MFM Emulator System Architecture
ROUGH DRAFT

Seth Morabito

June 17, 2014

Contents

1 Summary

The MFM Emulator is designed to act as a drop-in replacement for MFM disk drives. It is built around an Atmel ARM Cortex-M3 microcontroller and a Xilinx CoolRunner-II CPLD.

The MCU controls loading and storing raw MFM data into virtual disk files on an SD/MMC card, while the CPLD provides glue logic and real-time MFM data streaming.

2 Overview

2.1 Structure

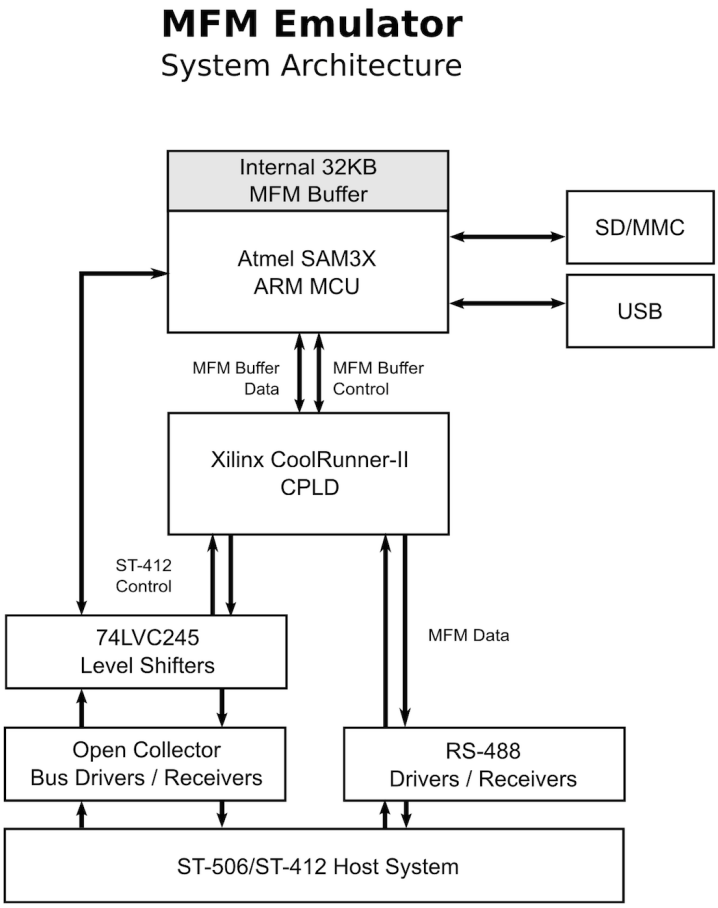

Figure 1: System Architecture

2.1.1 MCU

The MCU is an Atmel SAM3X8E operating at 84MHz. It fills several roles.

- It has an integrated 32KB buffer that holds one full cylinder of data in fast internal SRAM.
- It coordinates loading sector data from virtual disk files on a standard SD/MMC card.
- It responds to interrupts and maintains a state machine to handle read and write requests from the host.

2.1.2 CPLD

The CPLD is a Xilinx CoolRunner-II clocked at 50MHz. Its primary responsibility is real-time raw MFM streaming, but it handles quite a few other tasks.

- It has a small 64-byte FIFO buffer ("the MFM Buffer") that stores a cache of raw MFM data that is either being streamed into or out of the host system's disk controller.
- During host read operations, it shifts data written by the MCU out of the MFM buffer and into a serial MFM data stream with the correct timing.
- During host write operations, it shifts serial MFM data from the host into the MFM buffer, where it can be read by the MCU a byte at a time.
- It decodes the head select lines and passes an interrupt to the MCU when the selected head has changed.
- It counts steps, maintains the current cylinder number, and interrupts the MCU when stepping has finished.

2.1.3 Data Bus

The data bus is a bi-directional, tri-state bus with eight data lines that provides communication between the MCU and the CPLD MFM Buffer.

The primary purpose of the data bus is to transfer data between the MCU's track buffer and the CPLD's MFM buffer, and to communicate state changes from the CPLD to the MCU. The data transfer is coordinated by the control bus, described below.

2.1.4 Control Bus

The control bus coordinates data transfers between the CPLD and the MCU. It consists of the following four lines.

Buffer Ready (/BRDY)
: CPLD Output. During disk read (MFM buffer write) operations, a low level on this line indicates that the buffer is NOT FULL, and can be written to by the MCU. During emulated disk write (MFM buffer read) operations, a low level on this line indicates that the buffer is NOT EMPTY and can be read from by the MCU.

Buffer Error (/BERR)
: CPLD Output. During disk read (MFM buffer write) operations, a low level on this line indicates that the buffer has over-run. During disk write (MFM buffer read) operations, a low level indicates that the buffer has under-run.

Data Ready (/DRDY)
: CPLD Output. When asserted, the data on the data bus is valid.

Write Enable (/WE)
: CPLD Input. Indicates the direction of read/write operations on the data bus. A high level indicates that the MFM buffer may be read from (disk write operation). A low level indicates that the MFM buffer may be written into (disk read operation).

Output Enable (/OE)
: CPLD Input. When low, the data bus output is enabled. When high, the data bus is high impedence (tri-stated).

Address (A0, A1)
: CPLD Input. These two lines taken together are decoded as the address of the CPLD register to be read from or written to by the MCU.

A1	A0	Register
0	0	MFM Buffer
0	1	Cylinder Number (High)
1	0	Cylinder Number (Low)
1	1	Bit Count

2.1.5 Host Interface

The Host connection is a standard ST506/ST412 interface operating at 5V. Standard open-collector bus drivers and receivers are used, and lines are terminated in accordance with the ST506/ST412 specification.

The bus drivers and receivers interface the host to the 3.3V CoolRunner-II CPLD through a set of three 74LVC245 transceivers in a logic level shifting configuration.

2.1.6 ST506/ST412 Bus

The ST506/ST412 bus is described in the *ST412 OEM Manual* (Seagate, 1982) [1], and is summarized here. All lines except the differential line pairs **MFM Write Data** and **MFM Read Data** are driven by open-collector drivers, terminated by 220/330 Ohm resistor pairs, and are active-low.

Seek Complete
: Asserted by the MFM emulator to indicate that a seek operation has completed.

Ready
: Asserted by the MFM emulator to indicate that it is ready to read, write, or seek, and that I/O signals are valid. When this line is false, all writing and seeking is inhibited.

Track 0
: Asserted by the MFM emulator to indicate that the currently selected track is Track 0.

Write Fault
: This line is not used.

Index
: Asserted by the MFM emulator once every approximately 16.67ms to indicate the start of a track of data during read operations. The pulse is held low forl approximately 200us.

Step
: Asserted by the host controller to indicate that the selected track number should increase or decrease, depending on the value of the **Direction In** line.

[1] `http://bitsavers.informatik.uni-stuttgart.de/pdf/seagate/ST412_OEMmanual_Apr82.pdf`

Direction In
: Asserted by the host controller to indicate the direction of stepping. A high value on this line indicates that the track number should decrease with each **Step** pulse. A low value indicates that the track number should increase with each **Step** pulse.

Write Gate
: When asserted by the host, data from the host is written to the MFM emulator.

Head Select 0, 1, 2, 3
: These four lines are asserted by the host controller to select heads 0 - 15. Head Select 0 is the least significant line. Note that on older drives, Head Select 3 was the **Reduced Write Current** line, and not used as a head select at all. This behavior is not planned for the MFM emulator.

Drive Select 1, 2, 3, 4
: Any one of these lines can be asserted by the host to select a different virtual, emulated drive.

Drive Selected
: Asserted by the MFM emulator to indicate that the drive has been selected by one of the **Drive Select** lines.

MFM Write Data
: This differential pair is used by the host to write data to the emlulated MFM drive.

MFM Read Data
: This differential pair is used by the host to read data from the emulated MFM drive.

3 Operation Overview

3.1 Emulated Disk Read

When the host system wishes to read data from the MFM emulator, it begins by seeking for the desired head and cylinder. This head and cylinder are translated into an offset into a virtual drive file residing on an SD/MMC card. After the seek operation has completed, raw MFM data representing

the selected head and cylinder is continuously streamed to the host system, exactly as it would be if it were reading from a real disk drive. The host system is fully responsible for finding the desired sector in this data stream.

The full procedure for reading is as follows:

1. The host selects a drive with one of its **DRIVE SELECT** lines.
2. The MFM emulator responds by opening the correct virtual hard drive file on the SD/MMC media, and determining the currently selected cylinder and head number. This is translated into an offset into the file. The contents of the desired cylinder are then read into the MCU's Track Buffer.
3. After the load is complete, the MFM emulator asserts **READY**, **SEEK COMPLETE**, and, if the currently selected track is track 0, **TRACK 0**.
4. The MFM emulator begins transferring data from the MCU's Track Buffer to the CPLD's MFM Buffer, byte by byte, from which it is streamed to the host. This will loop continuously until the drive is stepped, the head is changed, or the **Drive Select** line is de-asserted. At the start of each loop of the Track Buffer (approximately once every 16.67ms), the **INDEX** line is asserted for 200us.
5. The host may choose to step to a new track, change the head number, or both, at any point. If so, one or more of the following occurs:
   - The appropriate **DIRECTION IN** is set.
   - The appropriate **HEAD SELECT** are asserted.
   - The **STEP** line is pulsed one or more times.
6. If the any of these conditions is met, the MFM Emulator responds as follows:
   - The MFM emulator immediately de-asserts the **SEEK COMPLETE** line and begins counting the the number of steps taken.
   - When it has been at least 5ms since the last step pulse, the MFM emulator makes note of the new head and track number, and then reads the new track from the virtual disk file into the MCU's track buffer.

- The MFM emulator re-asserts **SEEK COMPLETE** and, if the currently selected track is track 0, **TRACK 0**.
- The MFM emulator begins streaming data for the new track (see #4 above)

3.2 Emulated Disk Write

Writing to disk is a bit more complicated. The process starts identically to a read operation, with the host system selecting a head and cylinder to write to, and then reading the data stream until the desired sector is found.

The procedure for writing is as follows:

1. The host begins by reading the track that contains the sector it wishes to write to.
2. When the correct sector ID has been found, the host asserts **WRITE GATE**.
3. Immediately upon detecting the falling edge of **WRITE GATE**, the CPLD switches from READ to WRITE mode. Here, some gymnastics take place in the CPLD. It latches the current bit position in the cylinder stream it has been writing, and stores this in its **BIT POSITION** register. Then, it interrupts the MCU and switches it from read to write mode, and simultaneously begins to store the stream of MFM write data into its MFM buffer.
4. The MCU begins its read operation by asking the CPLD for the exact position in the bitstream where the write operation began.
5. After the bit position is transferred to the MCU, the MCU starts to read data from the CPLD's MFM Buffer and into it's track buffer, one byte at a time, starting at the correct bit offset.
6. When the CPLD detects the rising edge of **WRITE GATE**, it switches from WRITE to READ mode and interrupts the MCU again.
7. The MCU finishes any data transfer operations it was carrying out, and then starts streaming its newly written track buffer back to the CPLD.
8. The next time the CPLD's MFM buffer is full, the MCU will conduct housekeeping by flushing the dirty track buffer back out to the virtual disk file on the SD/MMC card.

4 Open Issues

4.1 Write Timing

The timing of switching from read to write mode is critical, and must be handled in real time. The MCU must know EXACTLY where in the bit stream the switch to write took place.

4.2 Write Precompensation and Postcompensation

The question of how to handle precompensation and postcompensation from the host controller – or even whether we *need* to handle it – is an open question. Host controllers typically have complex logic to manipulate the pattern of bits written to disk in order to compensate for the influence that one bit can have on another on a magnetic medium. It is unclear how this will interact with the controller. If there is an interaction, the CPLD will need to de-compensate on writes.

5 Implemenation TODO List

- ☐ CPLD/MCU Communication Protocol.
- ☐ Spec for all CPLD registers.
  - ☐ MFM Bit Count register.
  - ☐ Cylinder Number register.
- ☐ MCU Track Buffer implemenation with bit-wise offset.
- ☐ Verilog for MFM Buffer.
- ☐ Interrupt routing (CPLD)
- ☐ Interrupt handling (MCU)