SD Card Read/Write with the **Nexys4** FPGA Platform

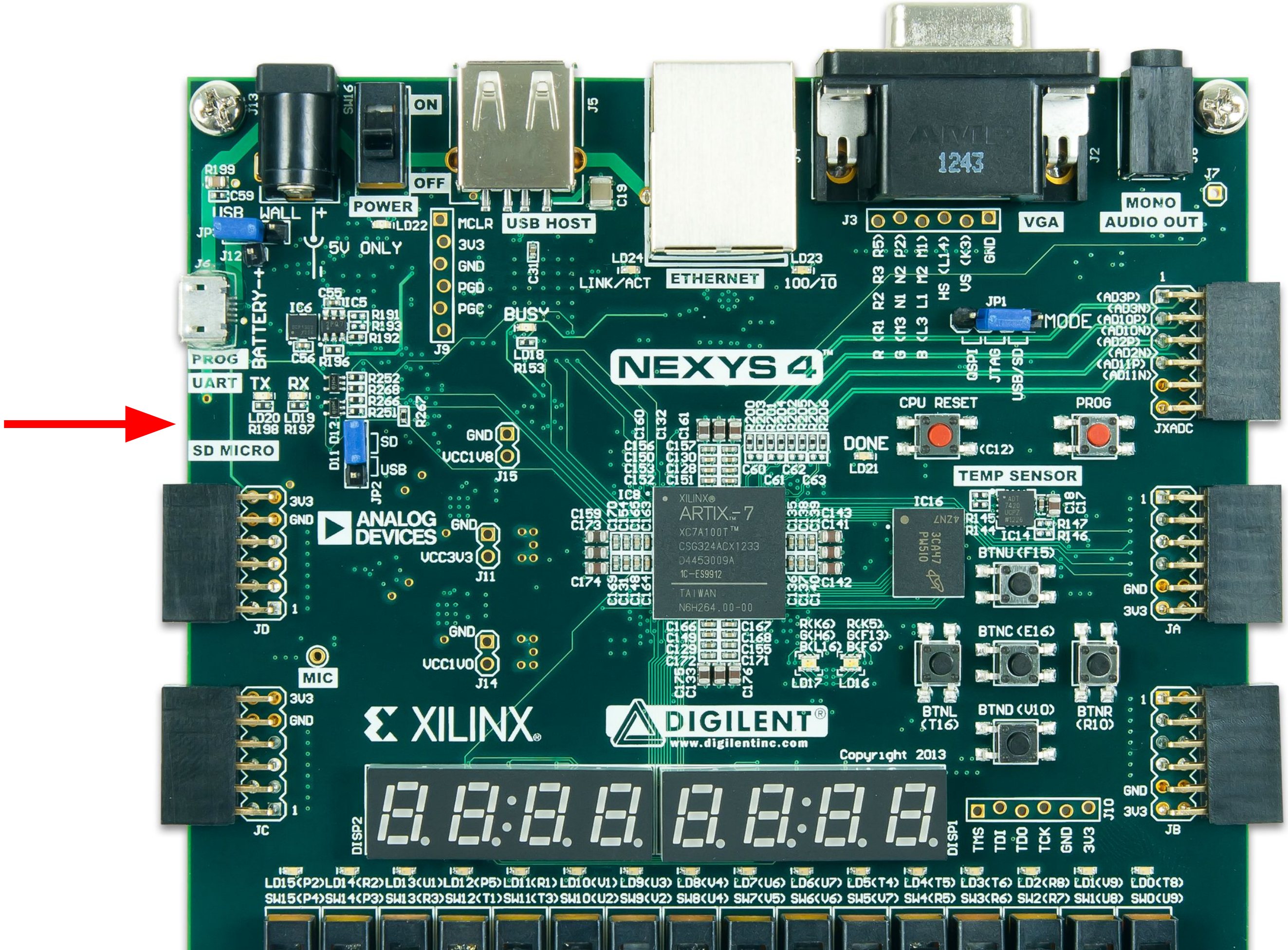
Jono Matthews | MIT EECS | 6.111 Fall 2015

Common Uses in Projects

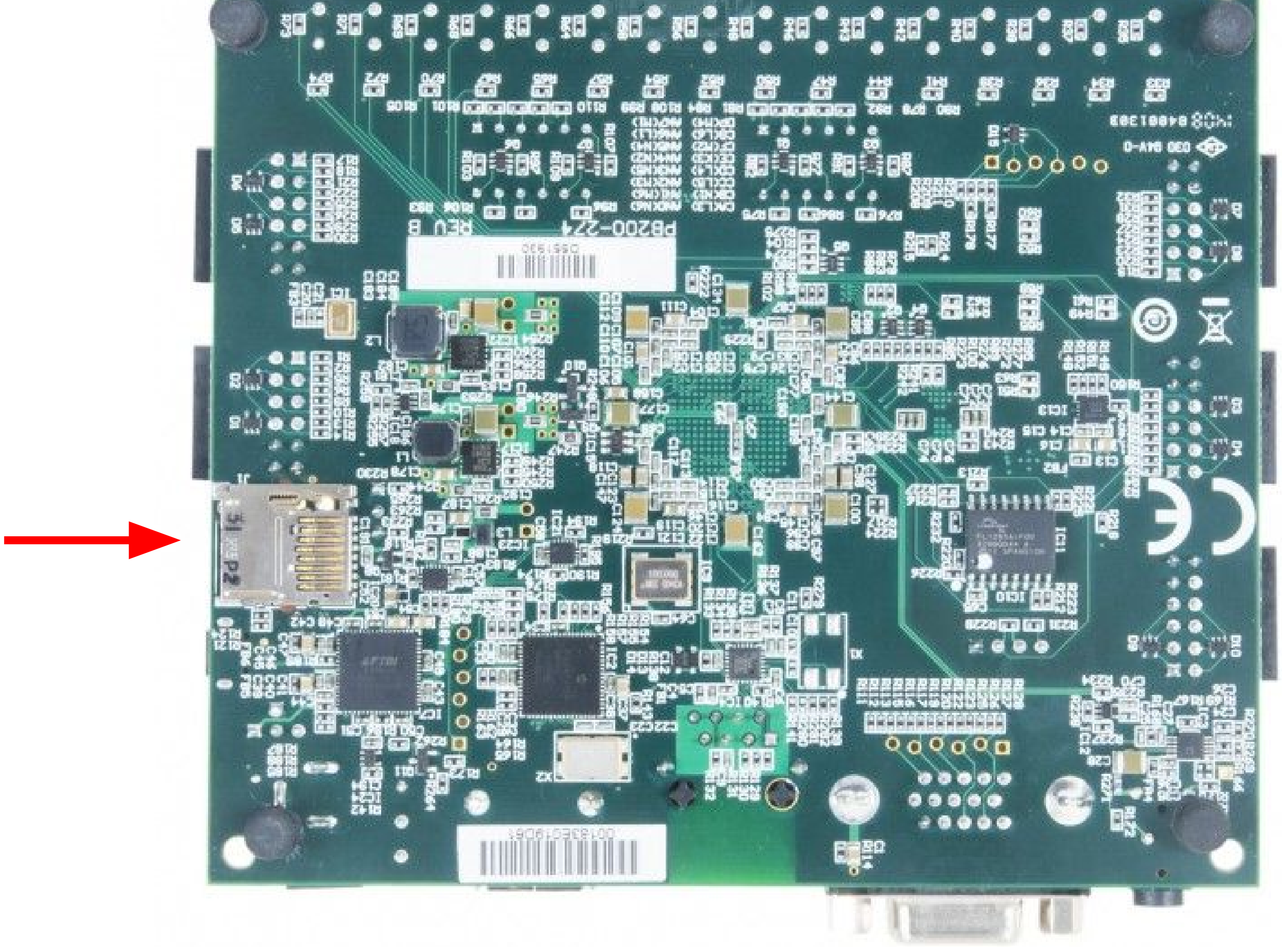
Video
Large color images

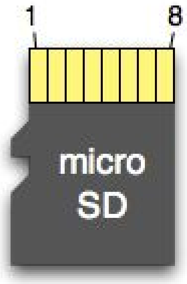
Audio
High-quality sound effects
Full-length songs

Misc.
Physiological recordings (ECG, EEG)
….



POWER
ON
OFF
USB HOST
ETHERNET
LINK/ACT
100/10
VGA
MONO
AUDIO OUT
USB
WALL
5V ONLY
MCLR
3V3
GND
PGD
PGC
BUSY
BATTERY
PROG
UART
TX
RX
SD MICRO
SD
USB
MODE
QSPI
JTAG
USB/SD
NEXYS 4
CPU RESET
PROG
DONE
TEMP SENSOR
XILINX ARTIX-7
BTNU (F15)
BTNC (E16)
BTNL (T16)
BTNR (R10)
BTND (V10)
JXADC
JA
JB
JC
JD
MIC
ANALOG DEVICES
XILINX
DIGILENT
www.digilentinc.com
Copyright 2013
DISP1
DISP2
TMS
TDI
TDO
TCK
GND
3V3
LD15(P2) LD14(R2) LD13(U1) LD12(P5) LD11(R1) LD10(V1) LD9(U3) LD8(V4) LD7(U6) LD6(U7) LD5(T4) LD4(T5) LD3(T6) LD2(R8) LD1(V9) LD0(T8)
SW15(P4) SW14(P3) SW13(R3) SW12(T1) SW11(T3) SW10(U2) SW9(V2) SW8(U4) SW7(V5) SW6(V6) SW5(V7) SW4(R5) SW3(R6) SW2(R7) SW1(U8) SW0(U9)





```
input SD_CD
output SD_RESET
output SD_SCK
output SD_CMD
inout SD_DAT[3:0]
```

Vivado Constraints

The sd_controller Module: Pins and Clock

```
module sd_controller(
    input reset,      // Resets controller on assertion.
    input clk,        // 25Mhz clock.
    output reg cs,    // Connect to SD_DAT[3].
    output mosi,          // Connect to SD_CMD.
    input miso,       // Connect to SD_DAT[0].
    output sclk,          // Connect to SD_SCK.
                      // For SPI mode, SD_DAT[2] and SD_DAT[1] should be held
                      // HIGH.
                      // SD_RESET should be held LOW.
    ....
```

The sd_controller Module: Reading

```
....
output ready,        // HIGH if the SD card is ready for a read or
                     // write operation.
input [31:0] address,// Memory address for read/write operation. This
                     // MUST be a multiple of 512 bytes, due to SD
                     // sectoring.
....
input rd,            // Read-enable. When [ready] is HIGH, asseting [rd]
                     // will begin a 512-byte READ operation at
                     // [address]. [byte_available] will transition HIGH
                     // as a new byte has been read from the SD card.
                 The
                     // byte is presented on [dout].

output reg [7:0] dout,        // Data output for READ operation.
output reg byte_available,    // A new byte has been presented on [dout].
....
```

The sd_controller Module: Writing

```
....
output ready,         // HIGH if the SD card is ready for a read or
                      // write operation.
input [31:0] address, // Memory address for read/write operation. This
                      // MUST be a multiple of 512 bytes, due to SD
                      // sectoring.
....
input wr,             // Write-enable. When [ready] is HIGH, asserting
                      // [wr] will begin a 512-byte WRITE operation at
                      // [address]. [ready_for_next_byte] will transition
                      // HIGH to request that the next byte to be written
                      // should be presentaed on [din].

input [7:0] din,                       // Data input for WRITE operation.
output reg ready_for_next_byte,    // A new byte should be presented on
                                   // [din].
....
```

A Useful Tool: HxD

Raw disk editing

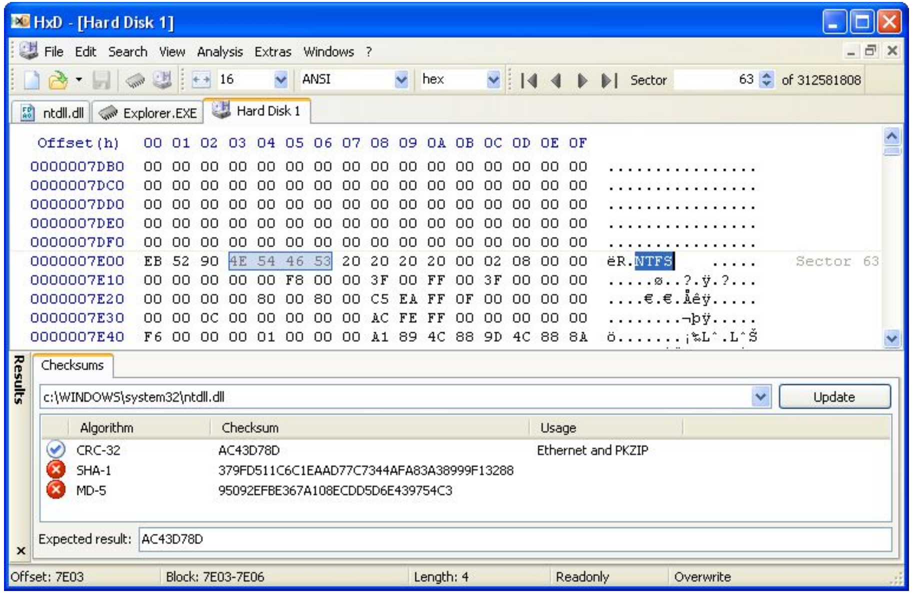
No filesystem needed

Windows:

http://mh-nexus.de/en/hxd/

Mac:

http://www.macupdate.com/app/mac/17562/hexedit



```
 Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

0000007DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000007DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000007DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000007DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000007DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000007E00  EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00  ëR.NTFS    .....   Sector 63
0000007E10  00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00  .....ø..?.ÿ.?...
0000007E20  00 00 00 00 80 00 80 00 C5 EA FF 0F 00 00 00 00  ....€.€.Åêÿ.....
0000007E30  00 00 0C 00 00 00 00 00 AC FE FF 00 00 00 00 00  ........¬þÿ.....
0000007E40  F6 00 00 00 01 00 00 00 A1 89 4C 88 9D 4C 88 8A  ö.......¡‰Lˆ.LˆŠ
```

c:\WINDOWS\system32\ntdll.dll

	Algorithm	Checksum	Usage
✓	CRC-32	AC43D78D	Ethernet and PKZIP
✗	SHA-1	379FD511C6C1EAAD77C7344AFA83A38999F13288	
✗	MD-5	95092EFBE367A108ECDD5D6E439754C3	

Expected result: AC43D78D

Offset: 7E03 Block: 7E03-7E06 Length: 4 Readonly Overwrite

Playing **Audio** with the **Nexys4** FPGA Platform

Jono Matthews | MIT EECS | 6.111 Fall 2015

Nexys4 Pins

`ampPWM` (audio PWM signal)

`ampSD` (keep asserted whenever sound should be playing)

The `audio_PWM` Module

```
module audio_PWM(
    input clk,                // 100MHz Clock.
    input reset,                  // Reset assertion.
    input [7:0] music_data,   // 8-bit music sample.
    output reg PWM_out        // Connect to ampPWM.
    );
```