Nick DiMartino, Belief Gratini, Max Matias, Zack Wannie

Professor Andras Moritz

ECE 332: Embedded Systems Lab

Due: 20 April 2020

Lab 4 Report: HW/SW Compression and Decompression of the Captured Image

Introduction

The objective of this lab was to understand how to leverage the FPGA based hardware acceleration in an SoC design. Communication between the FPGA fabric and the ARM hard core processor system (HPS) in an SoC board also had to be understood. Along with these, new components had to be added to an existing QSYS design so that new functionalities operated correctly.

Detailed Procedure

1. Open Quartus Prime and open the provided DE1_SoC_With_D5M QPF file.
2. Add the RLE file from Lab 2 and the provided FIFO buffer file.
3. Open QSYS and open the provided Computer_Syste.qsys file.
4. Using the provided slides, add the eight Parallel Inputs/Outputs (PIOs) below:
    a. Inputs
        i. RESULT_READY_PIO
        ii. IDATA_PIO[23:0]
        iii. FIFO_IN_FULL_PIO
    b. Outputs
        i. RLE_FLUSH_PIO

ii. RLE_RESET

iii. ODATA_PIO[7:0]

iv. FIFO_IN_WRITE_REQ_PIO

v. FIFO_OUT_READ_REQ_PIO

5. The eight PIOs above should be set with the correct I/O relation to the ARM core as well as correct settings for the bit-width to match the signal.
6. For each of the PIOs added, the *external_connection* should be set to *Export.*
7. Connect each PIO to the following components:
    a. clk - System_PLL.sys_clk
    b. reset - System_PLL.reset_clk and ARM_A9_HPS.h2f_reset
    c. s1 - ARM_A9_HPA.h2f_lw_axi_master
8. Assign Base Addresses in the System Menu.
9. Add wires to connect the modules together and to the ARM core in the top-level entity.
    a. The DE1_SoC_With_D5M.v file should be set as the top-level entity.
10. In the top-level entity file, instantiate the Verilog modules.
11. Connect the inputs and outputs of each module to the appropriate wire.
12. Add PIO connections to the top-level entity using the template.
    a. Template can be found in QSYS-Generate-Show Instantiation Template
13. Compile design.
14. Modify the C code to do the following:
    a. Communicate with the RLE hardware using the alt_write_byte(), alt_byte(), and alt_read_word() functions in the socal.h file.

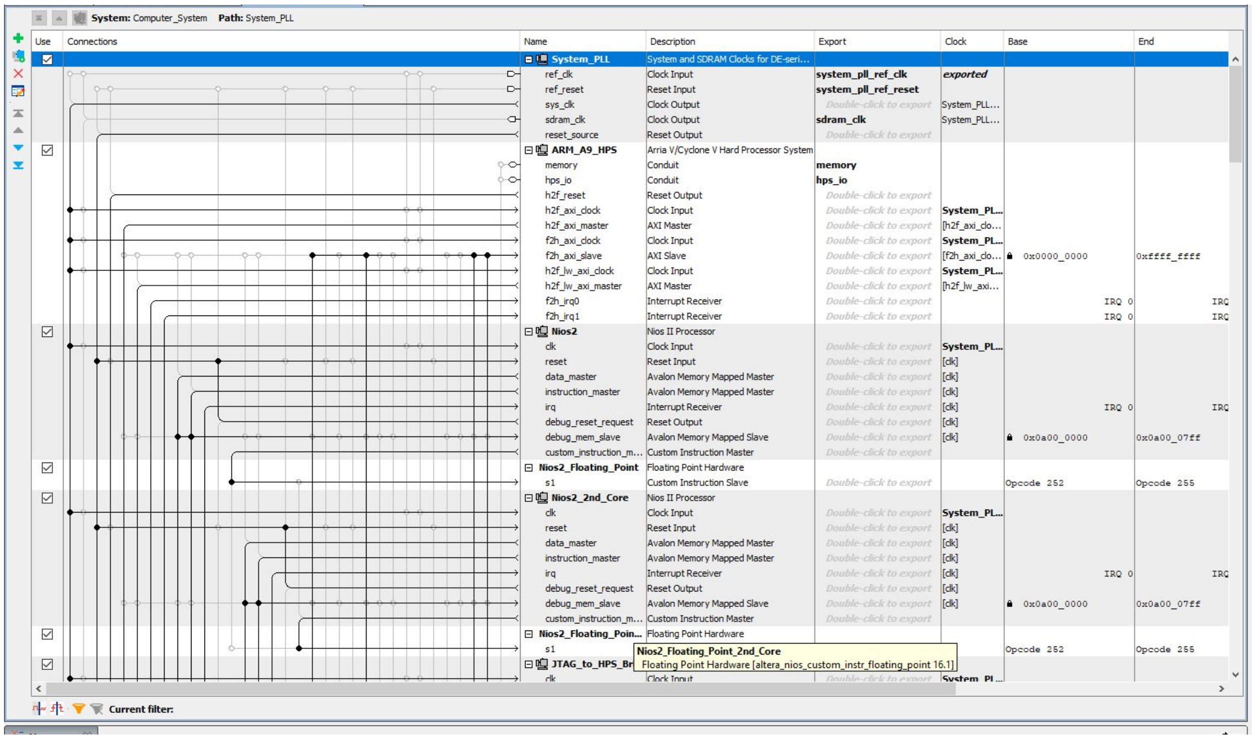
        i. This will perform compression. The result should be stored onto the SDRAM.

b. Convert the captured image to one-bit-per-pixel black and white representation before compression.
c. Write a function in C to decompress the RLE-compressed image.
d. Display the decompressed image with the compression ratio.

Hardware Changes

There were no hardware changes done in this lab due to recent events with COVID-19 hindering the use of the FPGA board. However, the lab used Quartus Prime, QSYS, and the Altera Monitor Program. Using these programs, the board could then be programmed with the *Software Changes* described below to then get a captured image which would have been compressed and decompressed. The following are screen shots of the QSYS connections and Quartus Prime compilation.

QSYS connections (note some screenshots may overlap with others in terms of components)

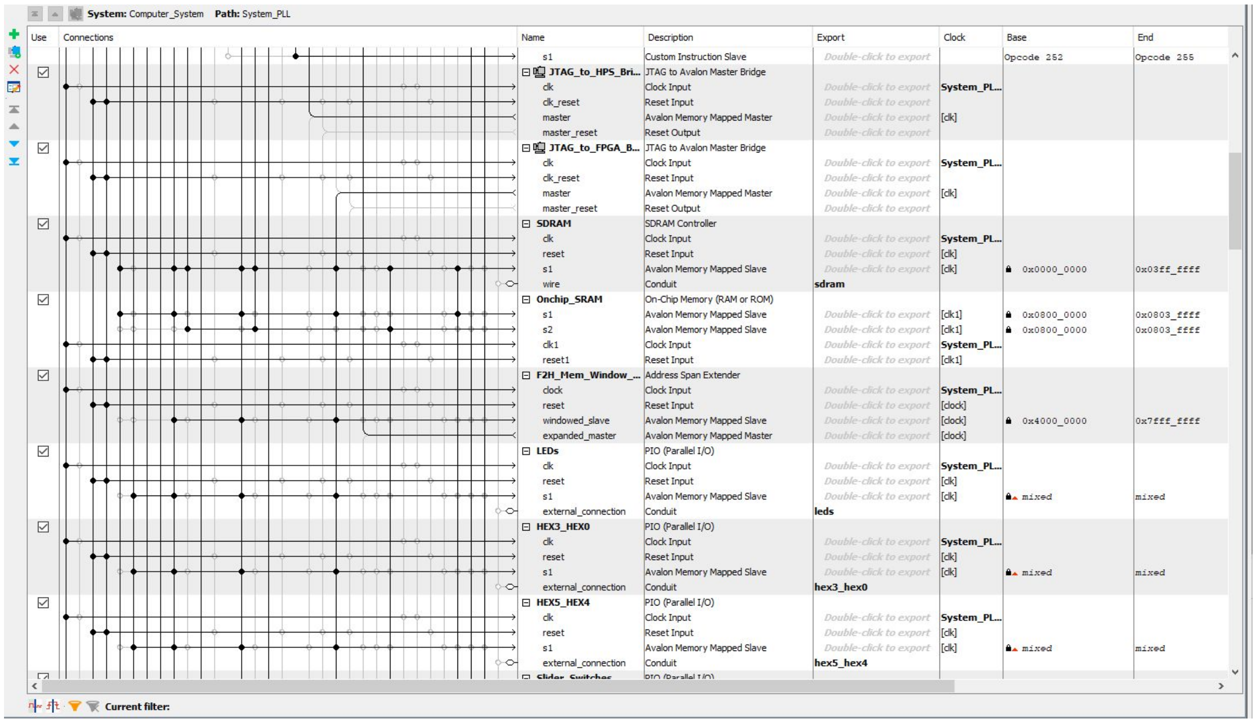


System: Computer_System **Path:** System_PLL

Use	Name	Description	Export	Clock	Base	End
✓	System_PLL	System and SDRAM Clocks for DE-seri...				
	ref_clk	Clock Input	system_pll_ref_clk	*exported*		
	ref_reset	Reset Input	system_pll_ref_reset			
	sys_clk	Clock Output	*Double-click to export*	System_PLL...		
	sdram_clk	Clock Output	sdram_clk	System_PLL...		
	reset_source	Reset Output	*Double-click to export*			
✓	ARM_A9_HPS	Arria V/Cyclone V Hard Processor System				
	memory	Conduit	memory			
	hps_io	Conduit	hps_io			
	h2f_reset	Reset Output	*Double-click to export*			
	h2f_axi_clock	Clock Input	*Double-click to export*	System_PL...		
	h2f_axi_master	AXI Master	*Double-click to export*	[h2f_axi_clo...		
	f2h_axi_clock	Clock Input	*Double-click to export*	System_PL...		
	f2h_axi_slave	AXI Slave	*Double-click to export*	[f2h_axi_clo...	0x0000_0000	0xffff_ffff
	h2f_lw_axi_clock	Clock Input	*Double-click to export*	System_PL...		
	h2f_lw_axi_master	AXI Master	*Double-click to export*	[h2f_lw_axi...		
	f2h_irq0	Interrupt Receiver	*Double-click to export*		IRQ 0	IRQ
	f2h_irq1	Interrupt Receiver	*Double-click to export*		IRQ 0	IRQ
✓	Nios2	Nios II Processor				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ
	debug_reset_request	Reset Output	*Double-click to export*	[clk]		
	debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0a00_0000	0x0a00_07ff
	custom_instruction_m...	Custom Instruction Master	*Double-click to export*			
✓	Nios2_Floating_Point	Floating Point Hardware				
	s1	Custom Instruction Slave	*Double-click to export*		Opcode 252	Opcode 255
✓	Nios2_2nd_Core	Nios II Processor				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ
	debug_reset_request	Reset Output	*Double-click to export*	[clk]		
	debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0a00_0000	0x0a00_07ff
	custom_instruction_m...	Custom Instruction Master	*Double-click to export*			
✓	Nios2_Floating_Poin...	Floating Point Hardware				
	s1				Opcode 252	Opcode 255
✓	JTAG_to_HPS_Br					
	clk	Clock Input	*Double-click to export*	System_PL...		

Nios2_Floating_Point_2nd_Core
Floating Point Hardware [altera_nios_custom_instr_floating_point 16.1]

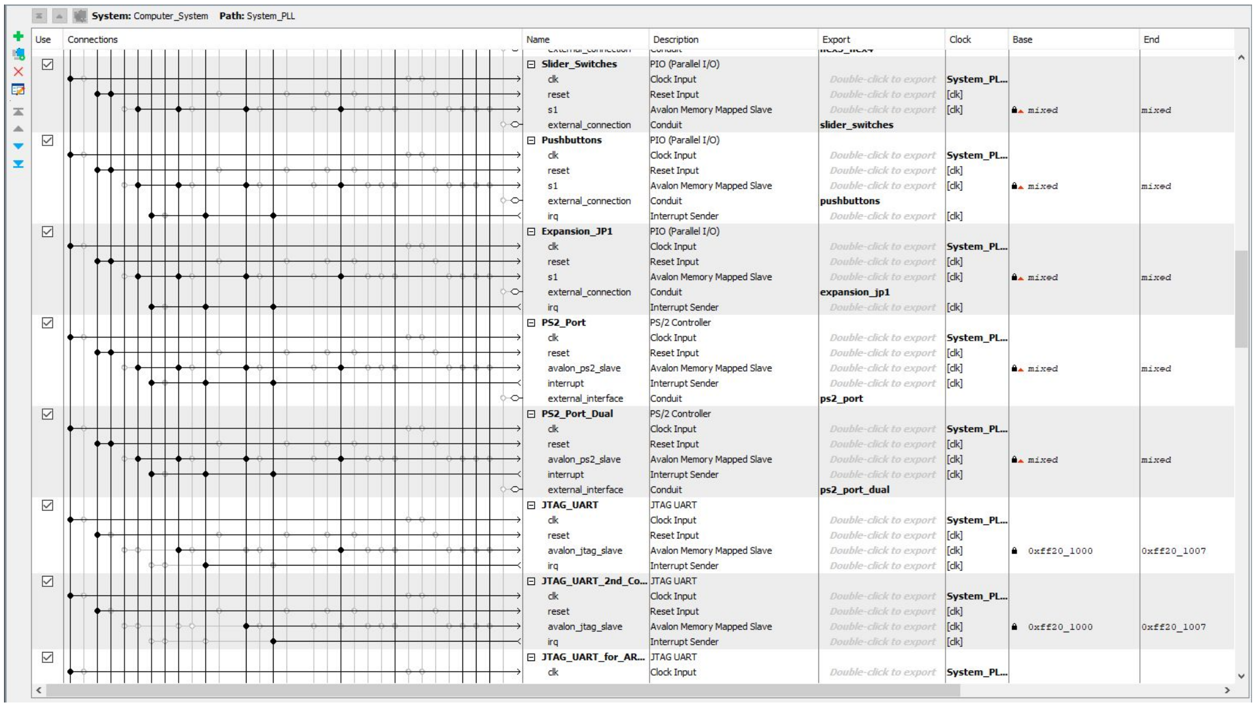
Current filter:



System: Computer_System **Path:** System_PLL

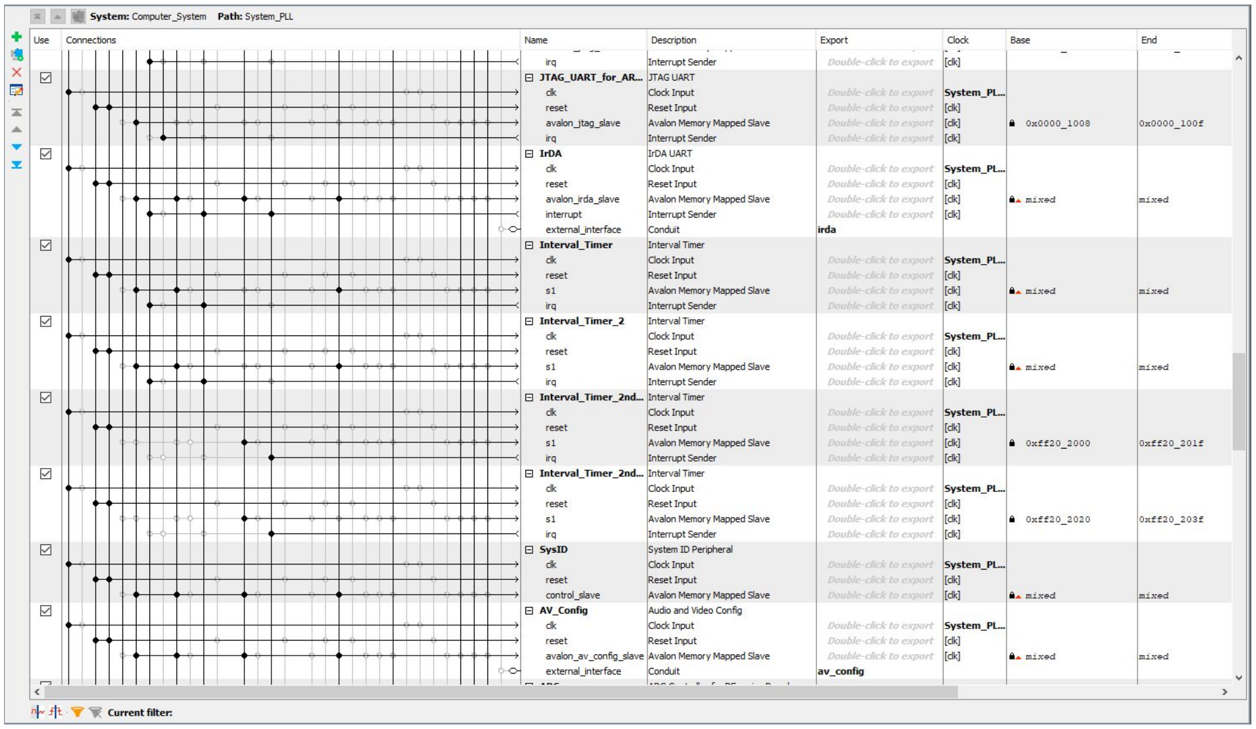
Use	Name	Description	Export	Clock	Base	End
	s1	Custom Instruction Slave	*Double-click to export*		Opcode 252	Opcode 255
✓	JTAG_to_HPS_Bri...	JTAG to Avalon Master Bridge				
	clk	Clock Input	*Double-click to export*	System_PL...		
	clk_reset	Reset Input	*Double-click to export*			
	master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	master_reset	Reset Output	*Double-click to export*			
✓	JTAG_to_FPGA_B...	JTAG to Avalon Master Bridge				
	clk	Clock Input	*Double-click to export*	System_PL...		
	clk_reset	Reset Input	*Double-click to export*			
	master	Avalon Memory Mapped Master	*Double-click to export*	[clk]		
	master_reset	Reset Output	*Double-click to export*			
✓	SDRAM	SDRAM Controller				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000_0000	0x03ff_ffff
	wire	Conduit	sdram			
✓	Onchip_SRAM	On-Chip Memory (RAM or ROM)				
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0800_0000	0x0803_ffff
	s2	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0800_0000	0x0803_ffff
	clk1	Clock Input	*Double-click to export*	System_PL...		
	reset1	Reset Input	*Double-click to export*	[clk1]		
✓	F2H_Mem_Window_...	Address Span Extender				
	clock	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clock]		
	windowed_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x4000_0000	0x7fff_ffff
	expanded_master	Avalon Memory Mapped Master	*Double-click to export*	[clock]		
✓	LEDs	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	leds			
✓	HEX3_HEX0	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	hex3_hex0			
✓	HEX5_HEX4	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	System_PL...		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	hex5_hex4			
✓	Slider_Switches	PIO (Parallel I/O)				

Current filter:



System: Computer_System **Path:** System_PLL

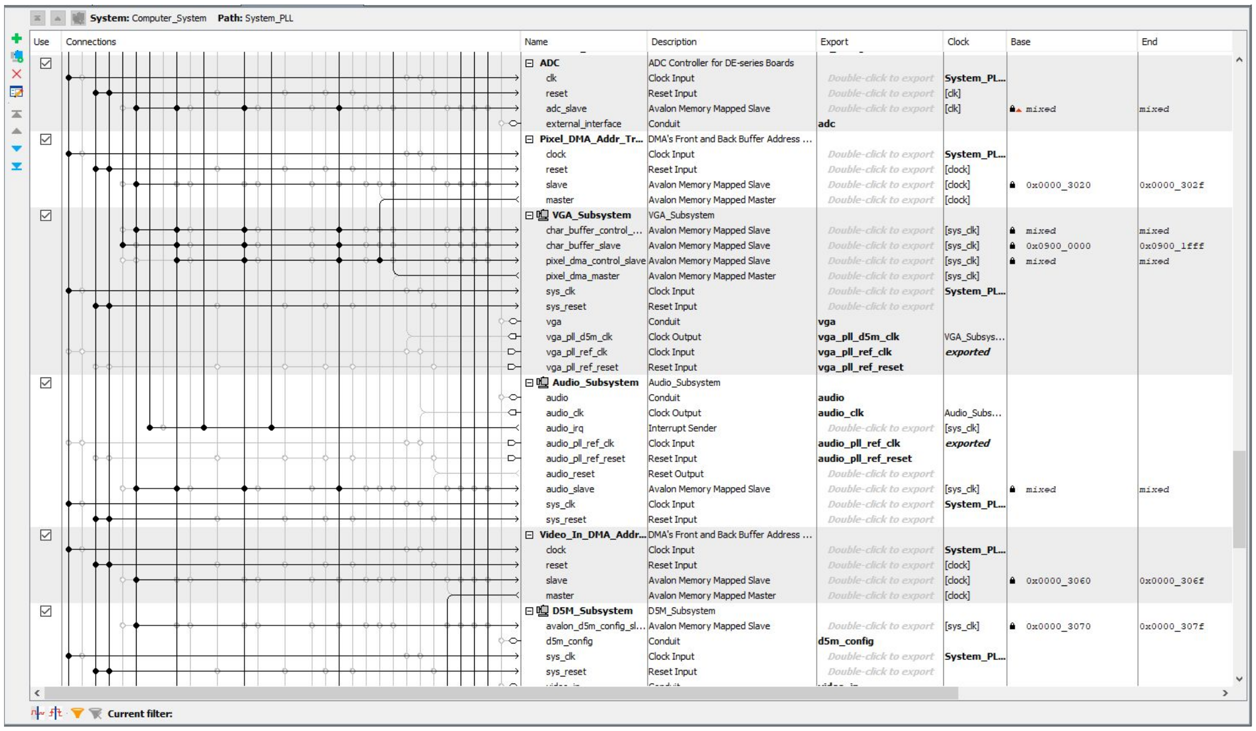
Use	Name	Description	Export	Clock	Base	End
☑	**Slider_Switches**	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	**slider_switches**			
☑	**Pushbuttons**	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	**pushbuttons**			
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**Expansion_JP1**	PIO (Parallel I/O)				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_connection	Conduit	**expansion_jp1**			
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**PS2_Port**	PS/2 Controller				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_ps2_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	interrupt	Interrupt Sender	*Double-click to export*	[clk]		
	external_interface	Conduit	**ps2_port**			
☑	**PS2_Port_Dual**	PS/2 Controller				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_ps2_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	interrupt	Interrupt Sender	*Double-click to export*	[clk]		
	external_interface	Conduit	**ps2_port_dual**			
☑	**JTAG_UART**	JTAG UART				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0xff20_1000	0xff20_1007
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**JTAG_UART_2nd_Co...**	JTAG UART				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0xff20_1000	0xff20_1007
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**JTAG_UART_for_AR...**	JTAG UART				
	clk	Clock Input	*Double-click to export*	**System_PL...**		



System: Computer_System **Path:** System_PLL

Use	Name	Description	Export	Clock	Base	End
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**JTAG_UART_for_AR...**	JTAG UART				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000_1008	0x0000_100f
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**IrDA**	IrDA UART				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_irda_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	interrupt	Interrupt Sender	*Double-click to export*	[clk]		
	external_interface	Conduit	**irda**			
☑	**Interval_Timer**	Interval Timer				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**Interval_Timer_2**	Interval Timer				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**Interval_Timer_2nd...**	Interval Timer				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0xff20_2000	0xff20_201f
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**Interval_Timer_2nd...**	Interval Timer				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0xff20_2020	0xff20_203f
	irq	Interrupt Sender	*Double-click to export*	[clk]		
☑	**SysID**	System ID Peripheral				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
☑	**AV_Config**	Audio and Video Config				
	clk	Clock Input	*Double-click to export*	**System_PL...**		
	reset	Reset Input	*Double-click to export*	[clk]		
	avalon_av_config_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
	external_interface	Conduit	**av_config**			

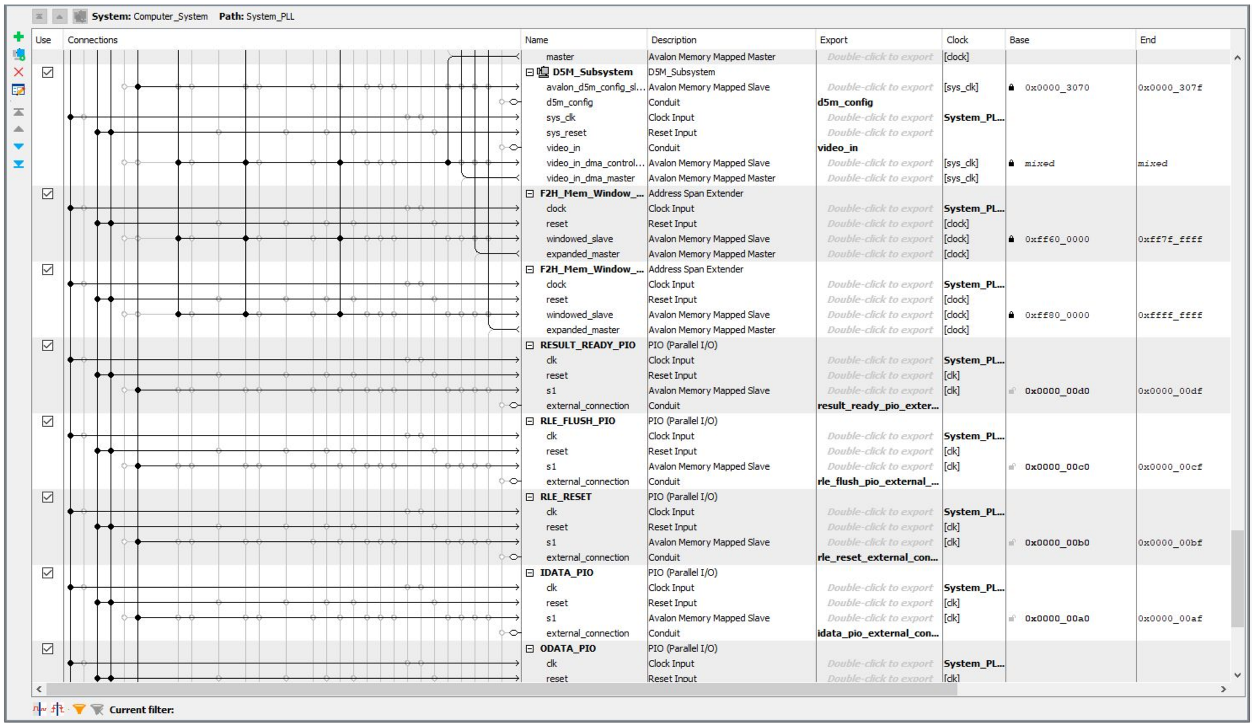
Current filter:



System: Computer_System **Path:** System_PLL

Use	Connections	Name	Description	Export	Clock	Base	End
☑		⊟ **ADC**	ADC Controller for DE-series Boards				
		clk	Clock Input	*Double-click to export*	**System_PL...**		
		reset	Reset Input	*Double-click to export*	[clk]		
		adc_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	mixed	mixed
		external_interface	Conduit	**adc**			
☑		⊟ **Pixel_DMA_Addr_Tr...**	DMA's Front and Back Buffer Address ...				
		clock	Clock Input	*Double-click to export*	**System_PL...**		
		reset	Reset Input	*Double-click to export*	[clock]		
		slave	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0000_3020	0x0000_302f
		master	Avalon Memory Mapped Master	*Double-click to export*	[clock]		
☑		⊟ **VGA_Subsystem**	VGA_Subsystem				
		char_buffer_control_...	Avalon Memory Mapped Slave	*Double-click to export*	[sys_clk]	mixed	mixed
		char_buffer_slave	Avalon Memory Mapped Slave	*Double-click to export*	[sys_clk]	0x0900_0000	0x0900_1fff
		pixel_dma_control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[sys_clk]	mixed	mixed
		pixel_dma_master	Avalon Memory Mapped Master	*Double-click to export*	[sys_clk]		
		sys_clk	Clock Input	*Double-click to export*	**System_PL...**		
		sys_reset	Reset Input	*Double-click to export*			
		vga	Conduit	**vga**			
		vga_pll_d5m_clk	Clock Output	**vga_pll_d5m_clk**	VGA_Subsys...		
		vga_pll_ref_clk	Clock Input	**vga_pll_ref_clk**	***exported***		
		vga_pll_ref_reset	Reset Input	**vga_pll_ref_reset**			
☑		⊟ **Audio_Subsystem**	Audio_Subsystem				
		audio	Conduit	**audio**			
		audio_clk	Clock Output	**audio_clk**	Audio_Subs...		
		audio_irq	Interrupt Sender	*Double-click to export*	[sys_clk]		
		audio_pll_ref_clk	Clock Input	**audio_pll_ref_clk**	***exported***		
		audio_pll_ref_reset	Reset Input	**audio_pll_ref_reset**			
		audio_reset	Reset Output	*Double-click to export*			
		audio_slave	Avalon Memory Mapped Slave	*Double-click to export*	[sys_clk]	mixed	mixed
		sys_clk	Clock Input	*Double-click to export*	**System_PL...**		
		sys_reset	Reset Input	*Double-click to export*			
☑		⊟ **Video_In_DMA_Addr...**	DMA's Front and Back Buffer Address ...				
		clock	Clock Input	*Double-click to export*	**System_PL...**		
		reset	Reset Input	*Double-click to export*	[clock]		
		slave	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0000_3060	0x0000_306f
		master	Avalon Memory Mapped Master	*Double-click to export*	[clock]		
☑		⊟ **D5M_Subsystem**	D5M_Subsystem				
		avalon_d5m_config_sl...	Avalon Memory Mapped Slave	*Double-click to export*	[sys_clk]	0x0000_3070	0x0000_307f
		d5m_config	Conduit	**d5m_config**			
		sys_clk	Clock Input	*Double-click to export*	**System_PL...**		
		sys_reset	Reset Input	*Double-click to export*			

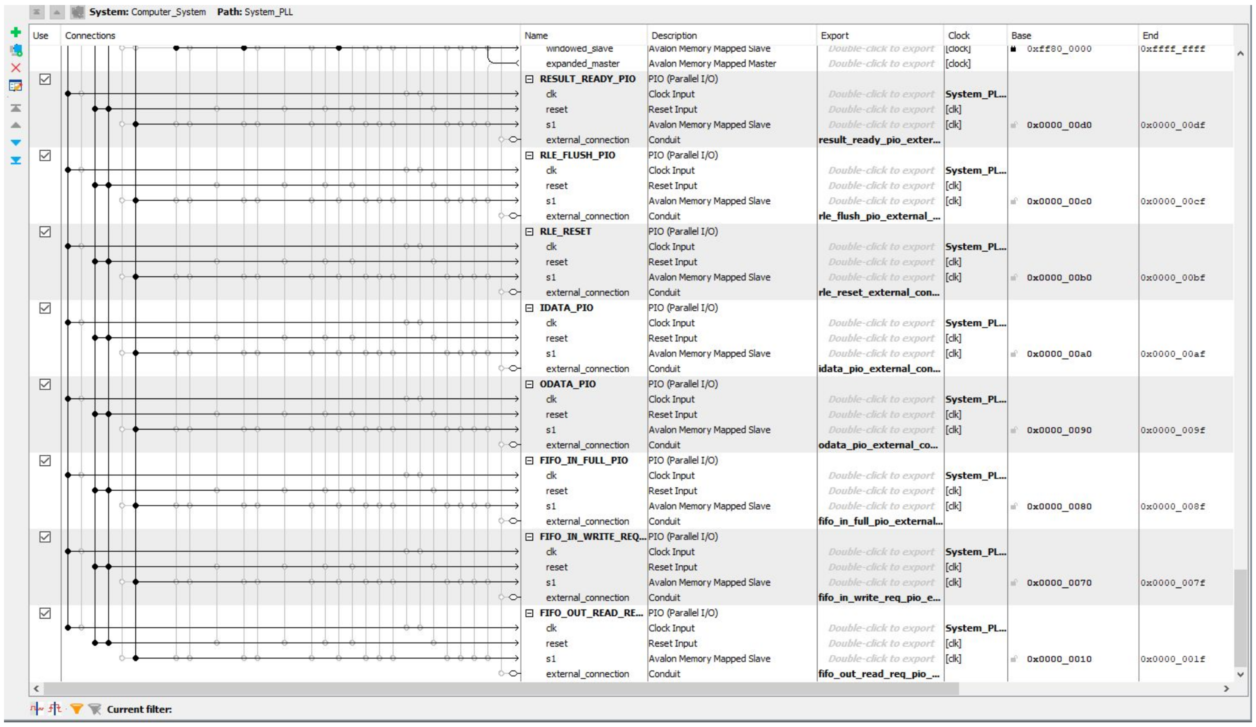
Current filter:



System: Computer_System **Path:** System_PLL

Use	Name	Description	Export	Clock	Base	End
	master	Avalon Memory Mapped Master	Double-click to export	[clock]		
✓	**D5M_Subsystem**	D5M_Subsystem				
	avalon_d5m_config_sl...	Avalon Memory Mapped Slave	Double-click to export	[sys_clk]	0x0000_3070	0x0000_307f
	d5m_config	Conduit	**d5m_config**			
	sys_clk	Clock Input	Double-click to export	**System_PL...**		
	sys_reset	Reset Input	Double-click to export			
	video_in	Conduit	**video_in**			
	video_in_dma_control...	Avalon Memory Mapped Slave	Double-click to export	[sys_clk]	mixed	mixed
	video_in_dma_master	Avalon Memory Mapped Master	Double-click to export	[sys_clk]		
✓	**F2H_Mem_Window_...**	Address Span Extender				
	clock	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clock]		
	windowed_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0xff60_0000	0xff7f_ffff
	expanded_master	Avalon Memory Mapped Master	Double-click to export	[clock]		
✓	**F2H_Mem_Window_...**	Address Span Extender				
	clock	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clock]		
	windowed_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0xff80_0000	0xffff_ffff
	expanded_master	Avalon Memory Mapped Master	Double-click to export	[clock]		
✓	**RESULT_READY_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00d0	0x0000_00df
	external_connection	Conduit	**result_ready_pio_exter...**			
✓	**RLE_FLUSH_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00c0	0x0000_00cf
	external_connection	Conduit	**rle_flush_pio_external_...**			
✓	**RLE_RESET**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00b0	0x0000_00bf
	external_connection	Conduit	**rle_reset_external_con...**			
✓	**IDATA_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00a0	0x0000_00af
	external_connection	Conduit	**idata_pio_external_con...**			
✓	**ODATA_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		

Current filter:



System: Computer_System **Path:** System_PLL

Use	Name	Description	Export	Clock	Base	End
	windowed_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0xff80_0000	0xffff_ffff
	expanded_master	Avalon Memory Mapped Master	Double-click to export	[clock]		
✓	**RESULT_READY_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00d0	0x0000_00df
	external_connection	Conduit	**result_ready_pio_exter...**			
✓	**RLE_FLUSH_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00c0	0x0000_00cf
	external_connection	Conduit	**rle_flush_pio_external_...**			
✓	**RLE_RESET**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00b0	0x0000_00bf
	external_connection	Conduit	**rle_reset_external_con...**			
✓	**IDATA_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_00a0	0x0000_00af
	external_connection	Conduit	**idata_pio_external_con...**			
✓	**ODATA_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0090	0x0000_009f
	external_connection	Conduit	**odata_pio_external_co...**			
✓	**FIFO_IN_FULL_PIO**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0080	0x0000_008f
	external_connection	Conduit	**fifo_in_full_pio_external...**			
✓	**FIFO_IN_WRITE_REQ...**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0070	0x0000_007f
	external_connection	Conduit	**fifo_in_write_req_pio_e...**			
✓	**FIFO_OUT_READ_RE...**	PIO (Parallel I/O)				
	clk	Clock Input	Double-click to export	**System_PL...**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0010	0x0000_001f
	external_connection	Conduit	**fifo_out_read_req_pio_...**			

Current filter:

Each PIO required a specific bit width. They are as follows

Result_Ready_PIO = 1bit

RLE_FLUSH_PIO = 1 bit

RLE_RESET = 1 bit

IDATA_PIO = 24 bits

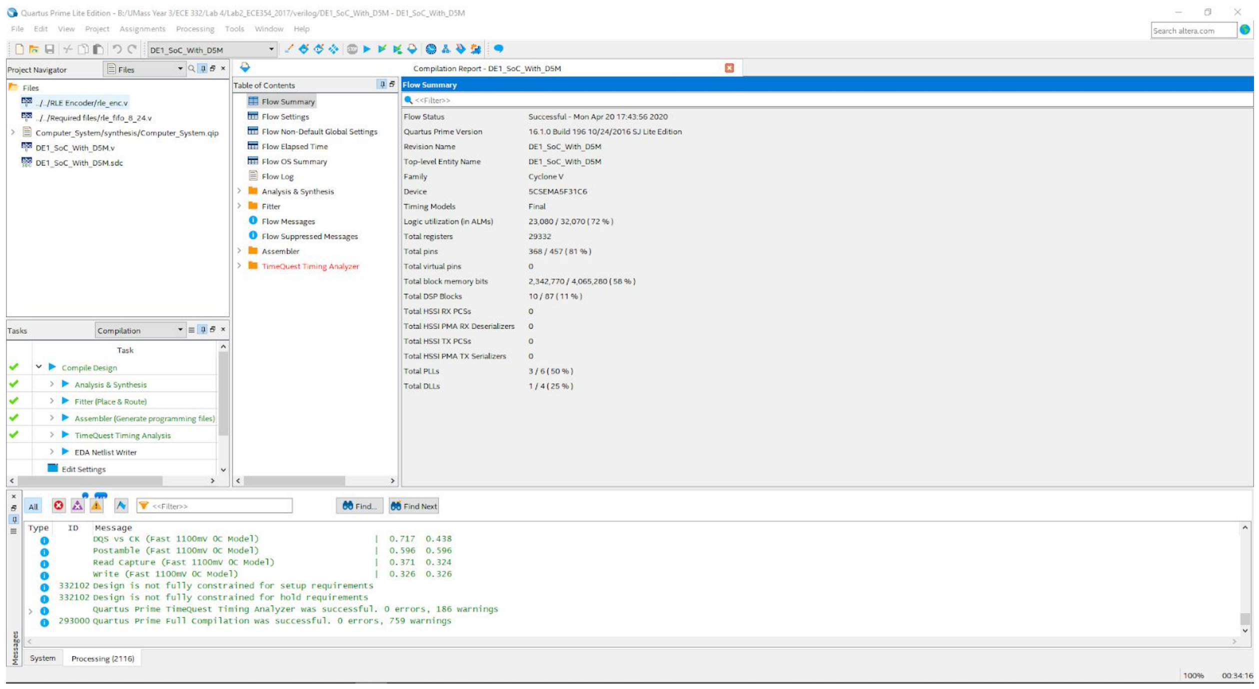
ODATA_PIO = 8 bits

FIFO_IN_FULL = 1bit

FIFO_IN_WRITE_REQ_PIO = 1 bit

FIFI_OUT_READ_REQ_PIO = 1 bit

Quartus Compilation and Added Wires

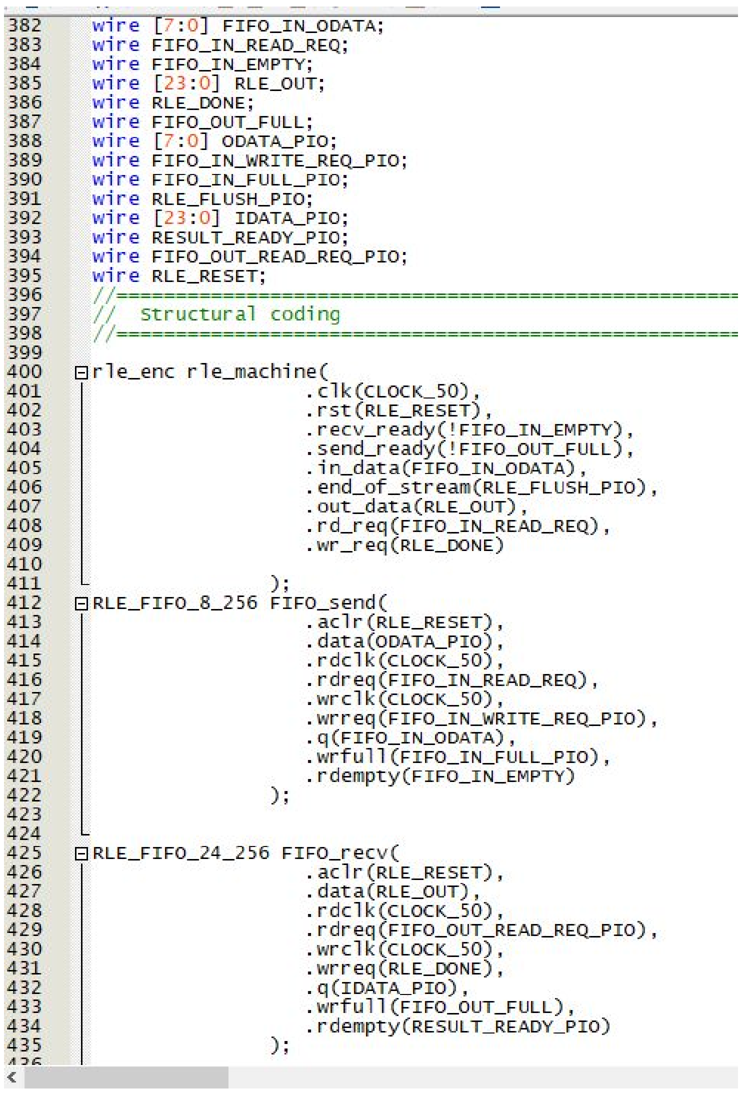


Compilation Report - DE1_SoC_With_D5M

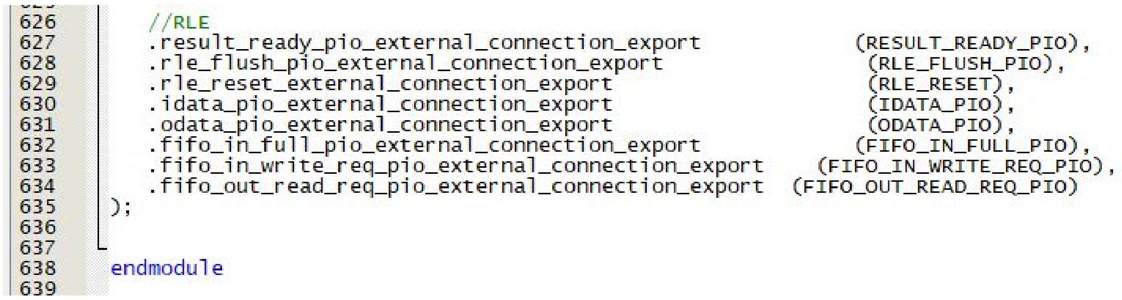
Flow Summary

Flow Status	Successful - Mon Apr 20 17:43:56 2020
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	DE1_SoC_With_D5M
Top-level Entity Name	DE1_SoC_With_D5M
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	23,080 / 32,070 (72 %)
Total registers	29332
Total pins	368 / 457 (81 %)
Total virtual pins	0
Total block memory bits	2,342,770 / 4,065,280 (58 %)
Total DSP Blocks	10 / 87 (11 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	3 / 6 (50 %)
Total DLLs	1 / 4 (25 %)

```
Type   ID     Message
              DQS vs CK (Fast 1100mV 0C Model)                    |  0.717  0.438
              Postamble (Fast 1100mV 0C Model)                    |  0.596  0.596
              Read Capture (Fast 1100mV 0C Model)                 |  0.371  0.324
              Write (Fast 1100mV 0C Model)                        |  0.326  0.326
       332102 Design is not fully constrained for setup requirements
       332102 Design is not fully constrained for hold requirements
              Quartus Prime TimeQuest Timing Analyzer was successful. 0 errors, 186 warnings
       293000 Quartus Prime Full Compilation was successful. 0 errors, 759 warnings
```



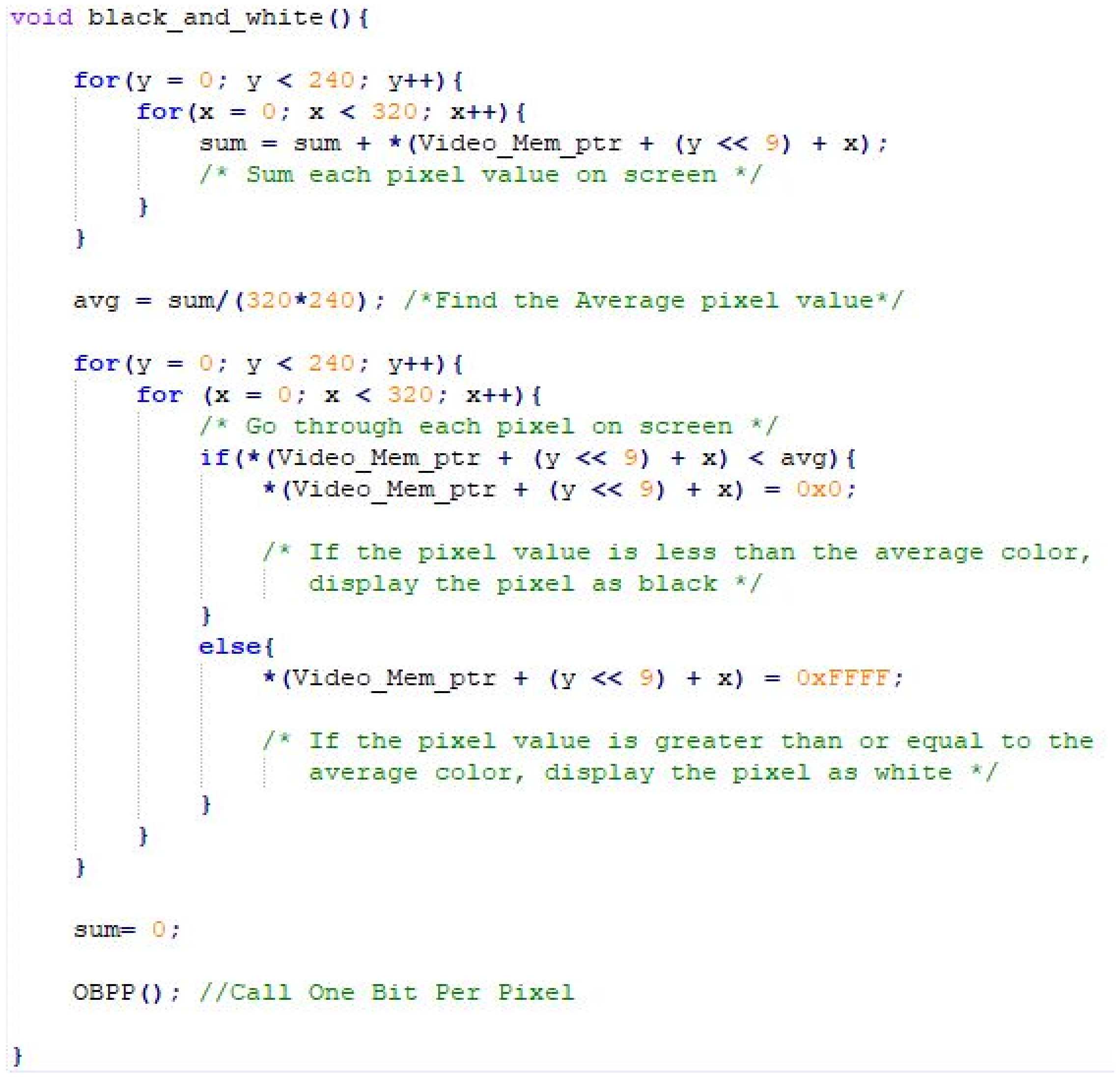
```
wire [7:0] FIFO_IN_ODATA;
wire FIFO_IN_READ_REQ;
wire FIFO_IN_EMPTY;
wire [23:0] RLE_OUT;
wire RLE_DONE;
wire FIFO_OUT_FULL;
wire [7:0] ODATA_PIO;
wire FIFO_IN_WRITE_REQ_PIO;
wire FIFO_IN_FULL_PIO;
wire RLE_FLUSH_PIO;
wire [23:0] IDATA_PIO;
wire RESULT_READY_PIO;
wire FIFO_OUT_READ_REQ_PIO;
wire RLE_RESET;
//=======================================================
//  Structural coding
//=======================================================

rle_enc rle_machine(
                        .clk(CLOCK_50),
                        .rst(RLE_RESET),
                        .recv_ready(!FIFO_IN_EMPTY),
                        .send_ready(!FIFO_OUT_FULL),
                        .in_data(FIFO_IN_ODATA),
                        .end_of_stream(RLE_FLUSH_PIO),
                        .out_data(RLE_OUT),
                        .rd_req(FIFO_IN_READ_REQ),
                        .wr_req(RLE_DONE)

                     );
RLE_FIFO_8_256 FIFO_send(
                        .aclr(RLE_RESET),
                        .data(ODATA_PIO),
                        .rdclk(CLOCK_50),
                        .rdreq(FIFO_IN_READ_REQ),
                        .wrclk(CLOCK_50),
                        .wrreq(FIFO_IN_WRITE_REQ_PIO),
                        .q(FIFO_IN_ODATA),
                        .wrfull(FIFO_IN_FULL_PIO),
                        .rdempty(FIFO_IN_EMPTY)
                     );


RLE_FIFO_24_256 FIFO_recv(
                        .aclr(RLE_RESET),
                        .data(RLE_OUT),
                        .rdclk(CLOCK_50),
                        .rdreq(FIFO_OUT_READ_REQ_PIO),
                        .wrclk(CLOCK_50),
                        .wrreq(RLE_DONE),
                        .q(IDATA_PIO),
                        .wrfull(FIFO_OUT_FULL),
                        .rdempty(RESULT_READY_PIO)
                     );
```



```
    //RLE
    .result_ready_pio_external_connection_export            (RESULT_READY_PIO),
    .rle_flush_pio_external_connection_export                (RLE_FLUSH_PIO),
    .rle_reset_external_connection_export                    (RLE_RESET),
    .idata_pio_external_connection_export                    (IDATA_PIO),
    .odata_pio_external_connection_export                    (ODATA_PIO),
    .fifo_in_full_pio_external_connection_export            (FIFO_IN_FULL_PIO),
    .fifo_in_write_req_pio_external_connection_export    (FIFO_IN_WRITE_REQ_PIO),
    .fifo_out_read_req_pio_external_connection_export  (FIFO_OUT_READ_REQ_PIO)
);


endmodule
```

Software Changes

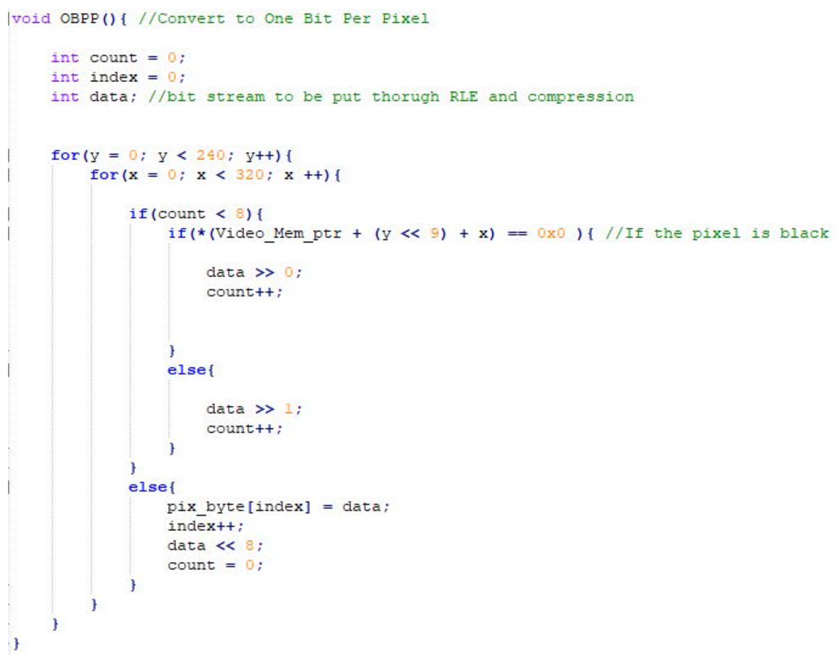
Black and White function:



```
void black_and_white(){

    for(y = 0; y < 240; y++){
        for(x = 0; x < 320; x++){
            sum = sum + *(Video_Mem_ptr + (y << 9) + x);
            /* Sum each pixel value on screen */
        }
    }

    avg = sum/(320*240); /*Find the Average pixel value*/

    for(y = 0; y < 240; y++){
        for (x = 0; x < 320; x++){
            /* Go through each pixel on screen */
            if(*(Video_Mem_ptr + (y << 9) + x) < avg){
                *(Video_Mem_ptr + (y << 9) + x) = 0x0;

                /* If the pixel value is less than the average color,
                   display the pixel as black */
            }
            else{
                *(Video_Mem_ptr + (y << 9) + x) = 0xFFFF;

                /* If the pixel value is greater than or equal to the
                   average color, display the pixel as white */
            }
        }
    }

    sum= 0;

    OBPP(); //Call One Bit Per Pixel

}
```

This function will iterate through each pixel on the screen and sums up the pixels to create an average. After getting the average, we go through each pixel and compare them to the average, and from there we set the pixel to black or white depending on how they compare to the average. Then, at the end of the function, we call our one bit per pixel function so that we can determine whether or not that pixel is going to be represented as a one or a zero when passed through the RLE encoder and compressor.

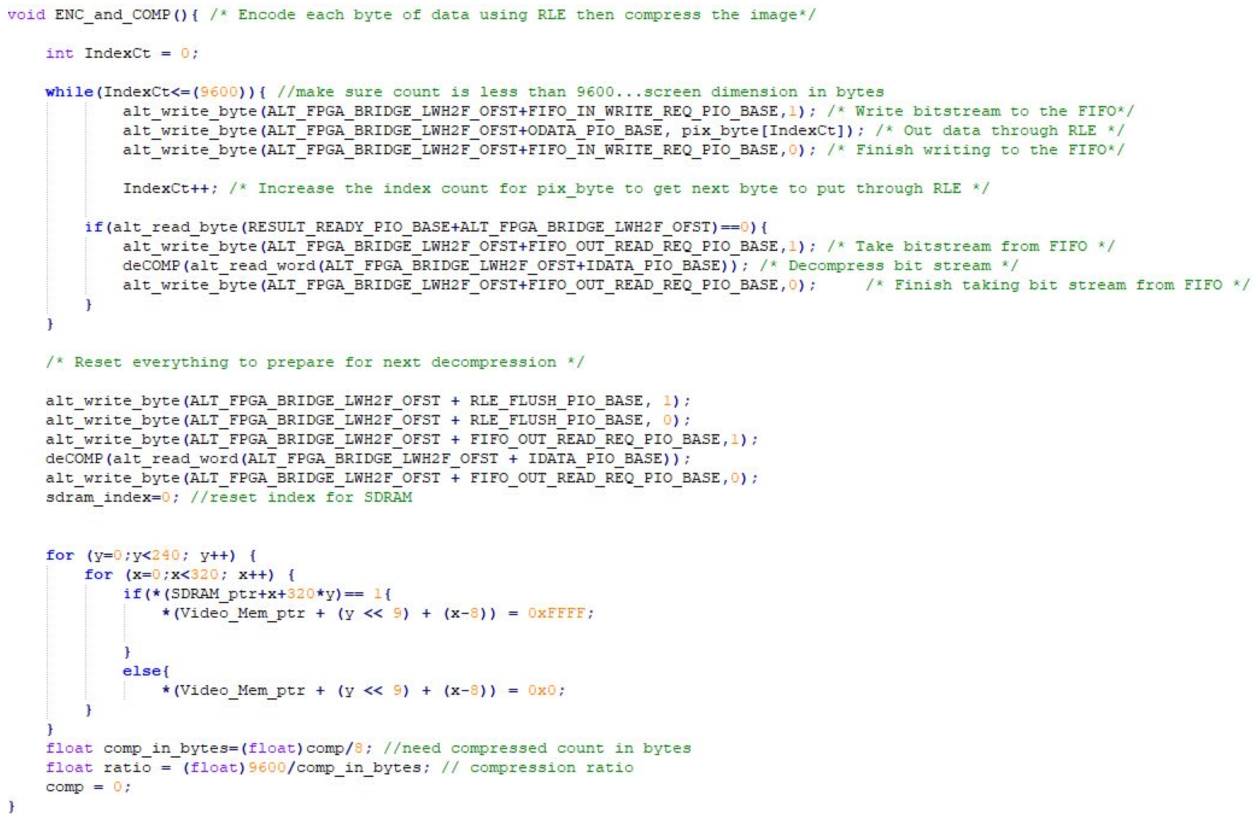
<u>One bit per Pixel function:</u>



```
void OBPP(){ //Convert to One Bit Per Pixel

    int count = 0;
    int index = 0;
    int data; //bit stream to be put thorugh RLE and compression


    for(y = 0; y < 240; y++){
        for(x = 0; x < 320; x ++){

            if(count < 8){
                if(*(Video_Mem_ptr + (y << 9) + x) == 0x0 ){ //If the pixel is black

                    data >> 0;
                    count++;


                }
                else{

                    data >> 1;
                    count++;
                }
            }
            else{
                pix_byte[index] = data;
                index++;
                data << 8;
                count = 0;
            }
        }
    }
}
```

The OBPP function determines whether or not a pixel will be passed through RLE as either a one or zero. Each run through adds either a one or zero to the end of a variable called data that, after going through this process eight times results in a byte of data to send to the RLE for encoding and compression.

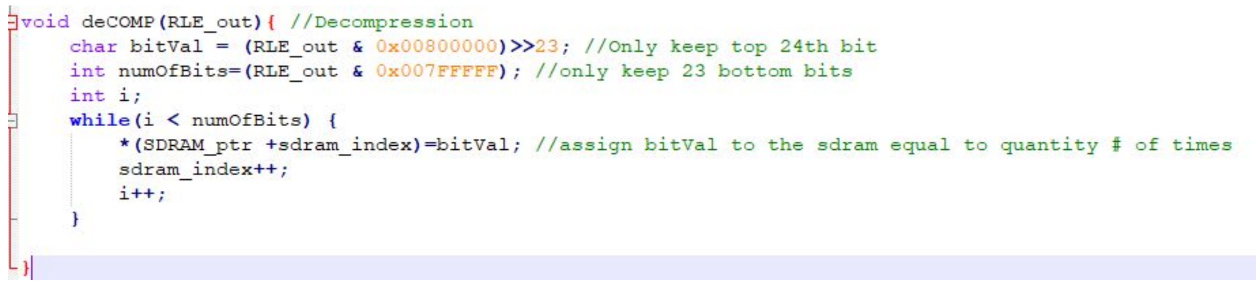
Encoding and compressing function:



```
void ENC_and_COMP(){ /* Encode each byte of data using RLE then compress the image*/

    int IndexCt = 0;

    while(IndexCt<=(9600)){ //make sure count is less than 9600...screen dimension in bytes
            alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST+FIFO_IN_WRITE_REQ_PIO_BASE,1); /* Write bitstream to the FIFO*/
            alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST+ODATA_PIO_BASE, pix_byte[IndexCt]); /* Out data through RLE */
            alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST+FIFO_IN_WRITE_REQ_PIO_BASE,0); /* Finish writing to the FIFO*/

            IndexCt++; /* Increase the index count for pix_byte to get next byte to put through RLE */

        if(alt_read_byte(RESULT_READY_PIO_BASE+ALT_FPGA_BRIDGE_LWH2F_OFST)==0){
            alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST+FIFO_OUT_READ_REQ_PIO_BASE,1); /* Take bitstream from FIFO */
            deCOMP(alt_read_word(ALT_FPGA_BRIDGE_LWH2F_OFST+IDATA_PIO_BASE)); /* Decompress bit stream */
            alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST+FIFO_OUT_READ_REQ_PIO_BASE,0);     /* Finish taking bit stream from FIFO */
        }
    }

    /* Reset everything to prepare for next decompression */

    alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST + RLE_FLUSH_PIO_BASE, 1);
    alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST + RLE_FLUSH_PIO_BASE, 0);
    alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST + FIFO_OUT_READ_REQ_PIO_BASE,1);
    deCOMP(alt_read_word(ALT_FPGA_BRIDGE_LWH2F_OFST + IDATA_PIO_BASE));
    alt_write_byte(ALT_FPGA_BRIDGE_LWH2F_OFST + FIFO_OUT_READ_REQ_PIO_BASE,0);
    sdram_index=0; //reset index for SDRAM


    for (y=0;y<240; y++) {
        for (x=0;x<320; x++) {
            if(*(SDRAM_ptr+x+320*y)== 1{
                *(Video_Mem_ptr + (y << 9) + (x-8)) = 0xFFFF;

            }
            else{
                *(Video_Mem_ptr + (y << 9) + (x-8)) = 0x0;
        }
    }
    float comp_in_bytes=(float)comp/8; //need compressed count in bytes
    float ratio = (float)9600/comp_in_bytes; // compression ratio
    comp = 0;
}
```

For the ENC_and_COMP() function, we assert and write to the FIFO buffers to encode and compress the bytes of data produced in OBPP. After that, we deassert and reset the FIFO buffers to prepare it for the next cycle. It then decompresses and displays the black and white image once again.

Decompressing function:



```
void deCOMP(RLE_out){ //Decompression
    char bitVal = (RLE_out & 0x00800000)>>23; //Only keep top 24th bit
    int numOfBits=(RLE_out & 0x007FFFFF); //only keep 23 bottom bits
    int i;
    while(i < numOfBits) {
        *(SDRAM_ptr +sdram_index)=bitVal; //assign bitVal to the sdram equal to quantity # of times
        sdram_index++;
        i++;
    }

}
```

The decompression function unencodes the RLE encoder by taking the first bit of the 24 bit encoded output, then attaching as many of that bit onto it until it reaches "quantity" number of that bit.

Problems Encountered and Solutions

The biggest problem we encountered through the duration of this lab was not being accessible to Duda hall and having no access to a camera to take the picture with to see if we can compress/decompress the image. This stopped us from being able to get a fully completed project with tested results. Another significant issue that we had was having the lab 2 files under the verilog folder. The order in which our directories had to be placed had a significant impact on the Quartus project completing a Full Compilation. It was not until we realized that there were two of the same .qsys files under the same directory and the lab 2 files being used were not under the correct lab 4 folder. The lab was also not able to be easily tested. Many different programs such as onlinegdb, ubuntu, and the Nios II Eclipse IDE were all tried and it was tough getting a tested and compiled code.

Results and Conclusion

The results of this lab were not able to be completely found due to recent events with COVID-19. With campus being closed, the FPGA Board was not able to be used in the lab. The QSYS design and the Quartus project were able to be fully compiled. The software was not able to be tested on a board so the code was not able to be run. After writing the code, the theory behind what is written makes sense and it is believed that the output of this code would be nearly correct. An image would be converted to black and white, compressed, stored on the SDRAM, decompressed, and then displayed on the screen.