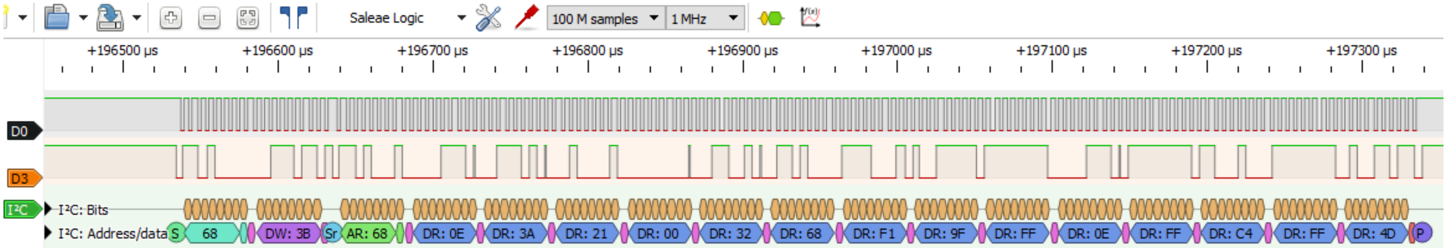
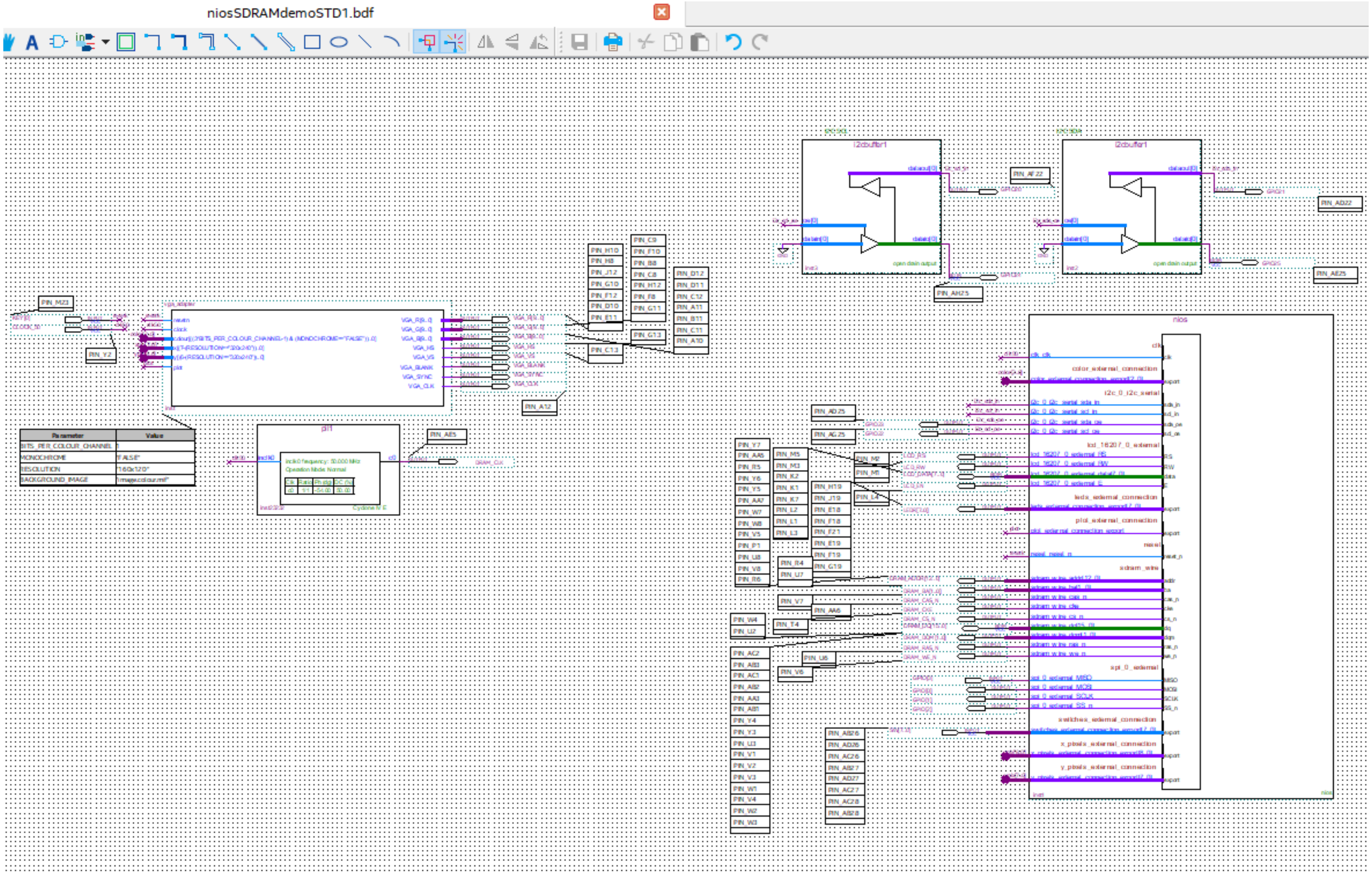
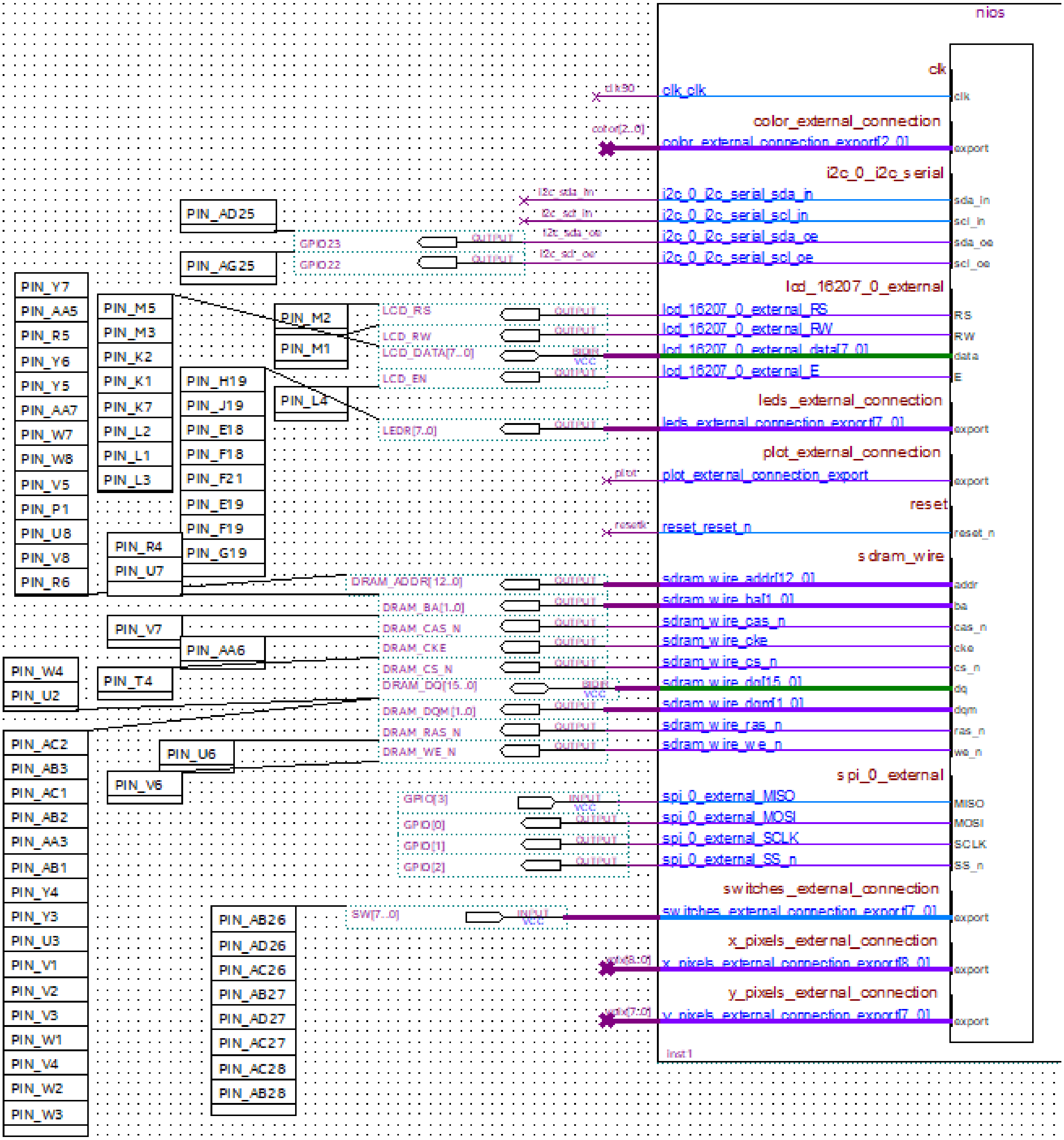
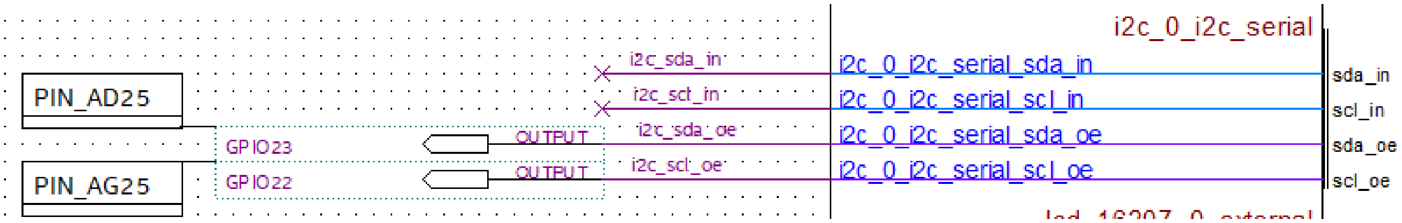
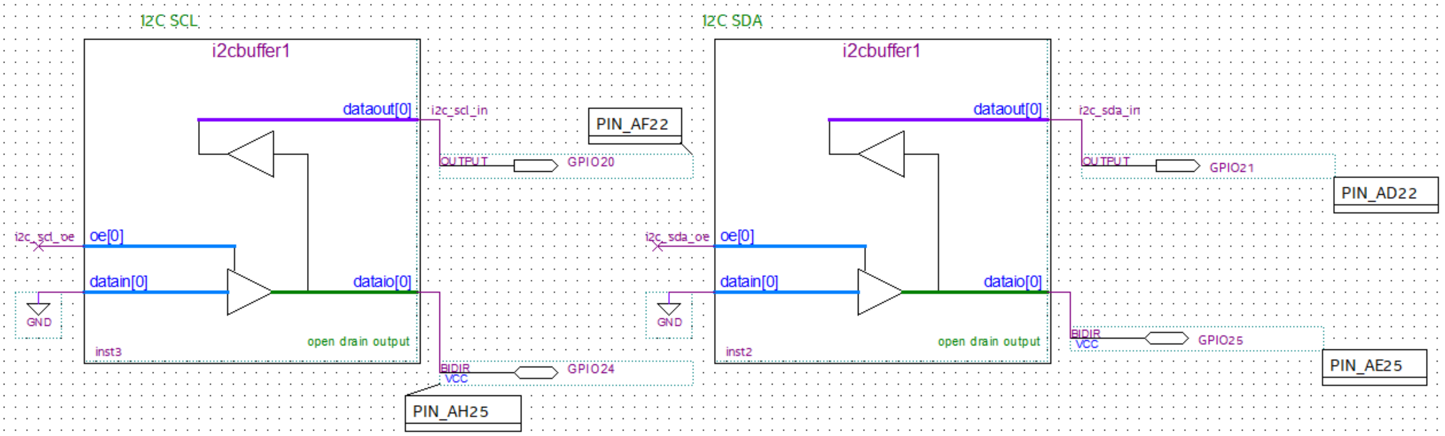


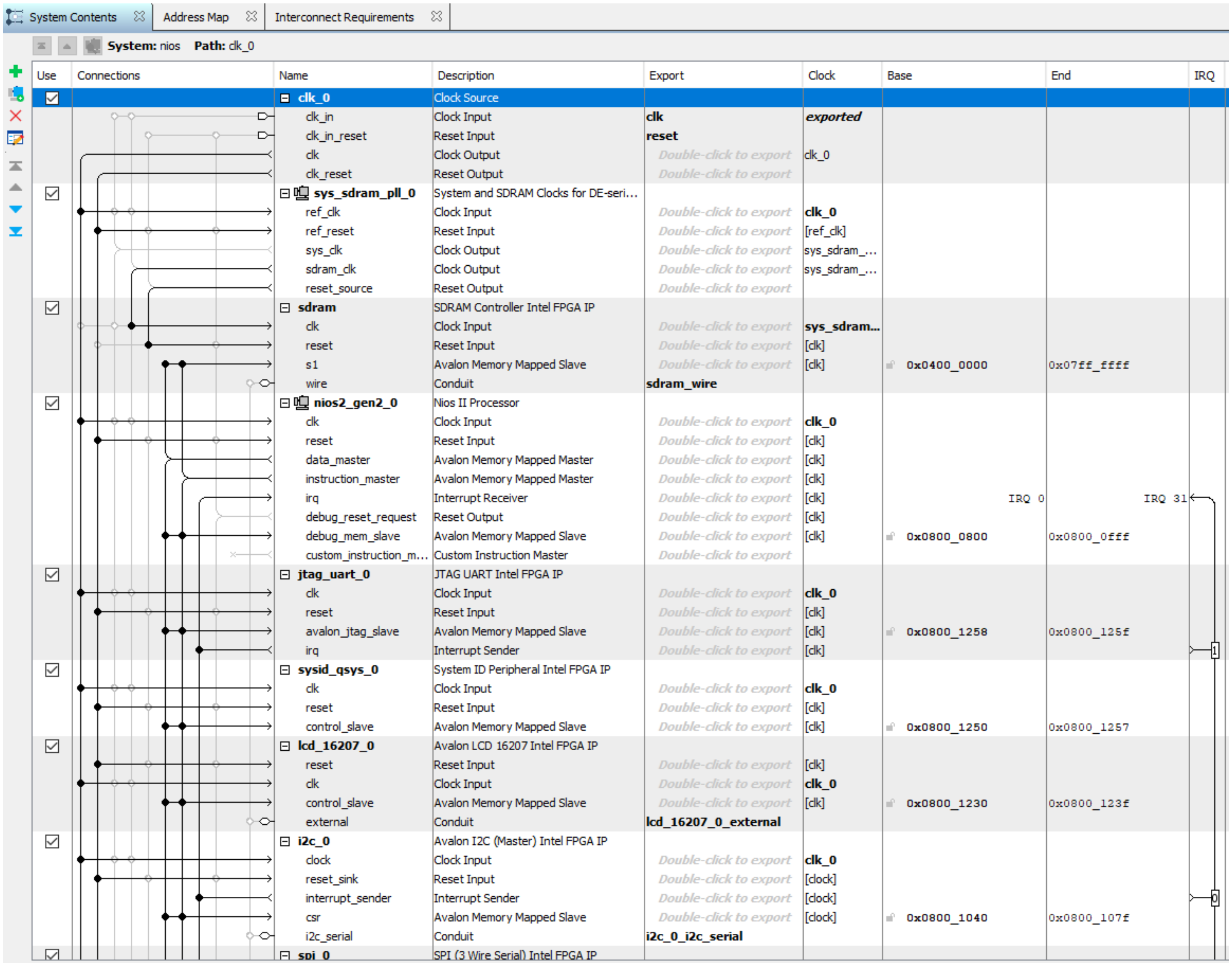


Parameter	Value
BITS_PER_COLOUR_CHANNEL	1
MONOCHROME	"FALSE"
RESOLUTION	"160x120"
BACKGROUND_IMAGE	"image.colour.mif"


nios
clk
clk50
clk_clk
clk
color_external_connection
color[2..0]
color_external_connection_export[2..0]
export
i2c_0_i2c_serial
i2c_sda_in
i2c_scl_in
i2c_sda_oe
i2c_scl_oe
i2c_0_i2c_serial_sda_in
i2c_0_i2c_serial_scl_in
i2c_0_i2c_serial_sda_oe
i2c_0_i2c_serial_scl_oe
sda_in
scl_in
sda_oe
scl_oe
PIN_AD25
PIN_AG25
GPIO23
GPIO22
OUTPUT
OUTPUT
lcd_16207_0_external
LCD_RS
LCD_RW
LCD_DATA[7..0]
LCD_EN
OUTPUT
OUTPUT
BIDIR
VCC
OUTPUT
lcd_16207_0_external_RS
lcd_16207_0_external_RW
lcd_16207_0_external_data[7..0]
lcd_16207_0_external_E
RS
RW
data
E
PIN_Y7
PIN_AA5
PIN_R5
PIN_Y6
PIN_Y5
PIN_AA7
PIN_W7
PIN_W8
PIN_V5
PIN_P1
PIN_U8
PIN_V8
PIN_R6
PIN_M5
PIN_M3
PIN_K2
PIN_K1
PIN_K7
PIN_L2
PIN_L1
PIN_L3
PIN_M2
PIN_M1
PIN_L4
PIN_H19
PIN_J19
PIN_E18
PIN_F18
PIN_F21
PIN_E19
PIN_F19
PIN_G19
leds_external_connection
LEDR[7..0]
OUTPUT
leds_external_connection_export[7..0]
export
plot_external_connection
plot
plot_external_connection_export
export
reset
resetk
reset_reset_n
reset_n
sdram_wire
PIN_R4
PIN_U7
PIN_V7
PIN_AA6
PIN_W4
PIN_U2
PIN_T4
PIN_U6
PIN_V6
DRAM_ADDR[12..0]
DRAM_BA[1..0]
DRAM_CAS_N
DRAM_CKE
DRAM_CS_N
DRAM_DQ[15..0]
DRAM_DQM[1..0]
DRAM_RAS_N
DRAM_WE_N
OUTPUT
OUTPUT
OUTPUT
OUTPUT
OUTPUT
BIDIR
VCC
OUTPUT
OUTPUT
OUTPUT
sdram_wire_addr[12..0]
sdram_wire_ba[1..0]
sdram_wire_cas_n
sdram_wire_cke
sdram_wire_cs_n
sdram_wire_dq[15..0]
sdram_wire_dqm[1..0]
sdram_wire_ras_n
sdram_wire_we_n
addr
ba
cas_n
cke
cs_n
dq
dqm
ras_n
we_n
PIN_AC2
PIN_AB3
PIN_AC1
PIN_AB2
PIN_AA3
PIN_AB1
PIN_Y4
PIN_Y3
PIN_U3
PIN_V1
PIN_V2
PIN_V3
PIN_W1
PIN_V4
PIN_W2
PIN_W3
spi_0_external
GPIO[3]
GPIO[0]
GPIO[1]
GPIO[2]
INPUT
VCC
OUTPUT
OUTPUT
OUTPUT
spi_0_external_MISO
spi_0_external_MOSI
spi_0_external_SCLK
spi_0_external_SS_n
MISO
MOSI
SCLK
SS_n
switches_external_connection
PIN_AB26
PIN_AD26
PIN_AC26
PIN_AB27
PIN_AD27
PIN_AC27
PIN_AC28
PIN_AB28
SW[7..0]
INPUT
VCC
switches_external_connection_export[7..0]
export
x_pixels_external_connection
xpix[8..0]
x_pixels_external_connection_export[8..0]
export
y_pixels_external_connection
ypix[7..0]
y_pixels_external_connection_export[7..0]
export
inst1

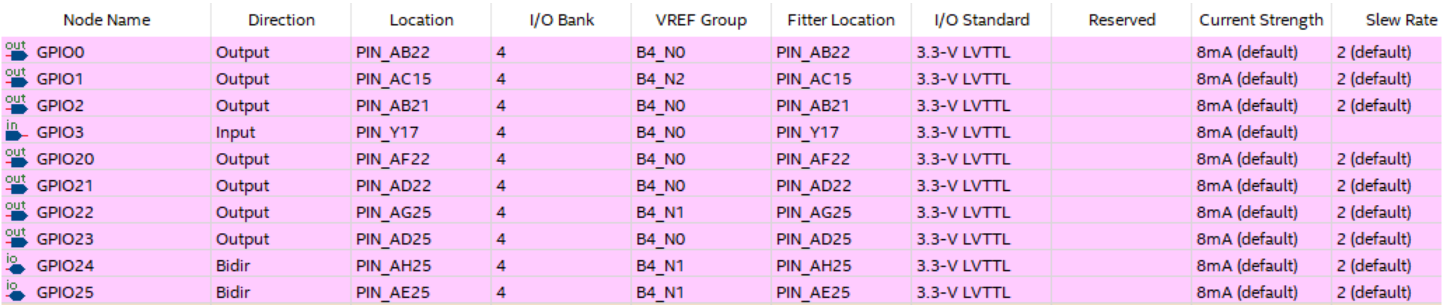

i2c_0_i2c_serial
i2c_sda_in
i2c_scl_in
i2c_sda_oe
i2c_scl_oe
i2c_0_i2c_serial_sda_in
i2c_0_i2c_serial_scl_in
i2c_0_i2c_serial_sda_oe
i2c_0_i2c_serial_scl_oe
sda_in
scl_in
sda_oe
scl_oe
PIN_AD25
PIN_AG25
GPIO23
GPIO22
OUTPUT
OUTPUT
lcd_16207_0_external




System Contents | Address Map | Interconnect Requirements

System: nios **Path:** clk_0

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
☑		⊟ **clk_0**	Clock Source					
		clk_in	Clock Input	**clk**	***exported***			
		clk_in_reset	Reset Input	**reset**				
		clk	Clock Output	*Double-click to export*	clk_0			
		clk_reset	Reset Output	*Double-click to export*				
☑		⊟ **sys_sdram_pll_0**	System and SDRAM Clocks for DE-seri…					
		ref_clk	Clock Input	*Double-click to export*	**clk_0**			
		ref_reset	Reset Input	*Double-click to export*	[ref_clk]			
		sys_clk	Clock Output	*Double-click to export*	sys_sdram_…			
		sdram_clk	Clock Output	*Double-click to export*	sys_sdram_…			
		reset_source	Reset Output	*Double-click to export*				
☑		⊟ **sdram**	SDRAM Controller Intel FPGA IP					
		clk	Clock Input	*Double-click to export*	**sys_sdram...**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0400_0000	0x07ff_ffff	
		wire	Conduit	**sdram_wire**				
☑		⊟ **nios2_gen2_0**	Nios II Processor					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		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 31	
		debug_reset_request	Reset Output	*Double-click to export*	[clk]			
		debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0800_0800	0x0800_0fff	
		custom_instruction_m…	Custom Instruction Master	*Double-click to export*				
☑		⊟ **jtag_uart_0**	JTAG UART Intel FPGA IP					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0800_1258	0x0800_125f	
		irq	Interrupt Sender	*Double-click to export*	[clk]			1
☑		⊟ **sysid_qsys_0**	System ID Peripheral Intel FPGA IP					
		clk	Clock Input	*Double-click to export*	**clk_0**			
		reset	Reset Input	*Double-click to export*	[clk]			
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0800_1250	0x0800_1257	
☑		⊟ **lcd_16207_0**	Avalon LCD 16207 Intel FPGA IP					
		reset	Reset Input	*Double-click to export*	[clk]			
		clk	Clock Input	*Double-click to export*	**clk_0**			
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0800_1230	0x0800_123f	
		external	Conduit	**lcd_16207_0_external**				
☑		⊟ **i2c_0**	Avalon I2C (Master) Intel FPGA IP					
		clock	Clock Input	*Double-click to export*	**clk_0**			
		reset_sink	Reset Input	*Double-click to export*	[clock]			
		interrupt_sender	Interrupt Sender	*Double-click to export*	[clock]			0
		csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0800_1040	0x0800_107f	
		i2c_serial	Conduit	**i2c_0_i2c_serial**				
☑		⊟ **spi_0**	SPI (3 Wire Serial) Intel FPGA IP					


Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
out GPIO0	Output	PIN_AB22	4	B4_N0	PIN_AB22	3.3-V LVTTL		8mA (default)	2 (default)
out GPIO1	Output	PIN_AC15	4	B4_N2	PIN_AC15	3.3-V LVTTL		8mA (default)	2 (default)
out GPIO2	Output	PIN_AB21	4	B4_N0	PIN_AB21	3.3-V LVTTL		8mA (default)	2 (default)
in GPIO3	Input	PIN_Y17	4	B4_N0	PIN_Y17	3.3-V LVTTL		8mA (default)	
out GPIO20	Output	PIN_AF22	4	B4_N0	PIN_AF22	3.3-V LVTTL		8mA (default)	2 (default)
out GPIO21	Output	PIN_AD22	4	B4_N0	PIN_AD22	3.3-V LVTTL		8mA (default)	2 (default)
out GPIO22	Output	PIN_AG25	4	B4_N1	PIN_AG25	3.3-V LVTTL		8mA (default)	2 (default)
out GPIO23	Output	PIN_AD25	4	B4_N0	PIN_AD25	3.3-V LVTTL		8mA (default)	2 (default)
io GPIO24	Bidir	PIN_AH25	4	B4_N1	PIN_AH25	3.3-V LVTTL		8mA (default)	2 (default)
io GPIO25	Bidir	PIN_AE25	4	B4_N1	PIN_AE25	3.3-V LVTTL		8mA (default)	2 (default)

```
/*
 * MPU6050 connection demo, based on Intel "Hello World" example for NiosII. Adapted by Tim Gilmour.
 *
 *  This code uses the Intel I2C controller... for another option of OpenCores core, see
 *       https://community.intel.com/t5/FPGA-Intellectual-Property/I2C-OpenCores-not-working/m-p/708254
 *         (For a different approach that uses VHDL for the I2C state machine, see
 *     https://github.com/danomora/mpu6050-vhdl/
 *
 *       See also https://github.com/alex-mous/MPU6050-C-CPP-Library-for-Raspberry-Pi/blob/master/MPU6050.cpp for example
 *       also https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
 *       and https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/RM-MPU-6000A.pdf (register map)
 *  and see the ug_embedded_ip pdf for Intel FPGA Avalon I2C (Host) Core API documentation and example code
 *
 *  Tips: put this Eclipse project and BSP project onto the C drive, not a network drive...
 */

#include <sys/alt_stdio.h>
#include <stdio.h>
#include "altera_avalon_pio_regs.h"
#include <unistd.h>
#include <system.h>
#include <stdlib.h>
#include <string.h>
#include "altera_avalon_i2c.h"
#include "io.h"


int main() {
        ALT_AVALON_I2C_DEV_t *i2c_dev; //pointer to instance structure
        ALT_AVALON_I2C_STATUS_CODE status;
        ALT_AVALON_I2C_MASTER_CONFIG_t cfg;
        alt_u8 txbuffer[0x200];
        alt_u8 rxbuffer[0x200];
        char in, out;
        int16_t X_accel, Y_accel, Z_accel, temperature, X_gyro, Y_gyro, Z_gyro;


        //get a pointer to the Avalon I2C Host Controller instance
        i2c_dev = alt_avalon_i2c_open("/dev/i2c_0");
        if (NULL == i2c_dev) {
                printf("Error: Cannot find /dev/i2c_0\n");
                return 1;
        } else {
                printf("Opened /dev/i2c_0 \n");
        }

        printf("Configuring MPU6050...");

        alt_avalon_i2c_master_config_get(i2c_dev, &cfg);
        // need to change the following line in the altera_avalon_i2c.h if you want to use 400 kHz:
        //  #define ALT_AVALON_I2C_DIFF_LCNT_HCNT 30 // 60 for 100kHz, 15 for 400 kHz, 30 for 200 kHz
        alt_avalon_i2c_master_config_speed_set(i2c_dev, &cfg, 200000);
        alt_avalon_i2c_master_config_set(i2c_dev, &cfg);

       //set the address of the device (MPU6050 has address 0x68 or 0x69 depending on ADDRESS pin)
       alt_avalon_i2c_master_target_set(i2c_dev, 0x68);

        usleep(1000);
        txbuffer[0] = 0x6b;  txbuffer[1] = 0x00; // power management: turn off sleep mode
        status = alt_avalon_i2c_master_tx(i2c_dev, txbuffer, 2, ALT_AVALON_I2C_NO_INTERRUPTS);

        usleep(1000);
        txbuffer[0] = 0x1a;  txbuffer[1] = 0x03; // frequency config
        status = alt_avalon_i2c_master_tx(i2c_dev, txbuffer, 2, ALT_AVALON_I2C_NO_INTERRUPTS);

        usleep(1000);
        txbuffer[0] = 0x19;  txbuffer[1] = 0x04; // sample rate
        status = alt_avalon_i2c_master_tx(i2c_dev, txbuffer, 2, ALT_AVALON_I2C_NO_INTERRUPTS);

        usleep(1000);
        txbuffer[0] = 0x1b;  txbuffer[1] = 0x00; // gyro config
        status = alt_avalon_i2c_master_tx(i2c_dev, txbuffer, 2, ALT_AVALON_I2C_NO_INTERRUPTS);

        usleep(1000);
        txbuffer[0] = 0x1c;  txbuffer[1] = 0x00; // accel config
        status = alt_avalon_i2c_master_tx(i2c_dev, txbuffer, 2, ALT_AVALON_I2C_NO_INTERRUPTS);

        printf("finished.\n");
        usleep(5000);

        while (1)
        {

                in = IORD_ALTERA_AVALON_PIO_DATA(SWITCHES_BASE); // for debugging only
                out = in;
                IOWR_ALTERA_AVALON_PIO_DATA(LEDS_BASE, out);

                //Read back the data into rxbuffer
                //This command sends the register address, then does a restart and receives the data.
                txbuffer[0] = 0x3B; // read accel_xout_H, accel_xout_L, accel_yout_H, etc
                status = alt_avalon_i2c_master_tx_rx(i2c_dev, txbuffer, 1, rxbuffer, 14, ALT_AVALON_I2C_NO_INTERRUPTS);
                if (status != ALT_AVALON_I2C_SUCCESS) {
                        printf("Error after alt_avalon_i2c_master_tx_rx: %d \n", status);
                } else {
                        //printf("%02X %02X %02X %02X %02X %02X \n", rxbuffer[0],
                                        rxbuffer[1], rxbuffer[2], rxbuffer[3], rxbuffer[4], rxbuffer[5] );

                        X_accel = rxbuffer[0] << 8 | rxbuffer[1];
                        Y_accel = rxbuffer[2] << 8 | rxbuffer[3];
                        Z_accel = rxbuffer[4] << 8 | rxbuffer[5];
                        temperature = rxbuffer[6] << 8 | rxbuffer[7]; // broken into separate steps for debugging,
                        temperature = ~temperature + 1;               // only using a small amount of the precision,
                                                                      // could extract more if needed
                        temperature = 37 - (temperature / 340);       // see the datasheet & register map
                        X_gyro = rxbuffer[8] << 8 | rxbuffer[9];
                        Y_gyro = rxbuffer[10] << 8 | rxbuffer[11];
                        Z_gyro = rxbuffer[12] << 8 | rxbuffer[13];
                        printf("%d,%d,%d,%d,%d,%d,%d\n", X_accel, Y_accel, Z_accel, temperature, X_gyro, Y_gyro, Z_gyro);
                }

                usleep(100000);
        }

        return 0;
}
```