

NIOS Pixel Display - SW

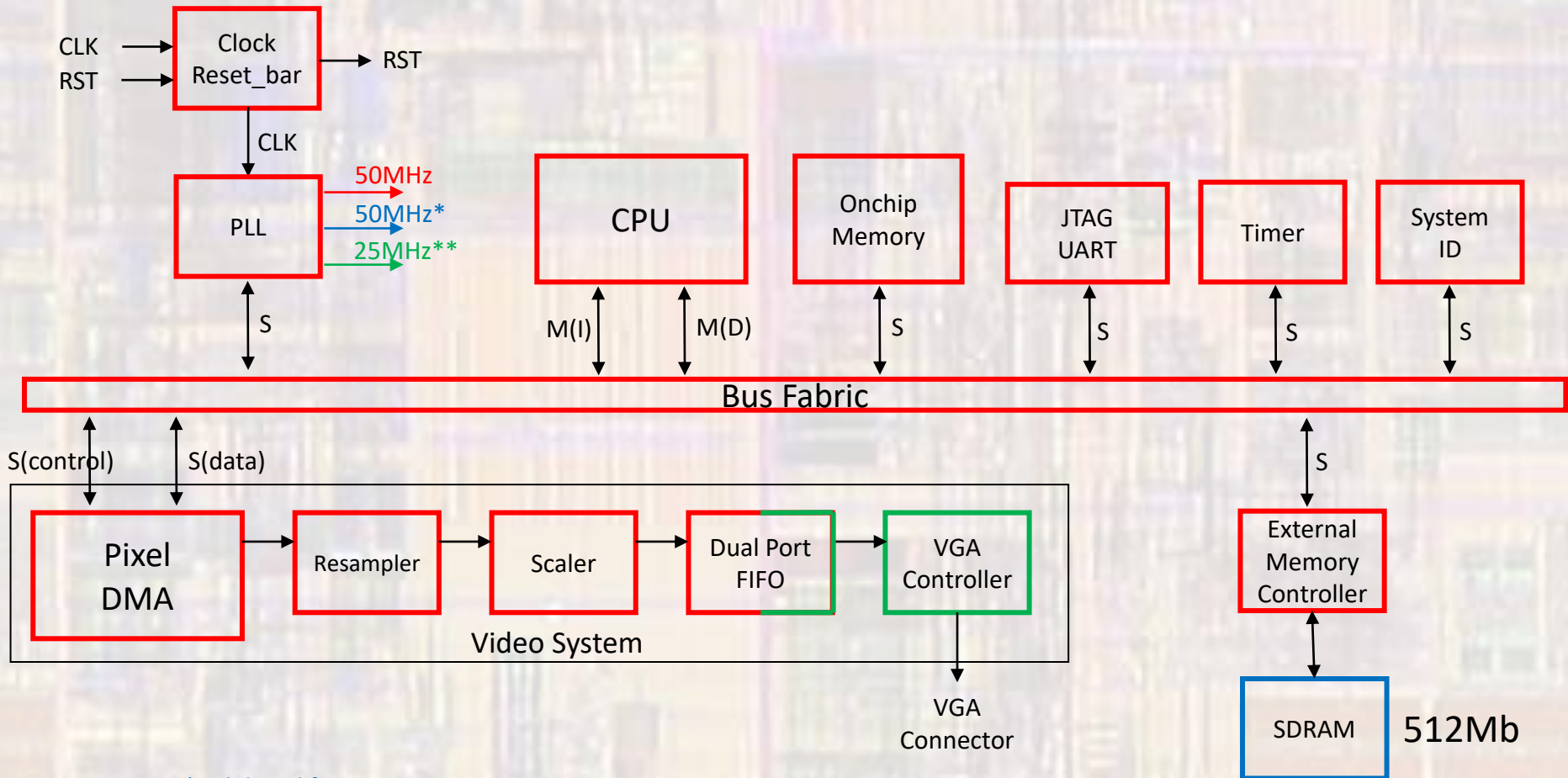
Last updated 10/12/20

NIOS II Pixel Display - SW

These slides describe the development of software for a moderately complex NIOS Processor using the Pixel Buffer IP

Upon completion: You should be able implement your own software on a NIOS processor using the Pixel Buffer IP

NIOS II Pixel Display - SW



50MHz* - delayed for SDRAM

25MHz** - VGA clk

NIOS II Pixel Display - SW

- Create Eclipse System
 - Open NIOSII software
 - [Tools](#) → [NIOSII Software Build Tools for Eclipse](#)
 - Create the BSP
 - [File](#) → [New](#) → [NIOSII Application and BSP from template](#)
 - Blank Template
 - Edit the BSP
 - Right click on the BSP, [NIOS II](#) → [BSP Editor](#)
 - Change the properties for small systems
 - [Small C library](#)
 - [Reduced device drivers](#)
 - Re-Generate the BSP

NIOS II Pixel Display - SW

- Create Eclipse System
 - Review the BSP Memory allocations
 - Right click on the BSP, **NIOS** → **BSP Editor** → **Linker Script** Tab
 - Most are in the SDRAM

BSP Editor - settings.bsp

File Edit Tools Help

Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory

Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	new_sdram_controller_0	new_sdram_controller_0
.entry	reset	onchip_memory2_0
.exceptions	onchip_memory2_0	onchip_memory2_0
.heap	new_sdram_controller_0	new_sdram_controller_0
.rodata	new_sdram_controller_0	new_sdram_controller_0
.rwd	new_sdram_controller_0	new_sdram_controller_0
.stack	new_sdram_controller_0	new_sdram_controller_0
.text	onchip_memory2_0	onchip_memory2_0

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory2_0	0x04020020 - 0x0403869F	onchip_memory2_0	99968	32
reset	0x04020000 - 0x0402001F	onchip_memory2_0	32	0
new_sdram_controller_0	0x00000000 - 0x03FFFFFF	new_sdram_controller_0	67108864	0

Grayed out entries are automatically created at generate time. They are not editable or persisted in the BSP settings file.

Exceptions

Operating Memory

Program Instructions

NIOS II Pixel Display - SW

- Create Eclipse System
 - In the BSP – under drivers/inc
 - Open altera_up_avalon_video_pixel_buffer_dma.h
 - Find the video pixel buffer structure name

```
21 @/*
22  * Device structure definition. Each instance of the driver uses one
23  * of these structures to hold its associated state.
24  */
25 typedef struct alt_up_pixel_buffer_dma_dev {
26     /// @brief Character device structure
27     /// @sa Developing Device Drivers for the HAL in Nios II Software Developer's Handbook
28     alt_dev dev;
29     /// @brief the pixel buffer's slave base address
30     unsigned int base;
31     /// @brief the memory buffer's start address
32     unsigned int buffer_start_address;
33     /// @brief the memory back buffer's start address
34     unsigned int back_buffer_start_address;
```

- Create a pointer of this type

```
// define a pointer of type pixel_buffer...
// to use as a reference in the dma functions
//
alt_up_pixel_buffer_dma_dev * pixel_buf_dma_dev;
```

NIOS II Pixel Display - SW

- Create Eclipse System

- In the BSP – under [drivers/inc](#)

- Open altera_up_avalon_video_pixel_buffer_dma.h
 - Find the function to open the pixel buffer dma device

```
56 //////////////////////////////////////////////////////////////////////////////////////
57 // direct operation functions
58 /**
59  * @brief Opens the pixel buffer device specified by <em> name </em>
60  *
61  * @param name -- the pixel buffer component name in SOPC Builder.
62  *
63  * @return The corresponding device structure, or NULL if the device is not found
64  */
65 alt_up_pixel_buffer_dma_dev* alt_up_pixel_buffer_dma_open_dev(const char* name);
66
```

- Open the device and assign it to the previously defined pointer

```
// open the Pixel Buffer port
// - command is in drivers/inc/altera..video_pixel_buffer_dma.h
// name reference is in system.h
// - "/dev/video_pixel_buffer_dma_0"
//
```

```
349 #define ALT_MODULE_CLASS_video_pixel_buffer_dma_0 altera_up_avalon_video_pixel_buffer_dma
350 #define VIDEO_PIXEL_BUFFER_DMA_0_BASE 0x4041020
351 #define VIDEO_PIXEL_BUFFER_DMA_0_IRQ -1
352 #define VIDEO_PIXEL_BUFFER_DMA_0_IRQ_INTERRUPT_CONTROLLER_ID -1
353 #define VIDEO_PIXEL_BUFFER_DMA_0_NAME "/dev/video_pixel_buffer_dma_0"
354 #define VIDEO_PIXEL_BUFFER_DMA_0_SPAN 16
355 #define VIDEO_PIXEL_BUFFER_DMA_0_TYPE "altera_up_avalon_video_pixel_buffer_dma"
356
```

```
pixel_buf_dma_dev = alt_up_pixel_buffer_dma_open_dev ("/dev/video_pixel_buffer_dma_0");
```

NIOS II Pixel Display - SW

- Create Eclipse System
 - In the BSP – under `drivers/inc`
 - Open `altera_up_avalon_video_pixel_buffer_dma.h`
 - The remainder of the pixel buffer dma commands are in this file

```
/**
 * @brief Draw a pixel at the location specified by <em>(x, y)</em> on the VGA monitor
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 * @param color-- the RGB color to be drawn
 * @param x -- the \em x coordinate
 * @param y -- the \em y coordinate
 *
 * @return 0 for success, -1 for error (such as out of bounds)
 **/
int altera_up_pixel_buffer_dma_draw(alt_up_pixel_buffer_dma_dev *pixel_buffer, unsigned int color, unsigned int x, unsigned int y);

/**
 * @brief Changes the back buffer's start address
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 * @param new_address -- the new start address of the back buffer
 *
 * @return 0 for success
 **/
int altera_up_pixel_buffer_dma_change_back_buffer_address(alt_up_pixel_buffer_dma_dev *pixel_buffer, unsigned int new_address);

/**
 * @brief Swaps which buffer is being sent to the VGA Controller
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 *
 * @return 0 for success
 **/
int altera_up_pixel_buffer_dma_swap_buffers(alt_up_pixel_buffer_dma_dev *pixel_buffer);
```

5 bits R
6 bits G
5 bits B

This actually writes to the back buffer

0 - 319

0 - 239

0xF800 = RED

NIOS II Pixel Display - SW

- Create Eclipse System

```
/**
 * @brief Check if swapping buffers has completed
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 *
 * @return 0 if complete, 1 if still processing
 */
int alt_up_pixel_buffer_dma_check_swap_buffers_status(alt_up_pixel_buffer_dma_dev *pixel_buffer);

/**
 * @brief This function clears the screen or the back buffer.
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 * @param backbuffer -- set to 1 to clear the back buffer, otherwise set to 0 to clear the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_clear_screen(alt_up_pixel_buffer_dma_dev *pixel_buffer, int backbuffer);

/**
 * @brief This function draws a box of a given color between points (x0,y0) and (x1,y1).
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 * @param x0,x1,y0,y1 -- coordinates of the top left (x0,y0) and bottom right (x1,y1) corner of the box
 * @param color -- color of the box to be drawn
 * @param backbuffer -- set to 1 to select the back buffer, otherwise set to 0 to select the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_draw_box(alt_up_pixel_buffer_dma_dev *pixel_buffer, int x0, int y0, int x1, int y1, int color, int backbuffer);

/**
 * @brief This function draws a horizontal line of a given color between points (x0,y) and (x1,y).
 *
 * @param pixel_buffer -- the pointer to the VGA structure
 * @param x0,x1,y -- coordinates of the left (x0,y) and the right (x1,y) end-points of the line
 * @param color -- color of the line to be drawn
 * @param backbuffer -- set to 1 to select the back buffer, otherwise set to 0 to select the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_draw_hline(alt_up_pixel_buffer_dma_dev *pixel_buffer, int x0, int x1, int y, int color, int backbuffer);
```

These actually write to the buffers

NIOS II Pixel Display - SW

- Create Eclipse System

These actually write to the buffers

```
/**
 * @brief This function draws a vertical line of a given color between points (x,y0) and (x,y1).
 *
 * @param pixel buffer -- the pointer to the VGA structure
 * @param x,y0,y1 -- coordinates of the top (x,y0) and the bottom (x,y1) end-points of the line
 * @param color -- color of the line to be drawn
 * @param backbuffer -- set to 1 to select the back buffer, otherwise set to 0 to select the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_draw_vline(alt_up_pixel_buffer_dma_dev *pixel_buffer, int x, int y0, int y1, int color, int backbuffer);

/**
 * @brief This function draws a rectangle of a given color between points (x0,y0) and (x1,y1).
 *
 * @param pixel buffer -- the pointer to the VGA structure
 * @param x0,x1,y0,y1 -- coordinates of the top left (x0,y0) and bottom right (x1,y1) corner of the rectangle
 * @param color -- color of the rectangle to be drawn
 * @param backbuffer -- set to 1 to select the back buffer, otherwise set to 0 to select the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_draw_rectangle(alt_up_pixel_buffer_dma_dev *pixel_buffer, int x0, int y0, int x1, int y1, int color, int backbuffer);

/**
 * @brief This function draws a line of a given color between points (x0,y0) and (x1,y1).
 *
 * @param pixel buffer -- the pointer to the VGA structure
 * @param x0,x1,y0,y1 -- coordinates (x0,y0) and (x1,y1) correspond to end points of the line
 * @param color -- color of the line to be drawn
 * @param backbuffer -- set to 1 to select the back buffer, otherwise set to 0 to select the current screen.
 *
 * @return 0 if complete, 1 if still processing
 */
void alt_up_pixel_buffer_dma_draw_line(alt_up_pixel_buffer_dma_dev *pixel_buffer, int x0, int y0, int x1, int y1, int color, int backbuffer);
```

NIOS II Pixel Display - SW

- Create Eclipse System
 - Write a program to print some Pixels to the screen

```
//////////////////////////////////
// Includes
//////////////////////////////////
#include "altera_up_avalon_video_pixel_buffer_dma.h"
#include <stdio.h>
#include <unistd.h>

int main(void){
    // define a pointer of type pixel_buffer...
    // to use as a reference in the dma functions
    //
    alt_up_pixel_buffer_dma_dev * pixel_buf_dma_dev;

    // open the Pixel Buffer port
    // - command is in drivers/inc/alter...video_pixel_buffer_dma.h
    // name reference is in system.h
    // - "/dev/video_pixel_buffer_dma 0"
    //
    pixel_buf_dma_dev = alt_up_pixel_buffer_dma_open_dev ("/dev/video_pixel_buffer_dma 0");

    // Check for error and output to the console
    //
    if ( pixel_buf_dma_dev == NULL)
        printf ("Error: could not open pixel buffer device \n");
    else
        printf ("Opened pixel buffer device \n");
}
```

Include the DMA functions

Debug code

NIOS II Pixel Display - SW

- Create Eclipse System
 - Write a program to print some Pixels to the screen

```
// Clear the screen
// - command is in drivers/inc/alter...video pixel buffer dma.h
// - wait until done before continuing
//
alt_up_pixel_buffer_dma_clear_screen (pixel_buf_dma_dev, 0);
usleep(1000000); // 1sec

// Draw a box
// - command is in drivers/inc/alter...video pixel buffer dma.h
//
alt_up_pixel_buffer_dma_draw_box (pixel_buf_dma_dev, 100, 50, 149, 99, 0xF800, 0);
alt_up_pixel_buffer_dma_draw_box (pixel_buf_dma_dev, 150, 100, 199, 149, 0x07E0, 0);
alt_up_pixel_buffer_dma_draw_box (pixel_buf_dma_dev, 200, 150, 249, 199, 0x001F, 0);
```

NIOS II Pixel Display - SW

- Create Eclipse System
 - Compile the software
 - Select the code file (box.c)
 - Project → Build Project
 - Right Click on the project → run as → Nios II Hardware

NIOS II Pixel Display - SW

- Create Eclipse System

