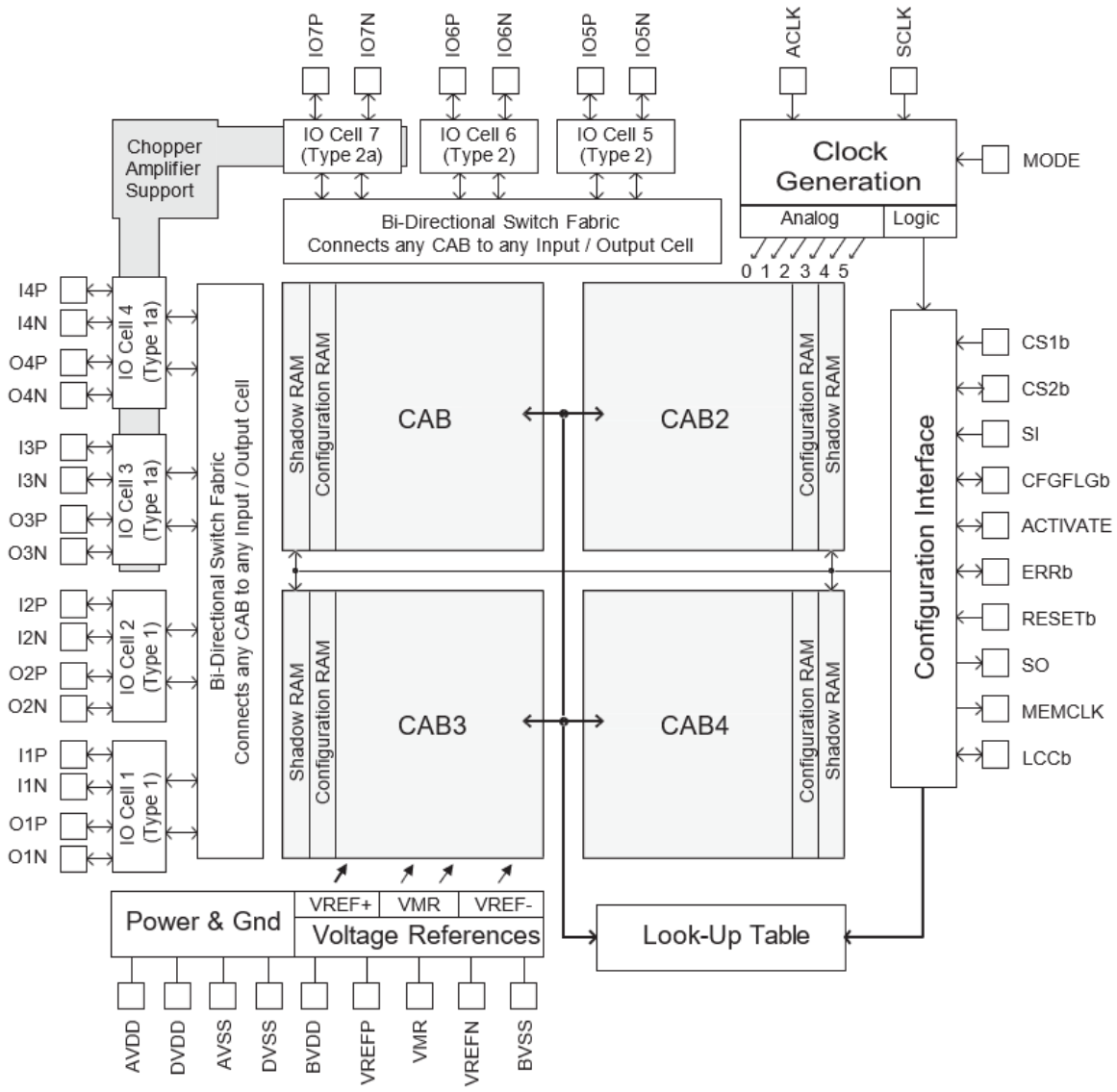
Okika Devices FlexAnalog™ technology, formerly known as Anadigm Apex dynamically programmable analog signal processors (dpASPs), represents the third generation of dynamically programmable field-programmable analog arrays (FPAA). The first product in the FlexAnalog™ family is the AN231E04, which has 7 analog I/O Cells and 4 Configurable Analog Blocks (CABs).

Compared to previous generations of FPAA products, FlexAnalog™ delivers improved analog performance and increased value. The configuration interface is enhanced to accommodate dynamic reconfiguration - a breakthrough capability that allows analog circuit functionality to be controlled by a companion host processor.



- Faster time-to-solution compared to discretes or ASICs
- High precision operation despite system degradation and aging
- Eliminates the need to source and maintain multiple inventories of product
- Ability to implement multiple chip configurations in a single device and to adapt functionality in the field

The FlexAnalog™ solution allows OEMs to deliver differentiated solutions faster and at lower overall system cost.

Legal Notice

Okika Devices reserves the right to make any changes without further notice to any products herein. Okika Devices makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Okika Devices assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Okika Devices does not in this manual convey any license under its patent rights nor the rights of others. Okika Devices software and associated products cannot be used except strictly in accordance with an Okika Devices software license. The terms of the appropriate Okika Devices software license shall prevail over the above terms to the extent of any inconsistency.

Table of Contents

Table of Figures

1. Architecture Overview

The FlexAnalog™ family of FPAAs includes the AN231 device. These FPAAs process analog signals in their IO Cells and Configurable Analog Blocks (CABs). These structures are constructed from a combination of conventional and switched capacitor circuit elements and are programmed from off-chip non-volatile memory or by a host processor. Programmable analog arrays enable adaptability and flexibility in analog circuits not previously possible.

The SRAM based AN231 is *dynamically reconfigurable*. The behavior of the FPAA can be modified partially or completely while operating. Dynamic Reconfiguration allows a companion host processor to send new configuration data to the FPAA while the old configuration is running. Once the new data load completes, the transfer to the new analog signal processing configuration happens in a single clock cycle. Dynamic Reconfiguration in the AN231 device allows the user to develop innovative analog systems that can be updated (fully or partially) on-the-fly, as often as needed.

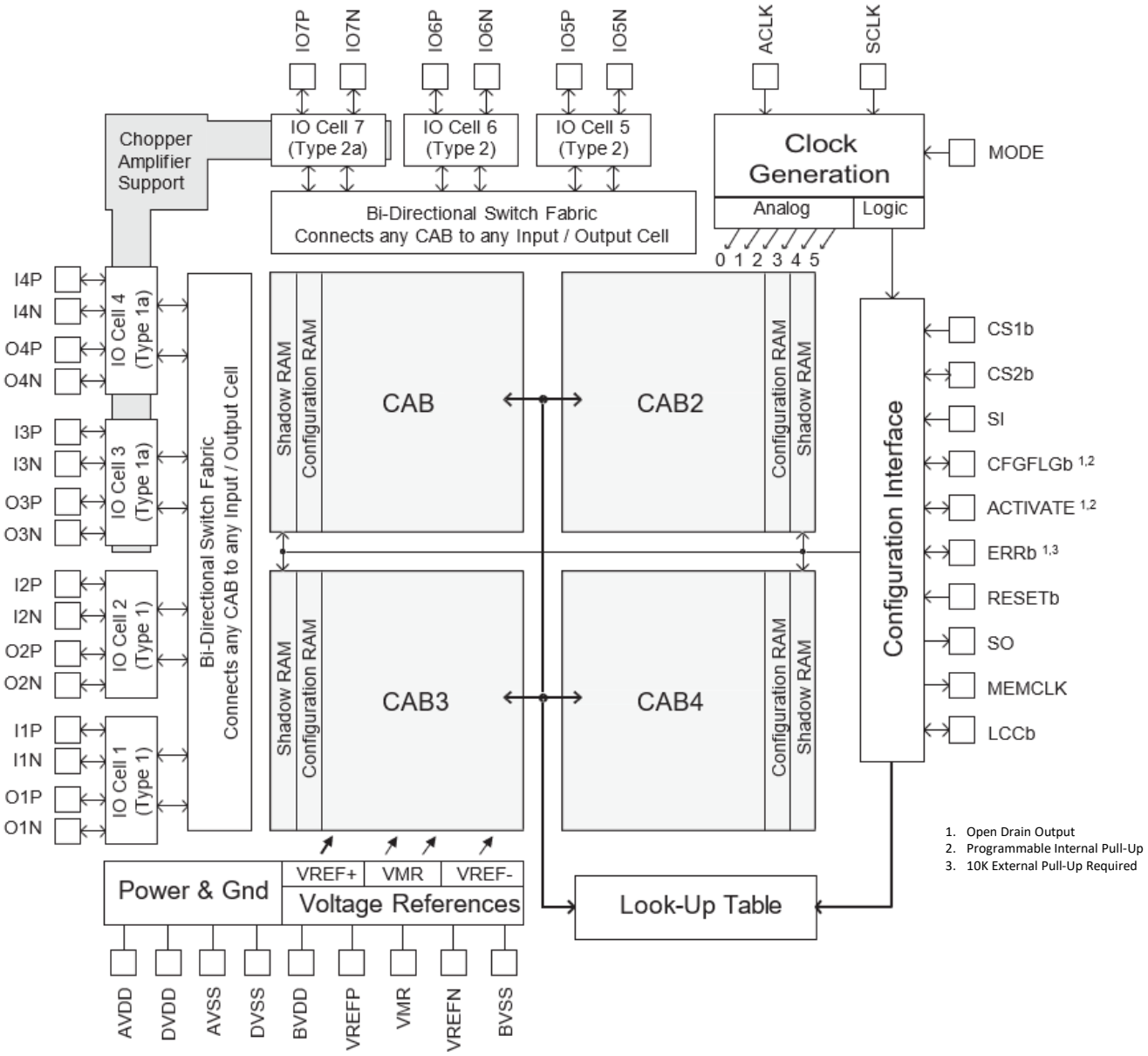


Figure 1 - Overview of the FlexAnalog™ dynamically programmable FPAA Architecture

AN231 FPAAs contain 4 Configurable Analog Blocks (CABs) in their cores. Most of the analog signal processing occurs within these CABs and is done with fully differential switched capacitor circuitry. The CABs share access to a single Look Up Table (LUT) which offers a method of adjusting any programmable AN231 element within the device in response to a signal or time base. The LUT can also be used to implement arbitrary input-to-output transfer functions (companding, sensor linearization), generate arbitrary signals, and construct voltage dependent filtering. A Voltage Reference Generator supplies reference voltages to each of the CABs within the device and has external pins for the connection of filtering capacitors.

Analog signals are routed in and out of the FPAA core via the available IO cells: two Type 1, two Type 1a, two Type 2, and one Type 2a. Type 1 and Type 1a IO cells contain both passive and active circuitry which allows direct signal input and output, building of active filters, sample and hold circuits, digital inputs, and digital outputs. The response of continuous time input and output filters is determined by a combination of internal programming and external components.

Type 2 and Type 2a IO cells are simpler and can implement direct input and output, reference voltage output, digital input, and digital output.

Any one of the Type 1a or Type 2a IO cells can have access to a specialized chopper amplifier resource which allows accurate amplification of very low energy inputs signals.

Look-Up Table (LUT) functionality facilitates the construction of arbitrary waveform generators and non-linear transfer functions. On-chip voltage reference generation precludes the need of any external reference voltage generation circuitry.

1.1. Using a dynamically programmable FPAA

The design of circuits for the FPAA is accomplished using AnadigmDesigner2. This software presents a graphical circuit design environment in which basic analog signal processing building blocks are dropped into place and wired together. Building blocks include: gain, filter, summing, rectification and many other more specialized behaviors. Specific parameters for each of the blocks used (e.g. Gain, Corner Frequency, etc.) are set by the user. AnadigmDesigner2 generates a configuration data file. The FPAA's configuration data file can be used to program a SPI PROM for stand-alone static operation, or compiled into a microcontroller's source program for hosted dynamic operation.

AnadigmDesigner2 also generates C source code for a host microcontroller which enables on-the-fly generation of new FPAA configuration data and subsequent dynamic reconfiguration. The FPAAs' signal processing behavior can be adjusted while your system remains continuously in mission mode.

The behavior of the analog circuitry is controlled by the contents of the FPAA's configuration memory. This memory is SRAM based and must be programmed after power-up. The FPAA's configuration interface provides a data port for this purpose. The configuration interface presents itself as a slave serial data port to a companion microprocessor, compatible with SPI signaling. The configuration interface can otherwise be configured to read data from an attached SPI PROM automatically after power-up or a device reset.

2. Typical Configuration Interface Connections

The behavior of the analog signal processing circuits within an AN231 device is dictated by the contents of its volatile (SRAM based) configuration memory. At power-on-reset, the FPAA clears its memory, placing the device in a benign condition. Once this power-on-reset sequence concludes, the device is ready to accept configuration data. The first configuration data set loaded into the device after a reset is called a Primary Configuration.

AN231 devices can be reconfigured (without intervening resets) using the Update format described later.

The configuration interface presents itself as either a serial data master or serial data slave. As a serial data master, the FPAA can automatically retrieve its configuration data set from any industry standard SPI EEPROM attached. As a serial data slave, the FPAA is compatible with SPI signaling from a host processor and can accept its configuration data from that host.

2.1. Dynamic Operation

The most powerful application scheme is when the FPAA is configured as a serial data slave. In this Dynamic Operation a companion host processor sends configuration data to the FPAA using SPI compatible signaling. This allows the creation of analog signal processing circuits that can be changed on-the-fly. The change may be as simple as a minor adjustment of a gain or corner frequency, or may involve wholesale transformation of behavior, say from a transmitter to a receiver configuration.

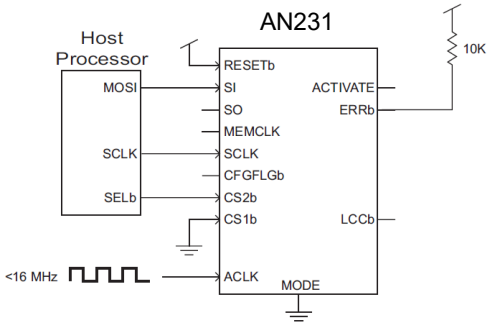


Figure 2 - Configuring a single FPAA from a Host Processor

Out of power-on-reset, the FPAA remains in a benign state, waiting for a configuration sequence. In order to configure the FPAA, the host processor drives CS2b low then streams a configuration data set out of its serial data port. Normally, the FPAA will enable analog signal processing automatically at the conclusion of configuration, though other options are discussed below. AN231 devices are highly flexible, allowing for on-the-fly reconfiguration any number of times after a reset.

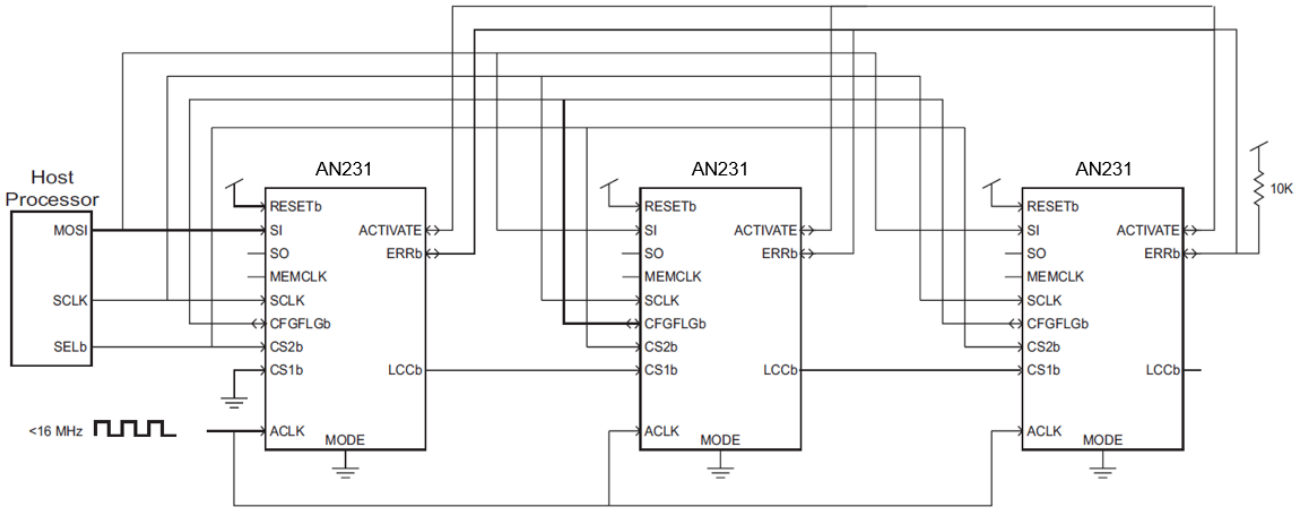


Figure 3 - Configuring multiple FPAAs from a Host Processor

Configuring several FPAAs from a Host Processor is a simple matter of busing the serial clock and data signals to each of the FPAAs. Figure 3, shows the parallel connection of the host's MOSI, SCLK and select signals to an arbitrary number of FPAAs.

The LCCb pin of the upstream device is tied to the CS1b pin of the downstream device, holding off down-stream configurations until the upstream device gets its configuration. This connection coupled with CFGFLGb pin behavior and logical addressing described later allows for conservation of host processor slave device select outputs. Note that the entire chain of FPAAs only requires a single select pin from the host.

ACTIVATE and ERRb nodes are tied together in order to facilitate the concurrent activation of analog circuitry and provide for configuration error handling. When connecting two or more FPAAs together as shown, it is important to bus together the CFGFLGb pins.The guidelines for the use of pull-up resistors on these nodes are given below in section 2.2.

Though not shown above, it is common practice to have the Host Processor monitor the ERRb node.

2.2. Static Operation

In static operation, the FPAA will automatically read in its configuration data from an SPI PROM after a manual reset or on power-up.

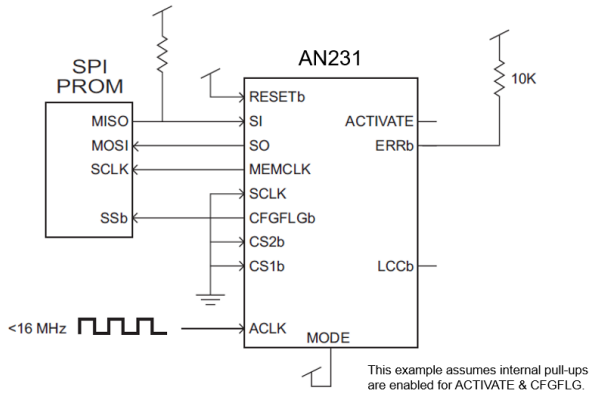


Figure 4 - A single FPAA self configuring from a SPI PROM

At the conclusion of the power-on-reset sequence, CFGFLGb will be low, selecting the attached SPI PROM.

The standard “read” command will be issued out of SO (clocked by MEMCLK). As MEMCLK continues, the SPI PROM responds with a serial data stream. This serial data stream is read by the SI pin.

Normally, the FPAA will enable analog signal processing automatically at the conclusion of configuration, though other options are discussed in more detail below.

In this simplest use model, the FPAA automatically: detects power-on, resets itself, reads in configuration data from a standard SPI PROM, and begins analog signal processing. A subsequent reset or power cycle will cause the sequence to repeat.

A slightly more advanced application of static configuration allows for the connection of several FPAA’s to a single SPI PROM. In this use scenario, the FPAAs are daisy chained; the *upstream* (closer to the PROM) device’s Local Configuration Complete (LCCb) pin feeds the *downstream* device’s CS1b enable pin. The SPI PROM’s MISO data output pin is bused to all FPAAs. All FPAA’s have their ACTIVATE and ERRb pins commoned.

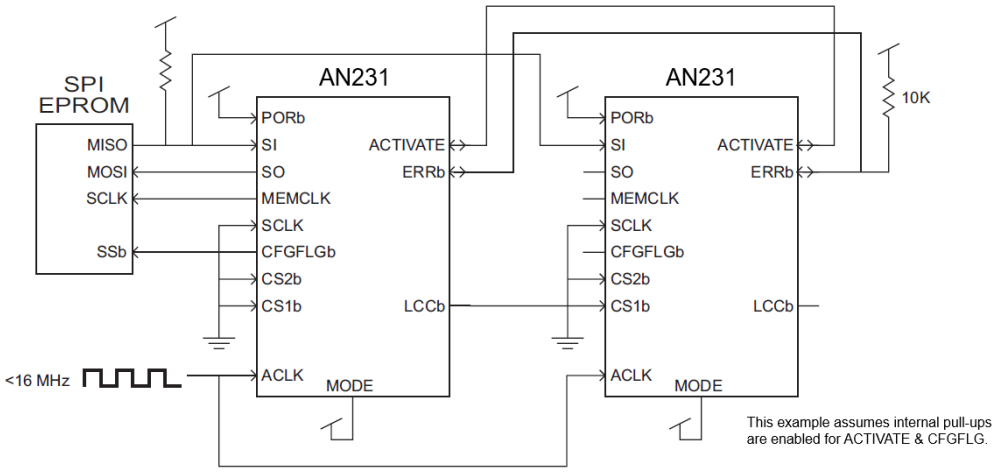


Figure 5 - Multiple FPAAs self configuring from a single SPI EPROM

As with the single FPAA example, the first FPAA in the chain still provides the “read” command to the SPI PROM but provides the necessary clocking for *all* the serial configuration data. As the configuration completes for an *upstream* device, its LCCb asserts low and enables the next device in the chain (*downstream)* to receive its data.

ACTIVATE is an open-drain bidirectional pin. During configuration the ACTIVATE pin is asserted low. Analog circuitry is not enabled (activated) until the ACTIVATE pin moves to a high state. As the local configuration completes, the FPAA de-asserts its ACTIVATE pin and monitors the ACTIVATE node. The daisy chained LCCb to CS1b sequence continues until all FPAAs in the chain have received configuration data. At that point all of the FPAAs will have de-asserted ACTIVATE and the commoned line will pull-high. Tying the ACTIVATE pin of all the devices in the configuration chain together ensures that analog signal processing does not begin in any of the FPAAs until all of them have received their configuration data.

ERRb is also an open-drain bidirectional pin. The ERRb pin will assert low if illegal or corrupted data is detected. Tying the ERRb pin of all the devices in the configuration chain together ensures that if any device detects an error, then all of the devices in the chain will reset (including the SPI PROM) and the configuration sequence will re-start automatically.

Pull-Ups on SI

Most SPI EPROMs hold their MISO pin in tri-state when the device is not selected. In order to ensure a valid logic signal is always presented to the FPAA a pull-up on the SI node is recommended.

Pull-Ups on ERRb

A 10K resistor is always required on the ERRb node.

Pull-Ups on ACTIVATE

The ACTIVATE pin has a programmable internal pull-up. In Master mode systems in which there are 3 or fewer FPAAs, it is recommended that the most upstream device enable its ACTIVATE pull-up. Master mode systems constructed with 4 or more FPAAs should use only an external pull-up on the ACTIVATE node.

Pull-Ups on CFGFLGb

The CFGFLGb pin has a programmable internal pull-up. The internal pull-ups for CFGFLGb and ACTIVATE are controlled with a single configuration bit; they are not separately programmable. Consequently, the same pull-up rules for ACTIVATE apply here to CFGFLGb.

3. Analog Architecture Details

3.1. Types 1 and 1a IO Cells

The device contains two Type 1 and two Type 1a IO cells. These IO sites provide tremendous flexibility in getting signals in and out of the CABs. The figure below summarizes the available options.

Bypass I/O
Differential Input
Differential Output
Digital I/O
Differential Input
Differential Output
Analog Input
Amplifier, or
Differential Low Offset Chopper Amplifier (type 1a IO cell only), or
Sample and Hold, with options for input:
Differential
Inverted Differential
Single Ended Positive
Single Ended Negative
Analog Output
Differential Amplifier
Differential Sample and Hold
VMR Output
Internal signal reference (1.5 V) presented on both pins.

Figure 6 - Types 1 and 1a IO Cell Options

Bypass

The Bypass setting of the IO cells provides direct, unbuffered access to CAB input and output ports. When using Bypass inputs, care should be taken to ensure that the differential signal and reference voltages are compatible with the CAB. Differential voltages should be maintained between 0 and 3 V, and centered about VMR (1.5 V).

Digital

Differential logic buffers are available to get signals into and out of the array. This IO configuration is most often used with Comparator and ADC-SAR CAMs.

Analog Input

The Amplifier's outputs are presented to the external pins for construction of anti-aliasing filters and external gain control.

The Sample and Hold setting provides a sampling circuit which can be used with either phase of the IO cell's clock. The Sample and Hold input can be configured as: Differential, Inverted Differential, Single Ended Positive, or Single Ended Negative.

On Type 1a IO cells, a Low Offset Chopper Amplifier setting is available. The Chopper Amplifier is spe- cially designed to provide very low offset voltages for weak external signals which must be gained up prior to processing in the CAMs. This setting includes programmable gain from 0, to 40 dB in 10 dB steps.

Only a single IO cell in a configuration is allowed access to the shared Chopper Amplifier resources.

Analog Output

Similar to Analog Input, Analog Output settings include: Amplifier, and Sample and Hold. Signalling from an Analog Output is always differential. The Amplifier setting of Analog Output is unique in that the IO cell presents its differential amplifier inputs and outputs to the device pins for external use only. There are no

signals from the core of the device involved. This setting is used for the creation of an output smoothing filter for a differential analog signal sourced by another IO site.

VMR Output

The VMR Output setting places the device's internal signal reference (1.5 V) on two pins of the IO site.

Amplifier Detail - Rauch Filter Designs

The Amplifier setting of the Type 1 and 1a IO cells accomodates construction of continuous time input anti-aliasing and output smoothing filters. Rauch (a.k.a. multiple feedback, MFB) differential filter con- struction is the recommended topology.

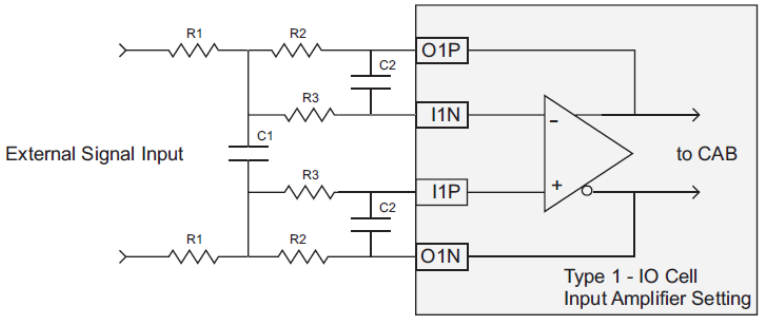


Figure 7 - Rauch Input Anti-Aliasing Filter

The same filter design technique can also be used in the construction of an output smoothing or recon- struction filter. In this case, the unfiltered output signal is sourced by a Bypass output and the amplifier used to construct the filter is provided by an adjacent IO cell using its output Amplifier setting.

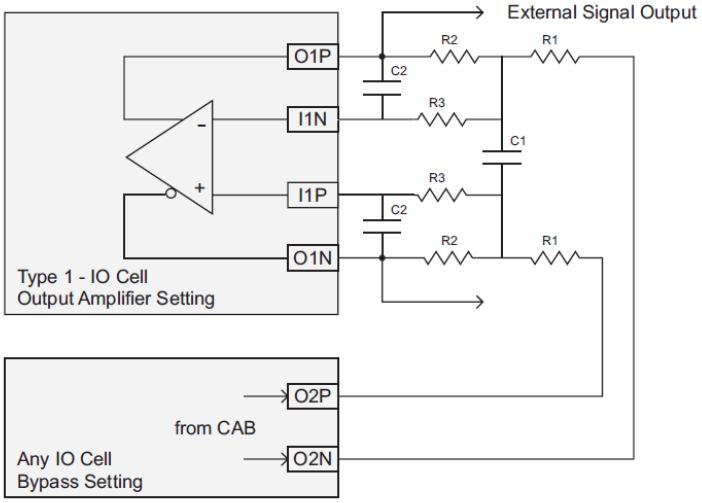


Figure 8 - Rauch Output Smoothing Filter

3.2. Types 2 and 2a IO Cells

The device contains two Type 2 and one Type 2a IO cells. These IO sites provide additional flexibility for getting signals in and out of the CABs. The figure below summarizes the available options.

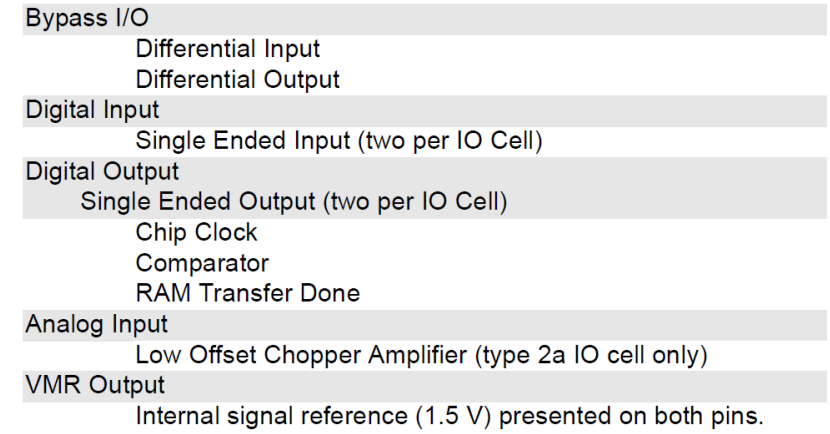


Figure 9 - Types 2 and 2a IO Cell Options

Digital Input

Two independent single ended logic control signals can be routed into the CAMs.

Digital Output

The Type 2 and 2a IO cells can be configured to provide two single-ended digital outputs. The outputs can reflect: any of the 6 internal clocks, a comparator or ADC-SAR output or a signal indicating the completion of a transfer from Shadow SRAM to Configuration SRAM. The polarity of these output signals is programmable.

Analog Input

Like the Type 1a IO cell, the Type 2a IO cell also offers a Low Offset Chopper Amplifier. The 2a Chopper Amplifier has programmable gain ranging from 0 up to 60 dB in 10 dB steps.

3.3. CAB

Most analog signal processing occurs in the Configurable Analog Block (CAB). Signal processing is accomplished using an architecture based on switched capacitor circuit design. Every CAB contains two op-amps, a comparator, banks of programmable capacitors, and a collection of configurable routing and clock resources. With switched capacitor signal processing, the absolute value of the components integrated into the chip is not important, but rather it is the *ratio* of the programmable capacitors employed and the clock frequencies that determine circuit response; both of which are well controlled. In order to further improve signal fidelity, all signal processing within the CABs is fully differential.

4. Configuration Interface Details

4.1. Pin Descriptions

MODE

The state of the MODE pin is read as part of the FPAA's power-on-reset sequence. If MODE is low out of reset the configuration interface establishes itself as a serial data slave. If MODE is high, the config- uration interface establishes itself as a serial data master. MODE should either be tied high to VDD or low to VSS. This is a static pin. Changing its state after power up is not allowed.

For more details on the effects of MODE, see section 4.2.4.

SCLK (Serial Clock)

If MODE is low, SCLK serves as the serial data clock input. The configuration state machine is driven from this input. SCLK may not exceed 40 MHz. SCLK may be free running or discontinuous.

If MODE is high, then SCLK is ignored and the configuration state machine is driven from an internally divided down ACLK (ACLK/16).

MEMCLK (Memory Clock)

If MODE is high, the FPAA establishes itself as a serial data master. MEMCLK serves as the serial data clock output.

Once configuration is complete, the MEMCLK pin may be used as a user programmable digital output.

ACLK (Analog Clock)

ACLK is the input for the analog clock source. All internal switched capacitor clocks are derived from the ACLK input. ACLK may not exceed 40 MHz.

If MODE is high, then SCLK is ignored and the configuration state machine is driven from an internally divided down ACLK (ACLK/16).

SI (Serial In)

The SI pin always serves as the configuration data input pin.

SO (Serial Out)

When MODE is high, the SO pin issues the "read" command to the attached SPI PROM.

LCCb (Local Configuration Complete)

LCCb (Local Configuration Complete) is high during power-on-reset and remains high until the local configuration completes, it then goes low.

In multi-FPAA systems, LCCb is normally connected to the CS1b input of the next FPAA down- stream.

Once configuration is complete, the LCCb pin may be used as a user programmable digital output. LCCB is active low.

CS1b (Chip Select 1)

The CS1b pin also serves as a chip select input, but its behavior is more involved than CS2b. In mul- tiple FPAA systems, the CS1b input is normally driven by the upstream device's LCCb pin. Out of reset, the LCCb pin of the upstream device will be high - suspending the configuration of the down- stream device. Once the upstream device completes its local configuration and drives its LCCb low, the downstream device will begin its configuration. CS1b is active low.

CS2b (Chip Select 2)

The CS2b pin serves as a regular chip select input. CS2b is active low.

CFGFLGb (Configuration Flag)
When connecting multiple FPAAs up as suggested in Figure 15, it is necessary to tie the CFGFLGb pins of the devices to a common node. The CFGFLGb node will be driven low whenever a device is being addressed for configuration or reconfiguration (AN23x devices only) and will pull-high when data transfer is complete. When the local device senses CFGFLGb as being pulled low (by another device), the local device ignores the data on SI. This CFGFLGb signaling prevents reconfiguration data intended for one device from being incorrectly intercepted by another.

There is a single configuration data bit which enables the CFGFLGb and ACTIVATE internal pull-up resistors.

ERRb (Error)
ERRb is an open-drain, bidirectional pin. The pin asserts low whenever a configuration error is detected.

In multi-FPAA systems, the ERRb pins should all be commoned. When the local device senses ERRb as being pulled low (by another device), the local device resets. Also, the ERRb signal is an enabling term in the power-on-reset cycle. Commoning the ERRb pins in a multi-FPAA system allows all the devices to complete their POR sequences concurrently (though not all device types complete their power-on-reset cycles in the same amount of time). More complete details on the behaviors associated with this pin are deferred to section 4.2.2.
ERRb must be pulled high with an external pull-up resistor. 10 K is the usual value, but may be less if the node is heavily loaded. ERRb is active low.

ACTIVATE
ACTIVATE is an open-drain, bidirectional pin. In a single FPAA system, ACTIVATE can be left floating. During a primary configuration, the ACTIVATE pin is asserted low. When the primary configuration is complete, ACTIVATE is release and monitored. When ACTIVATE is sensed as high, analog processing circuits are *activated*.

In multi-FPAA systems, the ACTIVATE pins are all commoned. When the local device completes its configuration, it quits driving ACTIVATE low and then monitors the state ACTIVATE line. Once all devices have completed their configuration, the ACTIVATE node will finally go high, allowing all devices to activate analog signal processing concurrently. More complete details on the behaviors associated with the ACTIVATE function are deferred to section 4.3.2.

There is a single configuration data bit which enables the CFGFLGb and ACTIVATE internal pull-up resistors.

RESETb
RESETb is an active low input. Normally the RESETb is tied high; internal power-on-reset circuitry senses brown-out or power up conditions and automatically resets the device. If the application dictates a manual reset capability, the pin may be driven low then high to re-initiate a complete power-on reset sequence.

4.1.1. Secondary Pin Functions

The LCCb and MEMCLK configuration pins can be converted to the same set of programmable polarity singled ended digital outputs available in a Type 2 (and 2a) IO cell, namely: Chip Clock, Comparator, AutoNull/ Osc, or RAM Transfer Done

4.2. Special Functions - Configuration and Control Features

4.2.1. Resets

There are two classes of reset in the device. A Primary Reset brings the configuration logic into a safe starting condition and clears all Shadow and Configuration SRAM with the exception of the LUT. A Secondary Reset only brings the configuration logic into a reset condition; all SRAM contents are left undisturbed.

During power-up, power-on-reset circuitry initiates a Primary Reset. This same circuitry initiates a Primary Reset whenever brown out conditions occur or the external RESETb pin is driven low. Holding the RESETb pin low keeps the device in reset until released.

A Secondary Reset only resets the configuration logic; no SRAM contents are affected. The sources of Secondary Resets are: Software Reset (see section 5.1.3 for further details) and a long ERRb pulse (see section 4.2.2 for further details).

4.2.2. ERRb

The ERRb pin serves several different functions.

As an output, a low asserted ERRb indicates a configuration error. If a configuration data error is detected during a Primary Configuration, ERRb will assert low for 19 configuration clock cycles. (19 SCLK cycles in Slave mode or 240 ACLK cycles in Master mode. Recall that ACLK/16 drives the configuration logic in Master mode.) The ERRb response to a configuration data error during a reconfiguration (AN23x devices only) is programmable. On error detection, the low assertion of ERRb may be for either 5 or 19 configuration data clocks. A “short” ERRb pulse indicates that the local device detected an error and that a reset occurred. A “long” ERRb pulse will cause all devices tied to the common ERRb node to reset.

As an input, holding ERRb low for 19 (or more) configuration clock cycles will cause a Secondary Reset. (Pulsing RESETb low is the recommended method for manually asserting a reset.)

ERRb is also a controlling term which holds off the completion of a power-on-reset sequence. Early in the power-on-reset sequence, ERRb is asserted low. Late in the power-on-reset sequence, ERRb is released and monitored. When ERRb reaches a valid logic high state, the power-on-reset sequence concludes. Since not all FPAA device types have the same length power-on-reset sequence, commoning the ERRb pins of all of the FPAAs within a system ensures that all devices will remain in reset until the slowest device finishes, at which point all the devices will come out of reset concurrently. (See section 4.2.1 for further details.)

4.2.3. Watchdog

The device contains an automatic power savings feature. When enabled, this Watchdog circuit monitors the frequency of the primary analog clock (ACLK pin). When ACLK falls below 31.25 KHz, the device will automatically shift into a powered-down condition. Resumption of ACLK will immediately bring the part back up into a normal powered state, though how fast normal signal processing resumes is a function of the current CAB configurations.

4.2.4. Analog and Configuration Clock Generation

All signal processing clocks within the device are derived from the analog master clock signal presented on the ACLK pin. The ACLK signal gets split and divided down into two system base clocks (SYS1 and SYS2), the divisor being between 1 and 510. These two system clocks are further divided down into 6 additional clock domains: Clock 0 through Clock 5. Each of these 6 analog processing clocks can use either SYS1 or SYS2 as its base, and will further divide that base clock down by 1 up to 510. Clock 5 and Clock 6 have an arbitrary phase delay setting which ranges from 0° to 360°.

Having two base clocks allows for creation of two unrelated analog signal processing circuits within a single device.

Using two unrelated base clocks can also be a hazard. If using two clock domains, be sure that all analog signals stay completely within their own domains. It is unlikely that an analog signal moving from one switched capacitor clock domain to another can do so without corruption

Clock frequency is a fundamental parameter in the function of switched capacitor analog circuitry. If clock frequencies are changed later in the course of the design, care must be taken to ensure that the CAMs with frequency parameters placed in the design are still operating as desired. For example, changing the frequency of the clock driving a filter CAM will change the frequency response of that filter.

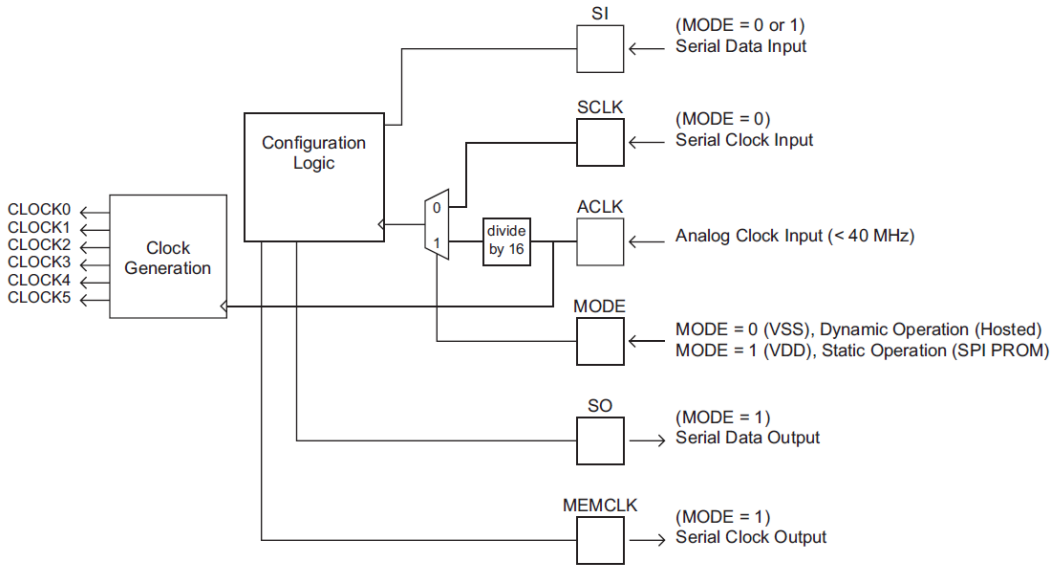


Figure 10 - MODE Controls the Behavior of the Configuration Interface

4.3. SRAM

There are three regions of volatile SRAM within the device. The first, Shadow SRAM, is the memory that gets written to during configuration or reconfiguration. Shadow SRAM serves as a temporary holding area for configuration data prior to its transfer into Configuration SRAM. This second region, Configuration SRAM, controls the behavior of the analog signal processing circuitry. The transfer from Shadow SRAM to Configuration SRAM happens in a single clock cycle, minimizing disturbance to the analog signal paths. The third region of memory is the Look-Up Table.

4.3.1. LUT

The device contains a Look-Up Table (LUT) memory. The LUT provides replacement values for Configuration SRAM locations. A CAB plus LUT combination can be used to create non-linear functions such as arbitrary waveform synthesis and table based sensor linearization functions.

4.3.2. Auxiliary Cell

Recall that the device contains both Shadow SRAM and Configuration SRAM. Configuration SRAM controls the behavior of the analog signal processing elements. Data written into the device, loads into Shadow SRAM and remains there until an internal signal enables the transfer to Configuration SRAM.

At the end of a Primary Configuration, the transfer takes place only when the Activate pin pulls high.

Once a Primary Configuration has completed, an AN23x device can accept subsequent reconfiguration data into its Shadow SRAM (see section 5.1.4 for further details on the Update protocol). The timing of the transfer of reconfiguration data from Shadow SRAM to Configuration SRAM can be tightly controlled using one of several methods made available via the RAM Transfer Cell:

Immediate
The Immediate setting of the RAM Transfer Cell forces the reconfiguration data to transfer into the Configuration SRAM as soon as the external write of the reconfiguration data completes.

Event Driven
The Event Driven setting of the RAM Transfer Cell allows the use of either an internal or external digital control signal to cause the transfer of data to occur. Internal event signals are typically sourced by a comparator output.

Event Driven and Armed
An arming term is available to hold off the Event Driven trigger of data transfer. When using the arming function, either the Trigger or Arm terms may be sourced externally, but not both.

Clock Synchronization
The Clock Synchronization setting forces the SRAM transfer to occur only at the point at which all chip clocks have a simultaneous rising edge. This ensures that clock synchronization is maintained when making changes to clock dividers, but will cause a delay between the end of the reconfiguration bit- stream and the transfer.

> *ACLK can be divided by up to 510for system clocks (SYS1 andSYS2). These CAB clocks can be divided further by up to another 510times. While extremely unlikely, such a scenario may result in a very long delay between the end of the reconfiguration bitstream and the SRAM transfer.*

5. Dynamic Operation Details

The most powerful use model of Anadigm FPAAs includes the use of a companion host processor. The FPAA is used with MODE tied low and consequently the configuration interface presents itself as a SPI compatible Slave. The host processor can be used to simply manage the configuration tasks, downloading data to the FPAAs from its SPI Master port. Hosted configuration is available with AN231 devices. The real potential of programmable analog, however, is best leveraged when the host processor is also used to generate and download new configuration data sets on-the-fly as analog signal processing requirements change. This dynamic reconfiguration is only available on AN23x devices. Please refer to AnadigmDesigner2 documentation to get more information on “C Code Generation” and this powerful application approach. This user manual focuses only on the physical and protocol requirements of the interface between a host processor and the FPAA.

5.1. Configuration Data Stream Protocol

Regardless of whether the FPAA is being applied in either the Master or Slave modes, the serial configuration data stream must adhere to the protocols defined in this section. AnadigmDesigner2 constructs a configuration data file compliant to this protocol so that even for the simplest case of self-booting from a slaved SPI PROM, all of the requisite information is contained in the data stream delivered to the device.

In dynamic applications, the host processor must generate the appropriate configuration data and transfer that data to the device using the protocols defined herein.

There are two data formats which comprise the configuration protocol: Primary Configuration format and Update format. Each is explained in detail in the following sections.

5.1.1. Primary Configuration Format

Primary Configuration is the format of the data that is generated by AnadigmDesigner2 and is the format that must be used exactly once to configure the device for the first time after power-on-reset. Out of reset, all Shadow SRAM locations are reset to “zeros”. A Primary Configuration is therefore only required to send data to Shadow SRAM locations requiring “ones”. The LUT SRAM is not expressly reset to zero. The Primary Configuration is therefore also required to initialize the LUT SRAM if the LUT is intended to be used.

Primary Configuration data sets presume the device has been reset (Shadow SRAM zeroed out). In order to make configuration as efficient as possible, Primary Configuration data sets only contain data where one’s need to be programmed.

The Primary Configuration format is comprised of a Header Block followed by one or more Data Blocks. A header block contains a Sync byte, Device ID, ADDR1 and Configuration Control bytes. A Data Block contains data destination address information, a configuration data byte count and from 1 to 256 configuration data bytes, followed by a single data block terminator byte or a two byte CRC 16 check word. Data must be shifted into the FPAA most significant bit first.

	Data	Byte Name	Description
Header Block	11010101 D5	SYNC	Synchronization byte, always D5
	10110111 B7	Device ID [31:24]	Device ID for AN231E04 Device ID is 0xB7200100
	00100000 20	Device ID [23:16]	
	00000001 01	Device ID [15:8]	
	00000000 00	Device ID [7:0]	
	XXXXXXXX	ADDR1	ADDR1 Byte, Primary Logical Address for the FPAA
	XXXXXXXX	CONTROL	Configuration Control Byte
Data Block (first)	11XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 1)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address + 1
	Remaining data bytes go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	XXXXXXXX or 00101010 2A	CRC_MSB or Data Block End	(depending on Bit 5 of BYTE ADDRESS) Most significant byte of CRC16 error code or Data Block End constant of 0x2A
	XXXXXXXX	CRC_LSB	Least significant byte of CRC16 error code (if used)
Remaining data blocks go in this region...			
Data Block (last)	10XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 0)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address + 1
	Remaining data bytes go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	XXXXXXXX or 00101010 2A	CRC_MSB or Data Block End	(depending on Bit 5 of BYTE ADDRESS) Most significant byte of CRC16 error code or Data Block End constant of 0x2A
	XXXXXXXX	CRC_LSB	Least significant byte of CRC16 error code (if used)

Figure 11 - Primary Configuration Data Stream Structure

5.1.2. Header Block

SYNC BYTE

The configuration logic always expects a synchronization header. For the Primary Configuration and Update formats, this sync header is always 11010101 (0xD5).

Device ID BYTE

Every Anadigm device type has a unique 32 bit Device ID. Requiring the Device ID to match during Primary Configuration is a way of ensuring that configuration data intended for another device type does not get accidentally loaded. If a Primary Configuration is attempted in which the Device ID is not as expected, the device will assert ERRb and no data will be loaded into the array. Incorrect data can cause high stress conditions to exist within the device, possibly causing damage.

Family Member	32 bit Device ID
AN231E04	0xB7200100

Figure 12 - Device IDs for the FlexAnalog™ Family

ADDR1 BYTE

The ADDR1 field establishes one of the two logical addresses for the device. ADDR1 is considered the primary logical address for the device and is loaded during a Primary Configuration. The alternate logical address (ADDR2) is not part of the Header Block, but rather it is established within the device's configuration data and is therefor delivered within a Data Block. Having logical addresses for every FPAA in the system allows the connection of many FPAAs in series (consuming no extra physical host connections) and once configured, communication only with the specifically addressed device(s). See section 5.2 for further details.

5.1.3. Data Block

CONTROL BYTE

Bit Number									
7	6	5	4	3	2	1	0		
								WRITE	**000:** WRITEDATA **001:** WRITEDATA_CLK **010:** (unused) **011:** (unused) **100:** (factory reserved) **101:** (factory reserved) **110:** (factory reserved) **111:** SRESET
								(0 constant)	**0:** Must be set to "0".
								(0 constant)	**0:** Must be set to "0".
								(1 constant)	**1:** Must be set to "1".
								PULLUPS	**1:** Enable internal pull-ups. **0:** Disable internal pull-ups. This bit is used to enable internal pull-ups on the CFGFLGb and ACTIVATE pins. PULLUPS is sticky, i.e.
								ENDEXECUTE	**1:** At the end of the data current transfer cycle, Shadow SRAM will be copied into Configuration SRAM. **0:** No action.

Write

WRITEDATA - This command instructs the configuration logic to handle the incoming data block as configuration data. Clock divisor settings in the data stream are ignored.

WRITEDATA_CLK - This command instructs the configuration logic to handle the incoming data block as configuration data. Clock divisor settings in the data stream are applied.

SRESET - This command causes a chip reset. Please see section 4.2.1 for further detail.

RESET_ALL

Please refer to section 4.2.1 for further detail.

PULLUPS

This bit is used to select internal pull-ups on the CFGFLGb and ACTIVATE external pins. Note that PULLUPS is sticky; once set it stays set until a reset. This is done to avoid having to track in software which device has PULLUPS set in a multi-FPAA system.

Only one device in a multi-FPAA system should have PULLUPS set. If a large number of FPAAs are being used and loading becomes an issue, external pull-up resistors on the CFGFLGb and ACTIVATE nets are appropriate in lieu of the internal pull-ups.

ENDEXECUTE

Please refer to section 4.3.2 for further detail.

BYTE ADDRESS BYTE

Bit Number								
7	6	5	4	3	2	1	0	
			BYTE ADDRESS					Starting byte address within a bank of Shadow SRAM.
		ENABLE CRC CHECK						**1:** CCITT-CRC16 error checking is enabled **0:** DATA BLOCK END checking is enabled
	DATA_FOLLOWS							**1:** A subsequent Data Block will be expected by the configuration logic. **0:** This block is presumed to be the final block of configuration data.
CONSTANT 1								**1:** Must always be set to "1". **0:** Undefined operation.

BANK ADDRESS BYTE

Bit Number								
7	6	5	4	3	2	1	0	
BANK ADDRESS								Starting bank address of Shadow SRAM.

The BYTE and BANK Address bytes taken together form the starting Shadow SRAM (or LUT SRAM) load address for the subsequent block of configuration data. (These bytes are also used to establish the starting read address for read commands.) The memory within the device is organized as 19 rows (banks) by 32 columns (bytes). No special handling is required to cross bank or byte boundaries, this is automatic. The address allocation of the device's Shadow and LUT SRAM is shown in the figure below..

BANK ADDRESS	BYTE ADDRESS 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 13 12 11 10 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
00	Auxiliary RAM 0 Clock Dividers and Power
01	Auxiliary RAM 1 LUT and I/O Control
02	Auxiliary RAM 2 I/O and Routing
03	CAB 1 Shadow SRAM Bank A
04	CAB 1 Shadow SRAM Bank B
05	CAB 2 Shadow SRAM Bank A
06	CAB 2 Shadow SRAM Bank B
07	CAB 3 Shadow SRAM Bank A
08	CAB 3 Shadow SRAM Bank B
09	CAB 4 Shadow SRAM Bank A
0A	CAB 4 Shadow SRAM Bank B
0B-0F	(Factory Reserved Address Range)
10	Look Up Table SRAM Bank 0
11	Look Up Table SRAM Bank 1
12	Look Up Table SRAM Bank 2
13	Look Up Table SRAM Bank 3
14	Look Up Table SRAM Bank 4
15	Look Up Table SRAM Bank 5
16	Look Up Table SRAM Bank 6
17	Look Up Table SRAM Bank 7
18-FF	(Factory Reserved Address Range)

Figure 13 - AN231 Memory Allocation

DATA COUNT BYTE

Setting this field to a value of 0x00 signifies that 256 data bytes follow in this data block. Setting this field to any integer value between 1 and 255 signifies that exactly that many data bytes follow. This byte count only represents the number of configuration data bytes that follow (data bytes destined for the Shadow SRAM or LUT SRAM); the count does not include the Data Block End or CRC bytes.

There are no special requirements for data writes that traverse bank boundaries. Crossing bank boundaries is handled automatically within the device.

DATA BYTE

Configuration data bytes. This is the data that gets loaded into the Shadow SRAM or LUT SRAM, starting at the address defined in BYTE and BANK address bytes defined just above. There may 1 up to 256 data bytes per block.

DATA BLOCK END BYTE

If bit 5 of BYTE ADDRESS is 0, the only byte expected after the DATA bytes is the DATA BLOCK END constant 0x2A. If any other value is read in, ERRb will assert and the configuration process will be aborted.

CRC_MSB and CRC_LSB BYTEs

If bit 5 of BYTE ADDRESS is 1, CRC_MSB byte followed by a CRC_LSB byte is expected at the conclusion of end of each data block. The 16 bit CRC is calculated using CCITT method with the polynomial:

x^16 + x^12 + x^5 + 1

> *Calculation of a CRC is a compute intensive chore for a host processor and is not often used. A pre-calculated look-up table based approach can be considerably faster but requires significant storage for all of the possible results*

5.1.4. Update Format

Once a Primary Configuration has been completed, there is no longer any requirement for the host to transmit Device ID information along the communication path. (A configuration data stream doing so would be considered an error, and devices would assert ERRb.) The host now only needs to send out a sync header and a valid logical address for the target device (or target devices). The remainder of the information is just as described above in section 5.1.1.

As with a Primary Configuration, it is not a requirement of the Update format to contain a complete data set for the device. Partial reconfiguration of the device is supported. It is most often the case that only a few Shadow SRAM or LUT SRAM addresses need new data. The Update format provides an efficient method for moving this new data into the device.

	Data	Byte Name	Description
Header Block	11010101 D5	SYNC	Synchronization byte, always D5
	XXXXXXXX	LOGICAL ADDRESS	ADDR1, ADDR2, or 0xFF - Logical address of the target device(s).
	XXXXXXXX	CONTROL	Configuration Control Byte
Data Block (first)	11XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 1)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address + 1
	Remaining data bytes (if any) go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	XXXXXXXX or 00101010 2A	CRC_MSB or Data Block End	(depending on Bit 5 of BYTE ADDRESS) Most significant byte of CRC16 error code or Data Block End constant of 0x2A
	XXXXXXXX	CRC_LSB	Least significant byte of CRC16 error code (if used)
Remaining data blocks (if any) go in this region...			
Data Block (last)	10XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 0)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address +1
	Remaining data bytes (if any) go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	XXXXXXXX or 00101010 2A	CRC_MSB or Data Block End	(depending on Bit 5 of BYTE ADDRESS) Most significant byte of CRC16 error code or Data Block End constant of 0x2A
	XXXXXXXX	CRC_LSB	Least significant byte of CRC16 error code (if used)

Figure 14 - Update Data Stream Structure

5.2. Configuration Examples

The following examples assume a microcontroller hosted interface. Data must be shifted into the FPAA most significant bit first. White space and comments are included only to improve readability for these examples.

Primary Configuration Format Example

```
00000000    //40 clocks are required to complete the internal power-up
00000000    //reset sequence. This is usually accomplished by sending out 5
00000000    //bytes of NULL prefix data. After the 40th clock,
00000000    //the configuration logic is ready and waiting for a sync.
00000000

11010101    //0xD5 is the required sync header.
10110111    //0xB7 is Most Significant Byte of Device ID word.
00100000    //0x20 is byte 3 of Device ID word.
00000001    //0x01 is byte 2 of Device ID word.
00000000    //0x00 is Least Significant Byte of Device ID word.

00000001    //Any user assigned ADDR1(Primary Logical Address), besides 0xFF.
11000001    //Control Byte
            //ENDEXECUTE is enabled.
            //PULLUPS are enabled.
            //WRITEDATA_CLK is the command.

11000000    //Constant 1, DATA_FOLLOWS, starting BYTE address is 0
00000000    //The starting BANK address is 0

00000000    //0x00 byte count field means 256 data bytes follow.
datadata    //The first configuration data byte.
datadata    //The second configuration data byte.
...
datadata    //the 256th configuration data byte.

00101010    //0x2A is the Data Block End constant expected.
            //This is the region that the other blocks of data
            //need to be sent to completely fill the Shadow SRAM.
            //These blocks do not need to be prefaced by additional
            //clocks, nor do they require a Device ID, ADDR1
            //or Control bytes. These intermediate blocks all have
            //the same form as the final block shown below. The
            //important point to note is that only the final
            //block of Primary Configuration has the DATA_FOLLOWS
            //bit cleared in the BYTE ADDRESS byte.

10011110    //DATA_FOLLOWS is cleared to 0, this means that at
            //the conclusion of the transfer of this final block,
            //Shadow SRAM will get copied into Configuration SRAM,
            //with no additional action required by the host.
            //0x1E is the example starting BYTE address.

00010111    //0x17 is the example starting BANK address.

00000010    //0x02 is byte count for this particular last block.

datadata    //Second to the last configuration data byte.
datadata    //The Last configuration data byte.

00101010    //0x2A is the Data Block End constant expected.
00000000    //8 clocks are required by the configuration state
            //machine to finish the transfer. This is usually
            //accomplished by sending out a single byte of NULL data.
```

Update Format Example

```
11010101      //0xD5 is the required sync header.

00000001      //LOGICAL ADDRESS - The ADDR1 or ADDR2 value of the target device
              //or the universal logical address, 0xFF.

11000000      //Control Byte
              //ENDEXECUTE is enabled.
              //PULLUPS are enabled.
              //WRITEDATA is the command.

10011110      //DATA_FOLLOWS is cleared to 0, so the configuration
              //logic will expect no more data after this final block
              //and because ENDEXECUTE is set = 1 in the Control Byte
              //Shadow SRAM will get copied into Configuration SRAM
              //as soon as the data block is download with no
              //additional action required by the host.
              //0x1E (decimal 30)is the starting BYTE address.

00000011      //0x03 is the starting BANK address.

00000011      //0x03 byte count field means 3 data bytes follow.

datadata      //The 1st updated data byte goes to bank 3 byte 300
datadata      //The 2nd updated data byte goes to bank 3 byte 31
datadata      //The 3rd updated data byte goes to bank 4 byte 0

00101010      //Data Block End constant 0x2A is expected.

00000000      //8 “don’t care” NULL bits to provide the necessary clocks
              //to complete the load. Because the EXECUTE bit was
              //set on this block’s control byte, the immediate
              //transfer from Shadow SRAM to the Configuration SRAM
              //will occur here.
```

The 8 clocks at the end of each of these configuration examples are necessary only to complete the transfer at that time. If it is not critical to complete the transfer at that particular moment, then the clocking associated with any subsequent block will be sufficient to complete the transfer. With no clocks, the configuration state machine simply freezes in place. There are no “unsafe” states.

5.3. Advanced Feature - Logical Addressing

It is possible to uniquely address every device in a hosted multi-FPAA system physically, addressing each one with a unique connection to its CS2b input. This however this consumes chip select output pins on the host processor and complicates host software design. Instead, Anadigm FPAAs include features which allow all FPAAs to reside on a single SPI bus, share a single common chip select and respond to logical addresses in the data stream. When designing with AnadigmDesigner, there are data entry fields associated with each chip instance used to establish the ADDR1 (the primary logical address) and ADDR2 (the alternate logical address).

The primary logical address of a device is established in the ADDR1 field of a Primary Configuration data stream. The device’s alternate logical address (ADDR2) is established within a subsequent configuration data block within that same data stream. Once the Primary Configuration is complete, a device will respond to any Update reconfiguration data stream which contains either: a matching ADDR1 value, a matching ADDR2 value or 0xFF in the LOGICAL ADDRESS byte field. The hex address of 0xFF serves as a global logical address; *all* devices respond to this ID, regardless of their ADDR1 or ADDR2 values.

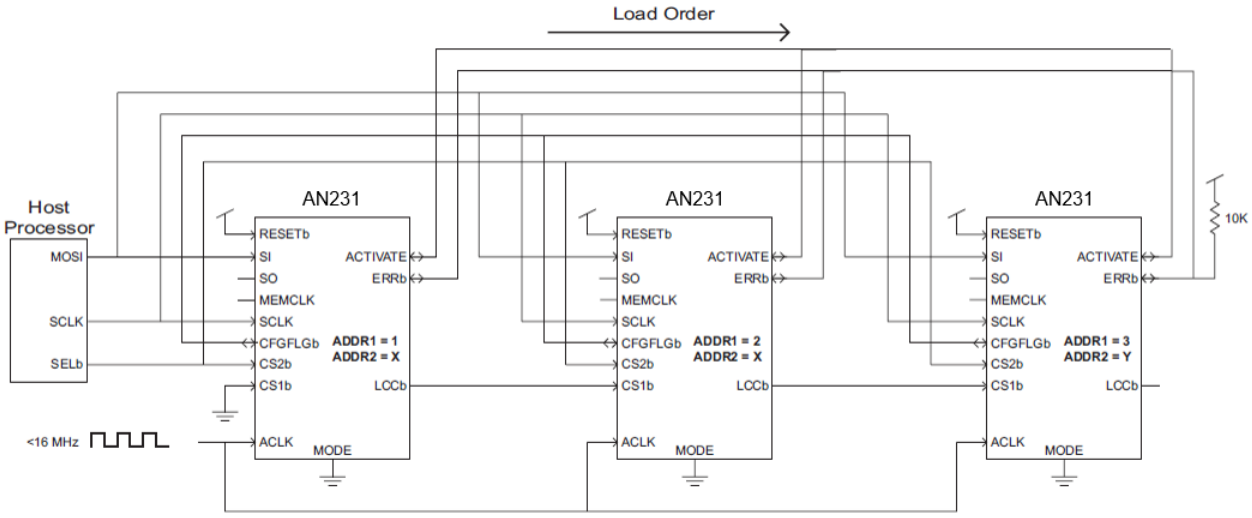


Figure 15 - Primary and Alternate Logical Addressing

Figure 15 shows a valid connection and configuration example for multiple FPAA's being hosted from a single SPI port. In this example, during Primary Configuration, each device in the chain received a unique ADDR1 and a non-unique ADDR2. The ADDR1's were assigned via the ADDR1 fields of the Primary Configuration data streams. The ADDR2's were established within the configuration data sets downloaded into each of the devices. Once Primary Configurations are complete for all of these devices, each will respond to the host SPI port only if the LOGICAL ADDRESS field of the Update configuration data stream contains either its ADDR1 or ADDR2 identifiers (or 0xFF).

Assume that two of the devices perform identical analog functions "X", and the third a unique function "Y". Also assume that the X and Y configuration data sets contain ADDR2 values of X and Y respectively. During the Primary Configuration, the host processor assigned the first device an ADDR1 of 1 and filled that device with the X configuration. Likewise, the ADDR1=2 device also got the X configuration. The ADDR1=3 device got the Y configuration. Once these Primary Configurations are complete and analog operations go active, the host processor may alter via Update both X devices concurrently by using LOGICAL ADDRESS = X rather than sequentially by addressing LOGICAL ADDRESS = 1 then LOGICAL ADDRESS = 2. If the host instead uses 0xFF in the LOGICAL ADDRESS field, then all three devices will concurrently accept the subsequent reconfiguration data.

Tieing CFGFLGb Pins Together in multi-FPAA Systems

When connecting multiple FPAAs up as suggested in Figure 15, it is necessary to tie the CFGFLGb pins of the devices to a common node. The CFGFLGb node will be driven low whenever a device is being addressed for configuration or reconfiguration (AN23x devices only) and will pull-high when data transfer is complete. When the local device senses CFGFLGb as being pulled low (by another device), it will ignore the data on SI. This CFGFLGb signaling prevents reconfiguration data intended for one device from being wrong intercepted by another.

6. Package and Pin Information

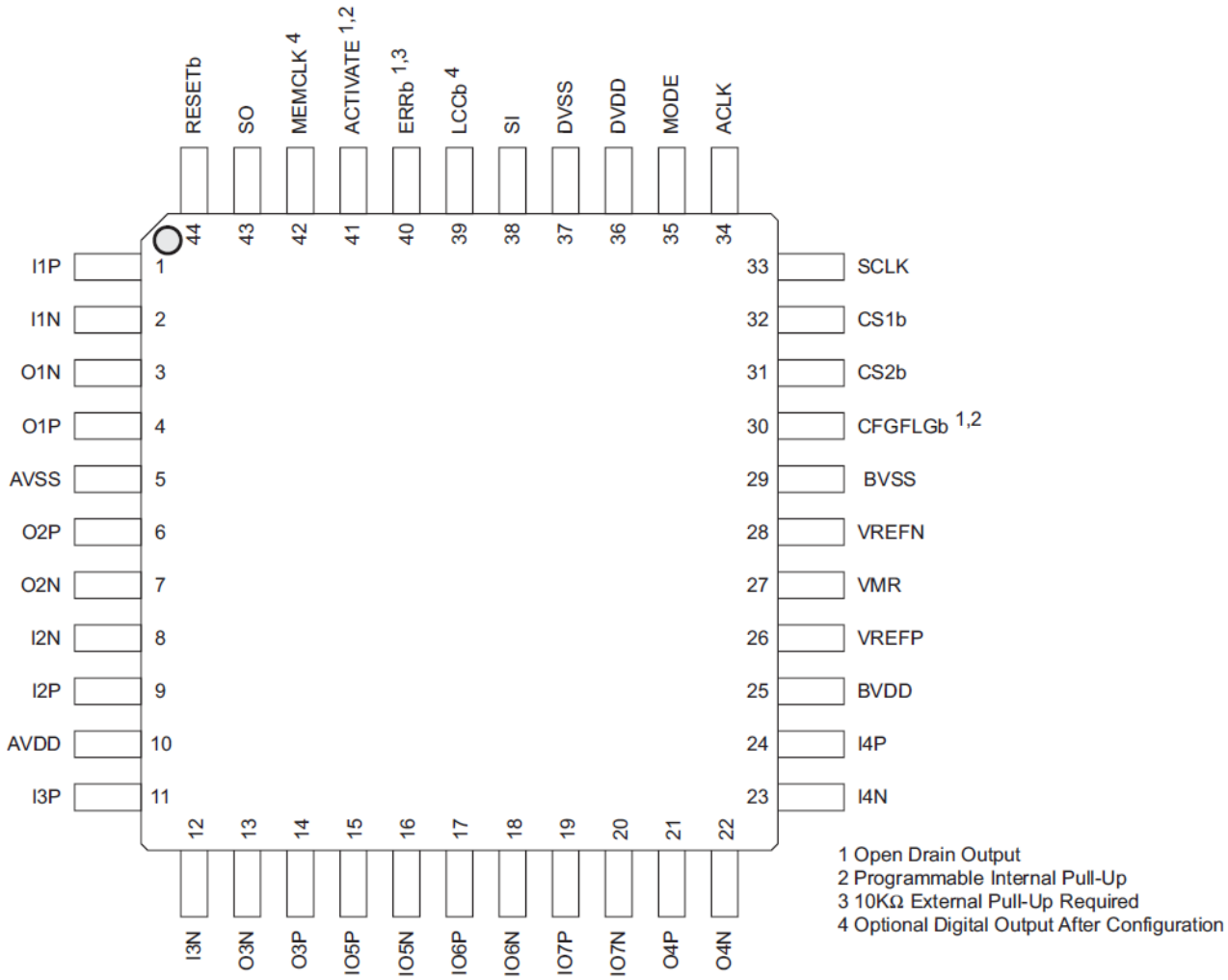


Figure 16 - 44 QFN Package

Pin	Name		Description	
1	I1P		Positive Input	IO Cell 1
2	I1N		Negative Input	(Type 1)
3	O1N		Negative Output	
4	O1P		Positive Output	
5	AVSS		Analog Ground (0 V)	
6	O2P		Positive Output	IO Cell 2
7	O2N		Negative Output	(Type 1)
8	I2N		Negative Input	
9	I2P		Positive Input	
10	AVD		D Analog Power (3.3 V)	
11	I3P		Positive Input	IO Cell 3
12	I3N		Negative Input	(Type 1a)
13	O3N		Negative Output	
14	O3P		Positive Output	
15	IO5		P Positive Input / Output	IO Cell 5
16	IO5		N Negative Input / Output	(Type 2)
17	IO6		P Positive Input / Output	IO Cell 6
18	IO6		N Negative Input / Output	(Type 2)
19	IO7		P Positive Input / Output	IO Cell 7
20	IO7		N Negative Input / Output	(Type 2a)
21	O4P		Positive Output	IO Cell 4
22	O4N		Negative Output	(Type 1a)
23	I4N		Negative Input	
24	I4P		Positive Input	
25	BVDD		Band Gap Power (3.3 V)	
26	VREFP		Noise Dump - attach a 100 nF capacitor between VREFP and BVSS	
27	VMR		Noise Dump - attach a 100 nF capacitor between VMR and BVSS	
28	VREFN		Noise Dump - attach a 100 nF capacitor between VREFN and BVSS	
29	BVSS		Band Gap Ground (0 V)	
30	CFGFLGb	1,2	Configuration Flag	
31	CS2b		Chip Select 2 (active low)	
32	CS1b		Chip Select 1 (active low)	
33	SCLK		Digital Clock Input	
34	ACLK		Analog Clock Input	
35	MODE	4	1 = Static Operation, 0 = Dynamic Operation	
36	DVDD		Digital Power (3.3V)	
37	DVSS		Digital Ground (0V)	
38	SI		Serial Input	
39	LCCb	5	Local Configuration Complete (active low)	
40	ERRb	1,3	Error Flag (active low)	
41	ACTIVATE	1,2	Activate Device	
42	MEMCLK	5	SPI master mode EPROM Data Clock Output	
43	SO	5	Serial Output	
44	RESETb		Power On Reset Input (active low)	

Figure 17 - Pin Numbering and Descriptions

Notes:
1. Open Drain Output
2. Programmable Internal Pull-Up
3. 10K External Pull-Up Required
4. Optional Digital Input After Configuration
5. Optional Digital Output After Configuration

6.1. Recommended PCB Design Practices

PCB design should include the following features to ensure good separation of the digital and analog signal environments in the target system. Good PCB design practices dictate that the digital and analog power and ground planes be separated. It is important to maintain these planes at the same basic potential, but care should be exercised to prevent the usual noise associated with a digital plane from coupling onto the analog plane. The electrical connection between the two planes is typically made at just one point, through ferrite bead choked wire. The ferrite beads act as low pass filters.

As with any mixed signal board design, it is good practice to keep digital signals (especially digital signals with high edge rates) routed only over digital power and ground planes. Care should be exercised to never route a high edge rate digital signal perpendicular to a plane boundary. Doing so will cause a noise wavefront to launch (left and right) onto both planes along the boundary.

It is recommended that the digital supply DVDD be bypassed to DVSS using ceramic capacitors. A 0.1 µF capacitor in parallel with a 0.01 µF capacitor is usually sufficient. The capacitor connections to the device should be made as close as practical to the package to reduce inductance. This same bypassing scheme will work sufficiently for BVDD - BVSS, AVDD - AVSS, pairs as well.

In multiple FPAA systems the VMR node for all devices should be tied to a common node. Just one of the devices should have its VMR enabled. All other devices should have VMR disabled - using the single device as their reference generator. Doing so ensures that all devices in the system use a common reference voltage, eliminating offset errors that might otherwise occur between devices.