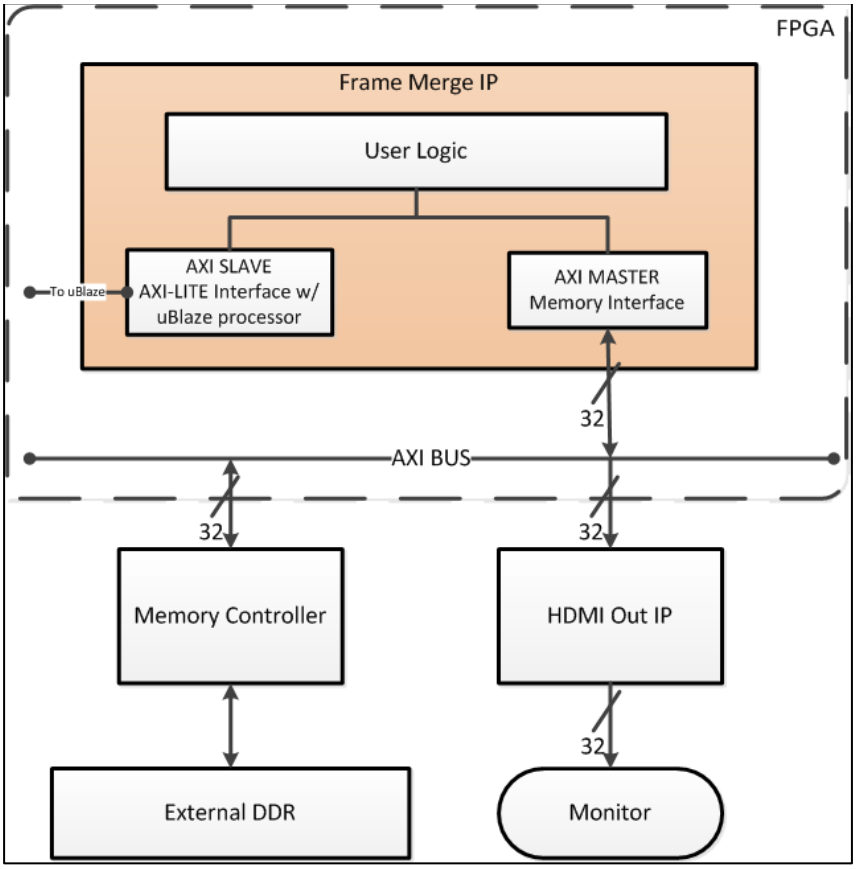


**Figure 12.2** shows the main blocks within the frame merge IP

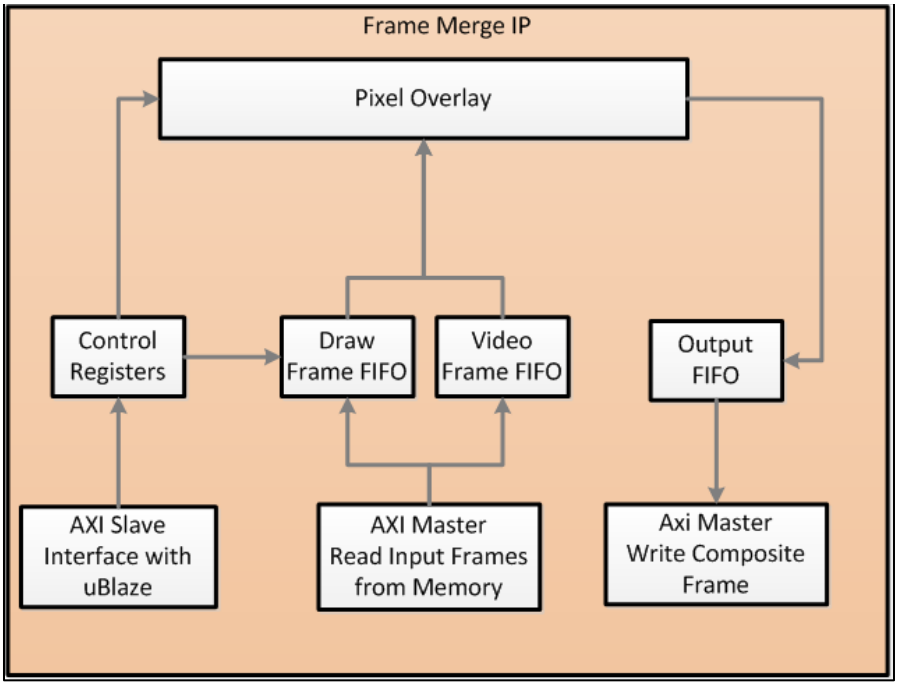


**Figure 12.3** shows the detailed implementation of the user logic that forms the IP.

## 12.5 Memory Mapped Registers

The registers map is abstracted away with the use of the Frame Merge API defined below, however, bare metal access to the core is available if needed.

Register 0: Go Signal → Used to tell the IP core to start processing.

Register 1: Input Draw Frame Address

Register 2: Input Video Frame Address

Register 3: Output Composite Frame Address

## 12.6 Frame Merge API and Usage

**Hardware API** → interfaces with the frame merge IP through AXI to handle frame merging.

```
void frame_merge_Initialize(Frame_Merge *InstancePtr, u32 Base_Addr)
void frame_merge_Go(Frame_Merge *InstancePtr)
void frame_merge_Stop(Frame_Merge *InstancePtr)
void frame_merge_SetAddr(Frame_Merge *InstancePtr, u32 Draw_Addr, u32 Video_Addr, u32
Comp_Addr)
```

### Parameters

`InstancePtr:` Pointer to Frame_Merge class.
`Base_Addr:` Address for frame_merge IP core
`Draw_Addr:` Address for Draw Frame in DDR3
`Video_Addr:` Address for VideoFrame in DDR3
`Comp_Addr:` Address for Composite Frame in DDR3

### Hardware Usage Example

```
// Address Setup, offset is calculated based on resolution and pixel byte size
#define DF_OFFSET   0x0
#define VF_OFFSET   0x400000
#define CF_OFFSET   0x800000

// Declare Frame Merge Class
Frame_Merge Merge_IP
```

```
// Define base DDR address and initialize
volatile u32 *ddr_addr = (volatile u32 *) XPAR_S6DDR_0_S0_AXI_BASEADDR;
frame_merge_Initialize (&Merge_IP, PAR_FRAME_MERGE_0_BASEADDR);

int df_addr = (int) ddr_addr + DF_OFFSET
int vf_addr = (int) ddr_addr + VF_OFFSET
int cf_addr = (int) ddr_addr + CF_OFFSET

// At this point, the address are configured, for the IP specified
frame_merge_SetAddr (&Merge_IP, df_addr, vf_addr, cf_addr)

// Can set the hardware block to start
frame_merge_Go (&Merge_IP)
```

## 12.7 Limitations

There are known issues with the frame merge API provided

- Software and hardware do not have checks to ensure they don't conflict. **Do not** use the software algorithm on the same addresses while the hardware is running.
- Merging will corrupt input frames Draw Frame and Video Frame. It will still apply the merge functionality as needed.