CO503-System on Chip(SoC) Design
E/16/203 - E/16/222

Part 1: First SoC - LED Counter

What we have done:

- We have created an SoC which can display an increasing counter on 8 LEDs as an Output.
- We followed the instructions given in the lab sheet and everything worked fine.
- We edited the provided code to use the base address of the led_out PIO device

New things learned:

- We learned how to work with Quartus 2 application to program the FPGA
- How to create and work around using a BDF design to create what we need
  - How to add symbols
  - Choose different symbols(Input and Outputs etc)
- How to use the Qsys tool
  - How to add components from the library
  - How to make connections between various components
  - How connections made as output
- Neos II processor variants
- How to use the pin planner to do the pin mapping

❖ What do you think the statement IOWR_8DIRECT(LED_BASE, OFFSET, count); does?

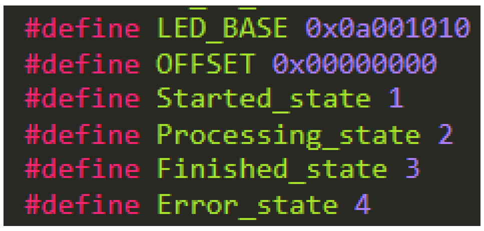
This statement is writing the count to the address given by adding LED_BASE and the OFFSET value.

Part 2: jpeg encoder

What we have done:

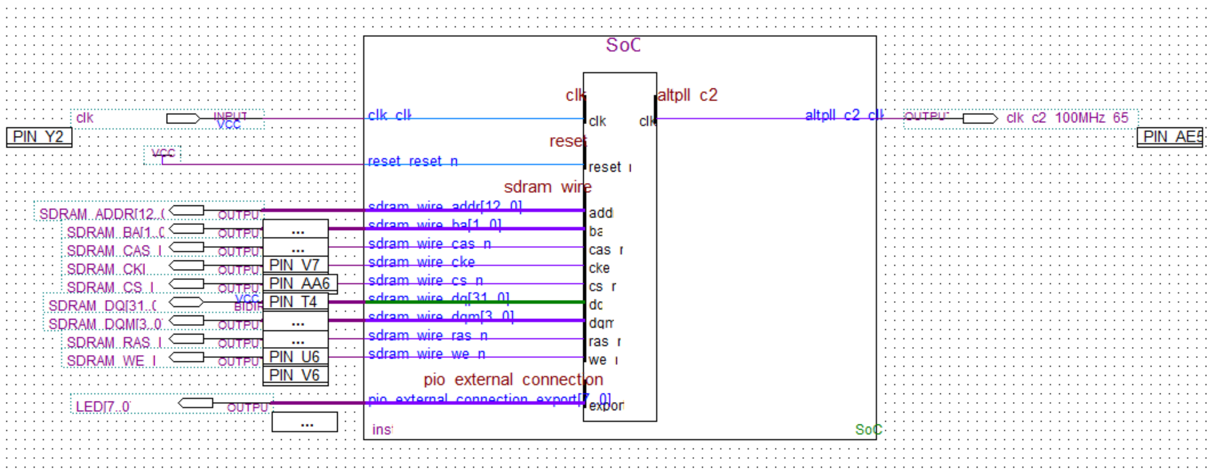
- We have created a jpeg encoder that uses onboard SDRAM of the FPGA as the main memory.
- Hardware configuration using Quartus II app is done as the diagram given in the lab sheet.

- The four states have defined as follows

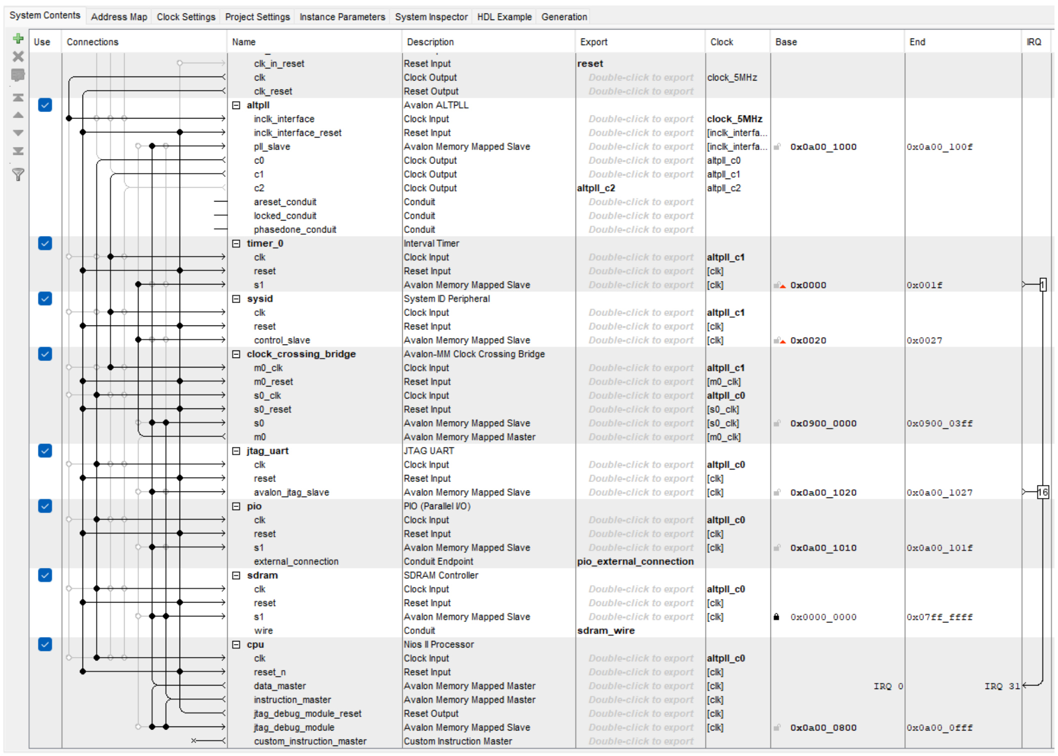


```
#define LED_BASE 0x0a001010
#define OFFSET 0x00000000
#define Started_state 1
#define Processing_state 2
#define Finished_state 3
#define Error_state 4
```

- The block diagram is drawn as follows.



- The Qsys tool is configured as follows. Used components and there connections are done according to the lab sheet.



Use	Name	Description	Export	Clock	Base	End	IRQ
	clk_in_reset	Reset Input	reset				
	clk	Clock Output	Double-click to export	clock_5MHz			
	clk_reset	Reset Output	Double-click to export				
✓	altpll	Avalon ALTPLL					
	inclk_interface	Clock Input	Double-click to export	clock_5MHz			
	inclk_interface_reset	Reset Input	Double-click to export	[inclk_interfa...			
	pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_interfa...	0x0a00_1000	0x0a00_100f	
	c0	Clock Output	Double-click to export	altpll_c0			
	c1	Clock Output	Double-click to export	altpll_c1			
	c2	Clock Output	altpll_c2	altpll_c2			
	areset_conduit	Conduit	Double-click to export				
	locked_conduit	Conduit	Double-click to export				
	phasedone_conduit	Conduit	Double-click to export				
✓	timer_0	Interval Timer					
	clk	Clock Input	Double-click to export	altpll_c1			
	reset	Reset Input	Double-click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000	0x001f	1
✓	sysid	System ID Peripheral					
	clk	Clock Input	Double-click to export	altpll_c1			
	reset	Reset Input	Double-click to export	[clk]			
	control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0020	0x0027	
✓	clock_crossing_bridge	Avalon-MM Clock Crossing Bridge					
	m0_clk	Clock Input	Double-click to export	altpll_c1			
	m0_reset	Reset Input	Double-click to export	[m0_clk]			
	s0_clk	Clock Input	Double-click to export	altpll_c0			
	s0_reset	Reset Input	Double-click to export	[s0_clk]			
	s0	Avalon Memory Mapped Slave	Double-click to export	[s0_clk]	0x0900_0000	0x0900_03ff	
	m0	Avalon Memory Mapped Master	Double-click to export	[m0_clk]			
✓	jtag_uart	JTAG UART					
	clk	Clock Input	Double-click to export	altpll_c0			
	reset	Reset Input	Double-click to export	[clk]			
	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0a00_1020	0x0a00_1027	16
✓	pio	PIO (Parallel I/O)					
	clk	Clock Input	Double-click to export	altpll_c0			
	reset	Reset Input	Double-click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0a00_1010	0x0a00_101f	
	external_connection	Conduit Endpoint	pio_external_connection				
✓	sdram	SDRAM Controller					
	clk	Clock Input	Double-click to export	altpll_c0			
	reset	Reset Input	Double-click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0000_0000	0x07ff_ffff	
	wire	Conduit	sdram_wire				
✓	cpu	Nios II Processor					
	clk	Clock Input	Double-click to export	altpll_c0			
	reset_n	Reset Input	Double-click to export	[clk]			
	data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	IRQ 0	IRQ 31	
	instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
	jtag_debug_module_reset	Reset Output	Double-click to export	[clk]			
	jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0a00_0800	0x0a00_0fff	
	custom_instruction_master	Custom Instruction Master	Double-click to export				

New things learned:

- Since we used an external memory (DRAM) as the memory component the clock given to it may be delayed than the internal clock because of the effect of delays like propagation delays. Therefore we learned that to use the external components we have to consider these delays, and hence the clock was given introducing -65 degrees phase shift to the original clock for the external DRAM.

- How to communicate with different components which use different frequency clocks