Hybrid FPGA/SOC Wireless Transmission

User Manual

Project Overview

This application is used to transmit data from a FPGA to a PC through WiFi (via a WiFi capable MCU, the CC3200 from TI), up to 8 Mbits/sec.

A FPGA receive data from several sensors, store them in memory and transmit them to a Microcontroller (MCU) through a SPI connexion. The MCU first connect itself to an Access point (WiFi) and then to a receiving server (in our case a python app). Afterward, the MCU start accepting the SPI data sent from the FPGA and store them in a Ring Buffer. Once the ring buffer is full enough, the MCU send data to a computer through the TCP/IP protocol. Meanwhile, it still receives data from the SPI. These parallel processes (SPI reception, TCP transmission) repeat endlessly as long as the (socket) connexion between the MCU and the PC stays alive.

Components and Software Used:

OS
- Windows (all tools are available on Linux)

FPGA
- Altera DE2-115
- Quartus II 13.1

SOC
- Ti CC3200 launchpad
- Code Composer Studio

Debug
- Putty
- Logic analyser

PC Receiver
- Linux (netcast)
- Python

Useful documents:

CC3200 Getting Started
Full Name: CC3200 SimpleLink Wi-Fi and IoT Solution With MCU LaunchPad Getting Started Guide User's Guide

CC3200 Launchpad User’s guide (hardware review)
CC3200 SimpleLink™ Wi-Fi® and IoT Solution with MCU LaunchPad Hardware User's Guide

CC3200 Reference Manuals (full doc about the chip peripheral)
CC3200 SimpleLink Wi-Fi and Internet-of Things Solution, a Single Chip Wireless MCU Technical Reference Manual

CC3200 SimpleLink User’s Guide
CC3100/CC3200 SimpleLink™ Wi-Fi® Internet on-a-Chip User's Guide

CC3200 Programmer’s guide
CC3200 SimpleLink ™ Wi-Fi ® and IoT Solution, a Single Chip Wireless MCU Programmer 's Guide

CC3200SDK documentation and examples (inside the SDK after download)

Uniflash V4 Quick guide (online resource)
CCS UniFlash v3.4.1 Release Notes (wiki)

Software installation

FPGA side

Quartus II:
Quartus II is the IDE used to develop the configuration (Verilog) and program (write the Ram) the FPG.

Quartus II web edition 13.1 Download (several method available)
http://dl.altera.com/13.1/?edition=web
The version 13.1 is used here for in the newer versions, a delay of 10 seconds has been implemented (for the free version) for each “analysis and elaboration” (pre compilation for simulation), which is slightly inconvenient
For the installation with the intention of using the DE2-115 board, the following components are required:

Quartus II Web Edition (Free)
- Quartus II Software
- ModelSim-Altera Edition (optional)
- Cyclone III, Cyclone IV device support

after installation (following Altera's guidelines). Open the Altera project in the file (Fpga_Soc_Interface.zip) Fpga_Soc_Interface.qfp with Quartus. On the file explorer, right click on top.v and set as top level entity. Compile it.

SOC (CC3200 LaunchPad) side:

Please refer to CC3200 Getting Started guide for deeper details:

Code Composer Studio (CCS):
CCS has been used for that project rather than the open sources tools (Make, GCC, GDB, and openOCD). The open method has been tested. It worked for few examples and generally the ones that didn't involve WiFi transmission and RTOS. More complex projects revealed to have a very limited support and open source debug tools were very limited too, hence unfortunately not well adapted for that project.

CCS Download:
http://processors.wiki.ti.com/index.php/Download_CCS

The installation is straightforward, thus not detailed.

Once CCS installed, We need to download the CC3200 SDK.
http://www.ti.com/tool/download/CC3200SDK

Opening and compiling the program:
Launch CCS and select a folder
Add the components (see Linux manual)
Jumpers
TI uniflash
debug mode
Putty

Set-Up and Testing

Debug tools:

By the nature of this project debugging through serial connection (USART) or any other heavily intrusive tool is not possible. Indeed the SOC's CPU is under heavy load during the transmission of the data because of the high throughout and the frequent interrupts. That is why tools as non-intrusive as possible has been used. Among them, logic analyser, data dump, breakpoints, core dump, register and memory read (in parallel to the CPU load) has been used. The logic analyser solution has been proved the most convenient one with memory read. The logic analyser used is the USB device from Saleae (Logic pro 16, 16 channels with 100 Msamples/sec on 4 channels).

Router Access point:

To simplify the communication and avoid protocol collision as much as possible, a dedicated access point has been used to make the bridge between the SOC and the receiver PC running the python program. The access point during the test wasn't connected to the internet in order to limit protocol collisions and minimise the error rate. The access point was configured in WPA2 password protection.
This application at the moment works with fixed IP address for the Server, where the emitting device must be aware of the IP address to send packets.

PC receiver side

The data reception and test has been made with Python (2.7).

Connexions

The link between the DE2-115 (FPGA) and the CC3200 launchpad (SOC) uses the SPI protocol through simple jump wires between the FPGA pins. 4 wires are dedicated for the SPI (SCLK, MISO, MOSI, and NSS), 1 wire for the SpiBlock signal, and 3 wires to connect the ground of the two boards (essential to maintain signal stability).

MCU ROM Flashing:

Most of the Tests done the MCU (CC3200 from TI) have been made by writing the compiled program under its binary form to the RAM of the chip. This method enables us to quickly program and debug the chip with the debug tools integrated to Code Composer Studio. However it obviously restrains since each time we power-up the device we need to reprogram it due to the volatile nature of the RAM.

On way to program durably the chip is to burn the code in its flash memory. For that, we need to use external tools not included to Code Composer Studio. Several tools can be used for that purpose (uniflash, OpenOCD, Serial Programing using the serial boot loader, and others...). In this manual, we will discuss about the Uniflash software from TI, which is the simplest solution.

Uniflash (for CC3200) download link:
http://processors.wiki.ti.com/index.php/CCS_UniFlash_v3.4.1_Release_Notes#Installation_Instructions
Ti Service Pack installer download link:
http://www.ti.com/tool/download/CC3200SDK

Download and install “Service Pack Installer for CC3200SDK” and “Uniflash”.
Run Uniflash
Quick tutorial for the CC3200:
http://processors.wiki.ti.com/index.php/CC3100_%26_CC3200_UniFlash_Quick_Start_Guide
According to the tutorial:
Test the connexion between Uniflash and the CC3200 launchpad (through the USB driver chip from FTDI on the board) by using “Get Version”
Format the chip,
Flash the service pack,
Flash the program.

Remark, in order to flash the CC3200 on the development board (CC3200 launchpad) set the jumper on position “100”: flash for the “SOP Jumper” (see CC3200 launchpad user’s guide). Once flash, remove the jumper (“000”: 4W JTAG) to run the code, and press the hard reset button on the board.

Improvement

The TCP/IP sending is disrupted sometime. the connexion however is not lost but due to the fast data rate, during the interruption, the data sent by the SPI are not processed, hence are lost. in details, the socket_send() function take usually few µs. But at some time (randomly every few seconds) it takes up to 0.2 ms. The fact that the send function get “stuck” for fractions of milliseconds is caused most likely by collisions due to the nature of the WiFi and TCP/IP protocols.

TI affirms that the chip is capable of 12 Mbit/sec and tests made can confirm it. However this is in an ideal case where no other peripheral on the chip are used. In our case, 5/6 Mbits per second is a more reasonable throughput.