da_platform Documentation

Release Initial

Michael Price

May 02, 2018

CONTENTS:

This page is the starting point for documentation about the project (often referred to as the “DA platform”).

CHAPTER

ONE

INTRODUCTION

1.1 Motivation

There are many options for DIY digital audio, especially 2-channel DACs. You can buy complete kits or building blocks like USB I2S interfaces, power supplies, and DAC boards. But there are some more complicated choices and compromises to make for other applications, like home theather, digital crossovers, music recording, and measurements. This project aims to provide a common platform to support this variety of applications with minimal hardware or software re-design. All of its PCB designs, FPGA firmware, and software code are open-source and included in the GitHub repository.

1.2 Features

The baseline design allows you to build a box with up to 32 channels of audio I/O at standard sample rates (44.1–192 kHz). The main capabilities I pursued are:

- **Removable DAC/ADC modules with a common mechanical and electrical specification**

  The modules can be swapped out based on system configuration needs, budget limitations, or listening preferences. DIYers can contribute novel converter and analog designs by designing a module, without "reinventing the wheel" for everything else. The platform carries up to 4 modules with up to 8 channels each. PCB designs are provided for 2-channel and 8-channel DACs and ADCs with recent AKM parts.

- **Support for FPGA-based asynchronous FIFO**

  The digital interfaces in the platform can be controlled by a ZTEX USB-FPGA 2.13a board containing a Xilinx XC7A35T FPGA and 256 MB of DRAM. I provide firmware for the FPGA to act as a set of large asynchronous FIFOs in both directions. This allows the use of a local master clock and makes performance independent of software and USB behavior.

- **Support for built-in music streaming server and real-time FIR filtering using Raspberry Pi**

  The Raspberry Pi can connect to a wired or wireless Ethernet network. MPD is used as a music server that can be controlled by an Android mobile app. It can play music stored on another computer or NAS, and stream music from another device. ALSA configuration options allow some channels to be directed through the speakers library or another filtering engine.

- **Support for galvanic isolation of modules**

  The complex digital portions of the system (typically Raspberry Pi and FPGA) have a separate power supply from the modules and clock sources. Digital signals (e.g. I2S, SPI) are routed through Analog Devices transformer isolators which allow these supplies to use separate grounds. The purpose of this is to reduce the currents flowing in the grounds (both digital and analog) used by the modules and clock sources.

- **Stuffing options for different budgets**

  In order to broaden the accessibility of this platform, different versions can be built with different capabilities. The only constant is the underlying PCBs, which are inexpensive. If an asynchronous FIFO and low-jitter clocking are not needed, a USBStreamer or Ministreamer I2S interface could be used. The baseline backplane can be built with jumpers in place of the galvanic isolators. Different DAC and ADC modules, clock sources, power supplies, and chassis can also be chosen for the desired system cost.

- **Moderate performance limitations**

  This platform probably won't achieve the same performance as a carefully designed and tweaked, purpose-built setup for a particular application. However, I believe it can get close. The backplane PCB does not carry analog audio signals. EMI can be suppressed by installing a conductive divider that places the ADC/DAC modules in a Faraday cage. Modules can provide their own supply regulation. The differential clock distribution scheme limits excess phase noise added to that of the clock sources.

These features are justified and explained in more detail in the following sections.

CHAPTER

TWO

HISTORY

This project has been in the works for a while. It was originally conceived in 2008 as a CD player built around a mini-ITX PC. In 2009 I designed a set of PCBs and a custom chassis to hold everything. Some of the design principles were established then: galvanic isolation of all digital signals, 2-channel and 8-channel DACs and ADCs, 4 modules per chassis. I acquired a Digilent Nexys2 FPGA board for the digital interfaces. However, my FPGA skills lagged behind my ambitions and the PCBs I had designed weren't very well thought out. I more-or-less abandoned the project and, for my own stereo system, continued using a CD player and an analog active crossover.

A few years later, I ended up doing research in digital circuits. This made the digital design for the FPGA seem much less intimidating. In 2013, I decided to finish the missing pieces and resurrected the original years-old hardware. I was able to demonstrate the concept of an FPGA-controlled USB DAC with the DSD1792A. However, I found many problems with the original PCB designs and my architecture was complex and unwieldy. Also, the Nexys2 FPGA platform was no longer supported by current tools; in 2015 I switched to a ZTEX board with a modern FPGA. In 2016, after finishing graduate school, I decided to revise the PCBs and construct a complete chassis. This became the "baseline" implementation of the project, with plans to add other interchangeable components in the future. In December 2017 I started using the device as a digital crossover and streaming server in my main stereo system. I rewrote some of the HDL code that was license-encumbered and published the project on GitHub in April 2018.

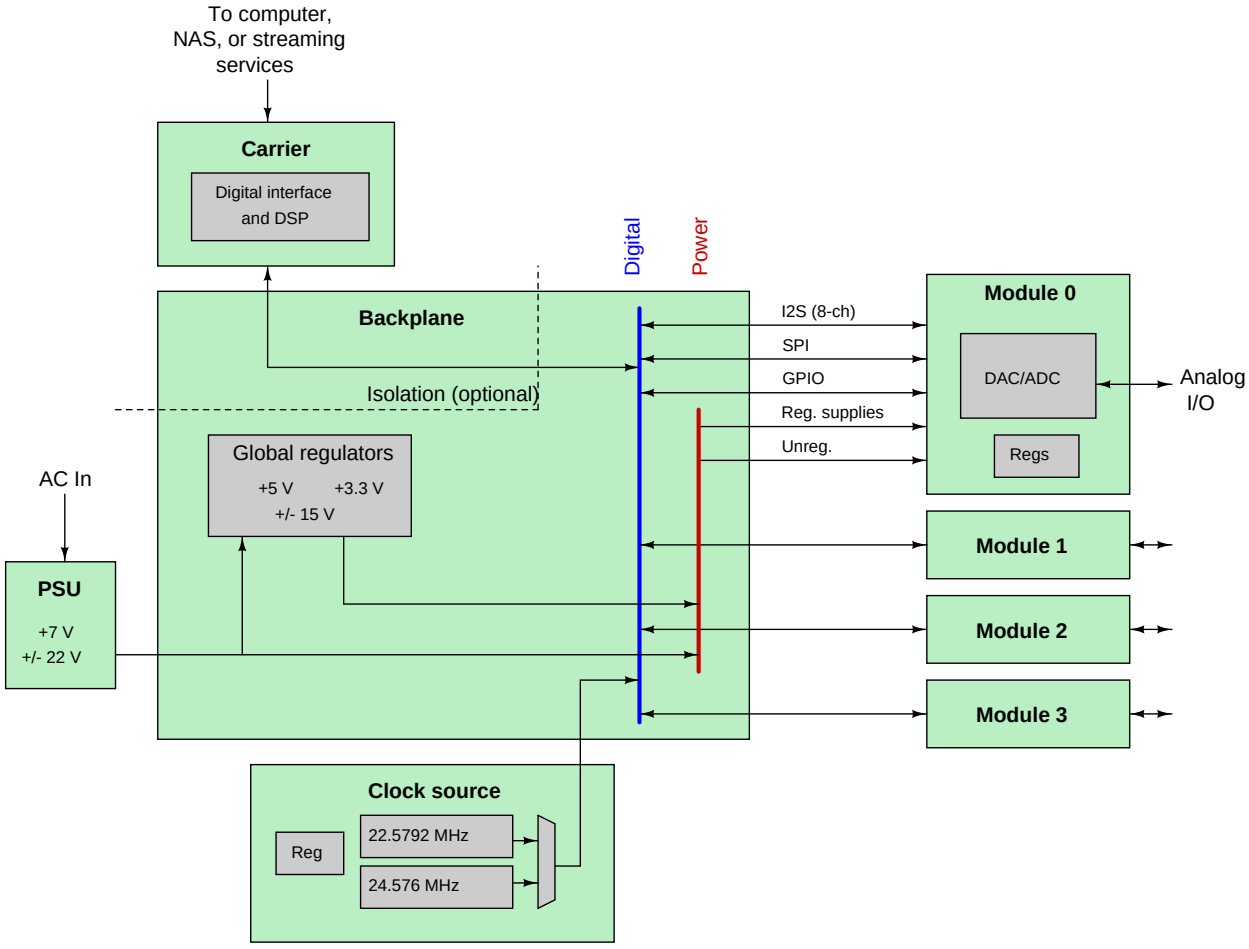
Progress has been slow due to work and family commitments, but I intend to continue soliciting community input, making improvements, and for those brave enough to build their own version, supporting the project indefinitely.

CHAPTER

THREE

HARDWARE ARCHITECTURE

The platform is divided into several components on PCBs. Different functions may be implemented on each PCB, but the interfaces should stay the same to keep them interoperable. These PCBs are illustrated in the figure below.



The following sections briefly describe the function of each PCB and their interfaces to the other PCBs.

Modules: An audio ADC or DAC. The basic requirement of a module is an I2S input (or output). The module interface provides for:

- Up to 4 lanes of I2S data (nominally 8 channels of audio)
- Regulated and unregulated power supplies
- SPI interface for software control of audio converters and other peripherals

- GPIO interface: 8 bits in and 8 bits out

If you design a new module complying to the module interface specification, you can write some high-level software to control it, but won't have to change any of the other links in the chain.

Carrier: A digital source/sink for the signals that go to each module. Can include or interface to a computing device (whether embedded or separate) to improve flexibility. Carriers don't necessarily have to support every possible feature.

Backplane: Interconnect between the carrier and the modules. Also distributes power from the PSU to the modules. The carrier can be powered by the same power supply as the modules, or have its own power supply. The backplane also contains supply regulators for any modules that don't have local regulation.

Clock source: Supply of the necessary clocks for audio converters, at a nominal frequency of 22.5792 MHz or 24.576 MHz. Clocks are distributed differentially by the backplane. The clock source has separate output pins for 5 loads: one for each of the four modules, and one for the carrier.

PSU: An unregulated power supply. Generates separate +7 V rails for the digital and analog sections, and +/- 22 V rails for the analog sections of modules.

3.1 Boards designed so far

This section describes the features of the PCB designs that have been completed and tested so far. Eagle 5.x files for the boards can be found in the `pcb` directory.

Some modules were developed for an earlier version of the platform, and have been tested with the current version using an adapter PCB. These older modules would need to be revised to meet the current interface specification, and don't fit within a reasonably sized chassis with the current backplane (because they are mounted vertically).

Modules

More detailed information about modules (explaining the design process) will be added later. For now, here is a short summary of what is on each board. The baseline implementations use AKM parts because they are readily available (with full datasheets), perform well, and the DACs conveniently have voltage outputs.

- DAC2 (2 channel DAC)
  - DSD1792 (tested on earlier platform only)
    - No local regulation; uses regulated supplies from backplane
    - Two iterations of a common-base I/V stage
      - Revision A with simple BJT current sources and MOSFET follower output
      - Revision B with improved PSRR, 3rd-order antialising filter and diamond buffer output
  - AK4490
    - LT3042 local regulators for VDDL/R and VREFL/R; ADM7160 regulators for AVDD and DVDD
    - AD9515 LVPECL clock receiver powered by ADM7154 regulator
    - ADA4899-1 output buffer/filter powered by ADP7142/ADP7182 regulators
    - 2 V differential (1 V single-ended) full-scale output
    - 0/10/20/30 dB resistive attenuator controlled by GPIO
- ADC2 (2 channel ADC)
  - AK5572
    - LT3042 local regulators for AVDD and VREFL/R; ADM7160 regulator for DVDD

        - AD9515 LVPECL clock receiver powered by ADM7154 regulator
        - ADA4004-4 input buffer; ADA4932-2 ADC driver powered by ADP7142/ADP7182 regulators
        - 2.25 V full-scale input (differential or single-ended)
- DAC8 (8 channel DAC)
    - AD1934 (tested on earlier platform only)
        - No local regulation; uses regulated supplies from backplane
        - Two iterations of a voltage gain / antialiasing filter stage

          Revision A, Revision B
    - AK4458
        - ADP7118 regulators for AVDD and VREF; ADM7160 regulator for DVDD
        - LS90LV012 LVDS clock receiver powered by ADP7118 regulator
        - OPA1664 output buffer/filter
        - 2 V single-ended full-scale output (no differential output)
        - 0/10/20 dB resistive attenuator controlled by SPI
- ADC8 (8 channel ADC)
    - AK5578
        - ADP7118 regulators for AVDD and VREF; ADM7160 regulator for DVDD
        - LS90LV012 LVDS clock receiver powered by ADP7118 regulator
        - OPA1664 input buffer; THS4524 ADC driver
        - 2.25 V full-scale input (differential or single-ended)

Clock source

- Initial version with multiple stuffing options
    - One clock is on and one is powered down at all times (selected by carrier)
    - Damped ferrite bead supply filters leading to each oscillator and to logic
    - Low cost stuffing example:
        - ADM7160 regulator
        - Epson SG-210STF oscillators
        - 74xx CMOS mux and FIN10xx LVDS buffer
    - High performance stuffing example:
        - ADM7154 regulator
        - Crystek CCHD-957 oscillators
        - ADCLK948 LVPECL clock mux/buffer

  Note that you can use any combination of oscillator, regulator and buffer. There are SMD footprints which would fit (for example) the NDK NZ2520SD, and DIP footprints that fit (for example) the Tentlabs XO.

Backplane

- Initial version

  - Stuffing options for digital interconnect:
    * ADUM340x / ADUM14xE0 transformer isolators
    * Resistor/jumper bypass option for non-isolated system
  - Global regulation (+5 V, +3.3 V, +/- 15 V) with stuffing options:
    * LM317/LM337 with Vadj bypass
    * Jung regulators with LM317/LM337 preregulator

Carrier

- Initial version contains:
  - ZTEX USB-FPGA 2.13a (Xilinx Artix-7 XC7A35T FPGA, 256 MB memory)

    This project includes an HDL design and build scripts for the FPGA to function as the carrier, exercising all features of the module interface. It also acts as a deep (8M sample) asynchronous FIFO, since the USB interface clock is unrelated to the audio clocks. Much more information about this can be found in the *Digital design* section.
  - Raspberry Pi (optional)

    I use MPD running on the Raspberry Pi as a streaming server for audio files stored on a NAS. The Raspberry Pi is mounted on the carrier card within the chassis, and has an extension cable to an Ethernet jack on the back. However, you can leave out the Raspberry Pi and connect any computing device to the FPGA board using USB (with a similar extension cable). It's possible to use a USB Wi-fi adapter.
  - External 5 V DC wall wart supply (can be isolated from audio supplies, if backplane is equipped with isolators)
  - Debug/expansion header (0.1" pitch) for prototyping

PSU

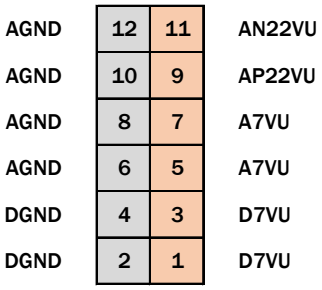
- Initial version
  - 60 Hz linear supply with diode rectifiers
  - Dual 5 V (Triad VPS10-2500) and 28 Vct (Triad FD7-28) transformers
  - +7 V unregulated outputs for analog and digital sections
  - +/- 22 V unregulated outputs for analog sections
  - 10,000 uF decoupling for each rail
  - At least 1 A load is allowed for all supplies simultaneously

3.2 Interfaces

This section describes the electrical interfaces in the initial version of the DA platform. The idea is that different PCBs can be designed for each function (carrier, backplane, modules, etc.) and safely interchanged if they meet the interface specifications. All of the board-to-board connections are made with standard 0.1" pitch headers. It's possible to stack boards, or (except for modules) to put the boards side by side and make the connection with an appropriately sized ribbon cable.

PSU to backplane : 12 pins (6x2 header)

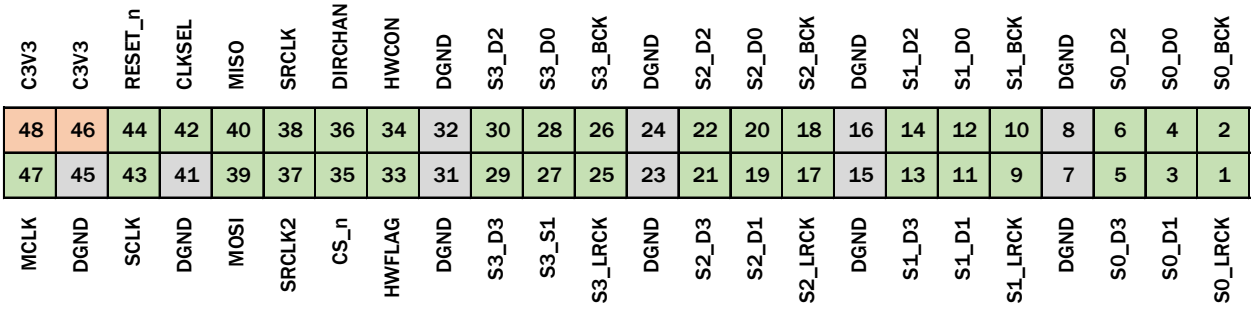
In the baseline design, the PSU has transformers directly screwed to the chassis, and a small rectifier/filter board that mounts to the right of the backplane. They are connected via a ribbon cable.



Pin number	Name	Description
1	D7VU	Unregulated +7 V for digital section
2	DGND	Digital ground
3	D7VU	Unregulated +7 V for digital section
4	DGND	Digital ground
5	A7VU	Unregulated +7 V for analog section
6	AGND	Analog ground
7	A7VU	Unregulated +7 V for analog section
8	AGND	Analog ground
9	AP22VU	Unregulated +22 V for analog section
10	AGND	Analog ground
11	AN22VU	Unregulated -22 V for analog section
12	AGND	Analog ground

Carrier to backplane:

In the baseline design, the backplane has a female header on top, and the carrier has a male header on the bottom. The carrier stacks on top of the backplane.



Pin number	Name	Description
1	S0_LRCK	I2S word clock for module slot 0
2	S0_BCK	I2S bit clock for module slot 0
3	S0_D1	I2S data lane 1 for module slot 0
4	S0_D0	I2S data lane 0 for module slot 0
5	S0_D3	I2S data lane 3 for module slot 0
6	S0_D2	I2S data lane 2 for module slot 0
7	DGND	Digital ground
8	DGND	Digital ground
9	S1_LRCK	I2S word clock for module slot 1
10	S1_BCK	I2S bit clock for module slot 1
11	S1_D1	I2S data lane 1 for module slot 1
12	S1_D0	I2S data lane 0 for module slot 1
13	S1_D3	I2S data lane 3 for module slot 1
14	S1_D2	I2S data lane 2 for module slot 1

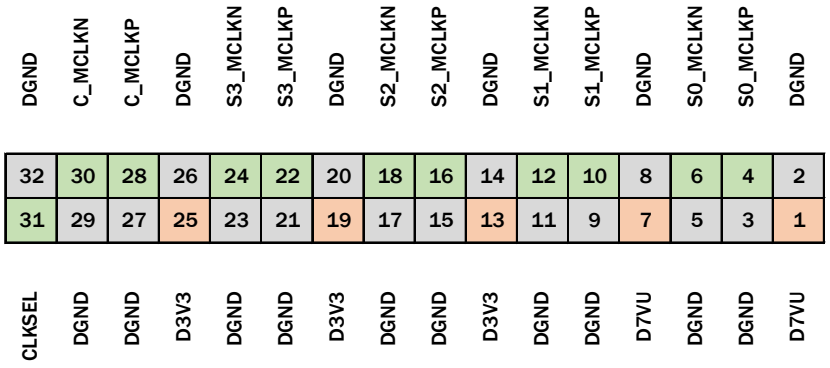
Continued on next page

Table 3.1 – continued from previous page

Pin number	Name	Description
15	DGND	Digital ground
16	DGND	Digital ground
17	S2_LRCK	I2S word clock for module slot 2
18	S2_BCK	I2S bit clock for module slot 2
19	S2_D1	I2S data lane 1 for module slot 2
20	S2_D0	I2S data lane 0 for module slot 2
21	S2_D3	I2S data lane 3 for module slot 2
22	S2_D2	I2S data lane 2 for module slot 2
23	DGND	Digital ground
24	DGND	Digital ground
25	S3_LRCK	I2S word clock for module slot 3
26	S3_BCK	I2S bit clock for module slot 3
27	S3_D1	I2S data lane 1 for module slot 3
28	S3_D0	I2S data lane 0 for module slot 3
29	S3_D3	I2S data lane 3 for module slot 3
30	S3_D2	I2S data lane 2 for module slot 3
31	DGND	Digital ground
32	DGND	Digital ground
33	HWFLAG	Serialized GPIO input from modules
34	HWCON	Serialized GPIO output to modules
35	CS_n	Serialized SPI chip select (active low)
36	DIRCHAN	Serialized module configuration indicator
37	SRCLK2	2nd level (64 bit) deserializer clock
38	SRCLK	1st level (8 bit) deserializer clock
39	MOSI	SPI data output to modules
40	MISO	SPI data input from modules
41	DGND	Digital ground
42	CLKSEL	Clock select (e.g. 22.5792 vs. 24.576 MHz)
43	SCLK	SPI and serializer clock
44	RESET_n	Module reset (active low)
45	DGND	Digital ground
46	C3V3	3.3 V digital supply from carrier
47	MCLK	Audio clock for I2S masters
48	C3V3	3.3 V digital supply from carrier

Clock source to backplane:

In the baseline design, the backplane has a female header on top, and the clock source has a male header on the bottom. The clock source stacks on top of the backplane, hanging over the lower edge.



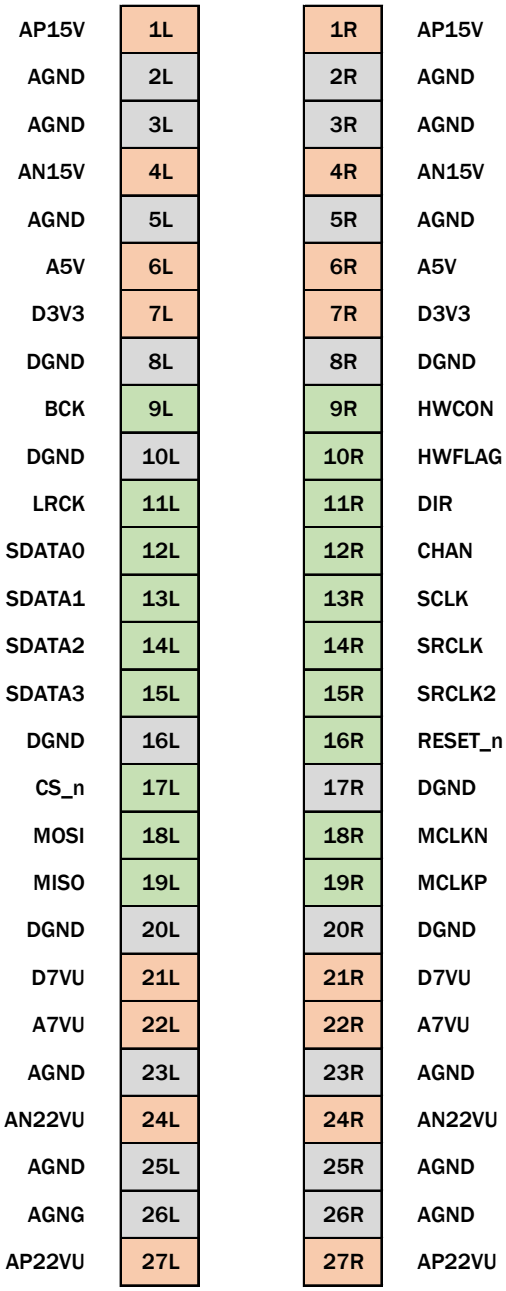
Pin number	Name	Description
1	D7VU	Unregulated 7 V supply
2	DGND	Ground
3	DGND	Ground
4	S0_MCLKP	Differential clock output to module slot 0
5	DGND	Ground
6	S0_MCLKP	Differential clock output to module slot 0
7	D7VU	Unregulated 7 V supply
8	DGND	Ground
9	DGND	Ground
10	S1_MCLKP	Differential clock output to module slot 1
11	DGND	Ground
12	S1_MCLKP	Differential clock output to module slot 1
13	D3V3	Regulated 3.3 V supply
14	DGND	Ground
15	DGND	Ground
16	S2_MCLKP	Differential clock output to module slot 2
17	DGND	Ground
18	S2_MCLKP	Differential clock output to module slot 2
19	D3V3	Regulated 3.3 V supply
20	DGND	Ground
21	DGND	Ground
22	S3_MCLKP	Differential clock output to module slot 3
23	DGND	Ground
24	S3_MCLKP	Differential clock output to module slot 3
25	D3V3	Regulated 3.3 V supply
26	DGND	Ground
27	DGND	Ground
28	C_MCLKP	Differential clock output to carrier
29	DGND	Ground
30	C_MCLKP	Differential clock output to carrier
31	CLKSEL	Selects active oscillator (22.5792 or 24.576 MHz)
32	DGND	Ground

Module to backplane : 54 pins (2x 27x1 header, with 2.0” spacing)

The module has some additional mechanical requirements because they sit on top of the backplane.

TODO: Drawing of mechanical footprint of module

If you can’t fit all the necessary circuitry on one board of this size, consider vertical stacking. You could also design an extra-wide module that takes up 2, 3, or 4 slots.



AP15V	1L	1R	AP15V
AGND	2L	2R	AGND
AGND	3L	3R	AGND
AN15V	4L	4R	AN15V
AGND	5L	5R	AGND
A5V	6L	6R	A5V
D3V3	7L	7R	D3V3
DGND	8L	8R	DGND
BCK	9L	9R	HWCON
DGND	10L	10R	HWFLAG
LRCK	11L	11R	DIR
SDATA0	12L	12R	CHAN
SDATA1	13L	13R	SCLK
SDATA2	14L	14R	SRCLK
SDATA3	15L	15R	SRCLK2
DGND	16L	16R	RESET_n
CS_n	17L	17R	DGND
MOSI	18L	18R	MCLKN
MISO	19L	19R	MCLKP
DGND	20L	20R	DGND
D7VU	21L	21R	D7VU
A7VU	22L	22R	A7VU
AGND	23L	23R	AGND
AN22VU	24L	24R	AN22VU
AGND	25L	25R	AGND
AGNG	26L	26R	AGND
AP22VU	27L	27R	AP22VU

Pin number	Name	Description
1L	AP15V	Regulated +15 V supply for analog section
2L	AGND	Analog ground
3L	AGND	Analog ground
4L	AN15V	Regulated -15 V supply for analog section
5L	AGND	Analog ground
6L	A5V	Regulated 5 V supply for analog section
7L	D3V3	Regulated 3.3 V supply for digital section
8L	DGND	Digital ground
9L	BCK	I2S bit clock
10L	DGND	Digital ground
11L	LRCK	I2S word clock
12L	SDATA0	I2S data lane 0
13L	SDATA1	I2S data lane 1
14L	SDATA2	I2S data lane 2
15L	SDATA3	I2S data lane 3
16L	DGND	Digital ground
17L	CS_n	SPI chip select (active low)
18L	MOSI	SPI data input
19L	MISO	SPI data output
20L	DGND	Digital ground
21L	D7VU	Unregulated 7 V supply for digital section
22L	A7VU	Unregulated 7 V supply for analog section
23L	AGND	Analog ground
24L	AN22VU	Unregulated -22 V supply for analog section
25L	AGND	Analog ground
26L	AGND	Analog ground
27L	AP22VU	Unregulated +22 V supply for analog section

Pin number	Name	Description
1R	AP15V	Regulated +15 V supply for analog section
2R	AGND	Analog ground
3R	AGND	Analog ground
4R	AN15V	Regulated -15 V supply for analog section
5R	AGND	Analog ground
6R	A5V	Regulated 5 V supply for analog section
7R	D3V3	Regulated 3.3 V supply for digital section
8R	DGND	Digital ground
9R	HWCON	Serialized GPIO input
10R	HWFLAG	Serialized GPIO output
11R	DIR	Module direction: 1 = DAC, 0 = ADC
12R	CHAN	Number of channels: 1 = 8-ch, 0 = 2-ch
13R	SCLK	SPI and serializer clock
14R	SRCLK	1st level deserializer clock
15R	SRCLK2	2nd level deserializer clock
16R	RESET_n	Reset (active low)
17R	DGND	Digital ground
18R	MCLKN	Differential audio clock input
19R	MCLKP	Differential audio clock input
20R	DGND	Digital ground
21R	D7VU	Unregulated 7 V supply for digital section
22R	A7VU	Unregulated 7 V supply for analog section
23R	AGND	Analog ground
24R	AN22VU	Unregulated -22 V supply for analog section
25R	AGND	Analog ground
26R	AGND	Analog ground
27R	AP22VU	Unregulated +22 V supply for analog section

3.3 Digital design

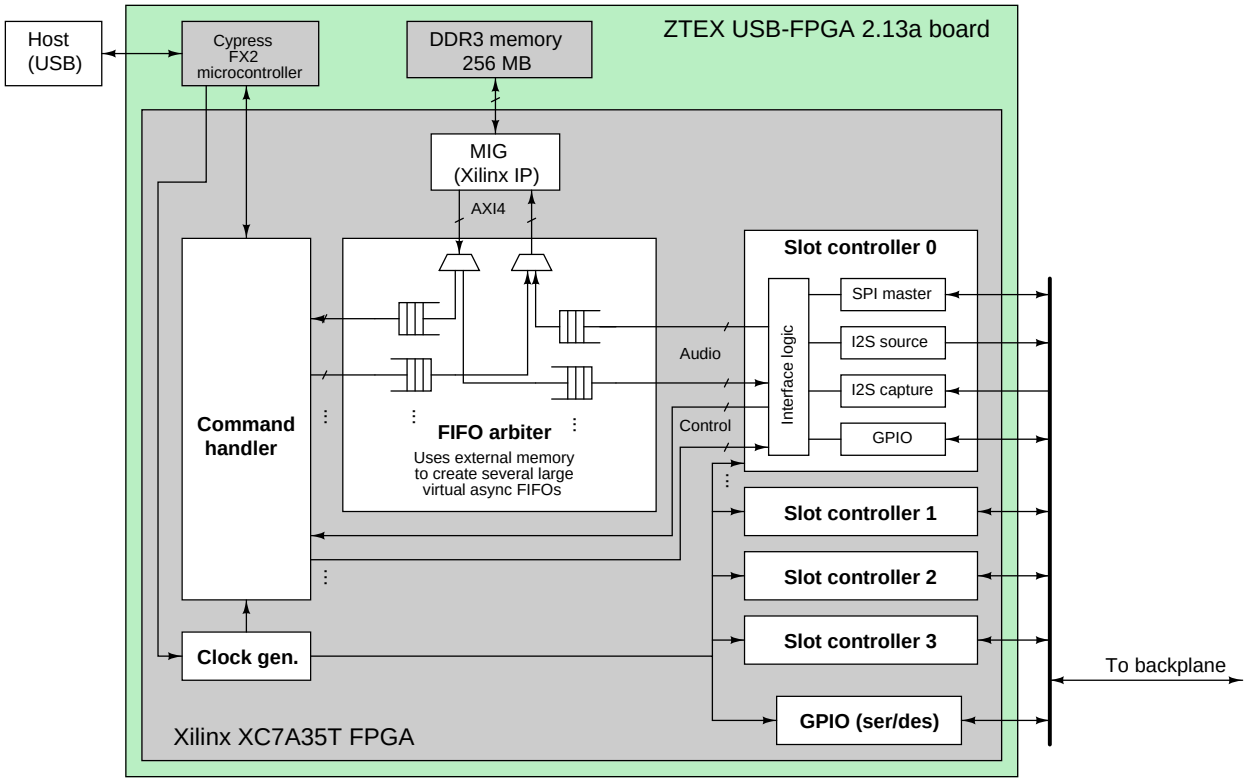
The baseline carrier design includes an FPGA, which gives it a lot of flexibility. This section describes the digital logic that runs on the FPGA, providing up to 32 channels of audio I/O through asynchronous FIFOs, and allowing a host computer to exercise all module features via USB.

The logic is defined in SystemVerilog hardware description language (HDL). Any future improvements will be shared with users who can simply reprogram their boards. The design could also be retargeted to other carrier boards. Please see the `verilog` directory of the repository for design and simulation sources, and the `xilinx` directory for build scripts.

I started out with the ZTEX USB-FPGA 2.13a board, which has all of the necessary features and comes with an elegant and open-source software/firmware stack for delivering data to and from the custom logic via USB. It is also affordable compared to other FPGA boards with similar capabilities.

The primary function of this design is to provide a bridge between a computing device and the physical interfaces (I2S, SPI, and GPIO) of the four DAC/ADC modules. (From a digital design perspective, it would be easy to change the number of modules at build time.) For now we are using a USB interface for the host computer, but other interfaces could be added in the future; for example, audio could be streamed over Ethernet, or control commands could be sent over a UART or SPI.

The architecture is shown in the following diagram.



Each module interfaces with a "slot controller" which accepts commands and audio samples specific to that module. The audio sample I/Os are 32 bits wide and the stream of samples can be rearranged according to runtime configuration (number of channels, I2S vs. left-justified vs. right-justified, etc). The command interface is 8 bits wide; commands are encoded into a stream of bytes, with the first byte representing the type of command and the remaining bytes containing parameters/arguments.

Data sent over USB (broken into 16-bit words) is directed to the command handler, a state machine which examines each word and takes the appropriate action. There are some global commands which can be dealt with immediately by the command handler, for example the command to reset all of the modules. Other commands are addressed to a specific module and thus are delivered to the appropriate slot controller; the response is returned to the host.

Audio samples are handled differently: we want to provide a large buffer (FIFO) so the software on the host doesn't need to keep up with the exact timing of playback or recording. A single buffer isn't enough since there are four modules, each of which could be functioning as a DAC or ADC, or not being used at all. But with some massaging, we can use a single fast memory (DDR3) to provide a set of 8 virtual FIFOs (one in each direction for each module). Access to this external memory is controlled by a block called the FIFO arbiter. To the outside world, it looks like we have several independent FIFOs. On the ZTEX FPGA board, the DDR3 memory is 256 MB, so each virtual FIFO is big enough for 8M samples–more than a minute of stereo audio at 44.1 kHz.

There are three clock domains in the design.

1. The USB interface provides a 48 MHz clock. The command handler and most of the control logic is driven by this clock; it is divided down to 3 MHz for SPI and GPIO signals.
2. The DAC/ADC modules run on a clock that is a multiple of the audio sample rate. Nominally this is 22.5792 MHz or 24.576 MHz. This clock must have very low phase noise at the DAC or ADC itself, but the data signals to/from the carrier don't need any special treatment as long as they meet the timing constraints of the chips they are talking to.
3. The DDR3 memory runs on a faster clock (nominally 400 MHz). Interfacing to this memory is handled by a specialized IP block provided by the FPGA vendor.

Within the design, synchronizers and asynchronous FIFOs are used to transfer signals between different clock domains.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search