

# Video Pipeline for Computer Vision

sddec24-06 V-PIPE

Team members: Liam Janda, Taylor Johnson, Ritwesh Kumar, Deniz Tazegul

Client: JR Spidell

Advisor: Dr. Phillip Jones

# Problem Statement

- Video pipeline for computer vision system
- Off the shelf components/open source software
- Wide range of applications
- Inspired by improving the quality of life for wheelchair bound individuals with limited mobility



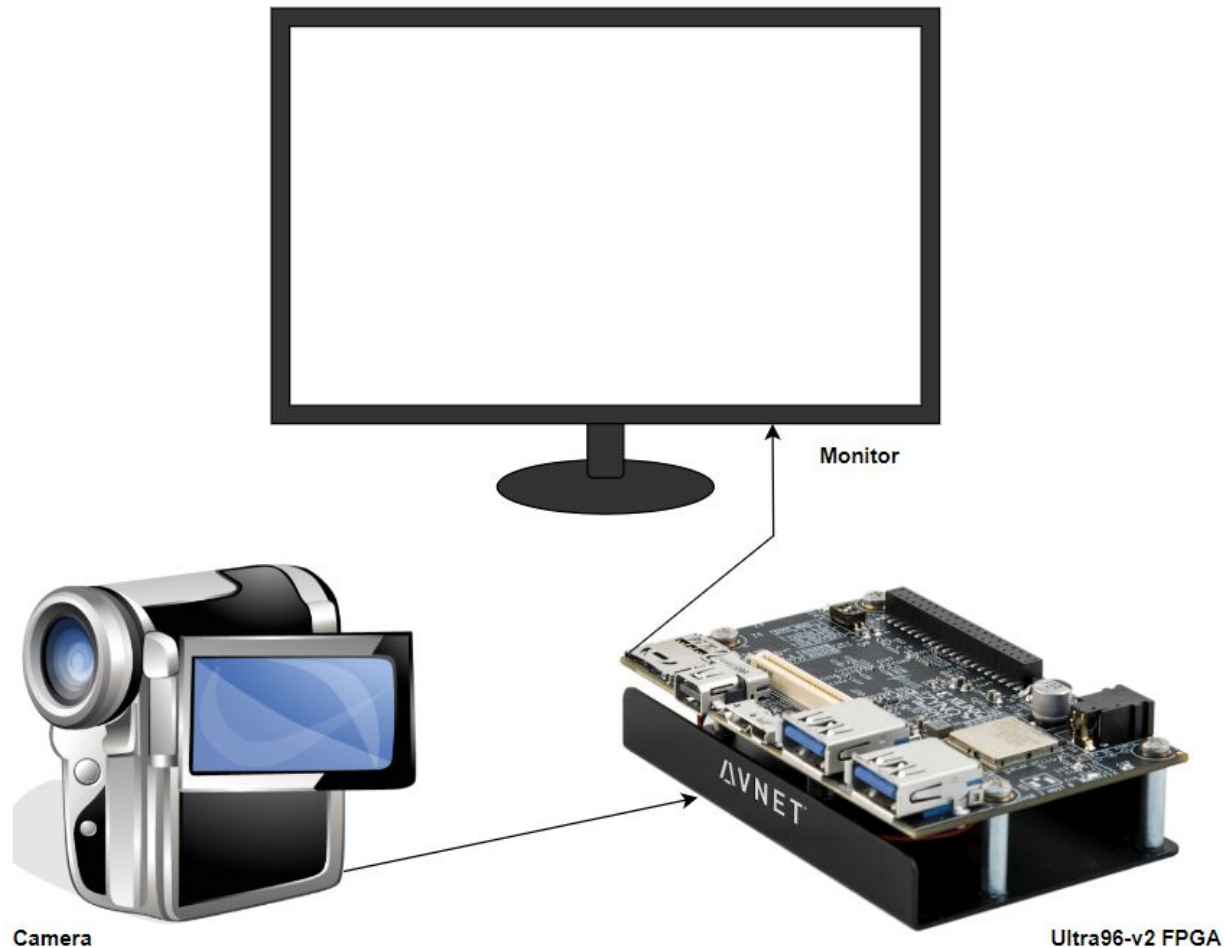
---

**V-PIPE**

**INNOVATE AT IOWA STATE**

# Project Plan

# Project Conceptual Sketch



# Project Overview

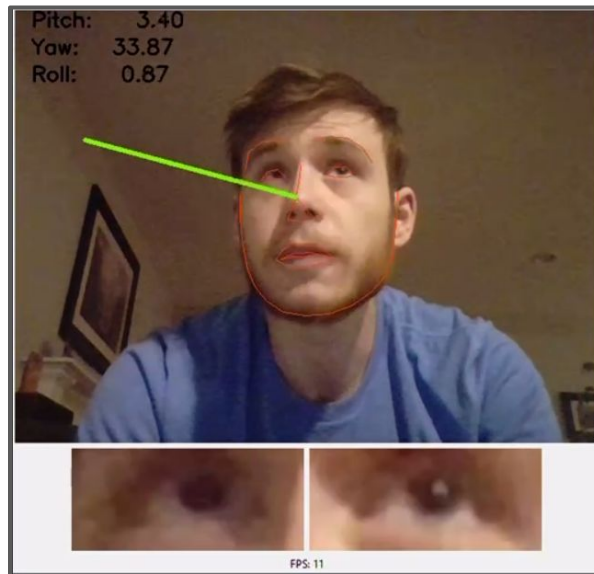
- Developing a FPGA-based video pipeline
  - MIPI-connected camera module
  - DisplayPort output
- Software executes in Ubuntu Linux OS
- Adapting previous team's implementation
- PYNQ libraries
- STRETCH GOAL: Pass video through a machine learning algorithm



**IMX219 Image Sensor**

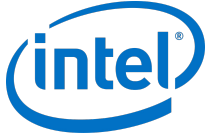


# Applications to Machine Learning (ML)

- Medical imaging
- Self-driving cars
- Search and rescue
- **Eye-tracking devices**



**Eye-Tracking Algorithm  
by a Previous SD Team**

# Market Research

Producer	Pros	Cons
V-PIPE (Ours)	Off-the-shelf hardware	Proper configuration
	Efficient	Specialized components with longer development time
	Attachable to existing wheelchairs	Niche market
	Easily configurable software	Requires own hardware

# Requirements

- **Constraints:**
  - Output video to a DisplayPort monitor in real-time
    - Latency video standards ~5-18 seconds
    - Real-time <100 ms
  - Route data using a Linux image in a PYNQ environment
  - Must execute using an **Ultra96-v2** FPGA board
- **Non-Constraints:**
  - Use a **Sony IMX219QH5-C** image sensor
  - **1920x1080p** standard Full HD resolution at **30 fps** standard live video format
  - Video stream in **RGB8** color format
  - Operate in a variety of lighting conditions



# Latency standards & Measurements

Type of Latency	Amount of Latency
Standard Broadcast Latency	5-18 seconds
Low Latency	1-5 seconds
Ultra-Low Latency	Less than 1 second
Real-Time Latency	Unperceivable to users

How will we measure the latency?

- Simple python script that takes the difference between the start and end of the data transmission

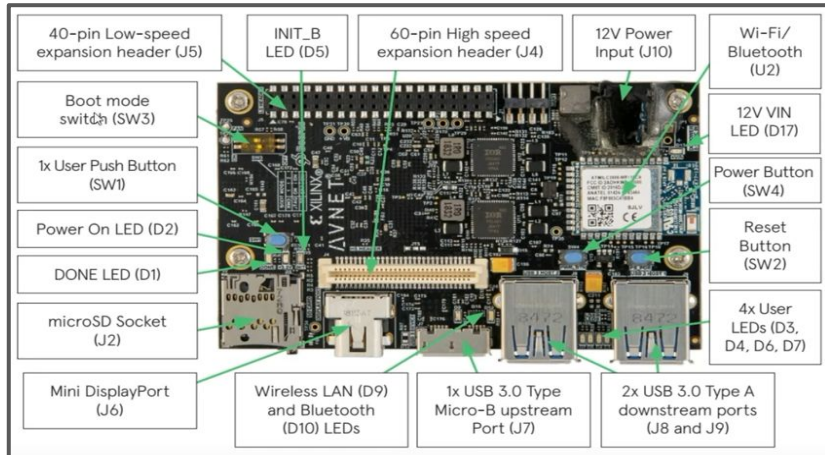
<https://www.pubnub.com/guides/whats-so-important-about-low-latency/>

# Engineering Standards

Engineering Standards	Relevance
IEEE Std 2977 2021	MIPI controller data conversion
IEEE Std 802.11 2016	Wifi connectivity: Jupyter Notebook server
I2C Protocol	Data communication with camera
AMBA AXI4 Protocol	Data transport within the FPGA

# System Design

# Technology Used



## Anatomy of Ultra96-v2 FPGA

### External project complexity

- Well-established technology
  - Ultra96-v2 FPGA made in 2018
  - IMX219 sensor made in 2016

### Internal project complexity

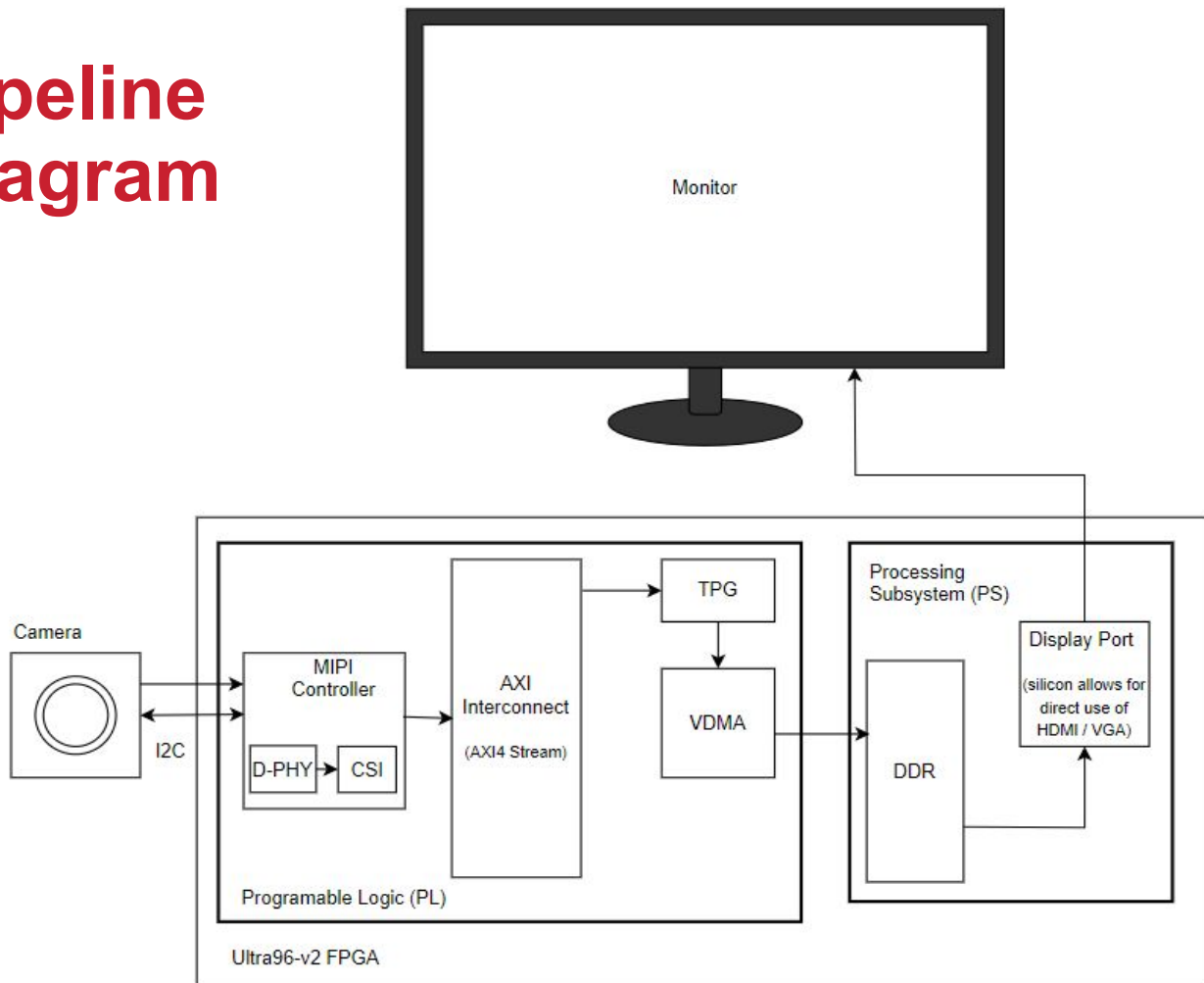
- Multiple hardware & software components & subsystems

# Hardware/Software

- **PYNQ** runs a Jupyter Notebook server
- **Non-PYNQ** runs on a terminal via TeraTerm
- **Vivado** custom hardware overlays
- **Debug:** ILA (Integrated Logic Analyzer) captures and analyzes I/O signals



# Video Pipeline Block Diagram



# Changes from Previous Teams


---

1. PYNQ implementation instead of shell scripts or C
2. IMX219 sensor instead of OV5647
3. 4 data lanes instead of 2
4. VDMA instead of frame buffer

# Test Plan

- Read from registers with known values
- Read from status registers
- Visual verification
  - DisplayPort to monitor
  - Jupyter Notebook
- Acceptance testing

```
frame_camera = cam_vdma.readchannel.readframe()  
frame_color=cv2.cvtColor(frame_camera,cv2.COLOR_BGR2RGB)  
pixels = np.array(frame_color)  
plt.imshow(pixels)  
plt.show()
```





# Potential Risks and Mitigation

---

- File/OS corruption mitigation
  - Backup SD cards
  - Shutdown Ultra96 using terminal
- Proper handling and storing of hardware



# Project Resource/Cost Estimate

Resource / Component	Cost
Budget	\$3,000
2 Ultra96-v2 FPGA Boards	2 * (\$300 to \$600) = \$600 to \$1200
IMX219/OV5647 Image Sensors	\$10 + \$20 = \$30
2 SD cards: 1 PYNQ and 1 non-PYNQ	\$15 + \$20 = \$35
(1 DisplayPort Cable + 2 USB Mini-to-USB A Cables + 1 Power Supply) * 2 sets	2 * (\$15 + \$10 * 2 + \$25) = \$120
(JTAG board + Ultra96-v2 Daughter Card) * 2 sets	2 * (\$40 + \$125) = \$330
<b>Total</b>	<b>\$1115 to \$1715</b>

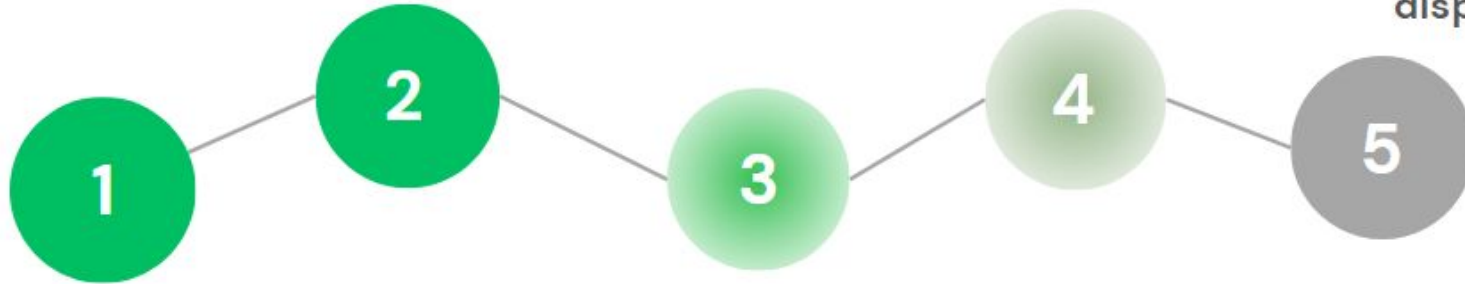
# Conclusion

# Project Milestones

Milestone #2  
Complete hardware  
setup and run  
existing code

Milestone #4  
Create software  
driver and test files to  
properly interface  
with the Ultra-96

Milestone #5  
Output the video  
stream to a  
display monitor



Milestone #1  
Illustrate & describe  
component  
connections

Milestone #3  
Design the video  
pipeline mapping  
inputs and outputs

# Current Project Status

---

- Developed understanding of architecture and component subsystems
- Tracking the inputs/outputs required for each component and the appropriate register values
- Setup hardware and run existing code
- Writing the code needed to implement our design

# Plan For Next Semester

---

- Transfer code to PYNQ environment
  - Configure image sensor and MIPI component
  - Configure board to output through DisplayPort
- Replace test pattern generator with VDMA to accept image sensor

# Team Member Responsibilities

---

## Team Members/Roles:

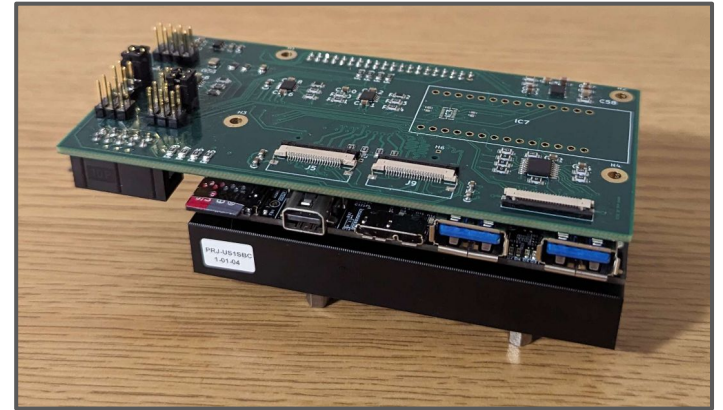
- Ritwesh Kumar - Image Sensor to MIPI Controller
- Deniz Tazegul - MIPI Controller to VDMA
- Liam Janda - VDMA to DDRM
- Taylor Johnson - DDRM to Output Display

**Questions?**



# Project Vocabulary

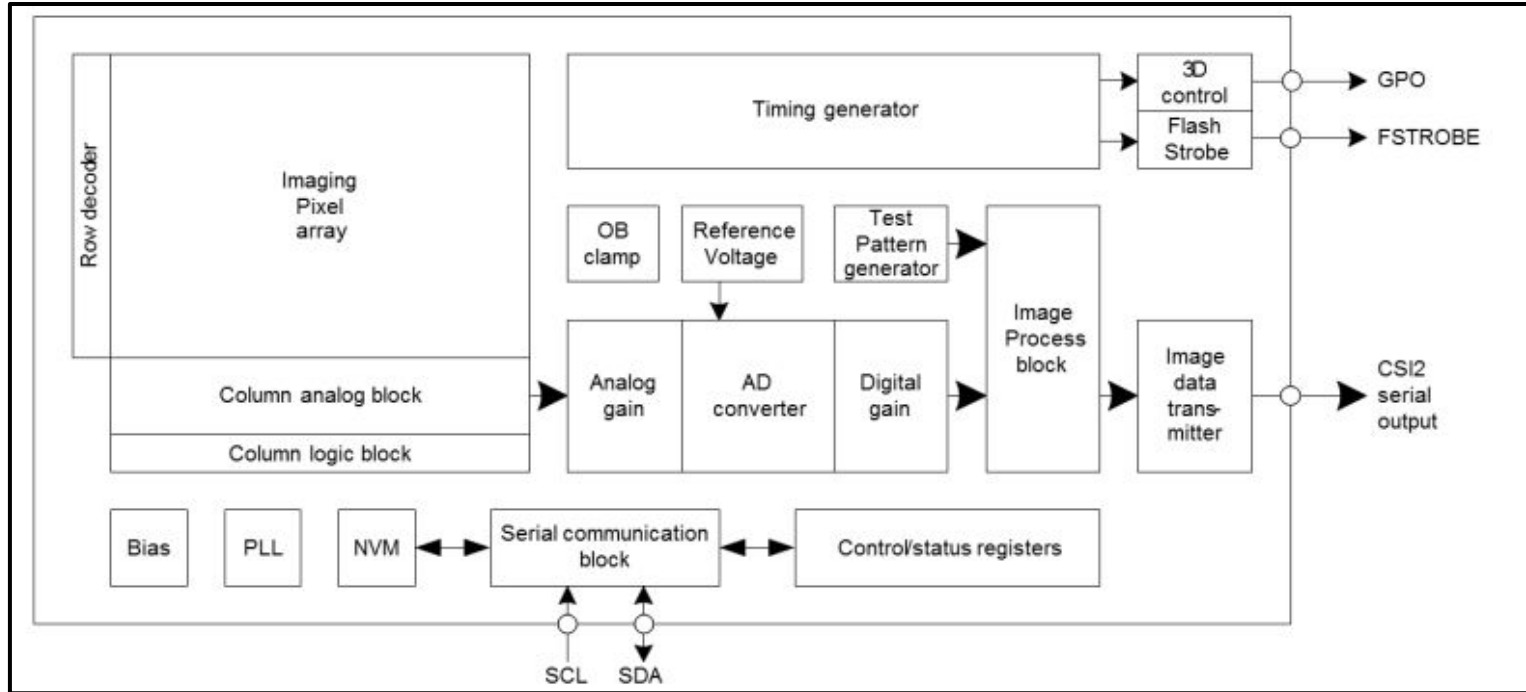
- **IMX219** image sensor: camera
- **MIPI**: mobile industry processor interface
- **CSI**: camera serial interface
- **D-PHY**: physical communication layer
- **VDMA**: video direct memory access
- **DDR**: double data rate (memory)
- **FPGA**: field programmable gate array
- **PYNQ**: python productivity for Zynq (python embedded systems developers)



**Ultra96-v2 FPGA Board**

# Backup Slides

# IMX219 Image Sensor



# RGB Bayer Filter & 1920x1080p

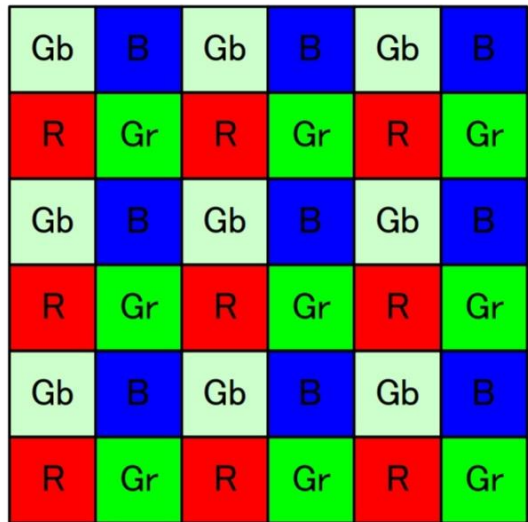
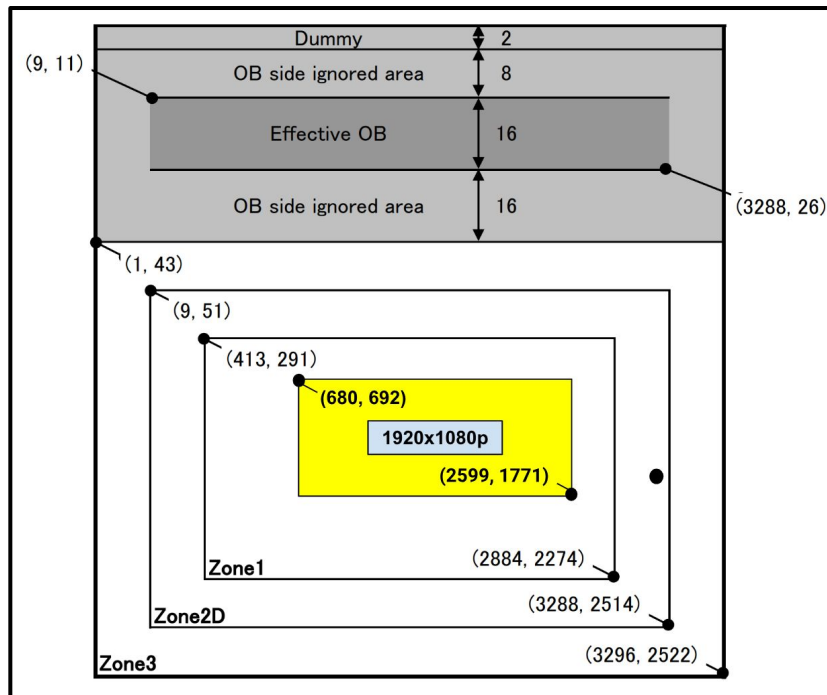
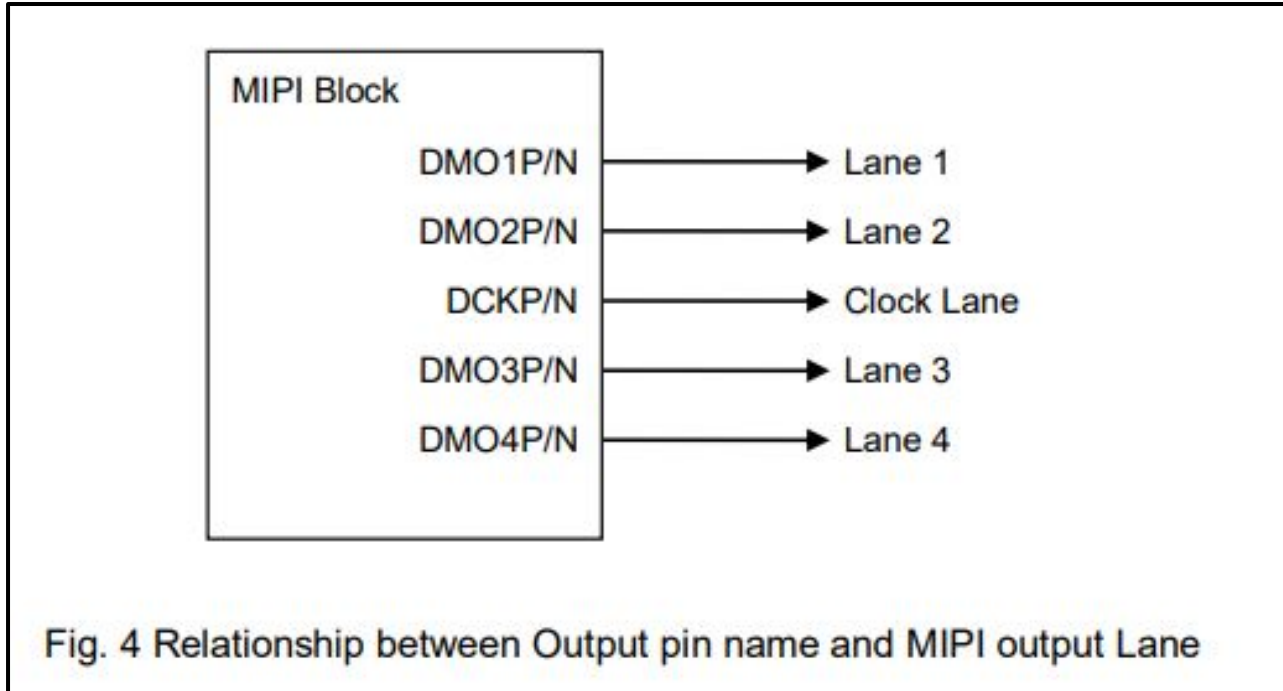


Fig. 47 Color coding alignment



# MIPI Transmit



# 4 Lane MIPI Transmit CSI-2

Table 12 Number of CSI lane Setting Registers

Index	Byte	Register Name	RW	Comment	Default (HEX)	Remark
0x0114	[1:0]	CSI_LANE_MODE	RW	03: 4Lane 01: 2Lane	03	Setting before "standby cancel"

◆ Data rate: Max. 755 Mbps/lane(@4lane), 912Mbps/Lane(@2lane)

# I/O Pixel and I/O System Clock (Code)

- Frame length: 256-65,534
  - **1113 in code**
- Line length: 3448-32752
  - **3448 in code**
- Min Line blanking: 168 (**IMX**)
- Min Frame blanking: 32 (**IMX**)

3-3-1-5 Frame Timing Parameter Limit Registers – [0x1140-0x114B]

Index	Byte	Register Name	RW	Comment	Re-Time	Default (HEX)	Embd DL
0x1140	[7:0]	min_frame_length_lines	RO	Minimum Frame Length allowed. Value both sensor dependent Format: 16-bit unsigned integer Units: Lines		01	
0x1141	[7:0]					00	
0x1142	[7:0]	max_frame_length_lines	RO	Maximum possible number of lines per Frame. Value sensor dependent Format: 16-bit unsigned integer Units: Lines		FF	
0x1143	[7:0]					FE	
0x1144	[7:0]	min_line_length_pck	RO	Minimum Line Length allowed. Value sensor dependent. But setup is possible at a smaller value * Format: 16-bit unsigned integer Units: Pixel Clock		0D	
0x1145	[7:0]					78	
0x1146	[7:0]	max_line_length_pck	RO	Maximum possible number of pixel clocks per line. Value sensor dependent Format: 16-bit unsigned integer Units: Pixel Clock		7F	
0x1147	[7:0]					F0	
0x1148	[7:0]	min_line_blanking_pck	RO	Minimum line blanking time in pixel clocks Format: 16-bit unsigned integer Units: Pixel Clock		00	
0x1149	[7:0]					A8	
0x114A	[7:0]	min_frame_blanking_lines	RO	Minimum frame blanking in video timing lines Format: 16-bit unsigned integer Units: Pixel Clock		00	
0x114B	[7:0]					20	

\* possible to setup up to D60 (HEX)

# Datasheet Clock Setting Examples

Interface	CSI-2	CSI-2	CSI-2
ADC bit width	10	10	10
Data Mode	Raw10	Raw10	Raw10
FPS	30 frame/s	180 frame/s	21 frame/s (*1)
Lanes	4	4	2
Bin	Full-Pel	2 (V) X2 (H) analog (special) binning	Full-Pel
Sub			
Output Size (H, V)	3280 x 2464	1408 x 792	3280 x 2464
pll1_vt_freq	702 MHz	702 MHz	456 MHz
EXCK_FREQ	12	12	12
VTSYCK_DIV	1	1	1
vt_pix_clk_div	5	5	10
Pix Rate	280.8M pix	280.8M pix	182.4M pix
Actual freq (VTCK)	140.4 MHz	140.4 MHz	91.2 MHz
pll2_op_freq	726 MHz	726 MHz	912 MHz
op_sys_clk_div	1	1	1
op_pix_clk_div	10	10	10
de-rating	1	1	1
Output Lanes	4	4	2
Actual freq (OPCK)	72.6 MHz	72.6 MHz	91.2 MHz
Actual MIPI-freq (byte)	90.75 MHz	90.75 MHz	114 MHz
Speed/Lane (Ch)	726 Mbps	726 Mbps	912 Mbps
Total Output Rate	2.904 Gbps	2.904 Gbps	1.824 Gbps



# Clock Diagram

- Max data transmission = (bits per CSI lane \* # of CSI lanes) / Data rate
- Data rate = length \* height \* frame rate \* bit pixel depth
- Bit pixel depth = bits per pixel \* # of channel => RGB = 8 bits \* 3 channels = 24
- => (755 Mbps \* 4 lanes) / (1920 \* 1080 \* 24) means that **3.02 Gbps** data rate and **60.68 fps** are possible for 4 Lane MIPI Tx at **1920x1080p**

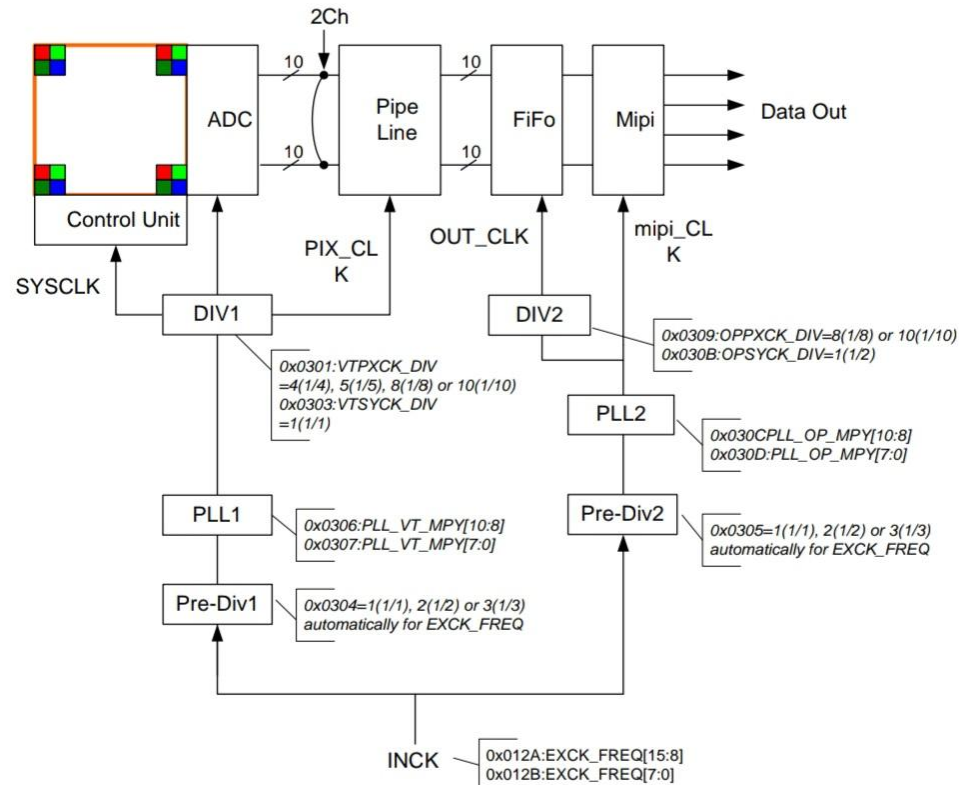


Fig. 43 Clock System Block Diagram

◆ Data rate: Max. 755 Mbps/lane(@4lane), 912Mbps/Lane(@2lane)

# Frame Rate

## 5-5 Frame Rate Calculation Formula

Frame rate in all-pixel scan mode is calculated by the followings.

$$\text{Frame\_Rate[frame/s]} = \frac{1}{\text{Time\_per\_Line[sec]} \times (\text{Frame\_Length})}$$

$$\text{Time\_Per\_Line\_}[sec] = \frac{\text{Line\_Length\_pck[pix]}}{2 \times \text{Pix\_Clock\_Freq[MHz]}}$$

# Min/max MIPI-freq

0x116C	[7:0]	min_op_pix_clk_freq_mhz	RO	Minimum output pixel clock frequency Format: IEEE 32-bit float Units: MHz 20 MHz
0x116D	[7:0]			
0x116E	[7:0]			
0x116F	[7:0]			
0x1170	[7:0]	max_op_pix_clk_freq_mhz	RO	Maximum output pixel clock frequency Format: IEEE 32-bit float Units: MHz 114.5 MHz
0x1171	[7:0]			
0x1172	[7:0]			
0x1173	[7:0]			

# MIPI Freq (Google IMX Driver Code)

- **Design metric:** 30 fps live video industry standard

Frame rate = 47.5 < 60.68 fps limit, frame length = 1113, line length = 3448

- **Verify**

=> Time per line =  $1 / (\text{Frame rate} * \text{Frame length}) = \mathbf{18915 \text{ ns}}$  (matches IMX code)

=> MIPI-freq (bytes) =  $\text{Line length} / (2 * \text{Time per line}) = \mathbf{91.15 \text{ MHz}}$

Speed =  $\text{MIPI-freq} * 8 * 4 \text{ lanes} = \mathbf{2.92 \text{ Gbps}} < 3.02 \text{ Gbps}$  4 Lane max data rate

# Clock Speeds (Code)

- EXCK (external clock) frequency: 6-27 MHz, **24 MHz in code**
- PLL input clock frequency: 6-27 MHz, **24-27 MHz in code**
- PLL multiplier: 8-2047, **57 in code for PLL1 and 114 in code for PLL2**
- PLL output clock frequency: 432-916 MHz
  - 432 Mhz < 729.15 MHz in code < 916 Mhz**

3-3-1-3 Pre-PLL and PLL Clock Set-up Capability Registers – [0x1100-0x111F]

Index	Byte	Register Name	RW	Comment	Re-Time	Default (HEX)	Embd DL
0x1100	[7:0]	min_ext_clk_freq_mhz	RO	Minimum external clock frequency Format: IEEE 32-bit float Units: MHz 6 MHz (= min_ext_clk_freq_mhz)		40	
0x1101	[7:0]					C0	
0x1102	[7:0]					00	
0x1103	[7:0]					00	
0x1104	[7:0]	max_ext_clk_freq_mhz	RO	Maximum external clock frequency Format: IEEE 32-bit float Units: MHz 27 MHz (= max_ext_clk_freq_mhz)		41	
0x1105	[7:0]					D8	
0x1106	[7:0]					00	
0x1107	[7:0]					00	
0x1108	[7:0]	min_pre_pll_clk_div	RO	Minimum Pre PLL divider value Format: 16-bit unsigned integer		01	
0x1109	[7:0]					00	
0x110A	[7:0]	max_pre_pll_clk_div	RO	Maximum Pre PLL divider value Format: 16-bit unsigned integer		00	
0x110B	[7:0]					0D	
0x110C	[7:0]	min_pll_ip_freq_mhz	RO	Minimum PLL input clock frequency Format: IEEE 32-bit float Units: MHz 6 MHz		40	
0x110D	[7:0]					C0	
0x110E	[7:0]					00	
0x110F	[7:0]					00	
0x1110	[7:0]	max_pll_ip_freq_mhz	RO	Maximum PLL input clock frequency Format: IEEE 32-bit float Units: MHz 27 MHz (= max_ext_clk_freq_mhz)		41	
0x1111	[7:0]					D8	
0x1112	[7:0]					00	
0x1113	[7:0]					00	
0x1114	[7:0]	min_pll_multiplier	RO	Minimum PLL multiplier Format: 16-bit unsigned integer		00	
0x1115	[7:0]					08	
0x1116	[7:0]	max_pll_multiplier	RO	Maximum PLL Multiplier Format: 16-bit unsigned integer		07	
0x1117	[7:0]					FF	
0x1118	[7:0]	min_pll_op_freq_mhz	RO	Minimum PLL output clock frequency Format: IEEE 32-bit float Units: MHz 432 MHz		43	
0x1119	[7:0]					D8	
0x111A	[7:0]					00	
0x111B	[7:0]					00	
0x111C	[7:0]	max_pll_op_freq_mhz	RO	Maximum PLL output clock frequency Format: IEEE 32-bit float Units: MHz 916 MHz		44	
0x111D	[7:0]					65	
0x111E	[7:0]					00	
0x111F	[7:0]					00	

# Output Freq / PLL2\_OP Freq (Code)

- Output pixel clock frequency (OPCK) = MIPI-freq (bytes) \* 8 / PLL2 multiplier (**10 in code**)

=> OPCK = **72.915 MHz**

=> OPCK is in range: 20 MHz min < **72.915 MHz** < 114.5 MHz max

- PLL2 Output Clock Frequency (PLL2\_OP) = OPCK \* PLL2 multiplier (**10 in code**)

=> PLL2\_OP = **729.15 MHz**

=> PLL2\_OP is in range: 432 Mhz < **729.15 MHz in code** < 916 Mhz

# Clock Division Input (Code)

- System clock division: 1-2, **1 in code**
- Video timing system clock frequency: 200-700 MHz, **700 MHz in code**
- Video timing pixel clock frequency: 80-140 MHz, **140 MHz in code**
- Video timing pixel clock division: 5, **5 in code**

3-3-1-4 Read Domain Clock Set-up Capability Registers – [0x1120-0x1137]

Index	Byte	Register Name	RW	Comment	Re-Time	Default (HEX)	Embd DL
0x1120	[7:0]	min_vt_sys_clk_div	RO	Minimum video timing system clock divider value Format: 16-bit unsigned integer		00	
0x1121	[7:0]					01	
0x1122	[7:0]	max_vt_sys_clk_div	RO	Maximum video timing system clock divider value Format: 16-bit unsigned integer		00	
0x1123	[7:0]					02	
0x1124	[7:0]	min_vt_sys_clk_freq_mhz	RO	Minimum video timing system clock frequency Format: IEEE 32-bit float Units: MHz 200 MHz		43	
0x1125	[7:0]					48	
0x1126	[7:0]					00	
0x1127	[7:0]					00	
0x1128	[7:0]	max_vt_sys_clk_freq_mhz	RO	Maximum video timing system clock frequency Format: IEEE 32-bit float Units: MHz 700 MHz		44	
0x1129	[7:0]					2F	
0x112A	[7:0]					00	
0x112B	[7:0]					00	
0x112C	[7:0]	min_vt_pix_clk_freq_mhz	RO	Minimum video timing pixel clock frequency Format: IEEE 32-bit float Units: MHz 80 MHz		42	
0x112D	[7:0]					A0	
0x112E	[7:0]					00	
0x112F	[7:0]	max_vt_pix_clk_freq_mhz	RO	Maximum video timing pixel clock frequency Format: IEEE 32-bit float Units: MHz 140 MHz		00	
0x1130	[7:0]					43	
0x1131	[7:0]					0C	
0x1132	[7:0]					00	
0x1133	[7:0]	min_vt_pix_clk_div	RO	Minimum video timing pixel clock divider value Format: 16-bit unsigned integer		00	
0x1135	[7:0]					05	
0x1136	[7:0]	max_vt_pix_clk_div	RO	Maximum video timing pixel clock divider value Format: 16-bit unsigned integer		00	
0x1137	[7:0]					05	

# Clock Division Output (Code)

- Output system clock division: 1-2, **1 in code**
- Output system clock frequency: 200-916 MHz, **729.15 MHz in code**
- Output pixel clock frequency: 20-114.5 MHz, **72.915 MHz in code**
- Output pixel clock divider: 8-10, **10 in code**

3-3-1-6 Output Clock Set-up Capability Registers – [0x1160-0x1177]

Index	Byte	Register Name	RW	Comment	Re-Time	Default (HEX)	Embd DL
0x1160	[7:0]	min_op_sys_clk_div	RO	Minimum output system clock divider value Format: 16-bit unsigned integer		00	
0x1161	[7:0]					01	
0x1162	[7:0]	max_op_sys_clk_div	RO	Maximum output system clock divider value Format: 16-bit unsigned integer		00	
0x1163	[7:0]					02	
0x1164	[7:0]	min_op_sys_clk_freq_mhz	RO	Minimum output system clock frequency Format: IEEE 32-bit float Units: MHz 200 MHz		43	
0x1165	[7:0]					48	
0x1166	[7:0]					00	
0x1167	[7:0]					00	
0x1168	[7:0]	max_op_sys_clk_freq_mhz	RO	Maximum output system clock frequency Format: IEEE 32-bit float Units: MHz 916 MHz		44	
0x1169	[7:0]					65	
0x116A	[7:0]					20	
0x116B	[7:0]					00	
0x116C	[7:0]	min_op_pix_clk_freq_mhz	RO	Minimum output pixel clock frequency Format: IEEE 32-bit float Units: MHz 20 MHz		41	
0x116D	[7:0]					A0	
0x116E	[7:0]					00	
0x116F	[7:0]					00	
0x1170	[7:0]	max_op_pix_clk_freq_mhz	RO	Maximum output pixel clock frequency Format: IEEE 32-bit float Units: MHz 114.5 MHz		42	
0x1171	[7:0]					E5	
0x1172	[7:0]					00	
0x1173	[7:0]					00	
0x1174	[7:0]	min_op_pix_clk_div	RO	Minimum output pixel clock divider value Format: 16-bit unsigned integer		00	
0x1175	[7:0]					08	
0x1176	[7:0]	max_op_pix_clk_div	RO	Maximum output pixel clock divider value Format: 16-bit unsigned integer		00	
0x1177	[7:0]					0A	



# Input Freq / PLL1 Freq (in Datasheet)

- **280 Mbps** Pix Rate for 4 Lanes

◆ Pixel rate: 280 [Mpixel/s] (All-pixels mode)

- => **140 MHz** Video Timing Clock Frequency (VTCK) from datasheet
- PLL1 Freq = VTCK \* PLL1 multiplier (**5 in code**) = **700 MHz**

# PLL1 Pix Rate vs PLL2 Data Rate

- 1 If, Pix Rate of PLL1 domain < Data Rate of PLL2 domain, data is always correctly output from the sensor
- 2 If Pix Rate of PLL1 domain > Data rate of PLL2 domain, Else If de-rating (binning and sub-sampling without resize), FiFo can handle.

- PLL1 Pix Rate (**280 Mbps**) < PLL2 Data Rate (**2.92 Gbps**)

=> “Data is always correctly output from the sensor”

# DisplayPort

