1) LVDS diagram

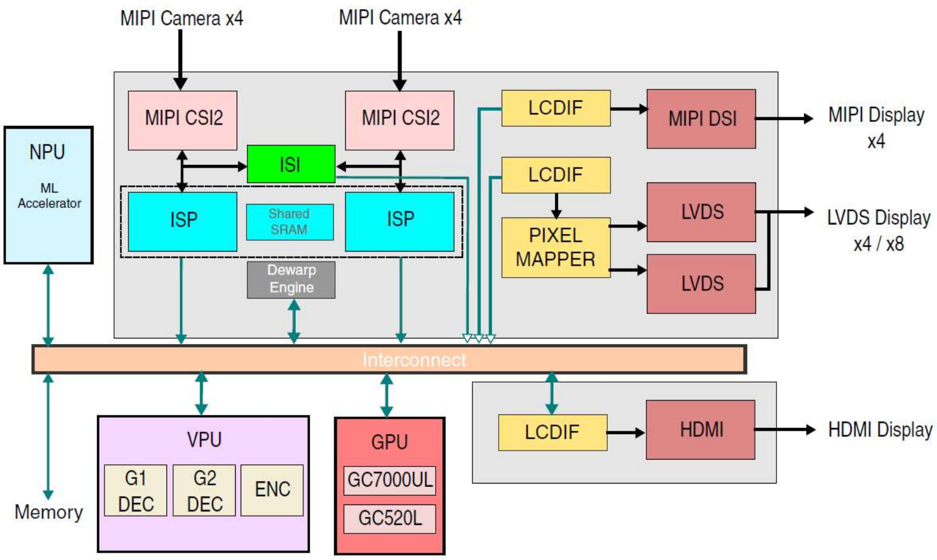


Figure 13-1. Display, Imaging, Camera I/F Diagram

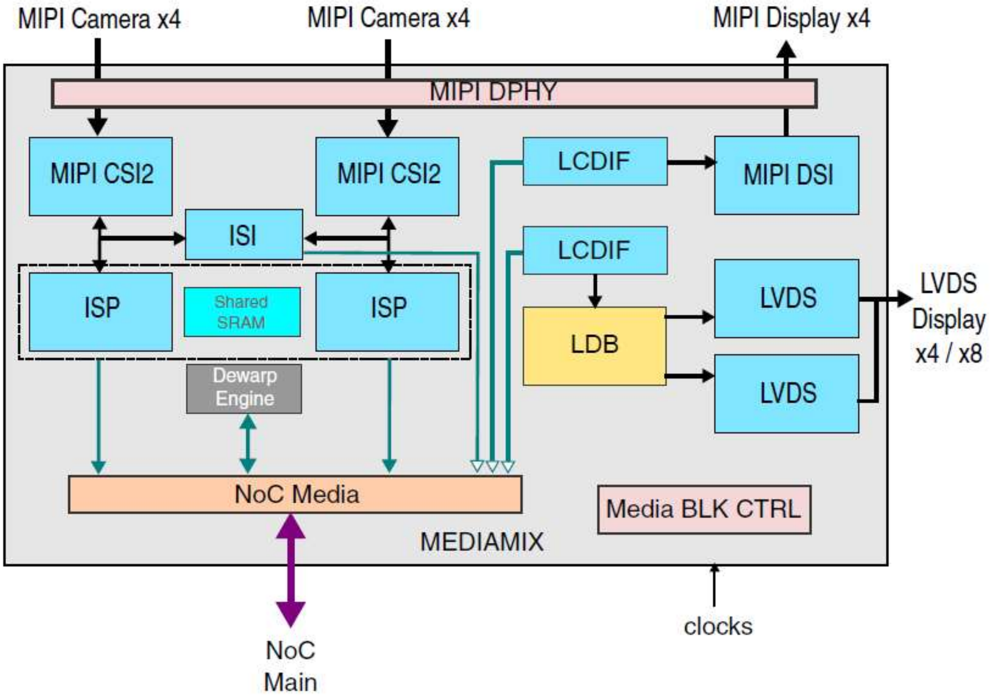


Figure 13-4. MEDIAMIX Block Diagram

The chip supports LVDS Tx display and Pixel Mapper, The Pixel Mapper splits and reorders the pixels from the single LCDIF display output into an odd and even pixel stream

LDB supports flow of synchronous RGB data to external display devices through the LVDS interface.

13.8.1.1 Block Diagram

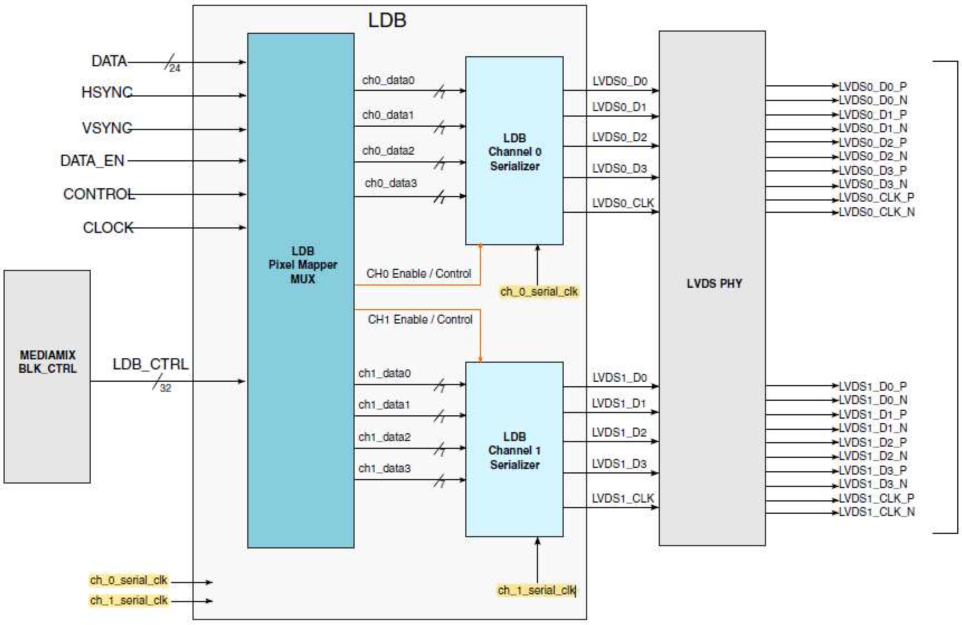


Figure 13-70. LDB Block Diagram

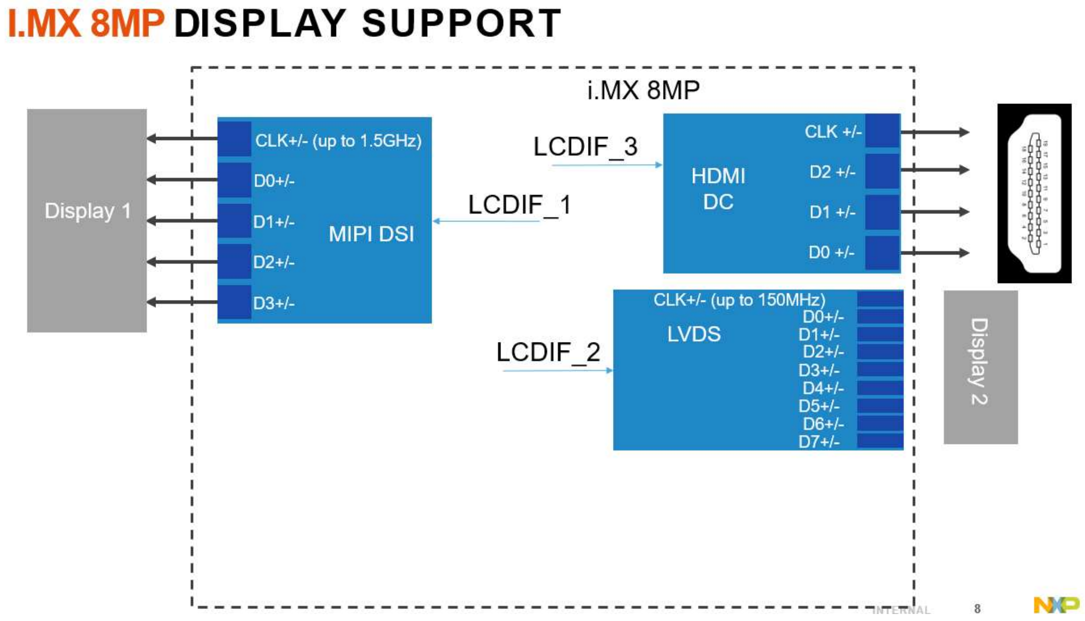
The chip supports LVDS Tx display and Pixel Mapper. The LVDS port can be used as follows:

- Single channel (4 lanes) output at up to 80MHz pixel clock and LVDS clock. This supports resolutions up to 1366x768p60.
- For 4-lane LVDS, channel 0 or channel 1 can be used.
- Dual asynchronous channels (8 data, 2 clocks). This is intended for a single panel with two interfaces, transferring across two channels (even pixel/odd pixel). This is supported at up to 160MHz pixel clock, which is up to 80MHz LVDS clock (due to 2 pixels per LVDS clock). This supports resolutions above 1366x768p60, up to

1080p60.

Refer to the imx8mp.dtsi file, imx8mp display supports three lcdif, lcdif1 is for mipi dsi, lcdif2 is for lvds and lcdif3 is for hdmi

```
display-subsystem {
                compatible = "fsl,imx-display-subsystem";
                ports = <&lcdif1_disp>,
                        <&lcdif2_disp>,
                        <&lcdif3_disp>;
};
```


I.MX 8MP DISPLAY SUPPORT

2)LVDS CLOCK

For lvds, current bsp use IMX8MP_CLK_MEDIA_LDB's parents IMX8MP_VIDEO_PLL1_OUT as source

```
ldb: ldb@32ec005c {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "fsl,imx8mp-ldb";
        clocks = <&clk IMX8MP_CLK_MEDIA_LDB_ROOT>;
        clock-names = "ldb";
        assigned-clocks = <&clk IMX8MP_CLK_MEDIA_LDB>;
        assigned-clock-parents = <&clk IMX8MP_VIDEO_PLL1_OUT>;
        gpr = <&media_blk_ctrl>;
        power-domains = <&mediamix_pd>;
        status = "disabled";

        lvds-channel@0 {
                #address-cells = <1>;
                #size-cells = <0>;
                reg = <0>;
                phys = <&ldb_phy1>;
                phy-names = "ldb_phy";
                status = "disabled";

                port@0 {
                        reg = <0>;

                        ldb_ch0: endpoint {
                                remote-endpoint = <&lcdif2_disp_ldb_ch0>;
                        };
                };
        };
```

Check the imx8mp clock driver clk-imx8mp.c,

```
hws[IMX8MP_VIDEO_PLL1] = imx_clk_hw_pll14xx("video_pll1", "video_pll1_ref_sel",
anatop_base + 0x28, &imx_1443x_pll);
```

the video_pll1 is defined in the imx_pll1443_tbl of clk-pll14xx.c

```
static const struct imx_pll14xx_rate_table imx_pll1443x_tbl[] = {
        PLL_1443X_RATE(1039500000U, 173, 2, 1, 16384),
        PLL_1443X_RATE(650000000U, 325, 3, 2, 0),
        PLL_1443X_RATE(594000000U, 198, 2, 2, 0),
        PLL_1443X_RATE(519750000U, 173, 2, 2, 16384),
        PLL_1443X_RATE(393216000U, 262, 2, 3, 9437),
        PLL_1443X_RATE(361267200U, 361, 3, 3, 17511),
};
```

Check the ldb driver in the ldb, the path is drivers/gpu/drm/imx/imx8mp-ldb.c

current bsp define the limited clock for lvds

```
	/*
	 * Due to limited video PLL frequency points on i.MX8mp,
	 * we do mode fixup here in case any mode is unsupported.
	 */
	if (ldb->dual)
		mode->clock = mode->clock > 100000 ? 148500 : 74250;
	else
		mode->clock = 74250;
```

current bsp fix the pixel clock up to the 74.25Mhz as single lvds channel and 148.5Mhz as dual lvds channel

```
serial_clk = mode->clock * (ldb->dual ? 3500UL : 7000UL);
	clk_set_rate(imx8mp_ldb->clk_root, serial_clk);
```

refer to the driver, the serial clock is the video pll1 clock and the formula is

video pll1 clock/serial clock = 7 x pixel clock (single channel)

video pll1 clock/serial clock = 3.5 x pixel clock (dual channel)

refer to this formula, current bsp uses 519.75Mhz as video pll1 frequency

```
PLL_1443X_RATE(519750000U, 173, 2, 2, 16384),
```

3) Different lvds clock support

Current bsp couldn’t support any clock display customer wants, and fix up the pixel clock to the 148.5Mhz for dual channel and 74.25Mhz for single channelbut if customer doesn’t want to use 74.25 for single channel clock, they can comment the below driver in the bsp

```
/*
         * Due to limited video PLL frequency points on i.MX8mp,
         * we do mode fixup here in case any mode is unsupported.
         */
        if (ldb->dual)
                mode->clock = mode->clock > 100000 ? 148500 : 74250;
        else
                mode->clock = 74250;
```

but even customer comment this, current bsp still couldn’t support any pixel clock they want, because the video pll1 is limited, The video pll1 only can be chosen from this table

```
static const struct imx_pll14xx_rate_table imx_pll1443x_tbl[] = {
        PLL_1443X_RATE(1039500000U, 173, 2, 1, 16384),
        PLL_1443X_RATE(650000000U, 325, 3, 2, 0),
        PLL_1443X_RATE(594000000U, 198, 2, 2, 0),
        PLL_1443X_RATE(519750000U, 173, 2, 2, 16384),
        PLL_1443X_RATE(393216000U, 262, 2, 3, 9437),
        PLL_1443X_RATE(361267200U, 361, 3, 3, 17511),
};
```

Maybe customer wants supports other pixel clock beside of these ones, then they can add new clock in the table, how to add these, and what pixel clock the video pll1 can support it, pls Refer to the chapter 5.1.5.4.4 SSCG and Fractional PLLs of reference manual, video pll1 is fractional pll

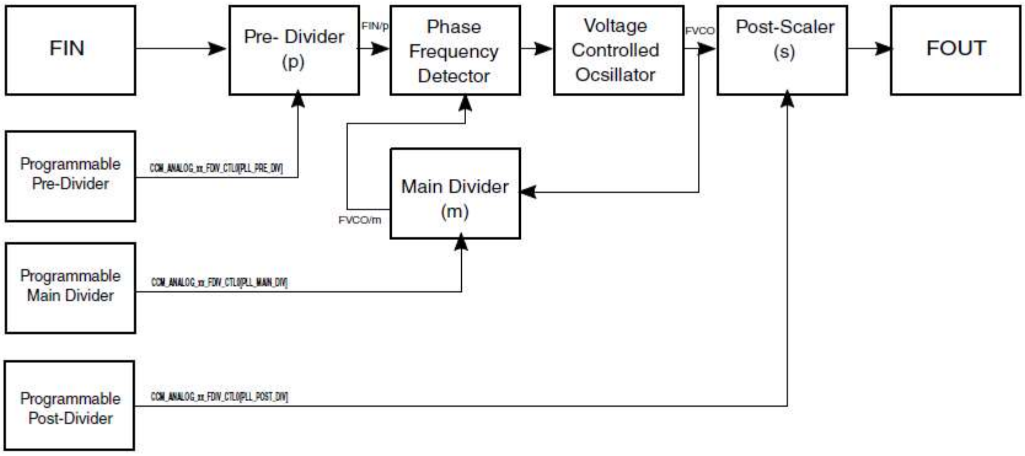

FIN
Pre- Divider (p)
FIN/p
Phase Frequency Detector
Voltage Controlled Ocsillator
FVCO
Post-Scaler (s)
FOUT
Programmable Pre-Divider
CCM_ANALOG_xx_FDIV_CTL0[PLL_PRE_DIV]
Main Divider (m)
FVCO/m
Programmable Main Divider
CCM_ANALOG_xx_FDIV_CTL0[PLL_MAIN_DIV]
Programmable Post-Divider
CCM_ANALOG_xx_FDIV_CTL0[PLL_POST_DIV]

Figure 5-9. Fractional PLL Block Diagram

Formula for Fraction PLLOUT:

- FOUT=((m + k/65536) × FIN) / (p × 2^s)
- Where, $1 \le p \le 63$, $64 \le m \le 1023$, $0 \le s \le 6$, $-32768 \le k \le 32767$

- p, m, and s are unsigned integers. k is a two's complement integer.
- Where, FOUT is the output frequency, FIN is the input frequency, and p,m,s and k are division values for pre-divider, main divider, scaler and DSM respectively

```
static const struct imx_pll14xx_rate_table imx_pll1443x_tbl[] = {
        PLL_1443X_RATE(1039500000U, 173, 2, 1, 16384),
        PLL_1443X_RATE(650000000U, 325, 3, 2, 0),
        PLL_1443X_RATE(594000000U, 198, 2, 2, 0),
        PLL_1443X_RATE(519750000U, 173, 2, 2, 16384),
        PLL_1443X_RATE(393216000U, 262, 2, 3, 9437),
        PLL_1443X_RATE(361267200U, 361, 3, 3, 17511),
};

#define PLL_1443X_RATE(_rate, _m, _p, _s, _k)           \
        {                                               \
                .rate   =       (_rate),        \
                .mdiv   =       (_m),                   \
                .pdiv   =       (_p),                   \
                .sdiv   =       (_s),                   \
                .kdiv   =       (_k),                   \
        }
```

For example *1039500000U*, here FIN is 24M as input frequency

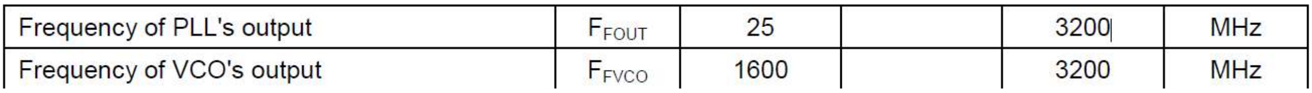
FOUT=((m + k/65536) × FIN) / (p × 2s)=((173 + 16384 / 65536) x 24) / (2 x

2)=1039.5

Refer to the PLL spec, besides of Fout, still need consider the Fvco

FFVCO = ((m+k/65536) x FFIN) / p

Let me remind, new pll output and vco output needs to meet the range as below



Frequency of PLL's output	F_{FOUT}	25		3200	MHz
Frequency of VCO's output	F_{FVCO}	1600		3200	MHz