

EDA 385: FPGA-pod

Pierre-Adrien Lefebvre: pieradrien.lefebvre@gmail.com

Kristian Samppa: et06ks7@student.lth.se

Matt Moles: mkmoles@ncsu.edu



Original Vs. Actual Design

-**The FPGA-pod:** A music player built off of the Diligent NEXYS-2 board and add-on modules

- **Necessary add-on modules include:**

-1 PModAMP1 (Speaker/Headphone Amplifier)

-1 PModCLS (LCD Screen)

-1 PmodSD (SD Card Interface)

- **Design:**

LCD Screen
(PModCLS)

Speaker/Headphone Amplifier
(PModAMP1)

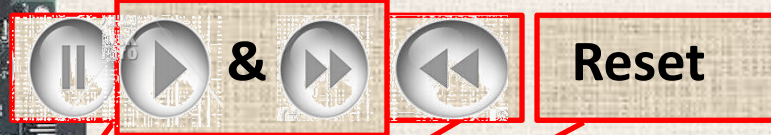
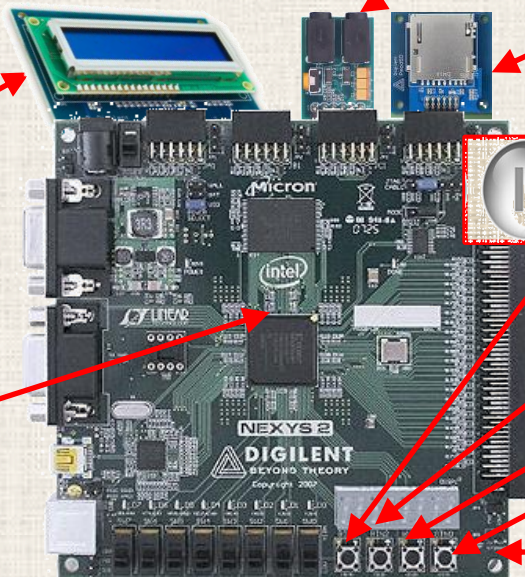
SD Card Interface
(PModSD)

Digilent
Nexys2 Board

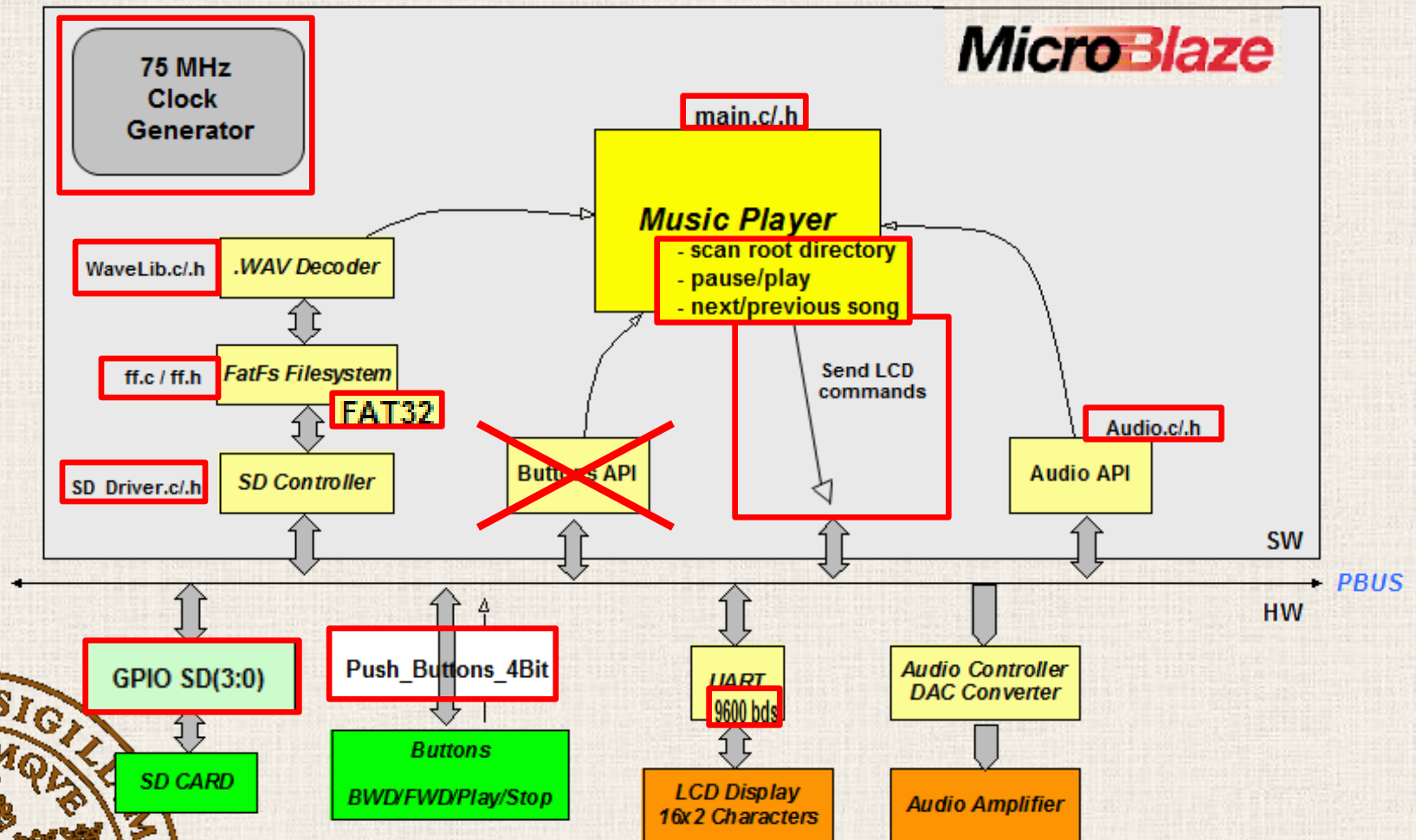
Reset

Push-Button
Controls

Lund University

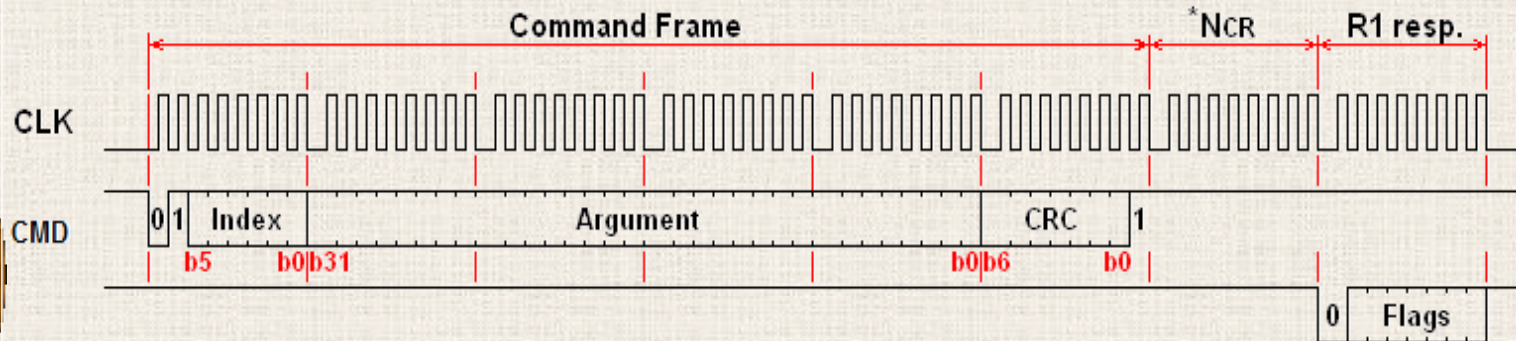
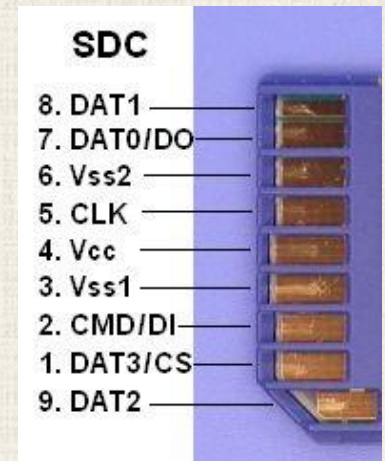


Proposed Vs. Actual Implementation



SD Card Communication

- SD Card:** 3 pins for power, 6 pins for communication
- SD Mode:** operating mode for SD cards with no native host interface
 - Data transfer made via synchronous byte oriented serial communication interface between SD card and host
 - Command frame: fixed length packet (6 bytes) is sent from host to card
 - Response from card can then be read after command response time (NCR)
 - Command response time (NCR) is 0-8 bytes



SD Controller

- Provides for reading data on SD card
- Originally implemented in hardware for speed but proved very difficult to debug
- Subsequently implemented in software
 - Software drives individual pins on SD card
 - SD card is first initialized
 - Data can then be read using various commands
- PROBLEM:** SD controller is too slow - each pin is driven directly via software
 - Easily solved with more time to debug hardware controller



FAT File System

- SD card was formatted with FAT32 file system
- Independent FAT file system library (software) used to read files and directories
 - Platform independent because no implementation of disk I/O
 - Disk I/O implemented by user - library can thus be instantiated on any kind of memory
- Operated using "black box" approach – easy to use

Main functions



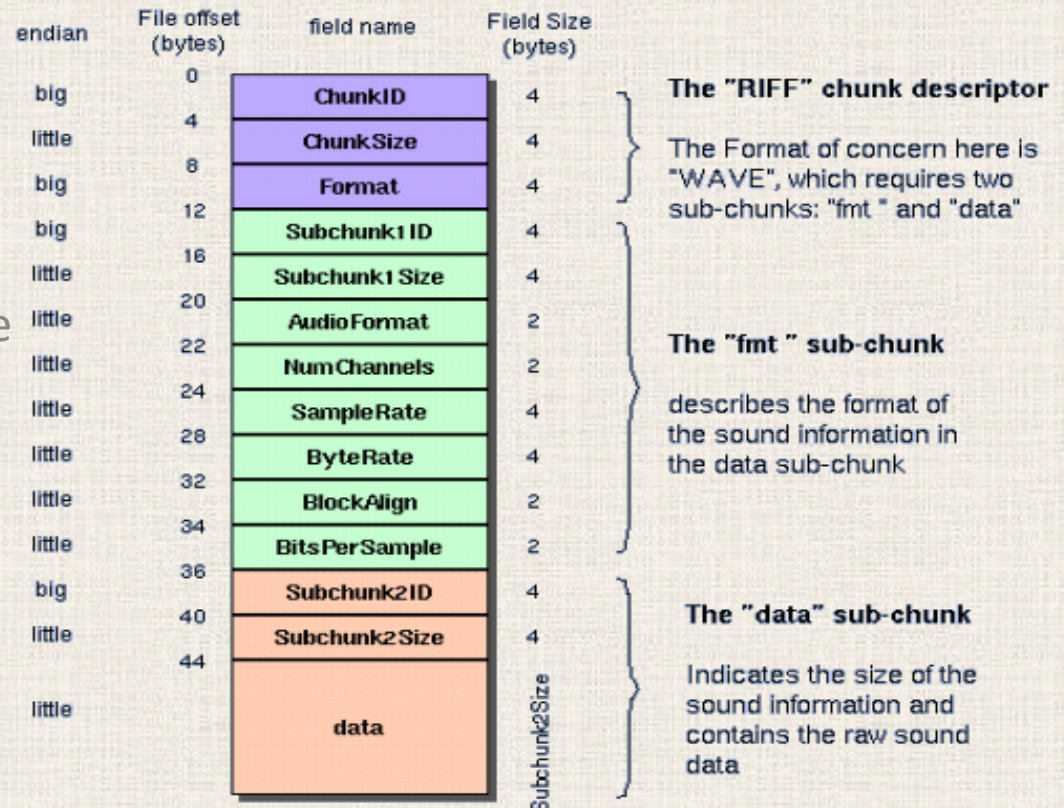
WAVE File Format

- WAVE files are a specialized subset of RIFF files
- ChunkSize**: file size in bytes (minus 8 bytes for ChunkID and ChunkSize fields)
- New WAVE files (extended WAVE file format) do not follow this standard exactly
- Most data stored in little-

endian standard

- PROBLEM**: Data stored in little-endian convention must be reversed before being used

- PROBLEM**: New "extended wave file format" –problems reading header to find data segment size



Audio Controller

-Little-endian WAVE file convention is handled here

- A splitter (hardware) rebuilds data in correct format for each audio channel before sending the data to the digital-to-analog converter

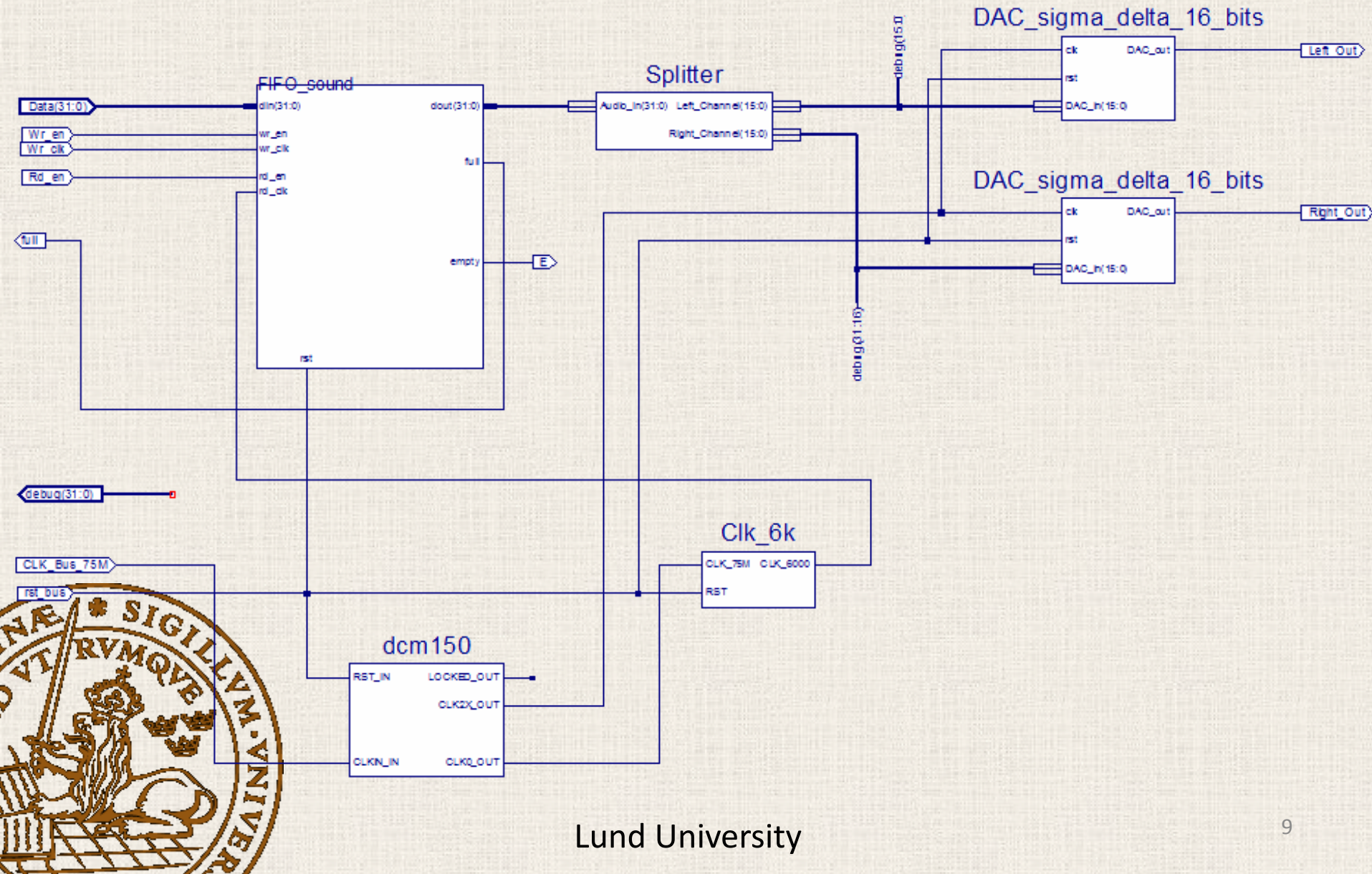
-At high sampling rate (CD quality – 44100 kHz) FIFO audio cue can be read faster than SD controller can write, producing cuts/loss in data

-Solution efforts:

- MicroBlaze clock frequency increased from 50 to 75 MHz via clock_generator IP-core (hardware)
- Audio FIFO queue size changed from 512 to 1024 32-bit samples
- Maximum sampling of 8000 kHz used

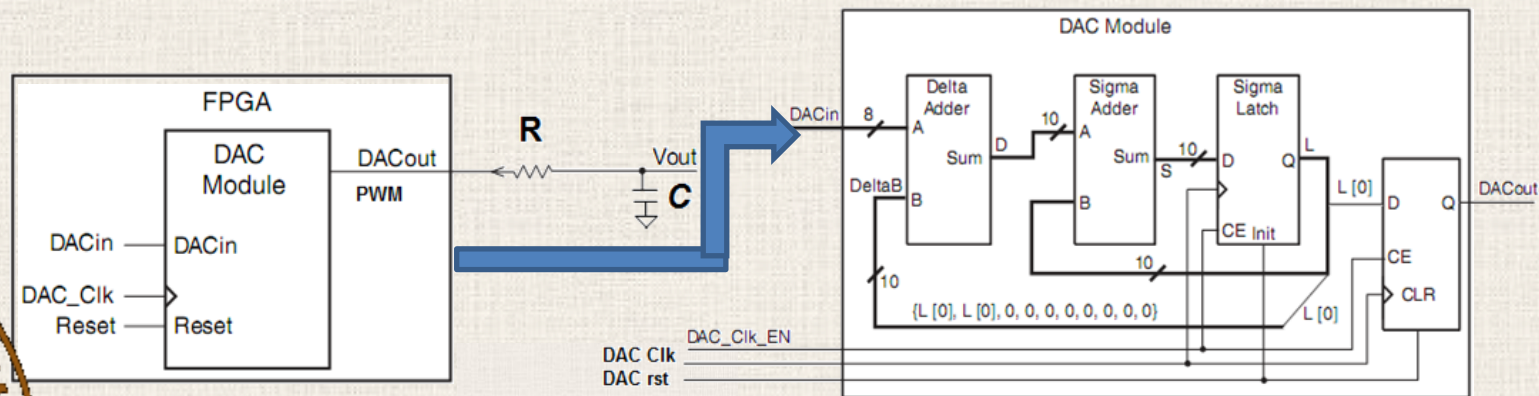


Audio Controller Schematic



Digital-to-Analog Converter

- Implemented in hardware
- No external Digital-to-analog converter on PModAmp, used on-board DAC
- **PROBLEM:** Lack of traditional external DAC means lack of proper filtering
 - PWM signals produced by DAC are linked directly to the inputs of PmodAMP
 - Signals are not filtered properly
 - A lot of noise exists in the audio output (PROBLEM – still unsolved)



User Interface

- Implemented almost entirely in software

- Push Buttons:**

 - Polling based

 - software starts/pauses the current song or skips to next/previous song

- Playlist:**

 - implemented in software through FatFS library

 - Playlist/user interface initialized by scanning root directory for WAVE files

 - Each entry in the playlist corresponds to a WAVE file in the root directory

- Song names are sent to LCD based on user push-button action

 - LCD driven by UART (hardware) at 9600 bds



Lessons Learned & Conclusions

- Estimating time required to complete embedded systems projects, technical projects in general, is difficult and time required is easily underestimated
- Pros and cons must be considered when choosing to implement a design primarily in either hardware or software
 - For certain applications hardware implementation is a much better choice
 - For other projects with demanding deadlines software implementation is sometimes easier to implement and debug
- Designing and successfully creating hardware in the real world is much more difficult, but also much more rewarding than simply designing hardware on paper



References

About SD, FATFS and Wave Files :

- <https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>
- http://elm-chan.org/fsw/ff/00index_e.html
- http://www.xilinx.com/support/documentation/ip_documentation/opb_deltasigma_dac.pdf
- http://www.sdcard.org/developers/tech/host_controller/simple_spec/Simplified_SD_Host_Controller_Spec.pdf

How to use Xilinx EDK :

- <http://www.labbookpages.co.uk/fpgas/edkHowTos/simple.html>
- http://courseware.ee.calpoly.edu/cpe-329/EDK_Resources.htm



FPGA-pod

Questions?

