EE318 Electronic Design Lab Project Report, EE Dept, IIT Bombay, April 2007

High speed USB interface for digital processor system

Group No: B1
Siddharth Tallur (04007020) <sidtallur@e.iitb.ac.in>
Akshay Bhatnagar (04007038) <abhatnagar@ee.iitb.ac.in>
Kanik Arora (04007024) <kanikarora@ee.iitb.ac.in>

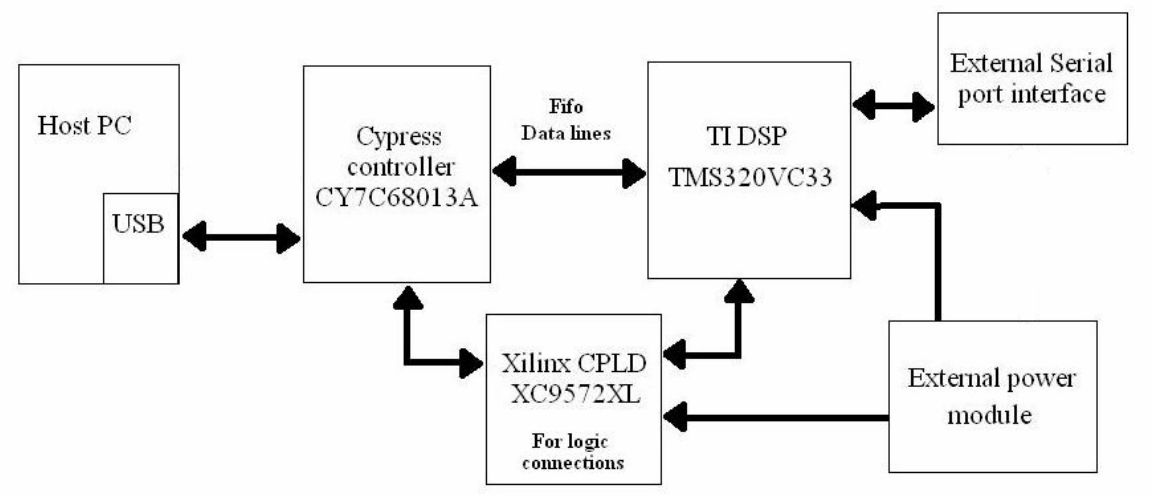
Supervisors: Prof. M. C. Chandorkar, Prof. D. K. Sharma

Abstract:

This project involved designing a high speed USB (480 Mbps) interface for a TMS320VC33 digital processor system. The interface will support user program download and debugging, as well as data exchange between a host computer and the processor while user programs are being executed. We propose the design of a high speed USB circuit using the Cypress Semiconductor USB IC CY7C68013, and its associated firmware.

1. Introduction

The goal behind designing such a system is to support user program download and debugging, as well as data exchange between a host computer and the DSP while user programs are being executed. The idea for using high speed USB interface is to allow transfer of large chunks of data between the host PC and the DSP with high throughput.

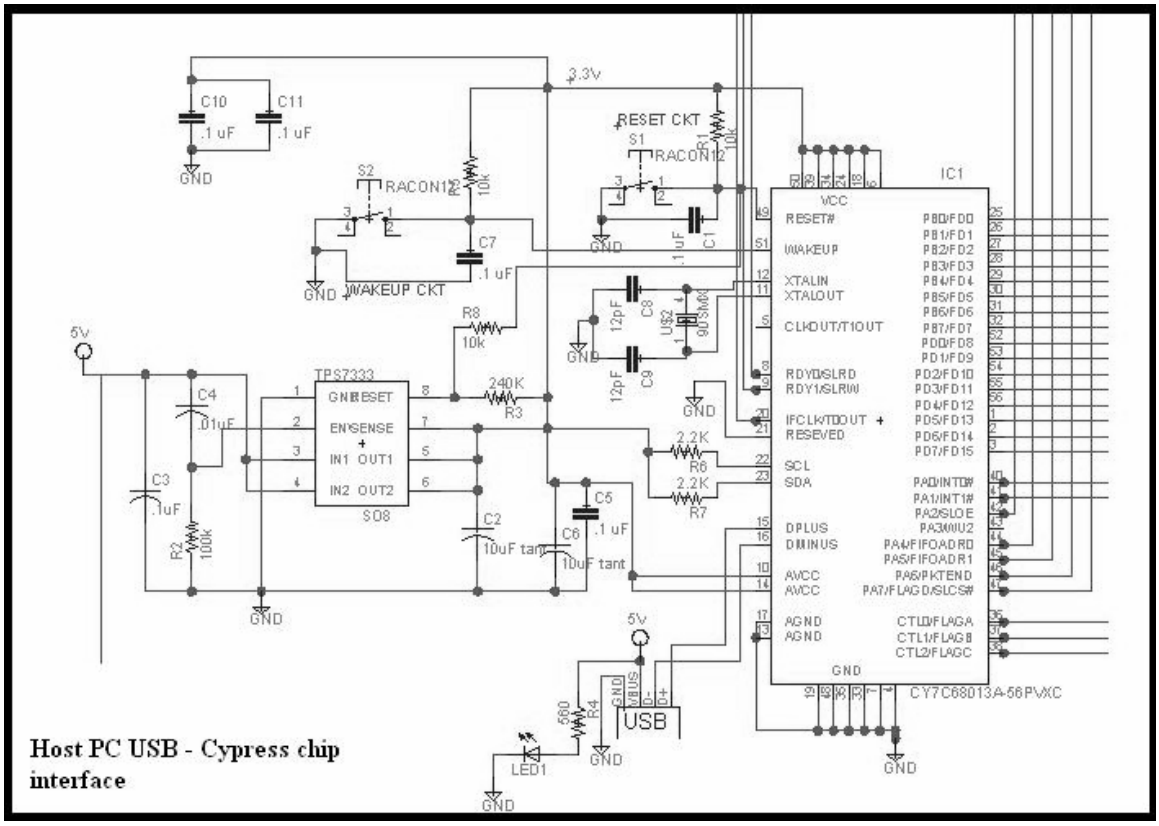


Basic block diagram of the proposed circuit

The block diagram of the circuit we have designed is as shown above.

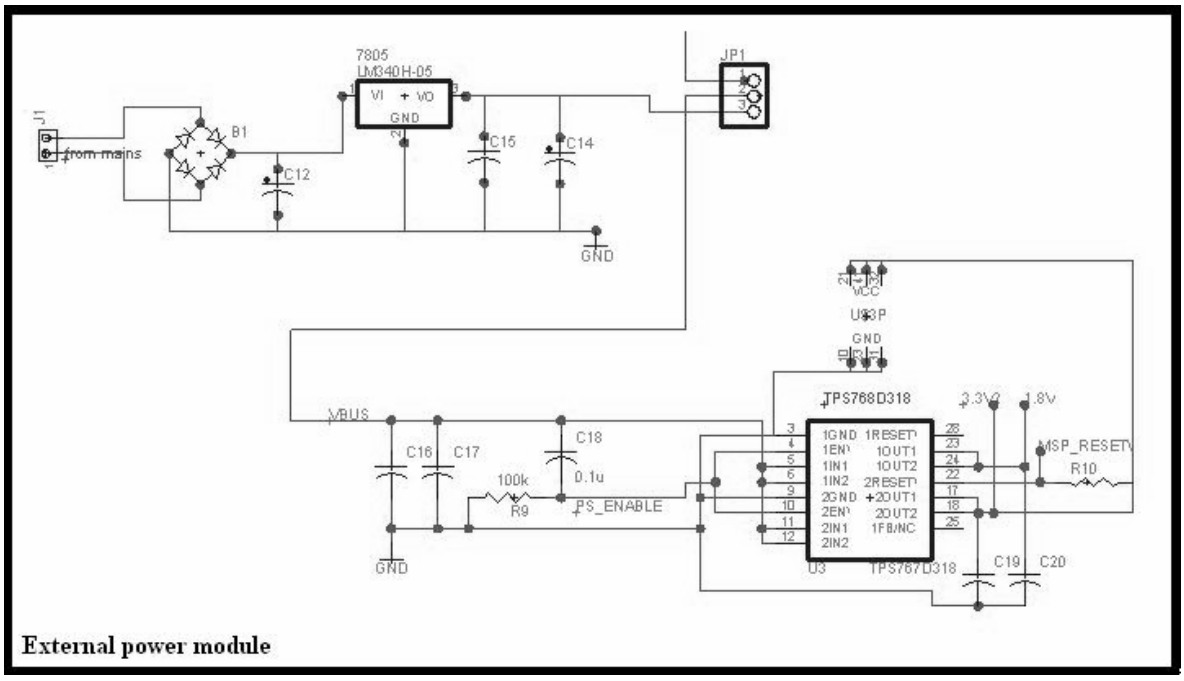
A brief description of each module with the accompanying circuit diagram is provided below:

1. Host PC – Cypress controller interface:



This interface has been designed and successfully tested. Sample firmware and test code available on the internet has been successfully tried out on this board. The Cypress chip is powered by the 5V available on the USB, the necessary 3.3V signal being generated by the TPS7333 voltage regulator IC. A 24MHz crystal is used along with the Cypress controller chip. Reset and wake up circuitry are also provided as shown in the figure. The firmware that we have developed has also been successfully tested with this module.

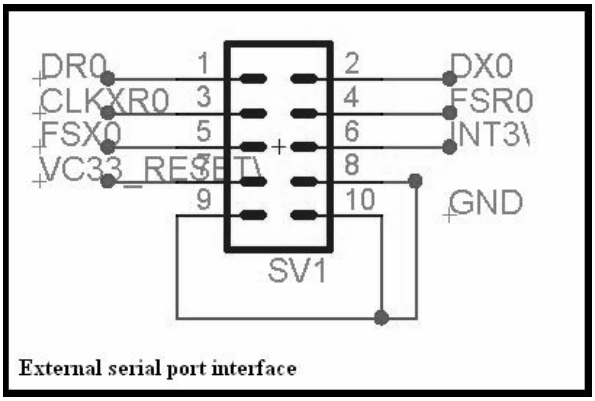
2. External power module:



External power module

This module is provided to meet the worst case current requirements of the circuit, which may well exceed the 500mA that the USB can supply. The regulator IC 7805 is used as shown to give a constant output of 5V, which is then fed to the regulator TPS768D318, which gives outputs of both 3.3V (used by the CPLD and I/O pins of the TMS processor), and 1.8V (used by the core CPU of the TMS processor). The jumper JP1 provides the flexibility of using either the external mains supply, or the USB voltage for powering the TMS processor and the CPLD.

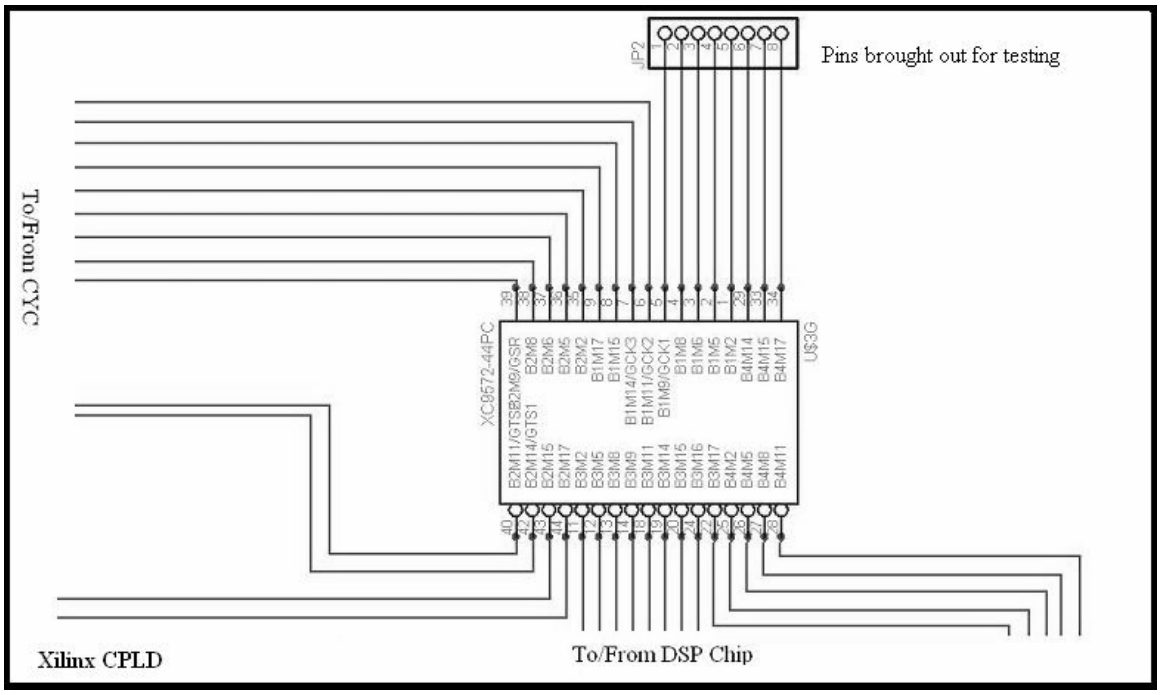
3. External serial port interface:



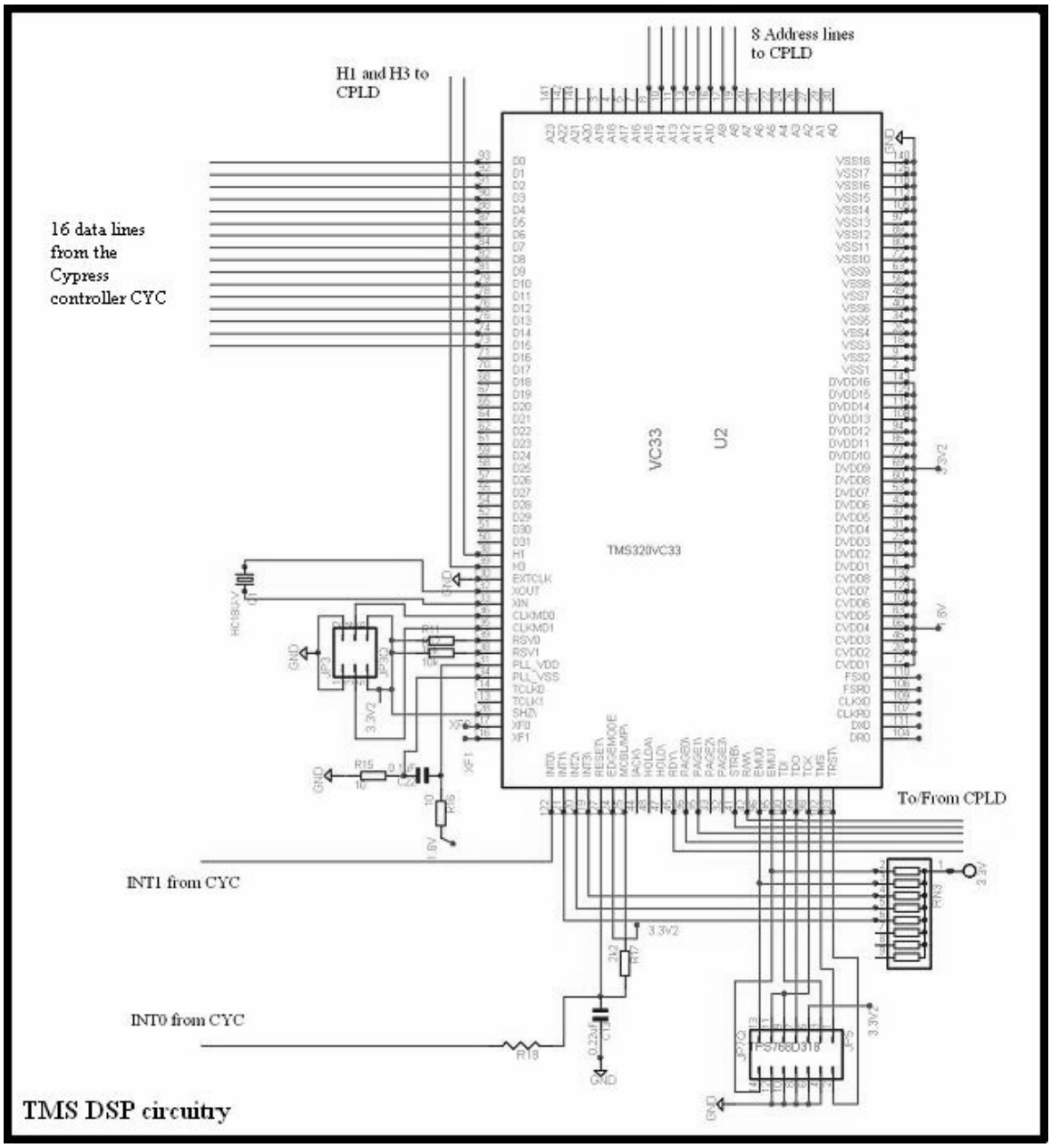
External serial port interface

This pin header is provided to allow an alternate window to load programs into the TMS DSP, which would be helpful in debugging of the board. This helps to serially boot load the DSP using an existing board developed by Prof. M. C. Chandorkar available in the APEL lab in our Department.

4. TMS DSP – Cypress controller interface via the Xilinx CPLD:



The figure above shows the CPLD circuit. The DSP circuit is as follows:



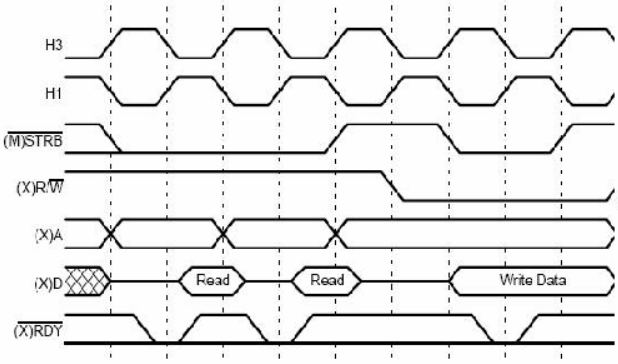
The interface is designed as shown above. A 15MHz crystal is used with the DSP chip. The PLL circuitry inside the DSP multiplies this by 5 and uses the resultant clock of 75MHz. The logic for the connections to the CPLD is worked out as follows:

SLWR\ -> STRB\.RD/W\
SLRD\ -> (STRB\)\.(RD/W\.(H1/2))
PKTEND -> XF1 (flag from the DSP)
RDY\ -> FULL + EMPTY
SLOE -> (RD/W\)\ + STRB\

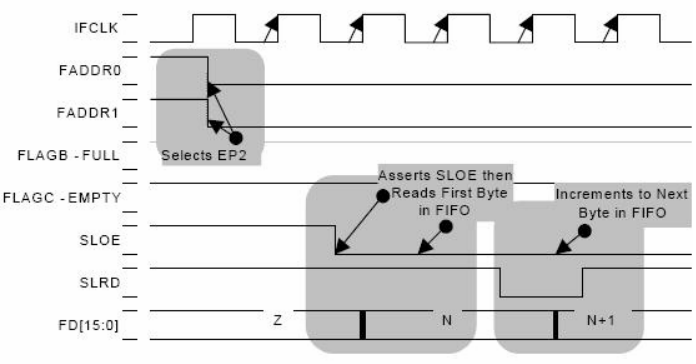
in accordance with the timing diagrams for the Cypress slave FIFO endpoint buffers and the TMS processor as shown in the figure on the next page.

The VHDL code for the CPLD has been written and simulation studies have been successfully carried out to check the correct functioning of the logic. The address lines

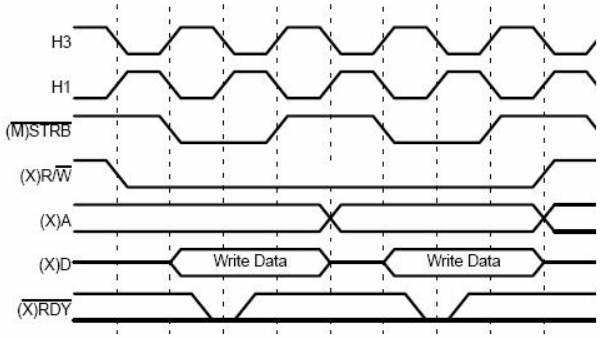
chosen to be connected to the CPLD are such that the address range for TMS reads lies in the range 1000h – 1100h, which allows bootstrapping to take place through the USB interface. The range for write addresses is chosen to be 1200h – 1300h. Using such a range allows us to use PAGE0\ instead of STRB\ in the CPLD logic equations shown earlier. Also, limiting our read and write address ranges to a small part of the DSP memory map, in the same page allows easy extension of the design to further applications, without losses of memory space.



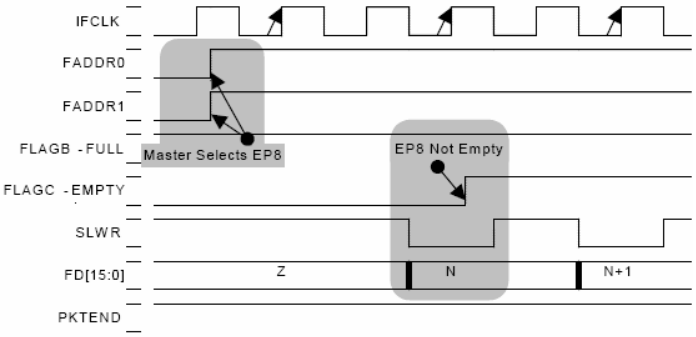
TMS read cycles



Synchronous FIFO reads



TMS write operations



Synchronous FIFO write - Cypress chip

Software:

We have written a code to test the transfer of data from the host PC through the Cypress controller to the DSP (cfirm.c). It involves transferring an array (array.txt) of code to flash a flag pin on the DSP (XF1). The basic idea is as follows:

The Cypress controller endpoint RAM is implemented as 4 FIFO buffers: EP2, EP4 (for master reads) and EP6, EP8 (for master writes). EP0 is used for control transfers by the host and EP1 is used for interrupt transfers (not used). We have used EP2 endpoint RAM buffer of the Cypress controller for Cypress->DSP writes and EP6 buffer for DSP->Cypress writes.

The DSP is set up for boot loading in mode 1 (bootload from the address 1000h): where the RESET\ pin has to be de-asserted followed by de-asserting the INT0\ pin. This is done by first loading the code array into the EP2 buffer and then de-asserting pins PA0

(RESET\) and PA1 (INT0\) respectively of the Cypress controller. The code successfully loads the FIFO buffer EP2 and successfully de-asserts the port A pins of the Cypress controller.

The code is compiled using the sdcc compiler. Also, one needs the libusb library for the usb functions. The binary code for the array to be transferred to the DSP is obtained by generating a listing file (.lst) by compiling the DSP code (test.asm). The list file provides the data in binary format. The asm file for the DSP is compiled using the tic4x assembler-compiler.

Conclusion:

The Cypress controller – host PC interface has been successfully tested. The DSP circuitry also works perfectly. The test firmware written also works perfectly. These two circuits are interfaced through the CPLD circuitry, which is not behaving properly. Future work would involve debugging the CPLD VHDL code and debugging the interface circuitry.

References:

1. EZ USB-FX2 Technical Reference manual, Cypress Semiconductors.

2. TMS320C3X User's guide, Texas Instruments.

3. TMS320C3x General-Purpose Applications User's Guide, Texas Instruments.

4. Datasheets:
    i. CY7C68013A EZ-USB FX2LP™ USB Microcontroller, Cypress Semiconductors.
    ii. TMS320VC33 Digital Signal Processor, Texas Instruments.
    iii. XC9572XL High Performance CPLD, Xilinx