DSPtronics Mixed Signal DSP-ESxx FPGA Boards

DSPtronics LLC

February 9, 2010

Contents

1 Overview

The DSPtronics E-series boards are mixed-signal FPGA signal processing development boards. The E-series have an audio CODEC for analog interfacing and a Xilinx FPGA for DSP computations.

2 Getting Started

First download the latest software from `www.dsptronics.com/downloads`. On the download page select the board type and download the software applications The software applications will allow users to interface with the default configuration and develop signal processing (DSP) applications.

2.1 Default Configuration

The board come with a default FPGA and USB configurations. To sample the default configuration connect an audio source to the line-in input (X2) and PC speakers to the line-out (X1) as shown in Figure 1. Then connect the USB to a PC the audio with an echo will be played on the speakers.

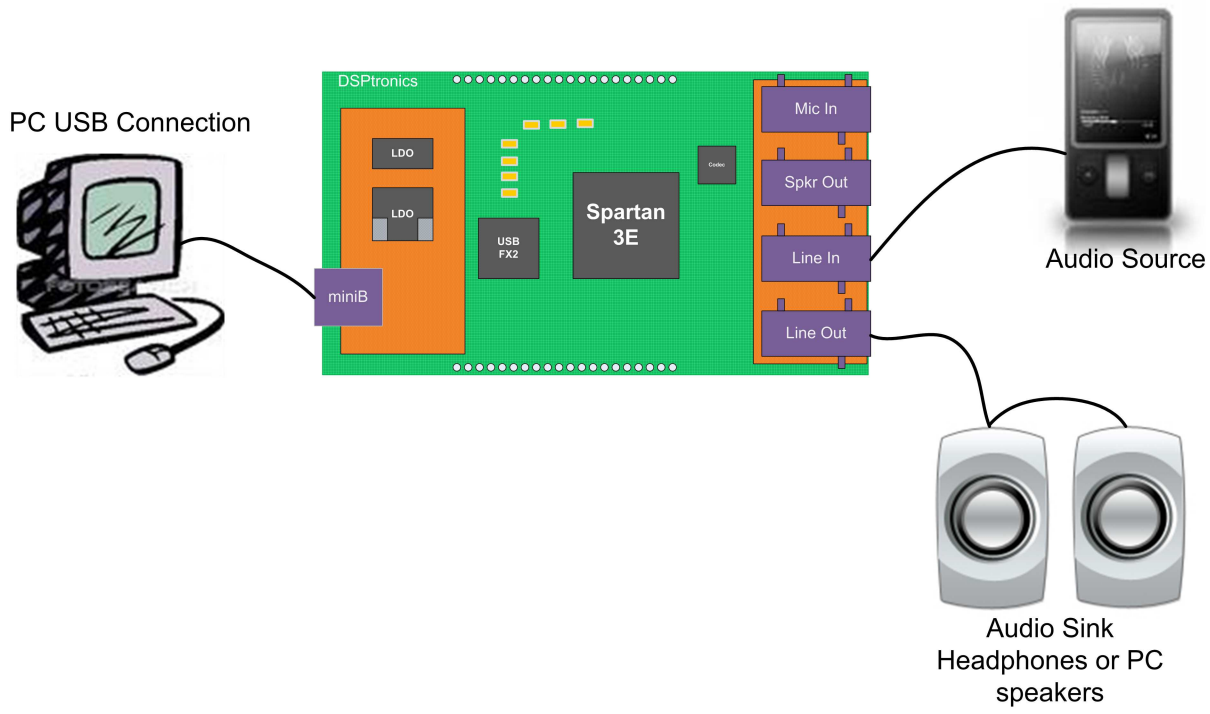

Figure 1: Setup for the Default Configuration

The default configuration will create an echo of the signal input. The echo delay can be adjusted with one of the applications installed.

Start→Programs→DSPtronics→Echo

The application will have the window shown in Figure 2. With this application use the slider bar to control the echo. Note the left and right channels can be controlled independently.

The design files for the echo application can be downloaded at `www.dsptronics.com/dsp_e_series/echo`. The website also includes step-by-step instructions how to create the echo in HDL, run the design through the FPGA tools, and load the configuration to the board.

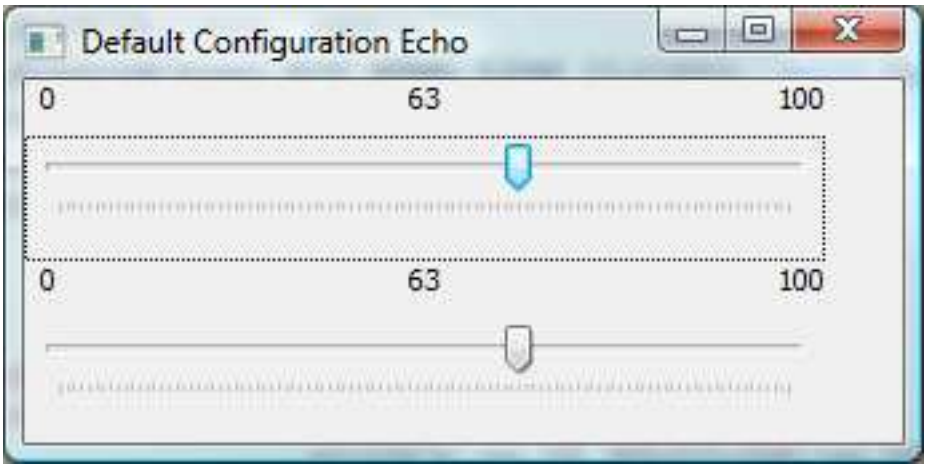

Figure 2: Application to Control the Echo Delay

2.2 Loading User FPGA Configuration

The following assumes the read is familiar with:

1. An HDL language for designing FPGA circuits (Verilog/VHDL/MyHDL).
2. Xilinx synthesis and PAR tools (ISE)
3. Output files created from the ISE Xilinx FPGA tools

If you are not familiar with the above see `www.dsptronics.com/newbee` for more information and an introduction to FPGA development.

Use the application provided in the downloaded installer to download a bit file. The installed applications is located at

Start→Programs→DSPtronics→Programmer

and should look similar to Figure 3. After a design and been synthesized, placed and routed, and the configuration file created. Browse to the ISE directory and select the bit file to be loaded. Hit the program button to configure the FPGA. After a few seconds the FPGA will be programmed. For more information see `www.dsptronics.com/newbee`.

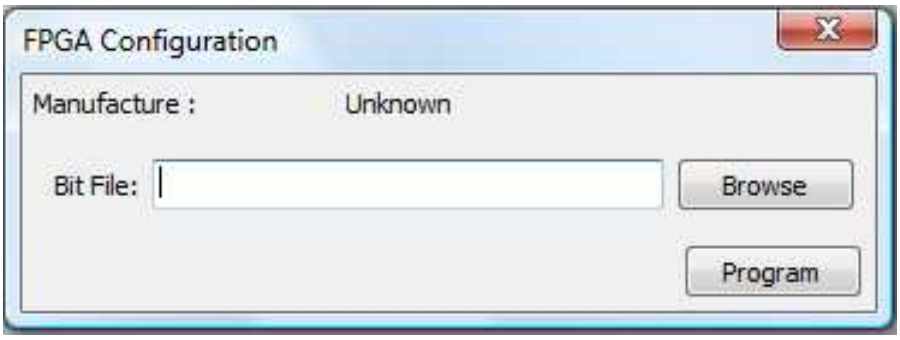

Figure 3: FPGA Programmer

3 Hardware

This following sections describe the hardware components of the DSP-ESxx boards.

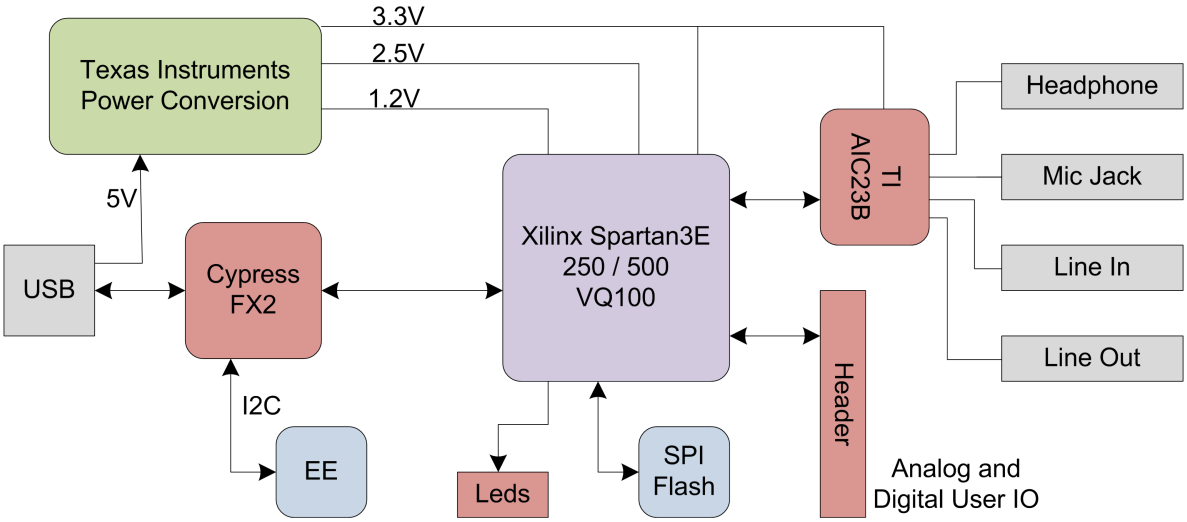

Figure 4: DSP-ESxx Block Diagram

3.1 USB Controller

The DSP-ESxx boards have a high speed (480Mbps) USB controller. The controller is used to configure the FPGA and for high-speed data transfers. The connection between the FPGA and USB controller is described in table 1.

The USB controller is a Cypress FX2 high-speed USB controller. For more information see the FX2 datasheet or `www.fpgaz.com/usbp`.

3.2 Power Supply

The DSP-ESxx board has an integrated power supply. The board runs completely off USB power. The power supply consists of efficient linear LDO regulators. The linear regulators provide the lowest noise susceptibility.

3.3 FPGA

The DSP-ESxx board is centered on a Xilinx FPGA. The Xilinx FPGA provides programmable logic to perform high computing DSP applications. See `www.dsptronics.com/dsp_e_series/dsp_`

Signal Name	FX2 Pin	FPGA Pin	I/O	Description
FX2_FLAGA	29	P3	I	Normally EP2 empty flag (can be programmed otherwise)
FX2_FLAGB	30	P2	I	Normally EP4 empty flag (can be programmed otherwise)
FX2_FLAGC	31	P5	I	Normally EP6 full flag (can be programmed otherwise)
FX2_FLAGD	40	P4	I	Normally EP8 full flag (can be programmed otherwise)
IFCLK	13	P35	I	48MHz clock from FX2
FX2_PKTEND	39	P36	O	Packet end signal to FX2 from FPGA
FX2_SLOE	35	P32	O	Slave FIFO output enable
FX2_SLRD	1	P26	O	Slave FIFO read
FX2_SLWR	2	P33	O	Slave FIFO write
FX2_FD[0]	18	P12	IO	USB controller FIFO data bus
FX2_FD[1]	19	P11	IO	USB controller FIFO data bus
FX2_FD[2]	20	P16	IO	USB controller FIFO data bus
FX2_FD[3]	21	P15	IO	USB controller FIFO data bus
FX2_FD[4]	22	P18	IO	USB controller FIFO data bus
FX2_FD[5]	23	P17	IO	USB controller FIFO data bus
FX2_FD[6]	24	P23	IO	USB controller FIFO data bus
FX2_FD[7]	25	P22	IO	USB controller FIFO data bus
FX2_FIFO_ADDR[0]	37	P10	O	FIFO address (select EP2, EP4, EP6, EP8 endpoint FIFOS)
FX2_FIFO_ADDR[0]	38	P9	O	FIFO address (select EP2, EP4, EP6, EP8 endpoint FIFOS)

Table 1: USB FX2 Controller and FPGA pin Connections

`intro.html` for more information on implementing DSP applications.

3.4 FPGA Configuration Flash

An SPI flash is connected to the FPGA and the FPGA will load its configuration from the SPI flash at power on. The SPI flash can be programmed with the Xilinx tools and an external programmer or with one of the applications provided.

Signal Name	SPI Pin	FPGA Pin	I/O	Description
FLASH_CSN		P24		Flash select
FLASH_SI		P27		Flash serial data input
FLASH_SCK		P50		Flash serial clock
FLASH_SO		P44		Flash serial data output

Table 2: SPI Configuration Flash and FPGA pin Connections

3.5 FPGA JTAG Connection

The FPGA and Flash can be programmed through the USB connection with no external programmer required. Also, the JTAG pins are exposed on the header and can be connected to a Xilinx

Platform cable if desired. See the "Getting Started" and "User IO Header" sections for more information. Figure 5 shows the location of the JTAG signals on the board.

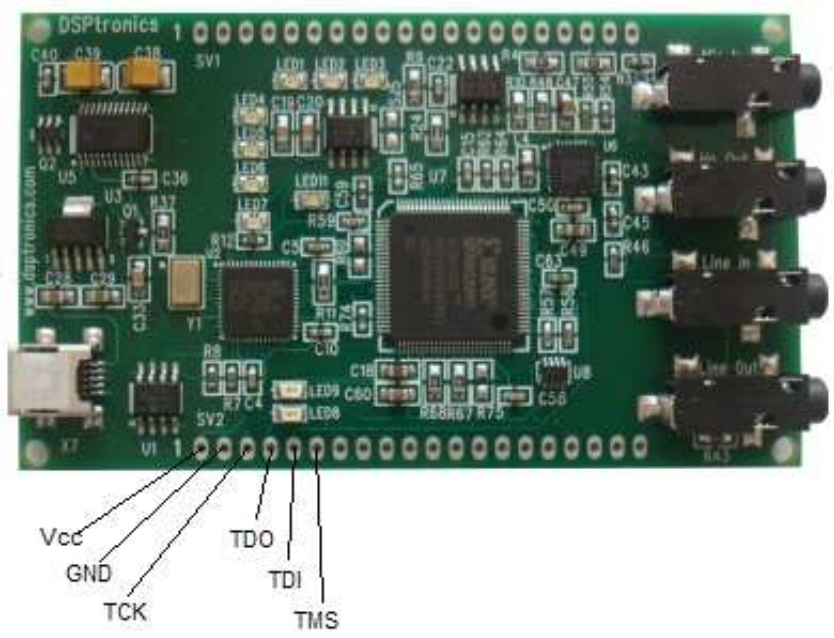

Figure 5: FPGA JTAG Connections

3.6 User LEDs

The DSP-ESxx board has seven user LEDs connected to the FPGA.

Signal Name	LED	FPGA Pin	I/O	Description
LED1	LED 1	P90	O	User LED
LED2	LED 2	P91	O	User LED
LED3	LED 3	P92	O	User LED
LED4	LED 4	P94	O	User LED
LED5	LED 5	P95	O	User LED
LED6	LED 6	P98	O	User LED
LED7	LED 7	P99	O	User LED

Table 3: User LEDs and FPGA pin Connections

3.7 Audio CODEC

The DSP-ESxx board has a Texas Instruments AIC23B audio codec. The audio CODEC adds an interface to analog signals. The table 4 describes the audio CODEC connections to the FPGA.

Signal Name	AIC23b Pin	FPGA Pin	I/O	Description
AUDIO_MODE	19	P66	O	Configuration mode SPI or TWI
AUDIO_CLK	22	P63	O	12MHz clock from FPGA (DCM)
AUDIO_BCLK	28	P85	I	Serial audio bit clock
AUDIO_DIN	1	P62	O	Serial audio data in (fpga out, CODEC in)
AUDIO_DOUT	3	P69	I	Serial audio data out (FPGA in, CODEC out)
AUDIO_LRCIN	2	P65	I	Serial audio in left / right
AUDIO_LRCOUT	4	P68	I	Serial audio out left / right
AUDIO_CSN	18	P84	O	Configuration chip select
AUDIO_SCLK	21	P78	O	Configuration serial clock
AUDIO_SDIN	20	P79	O	Configuration serial data in

Table 4: Audio CODEC and FPGA pin Connections

3.8 Audio Connections 3.5mm Audio Jacks

The DSP-ESxx board has 4 audio connectors, microphone in, headphone out, line in and line out.

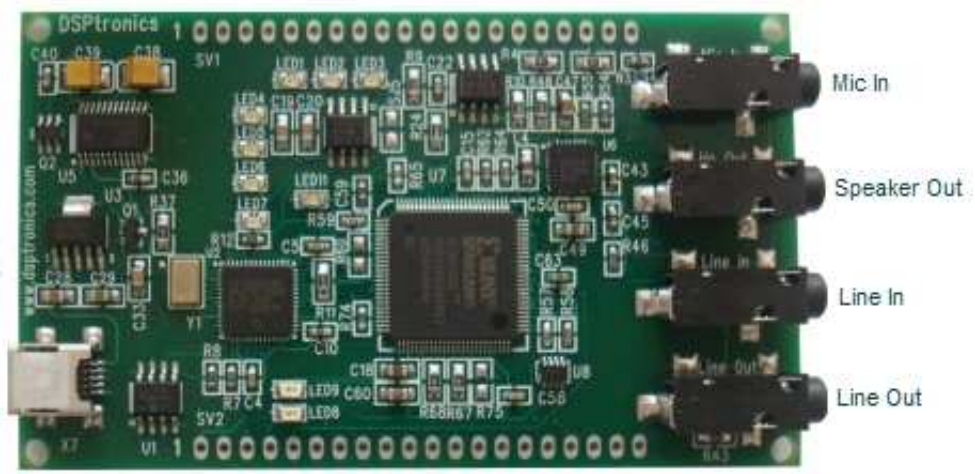

Figure 6: Audio Connectors

3.9 User IO Header

On the edges of the board are two test / interface headers. Table 5 describes which signals are available on the header. The header can be used to interface with custom hardware or various transducers.

The DSP-ESxx uses a generic IO header. To see the generic header definition see `www.dsptronics.com/io_header`.

Header SV1 Pin	Signal Name	FPGA Pin	Header SV2 Pin	Signal Name	FPGA Pin
1	3.3V		1	2.5V	
2	GND		2	GND	
3	NC		3	TCK	P77
4	LED1	P90	4	TDO	P76
5	GND		5	TDI	P100
6	LED2	P91	6	TMS	P75
7	LED3	P92	7	DIO_10	P53
8	NC		8	DIO_11	P54
9	NC		9	DIO_12	P57
10	GND		10	GND	
11	NC		11	DIO_13	P58
12	NC		12	DIO_14	P60
13	NC		13	DIO_15	P61
14	NC		14	DIO_16	P67
15	GND		15	GND	
16	AIO_1		16	DIO_17	P70
17	AIO_2		17	NC	
18	AIO_3		18	NC	
19	GND		19	GND	
20	3.3V		20	5V	

Table 5: User I/O Header and FPGA pin Connections

4 Developing Custom Software

A third party open-source framework for the USB interface is available at `www.fpgaz.com/usbp`. This framework can be used as a stand-alone C/C++ library or integrated with Python. Custom applications can easily be built from the framework. The software provided at `www.fpgaz.com/usbp` provides an encapsulated USB driver interface to the Cypress driver and the open-source LIBUSB driver. The framework also provides a Python interface / package for easy and rapid application development.

CAUTION: Custom software if installed incorrectly can overwrite parameters needed by the USB controller to enumerate on the USB bus. If these values are incorrectly written the EEPROM on the board will need to be replaced. DSPtronics is not responsible for incorrect programming of the FX2 configuration.

Other configurations can leave the board inoperable. If the PID/VID (USB parameters) are intact downloaded the latest installer and reprogram the non-volatile memories. More information can be found at `www.dsptronics.com/trouble_shooting`. Questions can be posted to the support forum at `www.dsptronics/support`.

The Figure 7 is an example of a custom application created using the open-source framework provided.

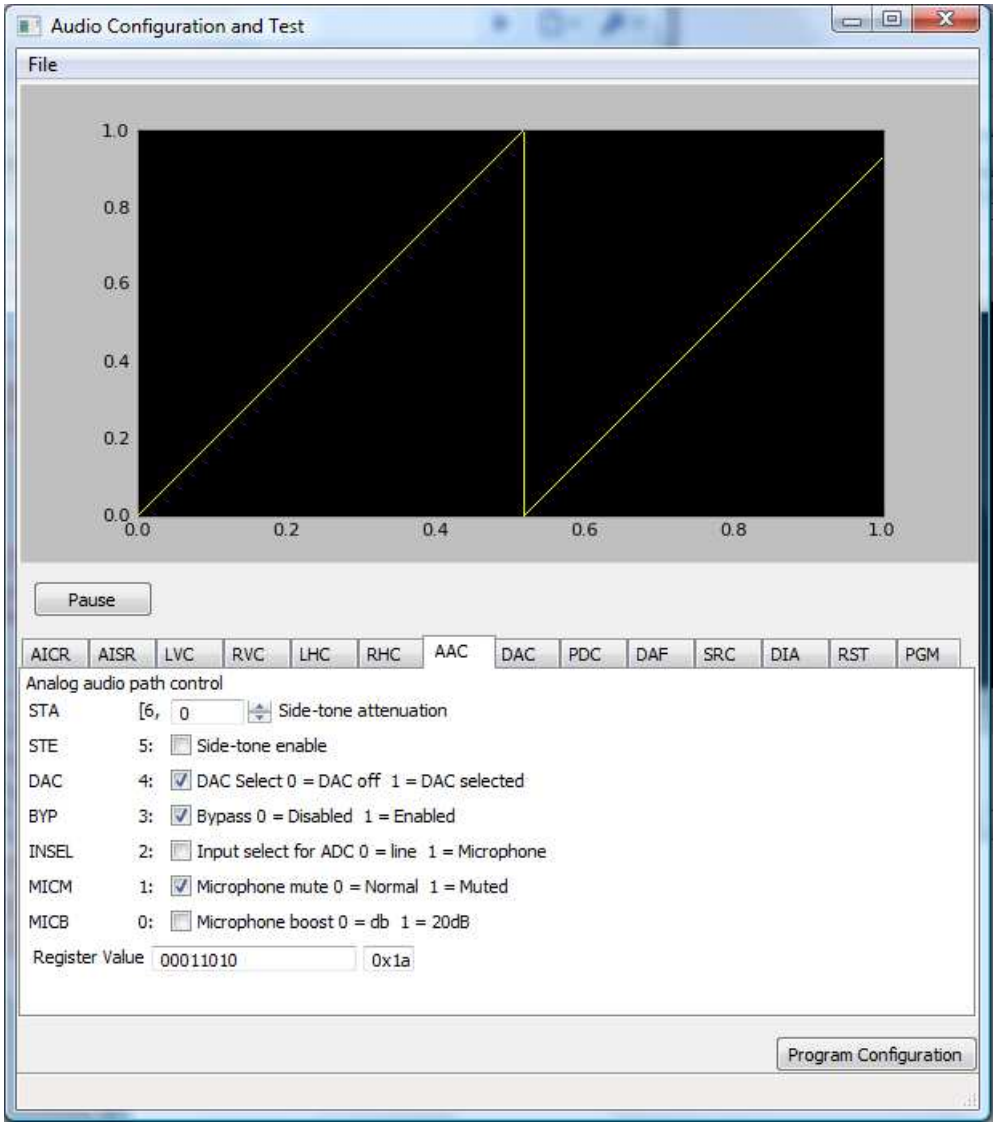

Figure 7: Custom Software Application for Rapid Development

5 Appendix A: UCF Listing

UCF file for the dsp-esxx boards.

```
NET "FD[0]"        LOC = P12  | IOSTANDARD = LVCMOS33;
NET "FD[1]"        LOC = P11  | IOSTANDARD = LVCMOS33;
NET "FD[2]"        LOC = P16  | IOSTANDARD = LVCMOS33;
NET "FD[3]"        LOC = P15  | IOSTANDARD = LVCMOS33;
NET "FD[4]"        LOC = P18  | IOSTANDARD = LVCMOS33;
NET "FD[5]"        LOC = P17  | IOSTANDARD = LVCMOS33;
NET "FD[6]"        LOC = P23  | IOSTANDARD = LVCMOS33;
```

```
NET "FD[7]"        LOC = P22  | IOSTANDARD = LVCMOS33;
NET "FIFOADR[0]"   LOC = P10  | IOSTANDARD = LVCMOS33;
NET "FIFOADR[1]"   LOC = P9   | IOSTANDARD = LVCMOS33;

NET "FLAGA"        LOC = P3   | IOSTANDARD = LVCMOS33;
NET "FLAGB"        LOC = P2   | IOSTANDARD = LVCMOS33;
NET "FLAGC"        LOC = P5   | IOSTANDARD = LVCMOS33;
NET "FLAGD"        LOC = P4   | IOSTANDARD = LVCMOS33;
NET "IFCLK"        LOC = P35  | IOSTANDARD = LVCMOS33;
NET "PKTEND"       LOC = P36  | IOSTANDARD = LVCMOS33;
NET "SLOE"         LOC = P32  | IOSTANDARD = LVCMOS33;
NET "SLRD"         LOC = P26  | IOSTANDARD = LVCMOS33;
NET "SLWR"         LOC = P33  | IOSTANDARD = LVCMOS33;

NET "LED[0]"       LOC = P90  | IOSTANDARD = LVCMOS33;
NET "LED[1]"       LOC = P91  | IOSTANDARD = LVCMOS33;
NET "LED[2]"       LOC = P92  | IOSTANDARD = LVCMOS33;
NET "LED[3]"       LOC = P94  | IOSTANDARD = LVCMOS33;
NET "LED[4]"       LOC = P95  | IOSTANDARD = LVCMOS33;
NET "LED[5]"       LOC = P98  | IOSTANDARD = LVCMOS33;
NET "LED[6]"       LOC = P99  | IOSTANDARD = LVCMOS33;

NET "FLASH_CSN"    LOC = P24  | IOSTANDARD = LVCMOS33;
NET "FLASH_SI"     LOC = P27  | IOSTANDARD = LVCMOS33;
NET "FLASH_SCK"    LOC = P50  | IOSTANDARD = LVCMOS33;
NET "FLASH_SO"     LOC = P44  | IOSTANDARD = LVCMOS33;

NET "TP_HDR[0]"    LOC = P53  | IOSTANDARD = LVCMOS33; # DIO_10
NET "TP_HDR[1]"    LOC = P54  | IOSTANDARD = LVCMOS33; # DIO_11
NET "TP_HDR[2]"    LOC = P57  | IOSTANDARD = LVCMOS33; # DIO_12
NET "TP_HDR[3]"    LOC = P58  | IOSTANDARD = LVCMOS33; # DIO_13
NET "TP_HDR[4]"    LOC = P60  | IOSTANDARD = LVCMOS33; # DIO_14
NET "TP_HDR[5]"    LOC = P61  | IOSTANDARD = LVCMOS33; # DIO_15
NET "TP_HDR[6]"    LOC = P67  | IOSTANDARD = LVCMOS33; # DIO_16
NET "TP_HDR[7]"    LOC = P70  | IOSTANDARD = LVCMOS33; # DIO_17

## Timing Constraints
TIMESPEC "TS_P2P" = FROM "PADS" TO "PADS" 80 ns;

NET "IFCLK" TNM_NET = "IFCLK" ;
TIMESPEC "TS_IFCLK" = PERIOD "IFCLK" 20 ns HIGH 50%;

# Note the following is too large!!
OFFSET = IN  10 ns BEFORE "IFCLK"  ;
```