Register	Offset	Size	Description
PHASE_STATUS	0x0	0x1	A high on bit 5 indicates that the phase-shift on adc1_clk is complete A high on bit 4 indicates that the phase-shift on adc0_clk is complete Bit 1 contains the value of adc0_clk sampled with adc1_clk[1] Bit 0 contains the value of adc1_clk sampled with adc0_clk
PHASE_CONTROL	0x1	0x1	Bit 5 control the DCM phase shift direction [PSINCDEC] of adc1_clk Bit 1 control the DCM phase shift direction [PSINCDEC] of adc0_clk Setting bit 4 will issue a single DCM phase shift step on adc1_clk Setting bit 0 will issue a single DCM phase shift step on adc0_clk[2]
MODE_CONTROL	0x2	0x1	Clearing bit 5 will disable the gateware autoconfig for adc1 Clearing bit 4 will disable the gateware autoconfig for adc0 Bit 1 is tied to the MODE pin for adc1 Bit 0 is tied to the MODE pin for adc0[3]
RESET_CONTROL	0x3	0x1	Setting bit 1 will issue a DDRB and a DCM reset for adc1 Setting bit 0 will issue a DDRB and a DCM reset for adc0
ADC0_DATA	0x4	0x2	This register contains the DATA to be used when configuring adc0 via the three-wire interface
ADC0_ADDR	0x6	0x1	This register contains the ADDRESS to be used when configuring adc0 via the three-wire interface
ADC0_CONFIG	0x7	0x1	Setting bit 0 will issue a three-wire configuration on adc0 Bit 0 will remain set until the configuration has been completed
ADC1_DATA	0x8	0x2	This register contains the DATA to be used when configuring adc1 via the three-wire interface
ADC1_ADDR	0xa	0x1	This register contains the ADDRESS to be used when configuring adc1 via the three-wire interface
ADC1_CONFIG	0xb	0x1	Setting bit 0 will issue a three-wire configuration on adc1 Bit 0 will remain set until the configuration has been completed

1 This is used for testing whether adc0_clk is aligned with adc1_clk
2 See Xilinx datasheet
3 See AT84AD001B datasheet for details on MODE behaviour

Python Use Cases

Increase phase of ADC0 clock by a single DCM tap without affecting auto-configuration
fpga.blindwrite('iadc_controller','%c%c%c%c'%(0x0, 0x03, 0xff, 0x0), offset=0x0)

Decrease phase of both ADC clocks by a single DCM tap without affecting auto-configuration
fpga.blindwrite('iadc_controller','%c%c%c%c'%(0x0, 0x11, 0xff, 0x0), offset=0x0)

Enable manual configuration and write an 'SPI' value to ADC0
spi_data = 0x1234
spi_addr = 0x02
#set mode bit high and disable autoconfiguration logic for adc0
#also ensure that autoconfiguration is left enable for adc1
fpga.blindwrite('iadc_controller','%c%c%c%c'%(0x0, 0x0, 0x21, 0x0), offset=0x0)
#write spi data into iadc data register
fpga.blindwrite('iadc_controller','%c%c%c%c'%((spi_data & 0xff00) >>8, spi_data & 0xff, spi_addr, 0x1), offset=0x4)
#reset both adc clock and dcms; make sure mode and autoconfig bits remain the same
fpga.blindwrite('iadc_controller','%c%c%c%c'%(0x0, 0x0, 0x21, 0x3), offset=0x0)
#there is no need to de-assert the reset