Design of a SPI IP Core and Linux Device Drivers

CSE 4356 - System On Chip Design

Deborah Jahaj
Nathan Fusselman
Dr. Losh
December 6th, 2021

Introduction

This project implements a SPI IP module on a Cyclone V FPGA, capable of being controlled and configured from the hard processor subsystem over an Avalon memory-mapped interface. Additionally, this project involved the implementation of Linux kernel modules to allow a user to write software to interface with the SPI IP module.

(Include General Photos: stop_go board, SoC, AD2, Platform Designer)

Theory of Operation

Memory Map Register

Top interface with the SPI IP module between the FPGA and HPS portions of the SoC Chip we use an Intel Avalon Memory Map Interface. This interface allows the system to map all memory and physical hardware to a memory address and allows us to create our own memory location at an offset 0x00008000 from the Avalon MM base address of 0xFF200000. With this we can then open a file to the devices memory and then read and write to portions of this memory that fall within our devices aperture.

Functions

This device will support many different functions and features to be able to interface with multiple SPI devices simultaneously. These functions are as follows.

SPI TX

As expected this SPI IP module must be capable of transmitting data to multiple SPI devices.

SPI RX

Additionally this SPI IP module must be able to receive incoming data from any device that it is connected to.

MODE

This SPI module shall be capable of supporting all possible SPI modes 00, 11, and the less common 01 and 10. This mode shall be selectable for each device individually meaning each of the 4 supported devices can be assigned its own unique mode as to prevent a user from having to switch the mode often.

CS_AUTO

The SPI module must be able to automatically assert and de-assert the CS line for the device being used so that the module triggers the CS line a clock before and releases the line after the transmutation is completed. When this is disabled a user can manually asserted by using the CS_ENABLE function of the SPI module.

WORD_SIZE

Our SPI module is designed to allow a user to send any word size between 1-32 bits as to work most effectively with a wide range of devices. This word size is global for the entire module and is not specific to each device. Therefor you must change it between transmutations if two devices use different word sizes.

BRD

The Baud Rate Divider that generates the SPI Baud Rate must be capable of transmitting data to the connected devices at any frequency between 1Hz-25MHz with a percent error of no greater than 0.1%.

CS_SELECT

To allow the user to select multiple devices a user can enter the device number that would like to be selected in this register. This is what CS will be automatically asserted for the transmission.

16 Level/Count FIFO

As to allow the user to send multiple transmutations in close succession and to allow for slow baud rates to work we have implemented a FIFO for the TX and RX functionality. This will allow the user to not need to wait for the device to become not busy.

FIFO Status

To allow the user to check on the status of each FIFO we will output a bit to represent if the FIFO is full, empty, and if it has overflown. These bits are available for each FIFO.

Module Status

To allow the use to enable and disable the entire module we has implemented a bit to enable and disable the module as a whole. This will stop the baud rate from being generated and will then cause the device to not transmit or receive any data. However the device will be capable of queuing transmissions in the TX FIFO and setting up settings.

MCP20S08

To demo this SPI module functioning software has been written to allow for the user to test and interface with MCP20S08 GPIO SPI Expander. This allows for us to interface with buttons and LEDs from a remote device with the use of a SPI interface.

HPS Virtual File System Control

To assist with potential development that uses the SPI module that has been created we have created and tested two SPI loadable kernel modules. These allow a user space user to read and write to pre-made files making controlling the SPI interface simpler for a high level user.

Kernel Tree for SPI Driver

```
/sys/kernel/spi/
├── baud_rate
├── word_size
├── cs_select
├── device0/
│   ├── mode0
│   ├── cs_auto0
│   └── cs_enable0
├── device1/
│   ├── mode1
│   ├── cs_auto1
│   └── cs_enable1
├── device2/
│   ├── mode2
│   ├── cs_auto2
│   └── cs_enable2
├── device3/
│   ├── mode3
│   ├── cs_auto3
│   └── cs_enable3
├── tx_data
└── rx_data
```

SPI Bus Expander Control

Similar to the SPI module we have created a kernel module to allow the user to easily interface with the SPI Expander. This supports the 3 most common functions including Direction, Pull-Up, and pin Data. These files allow a High level developer to read and write to specific files for each function rather than having to send specific commands to the SPI device.

Kernel Tree for GPIO Expander

```
/sys/kernel/spi-expander/
├── pin0/
│   ├── dir0
│   ├── pullup0
│   └── data0
├── pin1/
│   ├── dir1
│   ├── pullup1
│   └── data1
├── pin2/
│   ├── dir2
│   ├── pullup2
│   └── data2
├── pin2/
│   ├── dir2
│   ├── pullup2
│   └── data2
├── pin3/
│   ├── dir3
│   ├── pullup3
│   └── data3
├── pin4/
│   ├── dir4
│   ├── pullup4
│   └── data4
├── pin5/
│   ├── dir5
│   ├── pullup5
│   └── data5
├── pin6/
│   ├── dir6
│   ├── pullup6
│   └── data6
├── pin7/
│   ├── dir7
│   ├── pullup7
│   └── data7
```

Setup Environment

To demonstrate and test this module we have compiled a list of instruction that a user can use.

Program Board

To begin you must power on the DE-1 SoC board and program it. This can be done by opening the file in /FPGA/soc_system.qpf and then within the project opening the programer at Project>Programmer. Here everything has been setup and you can just hit Start. This will program the board and you will know if it is successful by seeing the characters GS in the lower left HEX5 and HEX4 displays.

SSH into Board

To then connect to the terminal for the board you can connect the device to the computer using ethernet. Then you can ssh into the device at root@192.168.1.123 with the password “password”. With this you can navigate to the folder that you have place the contents of the /HPS/ directory.

Connect SPI Device

To connect the SPI bus to the devices desired you can use the following pin map below.

SPI_TX: GPIO0 (pin8)

SPI_RX: GPIO0 (pin9)

SPI_CLK: GPIO0 (pin10)

SPI_CS0: GPIO0 (pin11)

SPI_CS1: GPIO0 (pin12)

SPI_CS2: GPIO0 (pin13)

SPI_CS3: GPIO0 (pin14)

Setup/Connect AD2 (Pins)

To be able to monitor the bus with the AD2 provided in this class you can connect DI0-6 to GPIO0 (pin8-14) like above and configure the AD2 for each respective pins function.

Available Programs

In the provided .zip file that this is contained in you will find some .c files with compiled .o and .ko files for the DE-1 SoC board in the /HPS/ directory.

/SPI/spi.c

This program allows you to interface with the SPI IP module using a CLI application. You can see a list of all commands by running ./spi -h in the /SPI/ directory.

/SPI/MCP20S08/stop_go.c

This program tests the SPI GPIO Expander by running a simple Stop Go example. This will start by setting only the Red LED on with the circuit that we use and then upon pressing the pushbutton it will change to illuminating the Green LED. This shows outputs and inputs, it also shows pull-up resistors, and finally it shows that we can poll the GPIO pins through SPI.

/SPI/MCP20S08/gpio.c

This program allows you to interface with the SPI GPIOP Expander using a CLI application. You can see a list of all commands by running ./gpio -h in the /SPI/ MCP02S08/ directory.

/SPI/spi_driver.ko

This kernel module will allow you to interface with the SPI IP module using the file directory provided above. This module can be loaded into the system with the command “insmod spi_driver.ko” while in the /SPI/ directory. Additionally it can be removed with “rmmod spi_driver.ko”.

/SPI/MCP20S08/gpio_expander_driver.ko

This kernel module will allow you to interface with the MCP20S08 SPI Expander using the file directory provided above. This module can be loaded into the system with the command “insmod gpio_expander_driver.ko” while in the /SPI/MCP20S08/ directory. Additionally it can be removed with “rmmod gpio_expander_driver.ko”.

Conclusion

This project introduces us to the processes in creating a custom SoC chip by designing Verilog designs for a Development SoC board with an FPGA as the I/O side of the chip as well as shows us how to interface and write low level code to support these modules that we create. This was done with the creation of a SPI module so a device is able to interface with others in a fast and commonly using protocol and interface.