App Note - 310

Application Note: Addressing Multiple FPAAs Using a SPI Interface

Rev:	1.0.0
Date:	23rd Jan 2015

TABLE OF CONTENTS

1 Purpose

The purpose of this document is to describe the SPI protocol and how it can be used to address multiple AN231E04 devices, henceforth referred to as FPAAs.

The SPI protocol is described in detail, then how to use it to address multiple FPAAs, both in series and parallel. Finally there is a section on buffering and layout considerations.

2 The SPI Interface

2.1 Overview

SPI stands for Serial Peripheral Interface and is a low-cost, low-speed communication option for short distances like those within a PC box. It is full duplex and very easy to implement between two hosts. SPI is capable of data rates in the tens of megahertz, but normal processor based SPI interfaces generally support data rates in the 1 to 3MHz range.

When using the SPI protocol, a Master and one or more Slaves are defined. In systems using Anadigm devices, the master is always the host processor. Anadigm devices are always SPI slaves. Figure 1 illustrates the single Master, single Slave case:

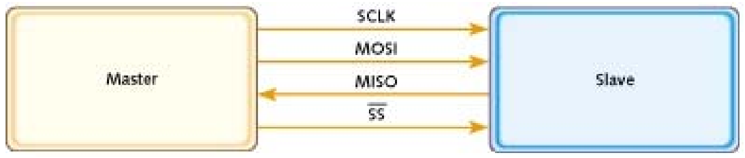

Figure 1: Single master / single slave

The four signals shown are in table 1:

Signal Name	Description	Anadigm Pin name
SCLK	Clock	SCLK
MOSI	Master out - Slave in	SI
MISO	Master in - Slave out	MEMSETUP_SO
SS	Slave select	CS2b

Table 1: Signal names

The clock is generated by the Master and sent to all slaves. The slave select line is used to indicate which Slave the Master is talking to. Figure 2 illustrates a case with multiple slaves, which could be Anadigm FPAAs.

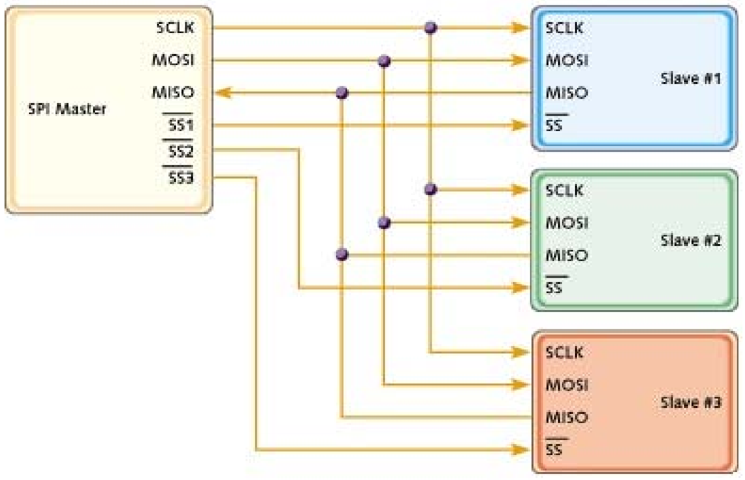

Figure 2: Single master / multiple slaves

As can be seen, all the signals except the slave selects are common to all slaves. Therefore the Master must indicate which Slave should be active when communicating. With many Slaves, the number of required outputs from the Master and the number of traces on a board become too great to justify use of this protocol. However, the Anadigm FPAA overcomes this issue by making use of ID fields embedded in the configuration data to the device.

With Anadigm FPAAs, all the CS2b chip selects can be tied together in parallel, and individual devices are communicated with only if CS2b is low AND the ID for that device is correct. It is of course also possible to address Anadigm devices using separate select lines if preferred.

NOTE: FPAAs have 2 select lines – CS1b and CS2b. CS2b is usually used as a SPI select because CS1b is used in chaining of FPAAs. This will be discussed in detail in section 3.

2.2 Detailed Description

SPI uses a couple of parameters called clock polarity (CPOL) and clock phase (CPHA) to determine when data is valid with respect to the clock signal. These are normally set the same on the Master and all the Slaves in order for communication to work. CPOL determines whether the leading edge is defined to be the rising or falling edge of the clock (and vice versa for the trailing edge). CPHA determines whether the leading edge is used for setup or sample (and vice versa for the trailing edge). Table 2 summarizes the various settings:

CPOL/CPHA	Leading Edge	Trailing Edge	SPI Mode
0/0	Sample, rising	Setup, falling	0
0/1	Setup, rising	Sample, falling	1
1/0	Sample, falling	Setup, rising	2
1/1	Setup, falling	Sample, rising	3

Table 2: SPI settings

Anadigm devices support SPI modes 0 and 3. These are the most commonly supported modes as reference to SPI EPROM datasheets will quickly tell you. Note that there is no need to physically select one of these modes on the Anadigm device. Both modes are supported by default. The master must however be set to transmit/receive data in either mode 0 or mode 3.

Figures 3 and 4 illustrate how these modes work:

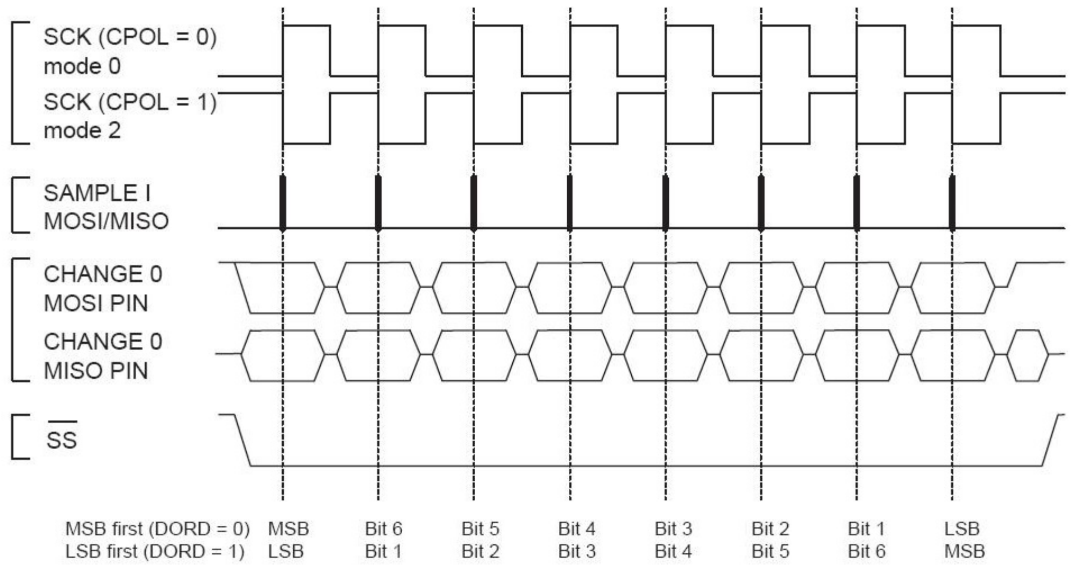
SCK (CPOL = 0)
mode 0
SCK (CPOL = 1)
mode 2
SAMPLE I
MOSI/MISO
CHANGE 0
MOSI PIN
CHANGE 0
MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

Figure 3: SPI formats with CPHA=0

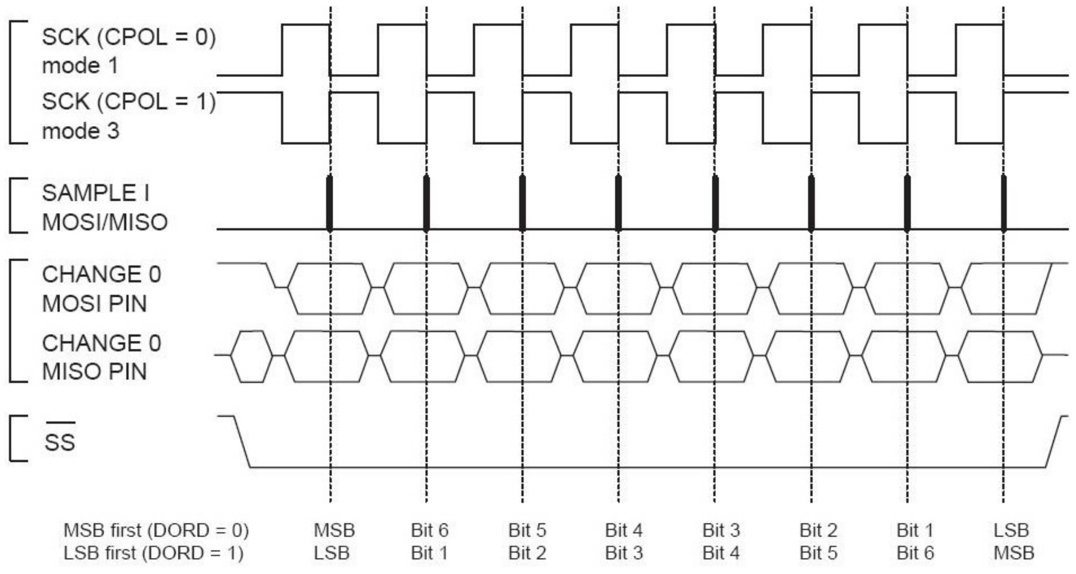

Figure 4: SPI formats with CPHA=1

In these examples note that either the MSB or the LSB can be sent first depending on how the SPI hardware is configured. In Anadigm Devices, the MSB is always sent first.

Communication between a master and slave occurs as follows:

1. Master pulls low the slave select line of the desired slave device. This indicates to each device to prepare for communication.

2. The Master generates the clock signal according to its SPI mode. Both Master and Slave transmit one bit per clock cycle.

3. After a byte has been sent the Master pulls the slave select line high.

What has been described here is all that the SPI protocol provides. There are no handshaking, acknowledgement or any advanced communication standards associated with it, so those would have to be implemented on top of this layer if needed.

Anadigm devices allow checking of the transmitted data from master to slave via CRC or basic error checking. Anadigm devices also incorporate an ID in their serial bitstreams which allows the individual select lines to be discarded if required and be replaced by a common select line. Devices are ‘selected’ by asserting the chip select line and sending data which includes an ID which matches one previously loaded during a primary configuration.

2.3 SPI Waveforms for Data Transfer in Anadigm Devices

Below are two waveforms taken directly from simulations of data transfer to an Anadigm device. In the waveform in figure 5, the Mode 0 SPI format is used. The bytes being transferred to the slave on SI are 2E,2F,30,2A (left to right). The bytes being read back from the slave to the master on MEMSETUP_SO are 00,2D,2E,2F(left to right). Note that when the Chip select line CS_B is taken high , the MEMSETUP_SO pin tristates. Observation will show that this format, used in simulations matches the Mode0 format above. Note though that the Cs_b signal goes low a nominal clk cycle before the first bit of data is set up. This has no effect on the simulations.

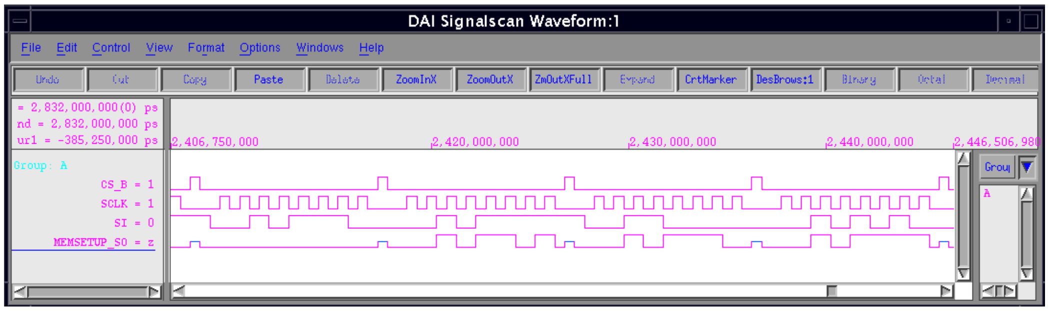

Figure 5: Mode 0 data transfer with FPAA

In the waveform in figure 6, Mode 3 SPI format is used. The bytes being transferred to the slave on SI are 2E,2F,30,2A (left to right). The bytes being read back from the slave to the master on MEMSETUP_SO are 00,2D,2E,2F(left to right). Note that when the Chip select line CS_B is taken high , the MEMSETUP_SO pin tristates. Observation will show that this format, used in simulations matches the Mode3 format above. Note though that the Cs_b signal goes low a nominal clk cycle before the first bit of data is set up. This has no effect on the simulations.

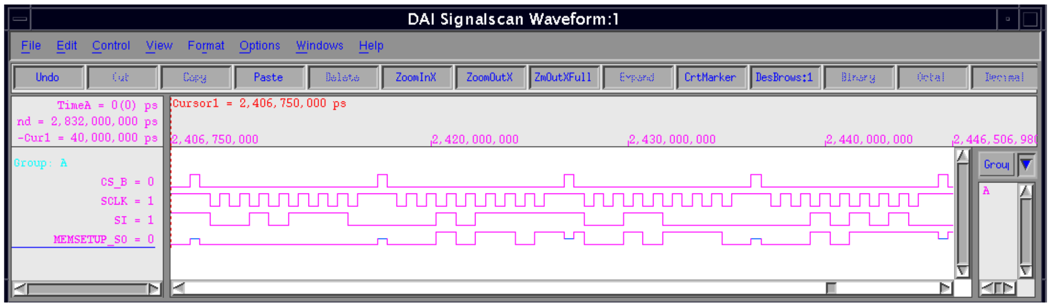

Figure 6: Mode 3 data transfer with FPAA

NOTE: the above section discusses bi-directional communication on a SPI port. Data is read from Anadigm FPAAs using its data out pin, called MEMSETP_SO. In practice it is rarely necessary or even useful to read back data from the FPAA. Consequently the next section will discuss uni-directional communication only i.e. configuring of FPAAs.

3 Addressing Multiple FPAAs

3.1 FPAA Chains

This section describes how to address multiple AN231E04 FPAAs from a SPI port. Figure 7 shows two chains of FPAAs. Each chain can consist of up to 255 FPAAs, each with its own 8bit address (0x00 to 0xFE). Address 0xFF is used to broadcast a reconfiguration update to all FPAAs in the chain.

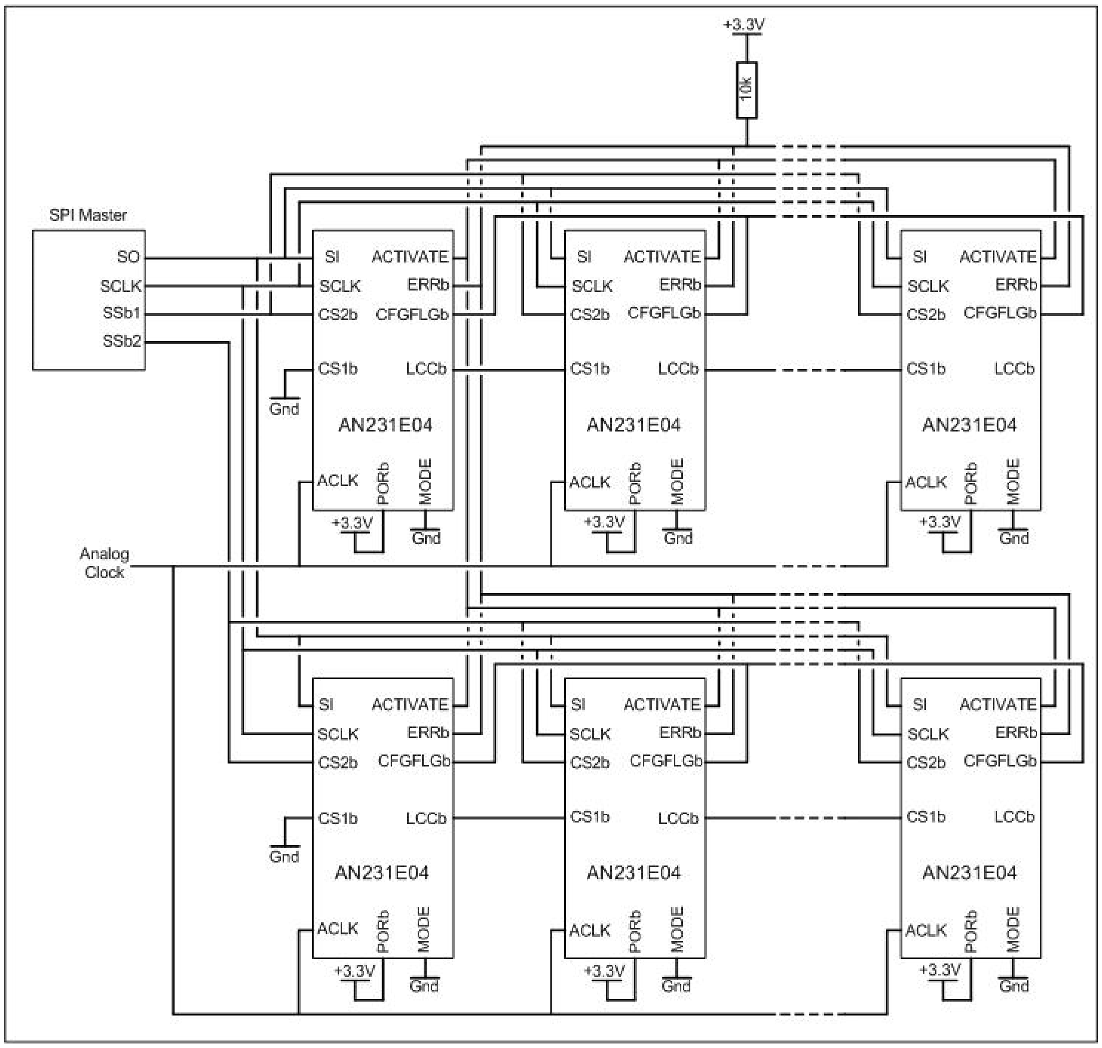

Figure 7: Multiple FPAAs in more than one chain

The FPAA addresses are allocated during primary configuration. During primary configuration of a chain, the appropriate select line will be activated (e.g. SSb1 for the upper chain in figure 7 will be driven low) and then N primary configurations will be sent where N is the number of FPAAs in that chain (<256). The primary circuits will be loaded into each FPAA in turn, starting at the left. As each FPAA is configured, it will adopt whatever address is allocated to it in the primary configuration header. Once primary configuration is complete, each FPAA will then respond to any reconfiguration update that is addressed to it.

A chain of FPAAs is characterised as follows:

1. A common select line to all CS2b pins.
2. A common clock line to all SCLK pins.
3. A common data line to all SI pins.
4. LCCb of each FPAA connected to CS1b of the next FPAA. The first FPAA has CS1b tied low, the last FPAA has LCCb open.
5. All CFGFLGb pins are tied together.
6. All ACTIVATE pins are tied together.
7. All ERRb pins are tied together (with a pull-up resistor).

As each FPAA in the chain is configured, it drives its LCCb pin low which selects the next FPAA in the chain. The next primary configuration will configure the newly selected FPAA (the FPAA already configured will ignore subsequent primary configurations).

The ACTIVATE pins, which are open drain, are commoned which means that this line will remain low until the last FPAA has been primary configured (each FPAA drives ACTIVATE low until it is configured, then it releases it). At that point all of the FPAAs will have released ACTIVATE and started monitoring it. This will allow the line to pull high and as the FPAAs detect it going high, they will activate their circuits. This process ensures that all of the circuits start up together. Only one device in a multi-FPAA system should have its interal PULLUPS set. If a large number of dpASPs are being used and loading becomes an issue, external pull-up resistors on the CFGFLGb and ACTIVATE nets are appropriate in lieu of one of the internal pull-ups. To turn off the internal pull-ups for ACTIVATE and CFGFLG, in each FPAA. Using AnadigmDesigner2, for each FPAA, one at a time select the FPAA “Chip Setting” then within the “Chip tab” deselect Pull-ups. If the CFGFLG net is also common together the same single internal Pull-up should be selected, the same control is sued for CFGFLGb and ACTIVATE.

If any FPAA sees an error in its primary configuration data then it will assert ERRb low for 19 SCLK cycles which will cause all FPAAs in the chain to reset (because ERRb is commoned). If an FPAA sees an error during a reconfiguration update, it will also assert ERRb low. In this case, the ERRb pulse is pre-programmable to be short or long – if short then only a local reset will occur, if long then all FPAAs in the chain will be reset.

Having been primary configured each FPAA will have an 8bit address. Any reconfiguration update sent to the chain will be received only by the FPAA with the appropriate address. All other FPAAs in the chain will ignore this update data. However, it may happen that a reconfiguration will contain data that looks like the start of a reconfiguration update for another FPAA. To avoid this accidental addressing, the CFGFLGb pins of all FPAAs in the chain are commoned. Once an FPAA has recognised that an incoming reconfiguration is intended for it, it will drive CFGFLGb low to tell all other FPAAs in the chain to ignore the current reconfiguration data.

This is how addressing of multiple FPAAs works. FPAAs are connected in chains of up to 255, each with a unique 8bit address. For more FPAAs, multiple chains can be set up, each connected to a different select line of the SPI master.

3.2 Parallel Configuring

If it is required to send identical reconfiguration updates to all FPAAs in a chain, then the address 0xFF should be used. If it is required to send identical reconfiguration updates to multiple chains, then simply select all chains to be reconfigured and send the data.

If it is required to send identical primary configurations to all FPAAs in a chain, then the same primary configuration should simply be sent multiple times, once for each FPAA in the chain. If it will always be required for the FPAAs to contain the same primary configuration, then it is simpler to change the hardware set-up to configure the FPAAs in parallel (figure 8).

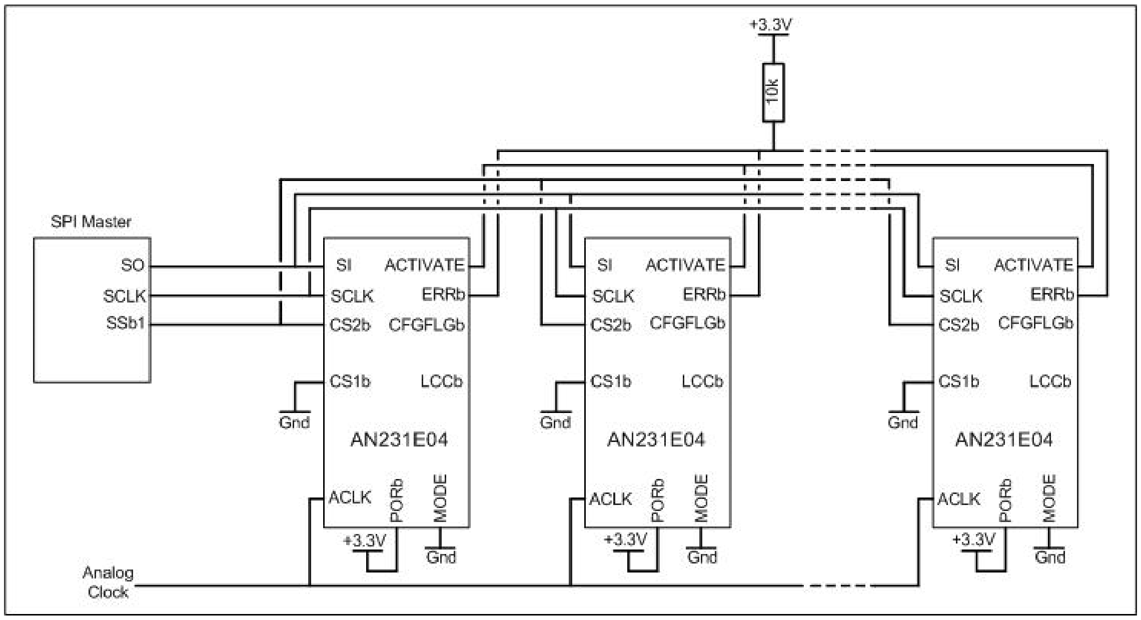

Figure 8: FPAAs configured in parallel

In this case, a single primary configuration can be sent to all FPAAs. One does not even need a select line, the CS2b pins could all be tied low, unless of course the SPI port is shared with other types of device in the system, in which case select lines may be required.

Note that primary configuring all FPAAs in parallel will mean that all FPAAs have the same address, which means that all FPAAs will respond to the same reconfiguration updates. The advantage of this set-up is that there is no theoretical limit to the number of FPAAs that can be configured in this way.

It is of course possible to use a combination of serial and parallel configuring. In figure 7, if the same select line was used for both chains, then both chains would be configured with the same circuits, even if the FPAAs within each chain had different circuits.

NOTES:

ACTIVATE: it has been described above how the ACTIVATE pins should be commoned to ensure that the circuits in the FPAAs start up together. It may be that it is not necessary for certain FPAAs or groups of FPAAs to start up together, in which case separate ACTIVATE lines should be used to reduce the loading. For example, in figure 7 it may be that all the circuits in a chain work together, but the chains are independent of each other. In this case it would be best to have separate ACTIVATE lines for each chain.

ERRb: a similar argument to above applies to ERRb. Different groups or domains of circuits should have separate ERRb lines. This will help reduce loading. It should also be noted that a lower value of pull-up resistor may be necessary for multiple FPAAs (see device manual).

4 Other Considerations

4.1 Clock Buffering

For large numbers of FPAAs it may be necessary to buffer clock lines, and possibly data lines too. Reasons for buffering clock lines are that heavy loading can cause slow edges. Also a long open ended clock line can have reflections on it leading to double edges on the clock. Both of these conditions can lead to multiple latching of the same data bits. In the past Anadigm has used the Texas Instruments CDCLVC11xx clock buffer which is both high speed and low skew. As an example, a CDCLCV1102 could be used in figure 7 to split the SCLK line between the 2 chains. Also the same with the ACLK line. If the chains are very long, then multiple buffers connected in a tree formation may be necessary.

4.2 Layout

A word needs to be said about layout which can become tricky with large numbers of FPAAs. The most important thing about layout is to keep analog and digital domains separate. Figure 9 shows a chain of 8 FPAAs laid out as 2 rows of 4.

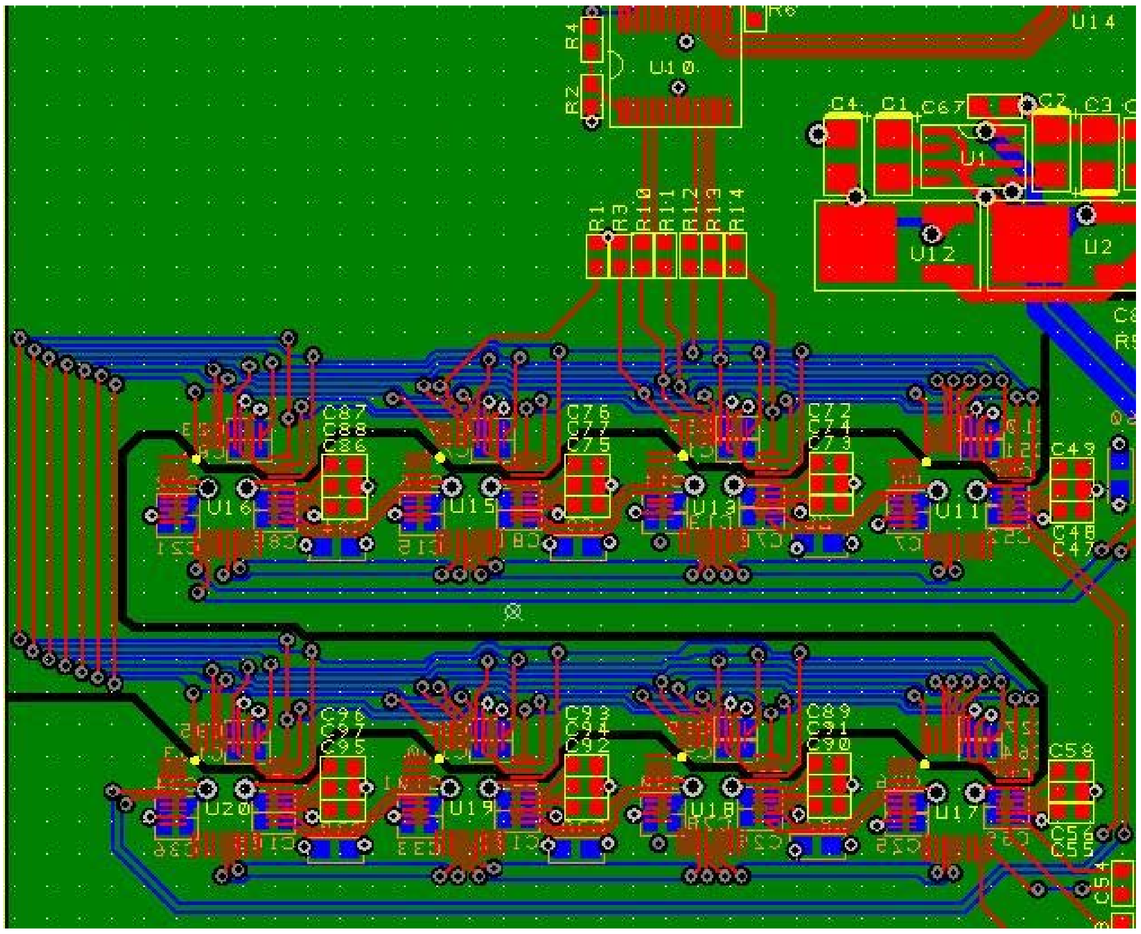

Figure 9: FPAA chain layout

The FPAAs are laid out with the digital pins on the top side and analog pins on the bottom side. These are routed together as a bus with the digital signals connected to the controller at the top, the analog signals taken out to the right. The green shows the ground plane which is split between analog and digital. Figure 9 shows how the digital signals have been kept over the digital ground section, and the analog signals kept over the analog ground. Note that in this design, the power plane was split in an identical way. The splits are normally run right across the board up to the point of entry of the power. Of course there should be plenty of decoupling, particularly in the analog domain.

It is also important to keep a separation between clock lines, and between clock and data lines, especially if they run over a long distance. Although it appears that the digital lines in figure 9 are all equally spaced, Anadigm likes to separate clock and data lines by alternating them with signals that are not critical in terms of cross-talk e.g. PORb, CFGFLGb, ACTIVATE, LCCb etc…