

# NIOS SPI

Last updated 10/12/20

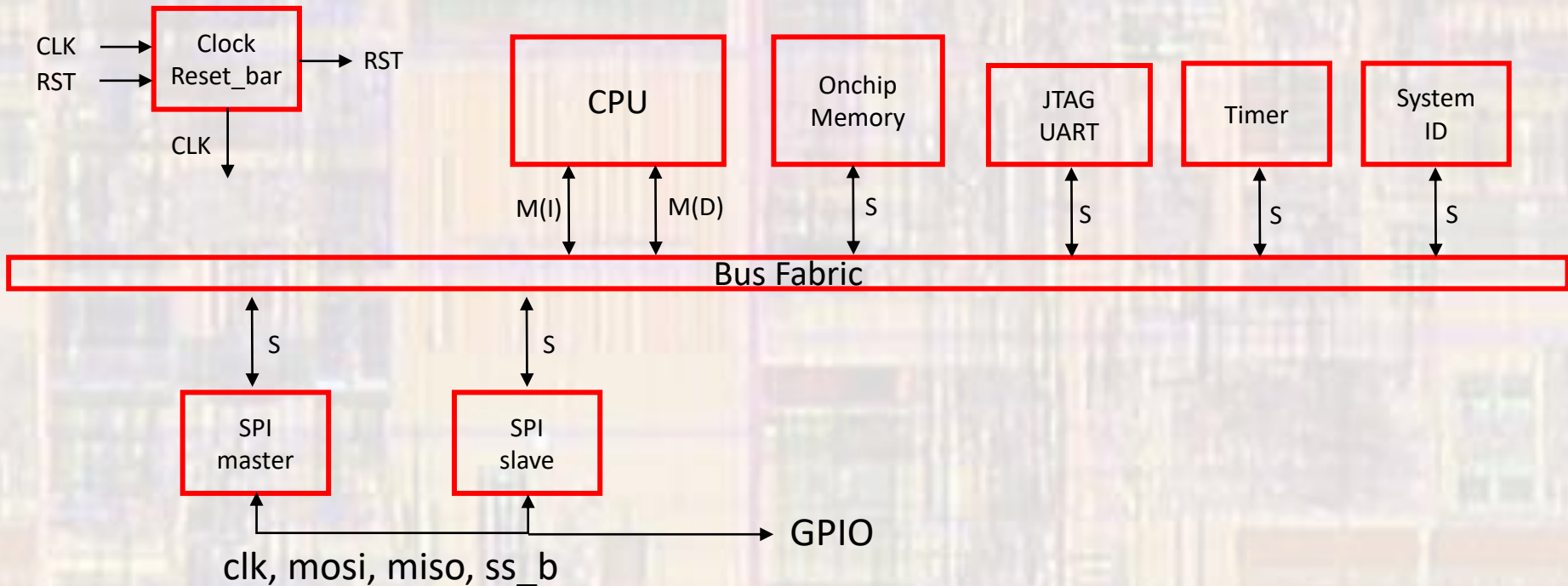
# NIOS SPI

These slides describe the operation of the Serial Peripheral Interface (SPI) in a NIOS system

Upon completion: You should be able to develop your own NIOS based SPI System

# NIOS SPI

- NIOS SPI System
  - Use 2 SPI modules, in loopback mode



# NIOS SPI

The screenshot shows the 'System Contents' window for a NIOS system named 'nios\_spi'. The path is 'clk\_0'. The window displays a tree view of components and their connections, a table of component properties, and a messages pane at the bottom.

Use	Connections	Name	Description	Export	Clock	Base	E
<input checked="" type="checkbox"/>		irq	Interrupt Sender	<i>Double-click to export</i>	[clk]		^
<input checked="" type="checkbox"/>		<b>timer_0</b>	Interval Timer Intel FPGA IP	<i>Double-click to export</i>	<b>clk_0</b>		
		clk	Clock Input	<i>Double-click to export</i>	[clk]		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		s1	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	#' 0x0001_1040	0
		irq	Interrupt Sender	<i>Double-click to export</i>	[clk]		
<input checked="" type="checkbox"/>		<b>sysid_qsys_0</b>	System ID Peripheral Intel FPGA IP	<i>Double-click to export</i>	<b>clk_0</b>		
		clk	Clock Input	<i>Double-click to export</i>	[clk]		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		control_slave	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	#' 0x0001_1060	0
<input checked="" type="checkbox"/>		<b>spi_master</b>	SPI (3 Wire Serial) Intel FPGA IP	<i>Double-click to export</i>	<b>clk_0</b>		
		clk	Clock Input	<i>Double-click to export</i>	[clk]		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		spi_control_port	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	#' 0x0001_1020	0
		irq	Interrupt Sender	<i>Double-click to export</i>	[clk]		
		external	Conduit	<b>spi_master_external</b>			
<input checked="" type="checkbox"/>		<b>spi_slave</b>	SPI (3 Wire Serial) Intel FPGA IP	<i>Double-click to export</i>	<b>clk_0</b>		
		clk	Clock Input	<i>Double-click to export</i>	[clk]		
		reset	Reset Input	<i>Double-click to export</i>	[clk]		
		spi_control_port	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clk]	#' 0x0001_1000	0
		irq	Interrupt Sender	<i>Double-click to export</i>	[clk]		
		external	Conduit	<b>spi_slave_external</b>			

Current filter:

Library/Interface Protocols/Serial/SPI(3 wire)...

Type	Path	Message
3 Info Messages		
Info	nios_spi.jtag_uart_0	JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board
Info	nios_spi.sysid_qsys_0	System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID
Info	nios_spi.sysid_qsys_0	Time stamp will be automatically updated when this component is generated.

Generate HDL... Finish

# NIOS SPI

- NIOS SPI System

The screenshot shows the 'Parameters' window for the 'SPI (3 Wire Serial) Intel FPGA IP'. The window title is 'Parameters' and the system path is 'nios\_spi spi\_slave'. The IP core is identified as 'altera\_avalon\_spi'. The 'Master/Slave' section is expanded, showing the following parameters:

- Type: Slave
- Number of select (SS\_n) signals (one for each slave): 1
- SPI clock (SCLK) rate: 128000 Hz
- Actual clock rate: 127551.0 Hz
- Specify delay:
- Target delay: 0.0 ns
- Actual delay: 0.0 ns

The 'Data register' section is also expanded, showing:

- Width: 8 bits
- Shift direction: MSB first

The 'Timing' section is expanded, showing:

- Clock polarity: 0
- Clock phase: 0

The 'Synchronizer Stages' section is expanded, showing:

- Insert Synchronizers:
- Depth: 2

A red circle highlights the 'SCLK rate' and 'Actual clock rate' fields. A callout box points to this circle with the text: 'Creates the required divide by circuitry in the Master'.

Creates the required divide by circuitry in the Master

# NIOS SPI

- NIOS SPI System

```
-----  
-- nios_spi_de10.vhd1  
-- Created 9/18/18  
-- by: johnsontimoj  
-- rev: 0  
-----  
--  
-- Basic Nios system - with spi - loopback  
-----  
  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity nios_spi_de10 is  
    port(  
        CLOCK_50 : in std_logic;  
        GPIO : out std_logic_vector(3 downto 0)  
    );  
end entity;
```

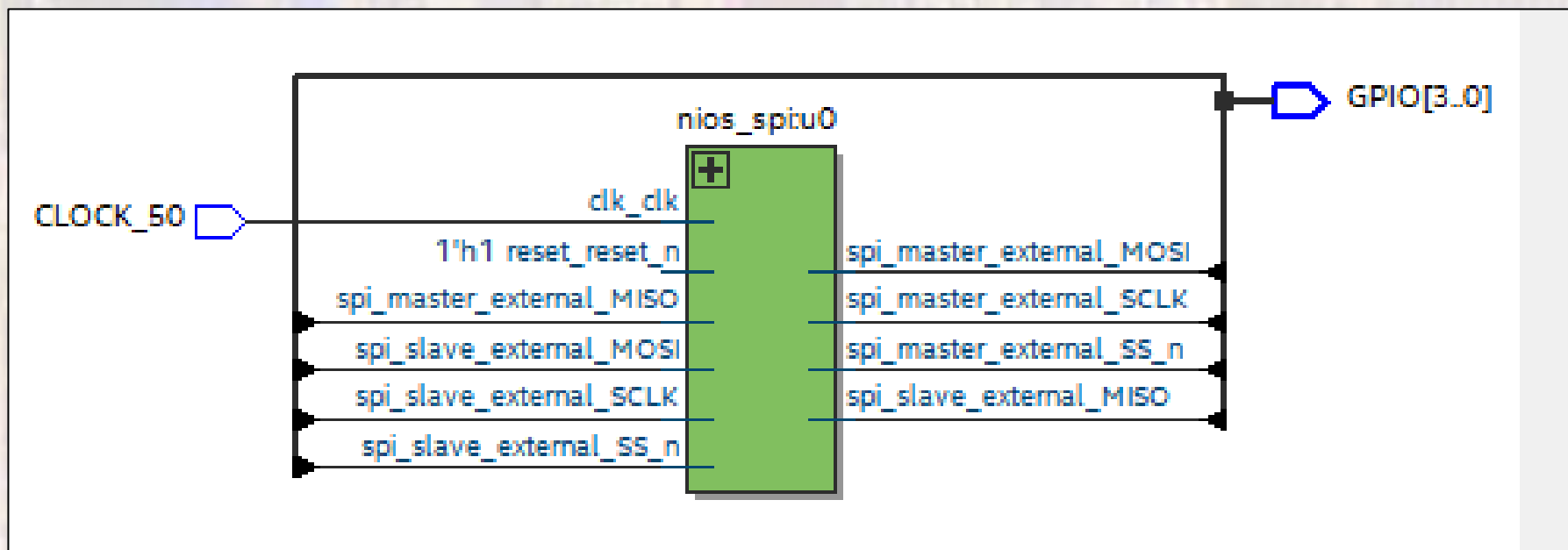
```
architecture behavioral of nios_spi_de10 is  
  
    component nios_spi is  
        port (  
            clk_clk : in std_logic := 'X'; -- clk  
            reset_reset_n : in std_logic := 'X'; -- reset_n  
            spi_slave_external_MISO : out std_logic; -- MISO  
            spi_slave_external_MOSI : in std_logic := 'X'; -- MOSI  
            spi_slave_external_SCLK : in std_logic := 'X'; -- SCLK  
            spi_slave_external_SS_n : in std_logic := 'X'; -- SS_n  
            spi_master_external_MISO : in std_logic := 'X'; -- MISO  
            spi_master_external_MOSI : out std_logic; -- MOSI  
            spi_master_external_SCLK : out std_logic; -- SCLK  
            spi_master_external_SS_n : out std_logic; -- SS_n  
        );  
    end component nios_spi;  
  
    signal MISO : std_logic;  
    signal MOSI : std_logic;  
    signal SCLK : std_logic;  
    signal SS_bar : std_logic;  
  
begin  
  
    u0 : component nios_spi  
        port map (  
            clk_clk => CLOCK_50, -- clk_clk  
            reset_reset_n => '1', -- reset.reset_n  
            spi_slave_external_MISO => MISO, -- spi_slave_external.MISO  
            spi_slave_external_MOSI => MOSI, -- .MOSI  
            spi_slave_external_SCLK => SCLK, -- .SCLK  
            spi_slave_external_SS_n => SS_bar, -- .SS_n  
            spi_master_external_MISO => MISO, -- spi_master_external.MISO  
            spi_master_external_MOSI => MOSI, -- .MOSI  
            spi_master_external_SCLK => SCLK, -- .SCLK  
            spi_master_external_SS_n => SS_bar -- .SS_n  
        );  
  
        GPIO(0) <= MISO;  
        GPIO(1) <= MOSI;  
        GPIO(2) <= SCLK;  
        GPIO(3) <= SS_bar;  
  
end architecture;
```

loopback signals

access for Analog Discovery

# NIOS SPI

- NIOS SPI System



# NIOS SPI

- NIOS SPI System

```
nios_spi_1.c altera_avalon_spi_regs.h
33
34 #include <io.h>
35
36 #define ALTERA_AVALON_SPI_RXDATA_REG 0
37 #define IOADDR_ALTERA_AVALON_SPI_RXDATA(base) __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_SPI_RXDATA_REG)
38 #define IORD_ALTERA_AVALON_SPI_RXDATA(base) IORD(base, ALTERA_AVALON_SPI_RXDATA_REG)
39 #define IOWR_ALTERA_AVALON_SPI_RXDATA(base, data) IOWR(base, ALTERA_AVALON_SPI_RXDATA_REG, data)
40
41 #define ALTERA_AVALON_SPI_TXDATA_REG 1
42 #define IOADDR_ALTERA_AVALON_SPI_TXDATA(base) __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_SPI_TXDATA_REG)
43 #define IORD_ALTERA_AVALON_SPI_TXDATA(base) IORD(base, ALTERA_AVALON_SPI_TXDATA_REG)
44 #define IOWR_ALTERA_AVALON_SPI_TXDATA(base, data) IOWR(base, ALTERA_AVALON_SPI_TXDATA_REG, data)
45
46 #define ALTERA_AVALON_SPI_STATUS_REG 2
47 #define IOADDR_ALTERA_AVALON_SPI_STATUS(base) __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_SPI_STATUS_REG)
48 #define IORD_ALTERA_AVALON_SPI_STATUS(base) IORD(base, ALTERA_AVALON_SPI_STATUS_REG)
49 #define IOWR_ALTERA_AVALON_SPI_STATUS(base, data) IOWR(base, ALTERA_AVALON_SPI_STATUS_REG, data)
50
51 #define ALTERA_AVALON_SPI_STATUS_ROE_MSK (0x8)
52 #define ALTERA_AVALON_SPI_STATUS_ROE_OFST (3)
53 #define ALTERA_AVALON_SPI_STATUS_TOE_MSK (0x10)
```



# NIOS SPI

- NIOS SPI System

```
/*
 * nios_spi_1.c
 *
 * Created on: Oct 22, 2018
 * Author: johnsontimoj
 */
////////////////////////////////////
//
// SPI with NIOS loopback example - no spi_command
//
////////////////////////////////////

#include "alt_types.h"
#include "altera_avalon_spi_regs.h"
#include "altera_avalon_spi.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h> // usleep

int main(void) {
    printf("Entered Main\n");
    alt_u8 send_M;
    alt_u8 send_S;
    alt_u8 rcvd_S;
    alt_u8 rcvd_M;
    alt_u8 count;
    count = 0;
```

```
while(1) {
    send_M = count;
    send_S = 255 - count;

    // Load Slave TX buffer THEN load Master TX buffer
    // loading master TX causes transmission to start
    //
    IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_SLAVE_BASE, send_S);
    IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_MASTER_BASE, send_M);

    // Wait for transfer to complete
    //
    usleep(75);

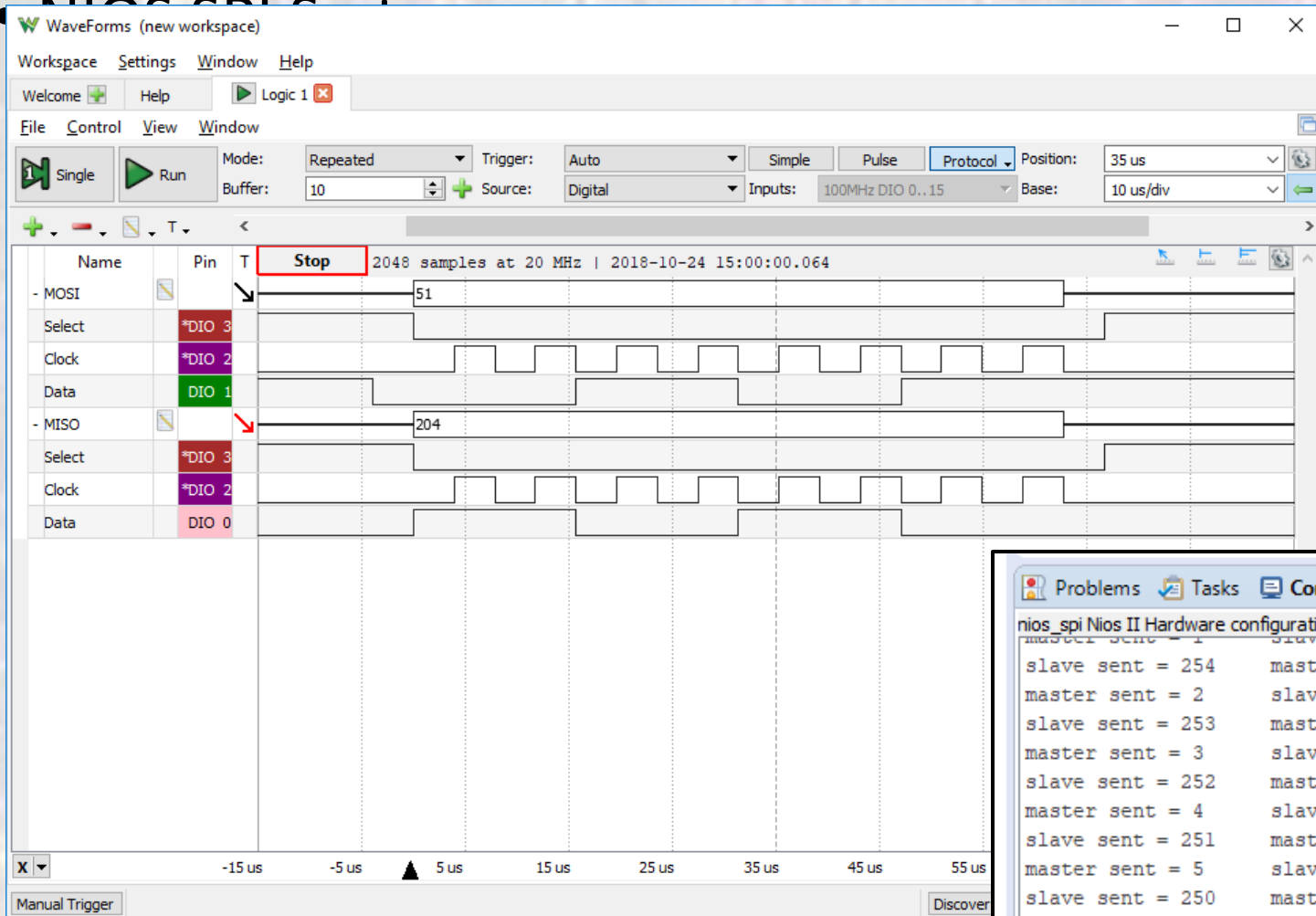
    // Read Master and Slave RX buffers
    //
    rcvd_S = IORD_ALTERA_AVALON_SPI_RXDATA(SPI_SLAVE_BASE);
    rcvd_M = IORD_ALTERA_AVALON_SPI_RXDATA(SPI_MASTER_BASE);

    // Print results
    printf("master sent = %i\tslave rcv'd = %i\n", send_M, rcvd_S);
    printf("slave sent = %i\tmaster rcv'd = %i\n", send_S, rcvd_M);

    // Setup for next loop
    count ++;
    usleep(1000000);
} // end while

return 0;
} // end main
```

# NIOS SPI



```
Problems Tasks Console Nios II Console
nios_spi Nios II Hardware configuration - cable: USB-Blaster on loc
master sent = 1 slave recv'd = 1
slave sent = 254 master recv'd = 254
master sent = 2 slave recv'd = 2
slave sent = 253 master recv'd = 253
master sent = 3 slave recv'd = 3
slave sent = 252 master recv'd = 252
master sent = 4 slave recv'd = 4
slave sent = 251 master recv'd = 251
master sent = 5 slave recv'd = 5
slave sent = 250 master recv'd = 250
```