The designs of the master and slave CCB FPGAs

[Document number: ?, revision 0]

Martin Shepherd, California Institute of Technology

March 26, 2004

This page intentionally left blank.

Abstract

The aim of this document is to detail the design of the CCB FPGA firmware, and define its interfaces to the rest of the CCB hardware. The design will be presented in a hierarchical manner, starting with block diagrams of major components and their interconnections, and ending with the VHDL code that sythesizes the lowest level components displayed, and connects them together.

[Only the top level of this heirarchy is shown so far]

Contents

List of Figures

Chapter 1

The slave FPGAs

There will be 4 slave FPGAs controlled by one master FPGA. All of the slave FPGAs will be identical, so this chapter documents the internal components, and external I/O connections of a single slave FPGA. Figure 1.1 shows the layout of a slave FPGA, showing the major logic components within the FPGA, the internal interconnections between these components, and all of the external I/O-pin connections to the 4 ADCs to the left, and to the master FPGA, shown at the bottom of the diagram.

1.1 An overview of the internals of a slave FPGA

Starting from the left hand side of the diagram, the 14 bit data-signals and 1-bit overflow signal of each of the 4 ADCs, are simultaneously latched into a corresponding 15-bit register at the start of every 10MHz clock cycle. Simultaneously, the previous values of these registers are read by integrator components. When in dump mode, the output of one of these 15-bit outputs is also siphoned off, to be read directly by the master FPGA. The outputs of the integrators are subsequently flash loaded into a FIFO, at the end of each integration, ready to be read out by the master FPGA.

1.1.1 Normal integration mode

When not in dump mode, the output of each ADC, delayed by one clock cycle by the input register, and by another 4 clock cycles by the pipeline within the ADC, is read into an integrator component.

The integrator component, which will be detailed shortly, either ignores the new sample, if the `drop` signal from the master FPGA is asserted, or adds it to the accumulation bin specified by the 2-bit `phase` signal, as received from the master FPGA.

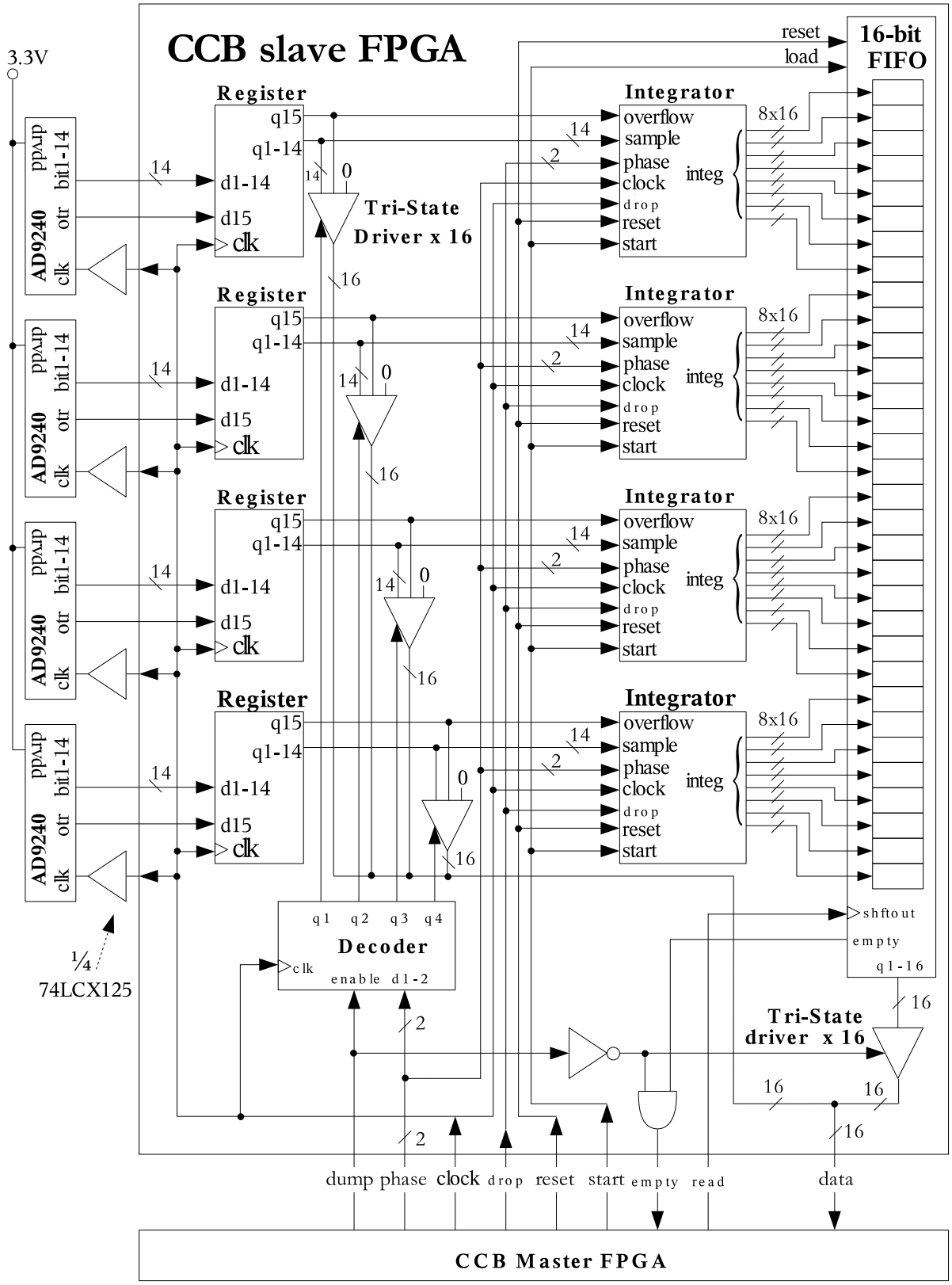

Figure 1.1: The top-level design of the slave FPGA

At the end of each integration period, the master FPGA asserts the `start` signal for one clock cyle. This causes the output FIFO, on the right of the diagram, to flash-load the contents of all of the 32-bit accumulator output bins of the integrators, and at the same time, the integrator components clear all of their accumulators, and add in the latest sample to the appropriate empty bin, unless the `drop` signal is asserted.

The accumulator bins are 32-bits wide, but there are only 16 pins available for clocking the data out of the slave FPGA. Therefore a 16-bit wide FIFO is used, with two entries assigned to each accumulator bin. Since there are 4 accumulator bins per integrator, 4 integrators, and 2 FIFO entries per accumiulator, the 16-bit wide FIFO will be 32 entries in length.

Once the FIFO has been flash loaded with the accumulator values, its `empty` signal becomes deasserted, which tells the master FPGA that there are new integrations waiting to be read into its output FIFO. When the master FPGA is ready to do this, it clocks the `read` line of the slave FPGA, in sync with the global 10MHz clock, to read out the FIFO. Once all data have been read from the FIFO, its `empty` signal is asserted, telling the master FPGA to direct its attention to reading data from another slave FPGA, if any.

1.1.2 Dump mode

In dump mode, although the integrators still integrate data, the tri-state output driver following the FIFO, is placed into a high impedance state, to enable one of the 16-bit tri-state drivers at the outputs of the ADC input registers, to instead direct the output of a chosen ADC, directly to the `data` output of the slave FPGA. In this mode, the 2-bit `phase` signal is re-interpretted as the address of the ADC to be dumped, so an address decoder, which is only enabled when the `dump` signal is asserted, is used to enable the chosen tri-state buffer, according to this address.

At the end of dump mode, the master FPGA briefly asserts the slave FPGA's `reset` signal, to discard the garbage contents of the integrators and the output FIFO, before normal integration mode resumes.

1.1.3 External connections

The FPGA I/O pins should be configured for 3.3V logic levels.

Provided that the `DRVDD` pins of the ADCs are connected to a 3.3V power-supply, then the output data pins of the ADCs can directly drive the corresponding input pins of the FPGA. This may not be the case if there are cables or filters in between. Unfortunately, the ADC clock input appears to require a TTL signal, regardless of the voltage at the ADC `DRVDD` pin. Thus I have shown individual 3.3V to TTL buffer amplifiers between the FPGA clock output pins and the ADC clock input pins. If the clock output pins of the FPGA are close

together, then presumably the number of FPGA clock output pins could be reduced; as could the number of separate buffer chips.

Note that an assumption in this design, which may not be correct, is that the interconnections between master and slave FPGAs, and between the slave FPGAs and the analog board, will be via daugther-board connectors, rather than via cables and EMI filters. The scenario in mind has a central board housing the master FPGA, the USB interface and the parallel port interface, then on the component side of this board, would be 4 daughter cards holding the slave FPGAs, and on the other side of the board, shielded by the ground plane of the master FPGA board, would be the daughter board housing the analog electronics. The connections from the ADCs to the slave FPGAs would thus go via two levels of daughter card connectors. If a significantly different scheme is adopted, which includes interboard cabling and/or EMI filtering, then extra buffer amplifiers may need to be added for the data lines, along with some means to compensate for any rise time delays added by EMI filters.

Chapter 2

The master FPGA

Figure 2.1 shows the layout of the master FPGA, showing its major internal components, along with their interconnections, and all of the external I/O-pin connections to external chips. The central brain of this design, is the *State Generator* component, which orchestrates the timing and values of all control signals going to the other components and the slave FPGAs. The *State Generator* is in turn told what to do by the computer, via the *Control Gateway* component, which handles all interactions with the parallel port interface. The *Data Dispatcher* component is responsible for sending integrated and dump-mode data to the computer, via the USB interface.

2.1 The Control Gateway

The *Control Gateway* handles all interactions with the CCB computer's EPP parallel port interface. It places commands and configuration data that it receives from the computer, within an array of registers, which are always visible to the *State Generator*. It also raises parallel-port interrupts, when so directed by the *State Generator*. Note that apart from the interrupt mask, which is read by the computer from its interrupt handler, all data is directed from the computer to the FPGA.

The reset signal of the EPP parallel port is used to reset the firmware and the USB chip. This can be asserted at any time by the device driver in the CCB computer, and this will automatically be done every time that the device driver is loaded or reloaded.

The *Control Gateway* component will be designed to present an 8-bit register based interface to the computer. This is simplified by built-in support for separate address and data cycles in standard EPP hardware. In particular, the receipt of an address-byte by the *Control Gateway* will be interpretted as the address of the 8-bit register into which to record subsequently received data-bytes.

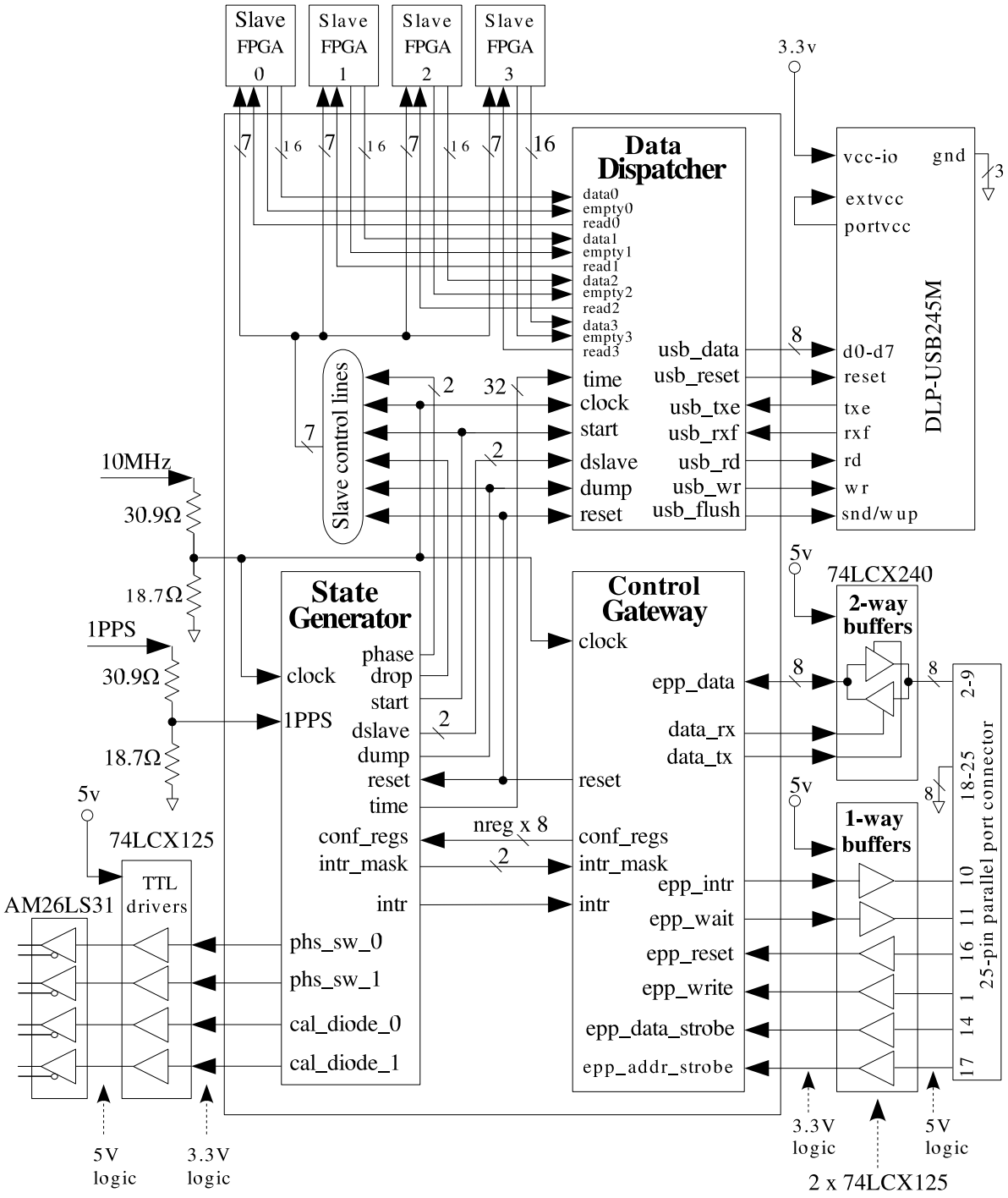

Figure 2.1: The top-level design of the master FPGA

Since no read-back of the internal registers is planned, any read request from the computer will be responded to with the interrupt mask, which will say which interrupts have been generated since the last time that this mask was read. This mask can thus be read from an interrupt handler with a single EPP read, without having to worry about other incomplete multi-byte read transactions.

There are only two times when data will be sent to the master FPGA by the computer.

1. When starting a new scan, a write to the control register will be used to prepare the *State Generator* for reconfiguration. This will be followed by multiple writes, to send the configuration data of the new scan, and terminated by the command which instructs the *State Generator* to start the new scan.

   Since the FPGA does nothing with the configuration data that it is sent, until it is told to start the next scan, it is safe to send the values of multi-byte configuration registers, a byte at a time.

2. During a scan, the configuration of the calibration diodes (for the integration after the next integration), will be sent, on receipt of each end-of-integration interrupt.

   Since between scans, only the calibration diode configuration register is written to, the device driver will send the address of this register once per scan, just after starting each new scan, such that the interrupt handler needn't specify the register to write its cal-diode reconfiguration info into, before writing said info. Thus at the end of each integration period, there will be one EPP read to get the interrupt mask, plus one EPP write to reconfigure the calibration diodes.

2.2 The Data Dispatcher

At the end of each integration period, and at the start of dump mode, the *Data Dispatcher* component, will read integrated or dump-mode data from the slave FPGAs, into a large FIFO, after prepending a start-of-frame header and timestamp, then stream the contents of this FIFO to the computer, via the USB link. Note that all communications over the USB bus will be directed from the FPGA to the computer. Thus, although the read (rd) and read-enable (rxf) pins of the USB interface are shown as inputs to the *Data Dispatcher*, there are no plans to use them at the moment.

Note the use of the DPL-USB245M module. This is a tiny PCB module containing a 6MHz crystal, a surface-mount FT245BM USB1.1 chip, a USB connector and all the interconnections needed between these parts. The PCB is just 1.5×0.7 inches in size, and the USB connector sticks out a further third of an inch from one end. The module can be soldered onto the CCB PCB, via 24 dual in-line pins. Its data-sheet can be downloaded from:

`http://www.dlpdesign.com/usb/dlp-usb245m12.pdf`

The two of these modules that I bought for testing the FT245BM, I got from a company called Saelig (`www.saelig.com`), which is an official US distributor for the FT245BM. The modules arrived overnight. Since then, I have noticed that Mouser Electronics carries them as well. Their catalog number at Mouser is 626-DLP-USB245M, and they cost $25.

2.3 The State Generator

TBD

[Note that the clock and reset lines aren't currently shown as being distributed to the slaves by the State Generator*, whereas they must be, to allow the* State Generator *to reset and halt the slave FPGAs, while it is in the process of receiving reconfiguration information for a new scan. I'll fix this soon]*