

## LA-UR-15-26312

Approved for public release; distribution is unlimited.

Title: Payload Communications Interface for CubeSat Platform: Design Review

Author(s): Akins, Alexander Brooks

Intended for: Report

Issued: 2015-08-10

---

**Disclaimer:**

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.



# Payload Communications Interface for CubeSat Platform: Design Review

July 2015

# Contents

---

- ▶ Objectives
- ▶ System Overview
- ▶ Breakdown of Individual Components
- ▶ Tracing Data Path
- ▶ Testing
- ▶ Special Considerations



# Objectives

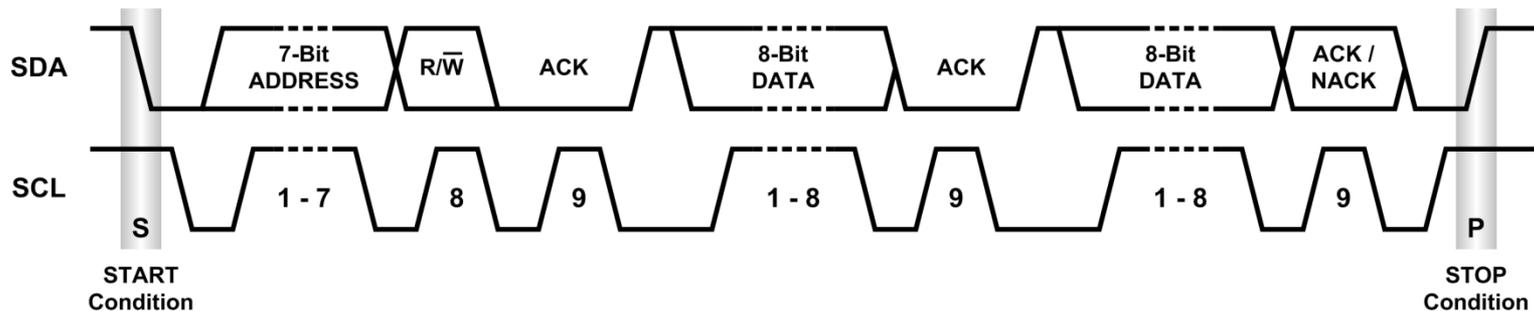
---

- ▶ **Primary Goal:** Send important sensor data from payload to SV following an event trigger as quickly as possible with high data integrity
- ▶ Control a variety of payload functions with reliability
- ▶ Reduce likelihood of error states
- ▶ Transmit data to SV via UART or I<sup>2</sup>C serial interface



# I<sup>2</sup>C Protocol

- ▶ Two-wire, serial interface consisting of a data line (SDA) and a clock line (SCK)
  - ▶ Supports 100 kHz, 400 kHz, 1 MHz clock speeds
  - ▶ Currently using 400 kHz (faster than SV UART)
- ▶ Idle bus is held high by pull-up resistors



Example I<sup>2</sup>C bus transaction

# Data Types: Read-Only

---

## ▶ Sensor Data

- ▶ Output voltages from high and low gain ADC channels in response to a trigger event
- ▶ Packaged with some trigger information (timing, etc.)

## ▶ State of Health Data

- ▶ Includes voltages from supplies, temperature, etc.
- ▶ Occurs every second

## ▶ FPGA Version Number

- ▶ Constant value indicating flight FPGA serial number



# Data Types: Read-Write

---

- ▶ **Test Pulse Settings**

- ▶ Settings to exercise the sensor system
- ▶ Includes number, time between pulses, etc.

- ▶ **DAC Settings**

- ▶ Configurable discriminator threshold levels realized via DACs

- ▶ **HV Settings**

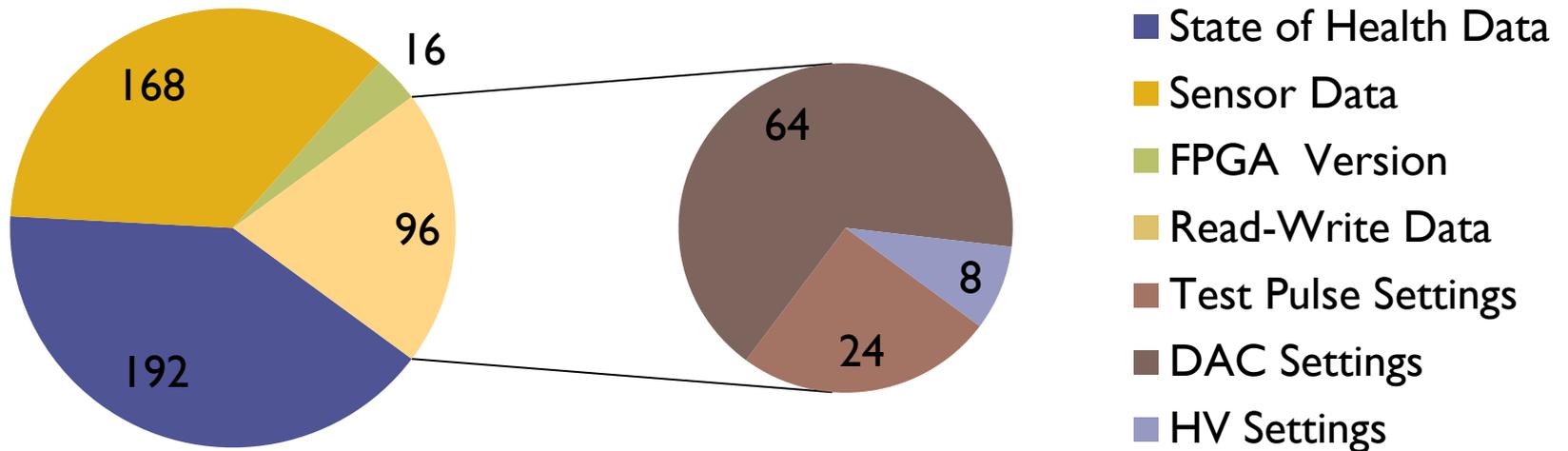
- ▶ Enable bits for high voltage sensor supply



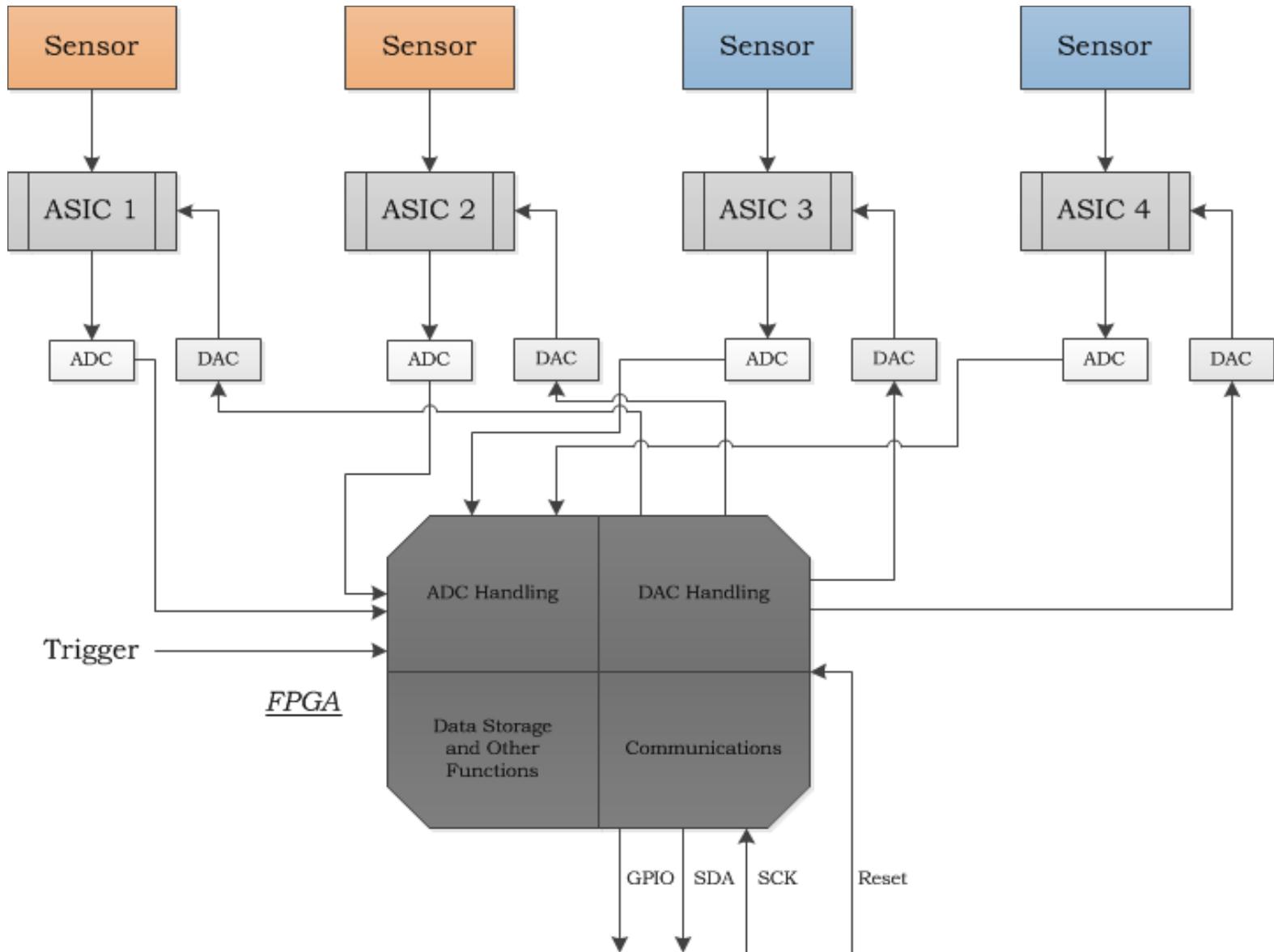
# Data Footprint

---

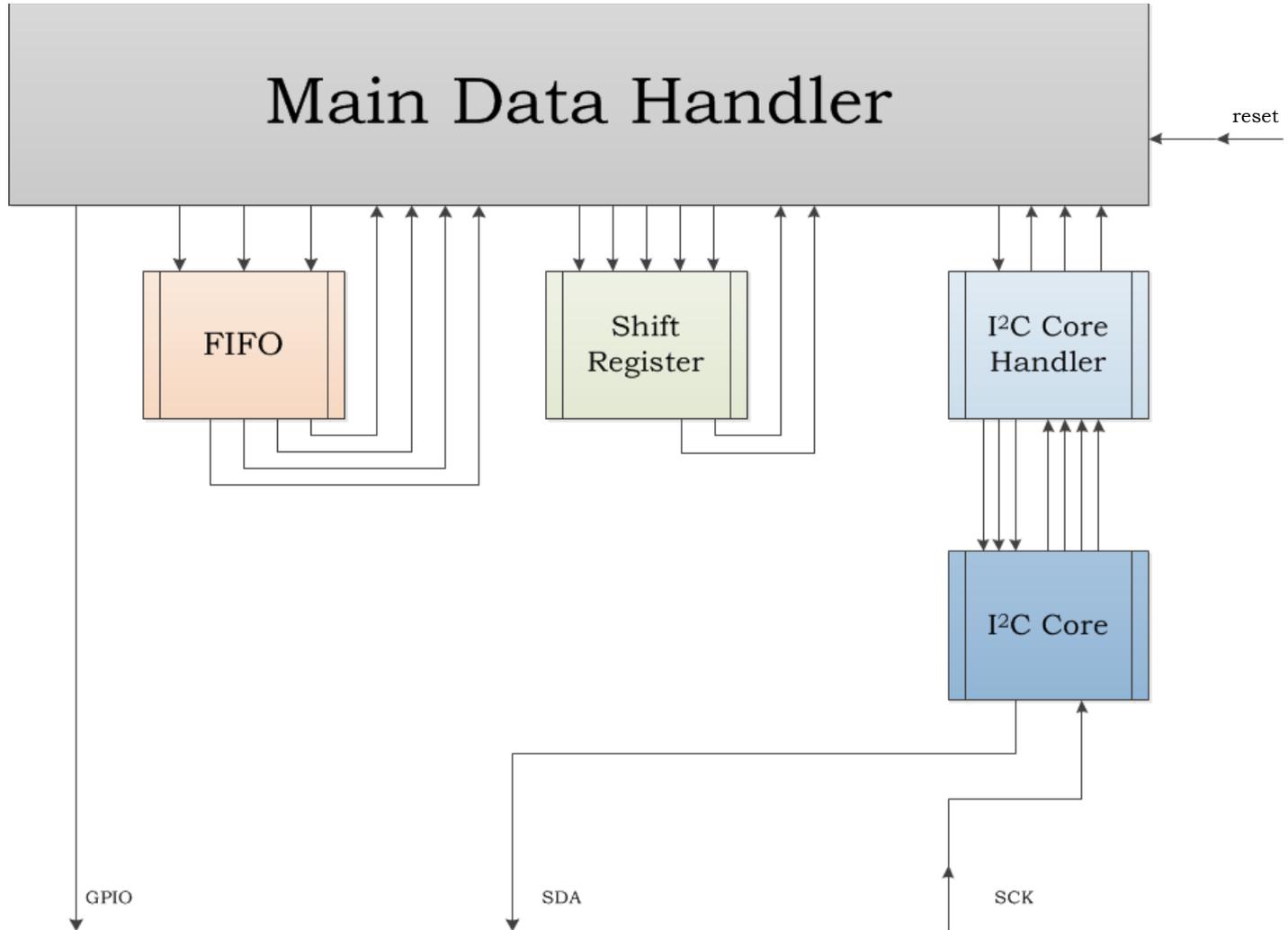
## Data Bits



# System Overview

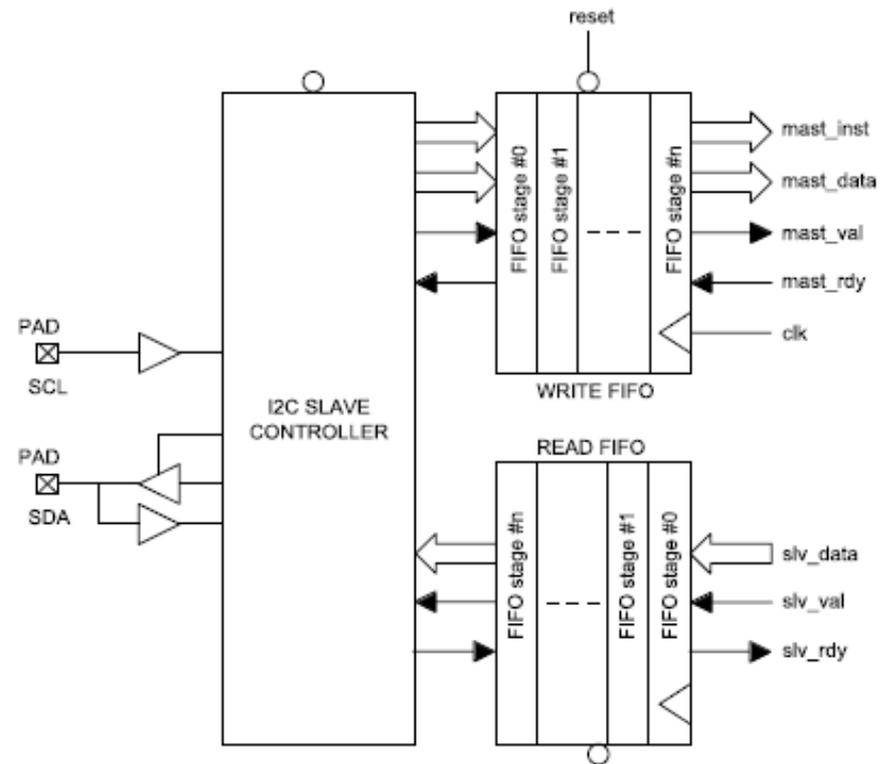


# Communications Interface



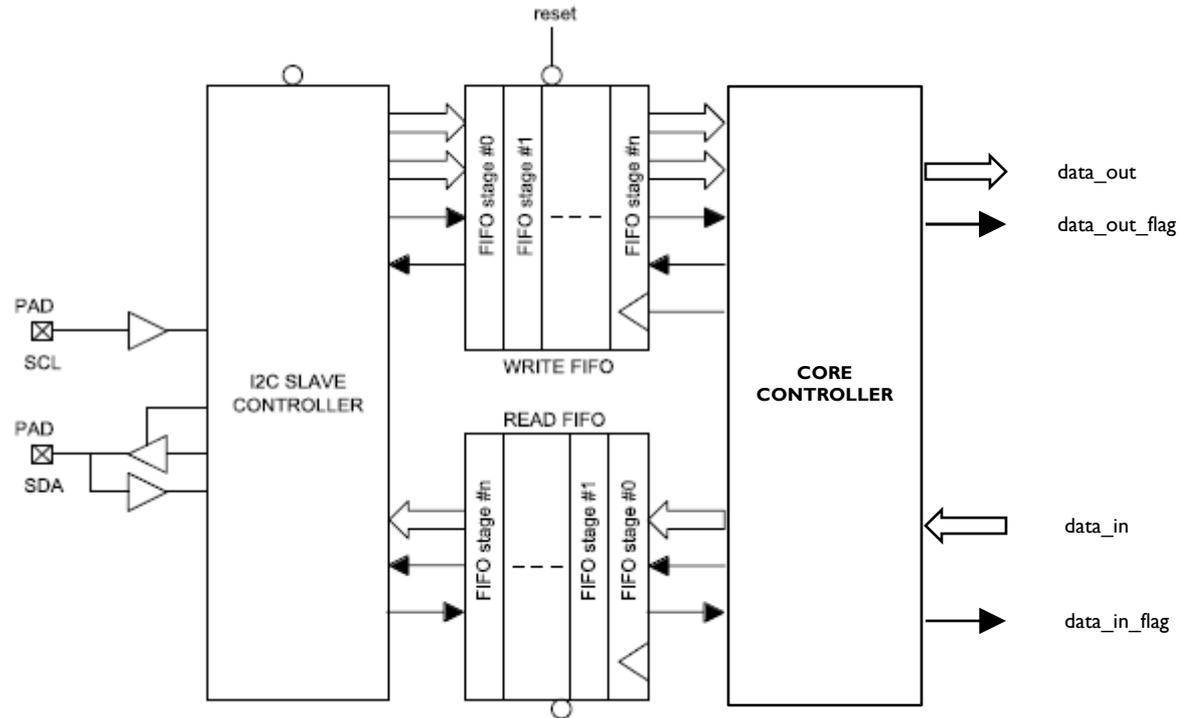
# ZIPCores I<sup>2</sup>C Slave Serial Interface Core

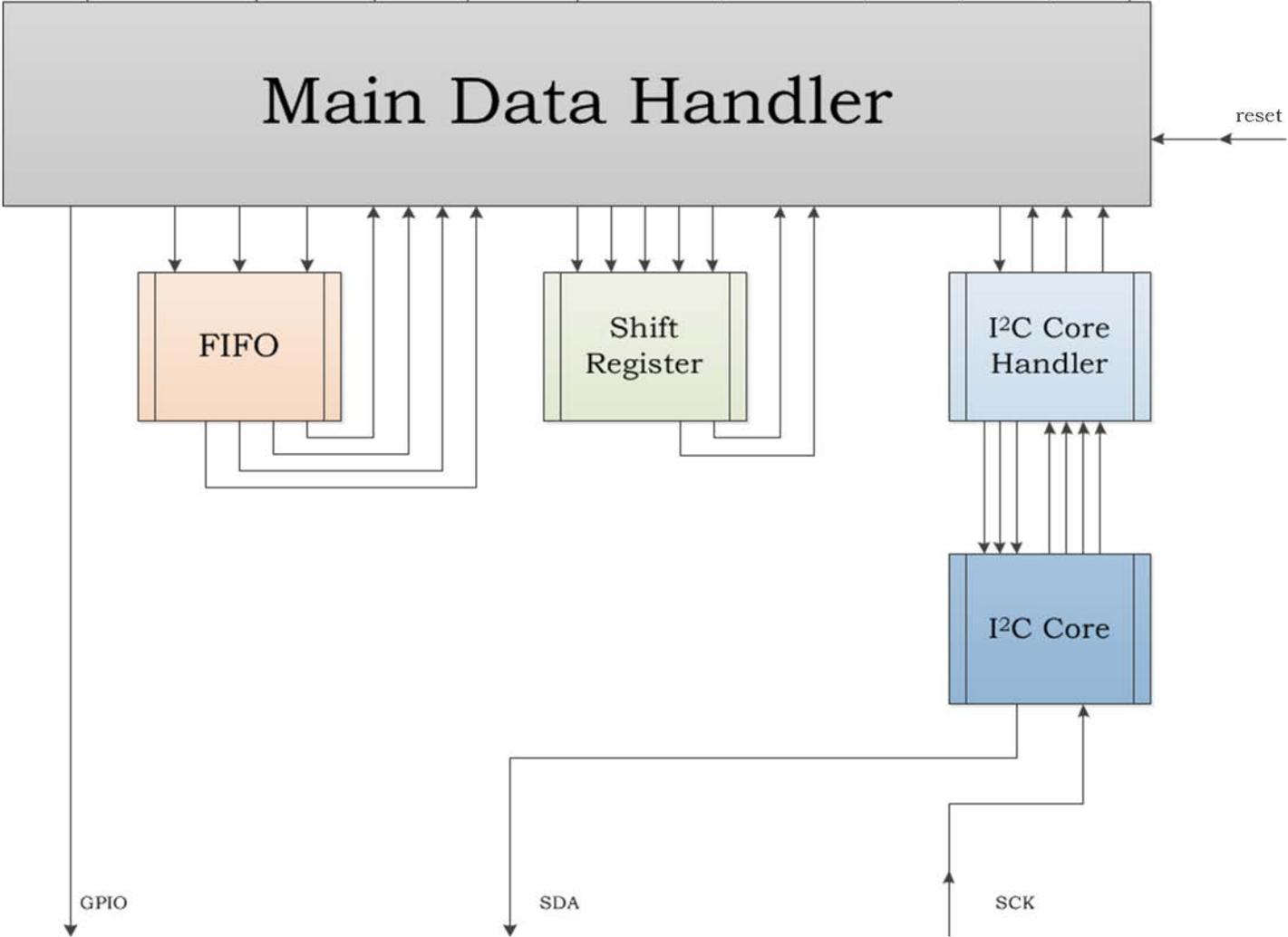
- ▶ Handles all low-level I<sup>2</sup>C functionality
  - ▶ Parses instructions, acknowledgements, start/stop conditions
- ▶ Data loading/unloading via two FIFOs
- ▶ Uses “valid-ready” pipeline protocol



# Slave Core Controller

---





# Main Data Handler Operation

---

- ▶ State machine
  - ▶ States accessed via initial master write
  - ▶ Operation States
    - ▶ INTERPRET\_INSTRUCTION
      - ▶ Idle state where system is waiting for an instruction from the master
    - ▶ SENSOR\_DATA
    - ▶ SOH\_DATA
    - ▶ FPGA\_VERSION
    - ▶ TEST\_PULSE\_CONFIGURE
    - ▶ DAC\_CONFIGURE
    - ▶ HV\_CONFIGURE
- 
- Read-Only
- Read-Write



# Master Instructions

---

Register Data	Register Address
SOH Data	0x01
Sensor Data	0x02
Test Pulse Settings	0x03
DAC Settings	0x04
HV Settings	0x05
FPGA Version	0x06



# Problem: How do we handle ADC data?

---

- ▶ Answer: FIFO loading
- ▶ Data is loaded into the FIFO as soon as it becomes available, forming a queue of information
- ▶ At lower DAC discriminator levels, event pileup can occur, requiring event storage for later read out
- ▶ Counter index will let the master know how much data needs to be read
  - ▶ Currently one byte (up to 255 events stored), but more bytes could be included



# Tracing Path: Sensor Data

- ▶ Start: Event processed by FPGA ADC handling modules

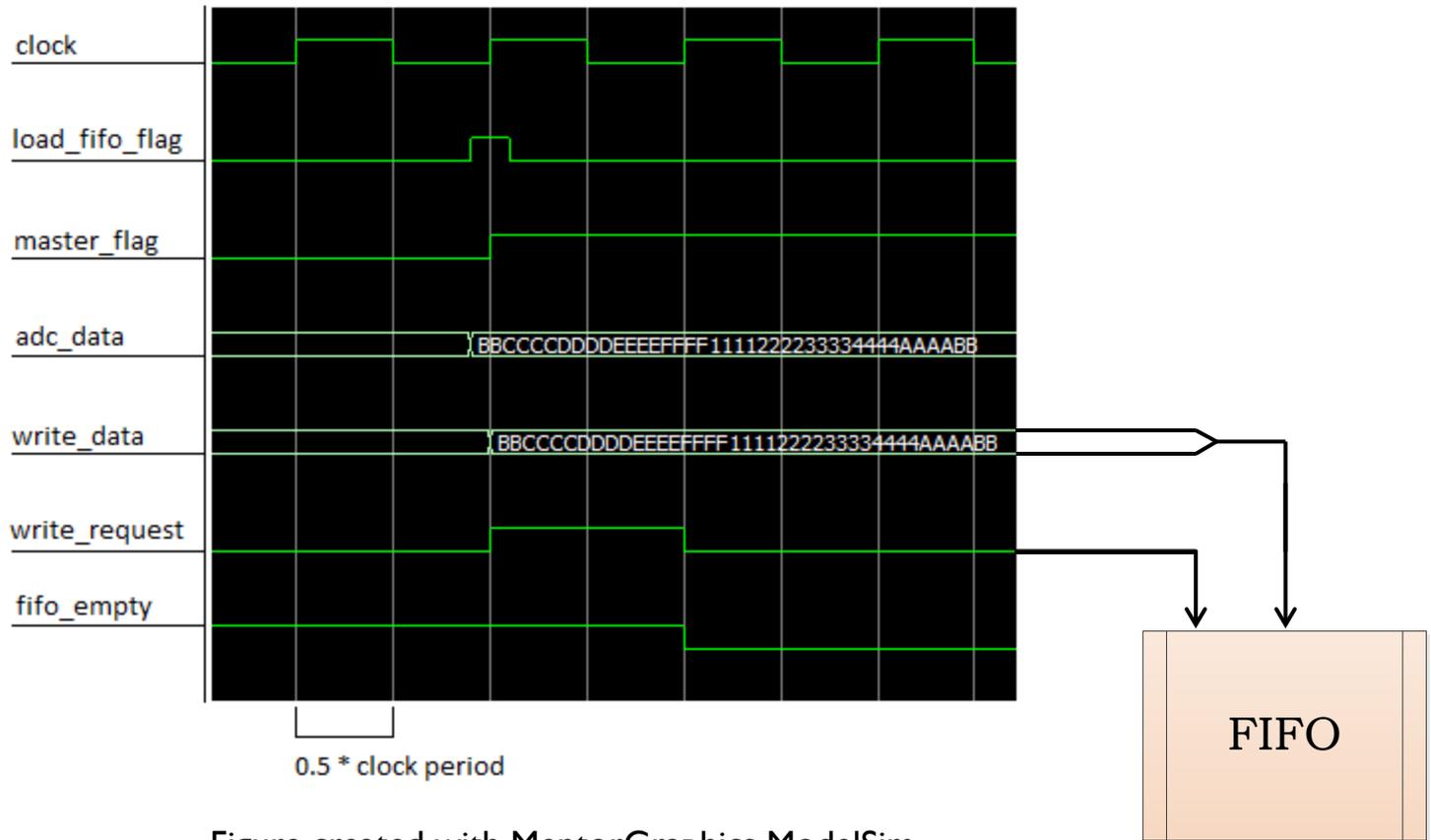
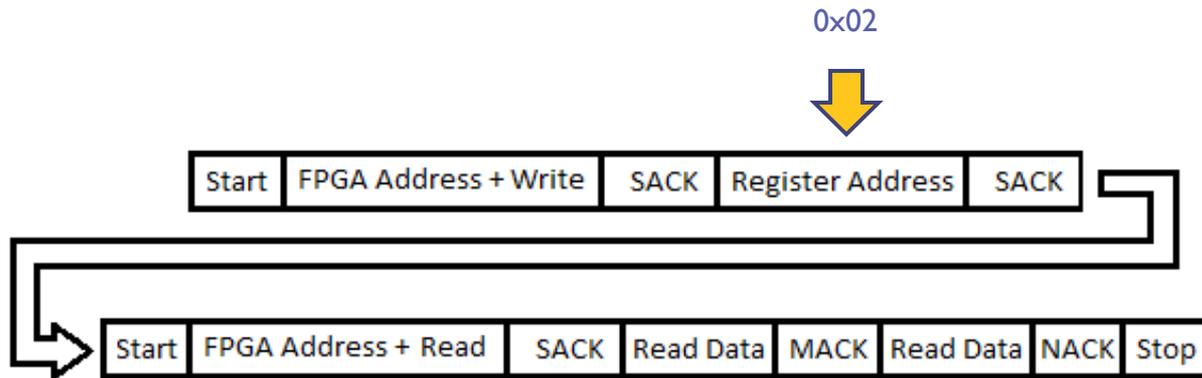


Figure created with MentorGraphics ModelSim

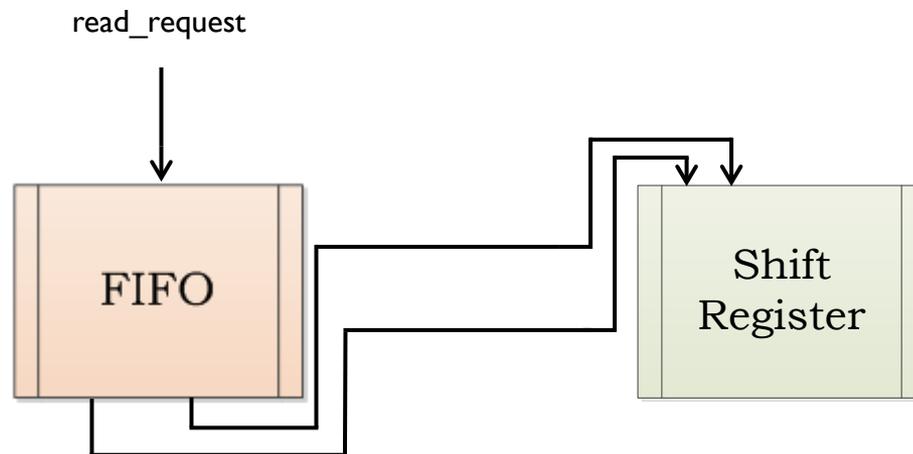
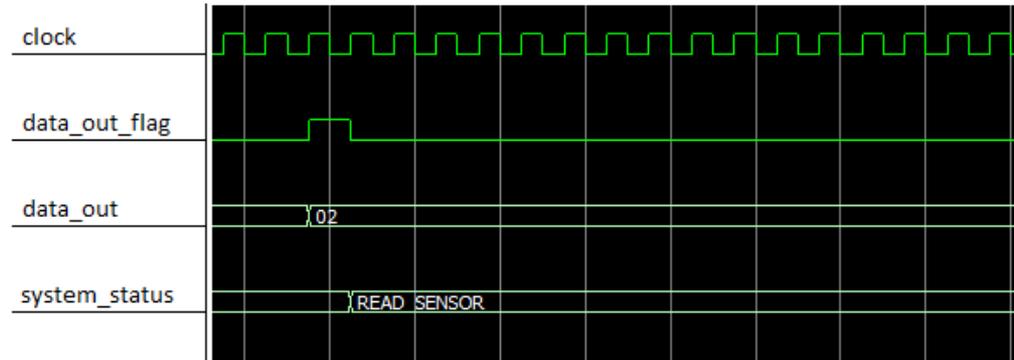
# Example Master Read Operation

---



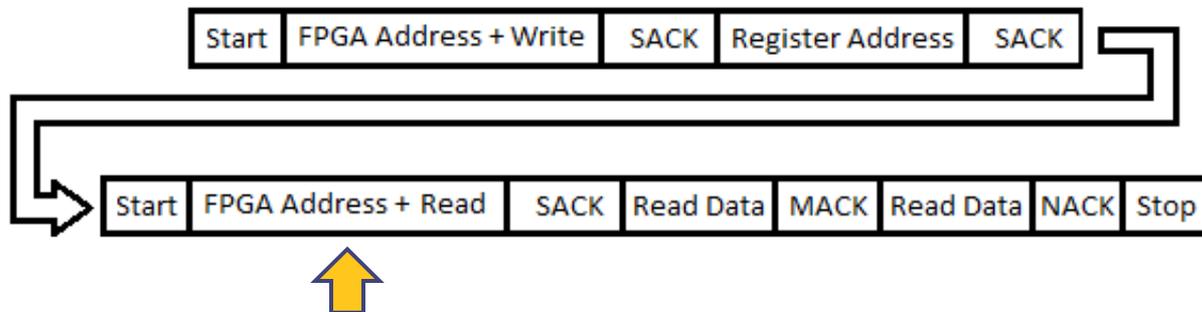
# Tracing Path: Sensor Data

---



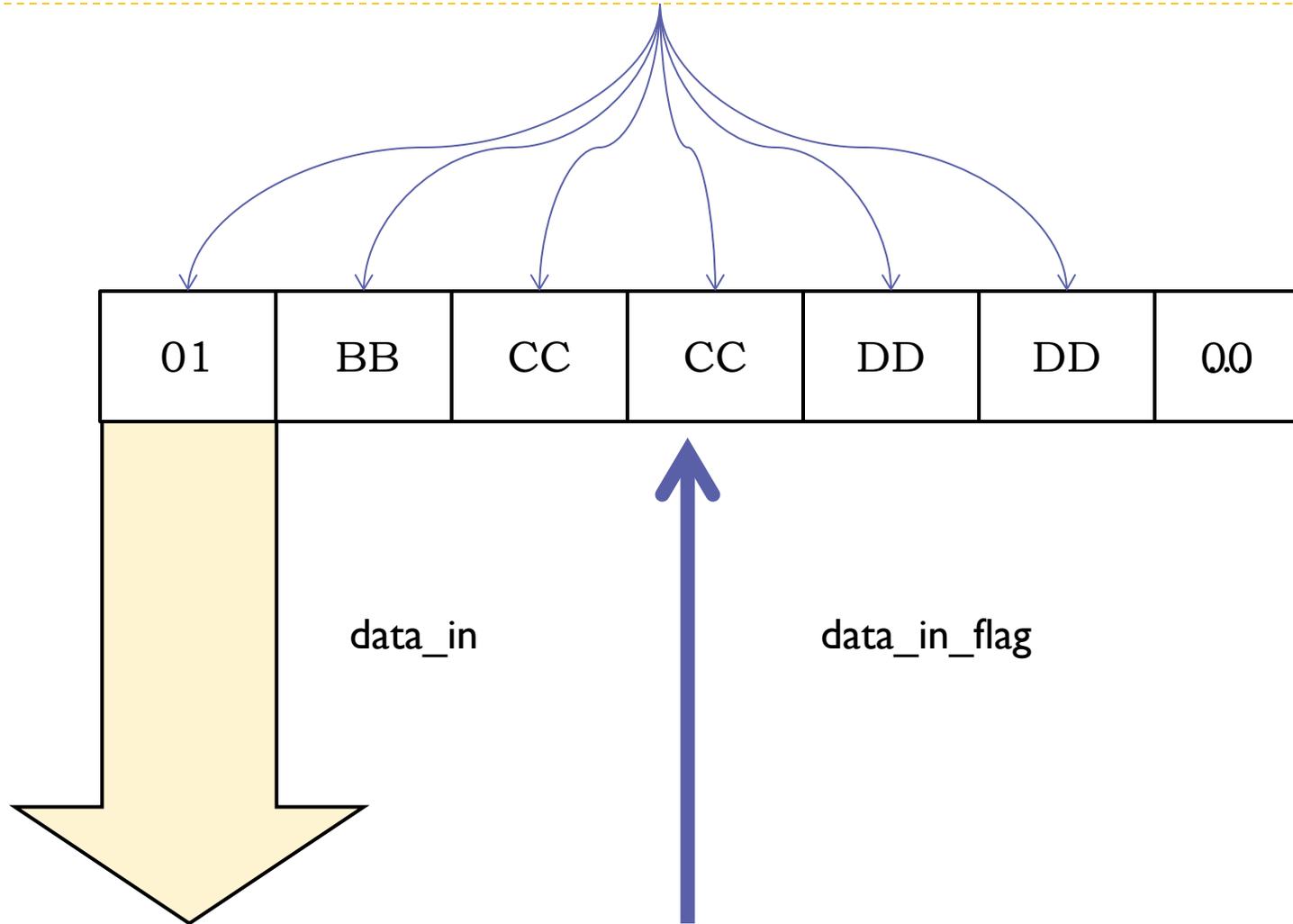
# Example Master Read Operation

---



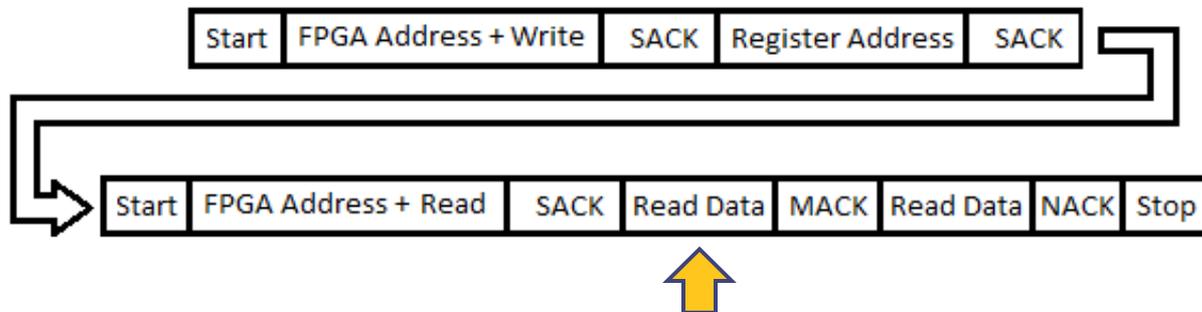
# Shift Register

---



# Example Master Read Operation

---



- ▶ After Stopping, the state will return to INTERPRET\_INSTRUCTION



# Lowering the Master Flag

---

- ▶ After the sensor read operation, the master\_flag signal will still be high.
- ▶ Master should write byte **0x0F** on the I<sup>2</sup>C bus to lower the flag
- ▶ If new events have been counted since the conclusion of the transfer, this operation will not succeed
- ▶ Flag will stay high if new data is ready, and the master should restart the read process



# Reading Data

---

- ▶ Only the sensor data uses the FIFO
- ▶ All data piped through shift register
- ▶ Shift register is 192 bits wide, since largest data size (state of health data) is 192 bits
- ▶ Other read processes require no GPIO flagging
  - ▶ Respond to master instruction, write out X bits, done



# Writing Data

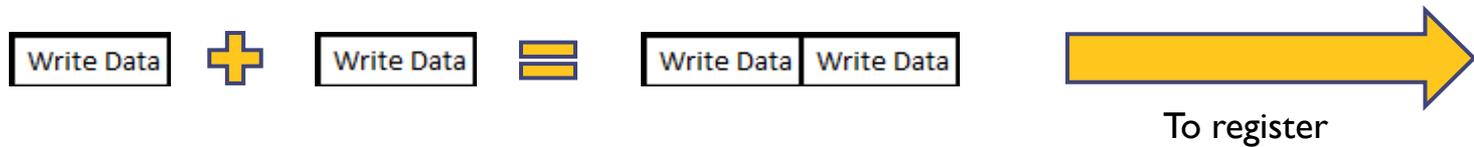
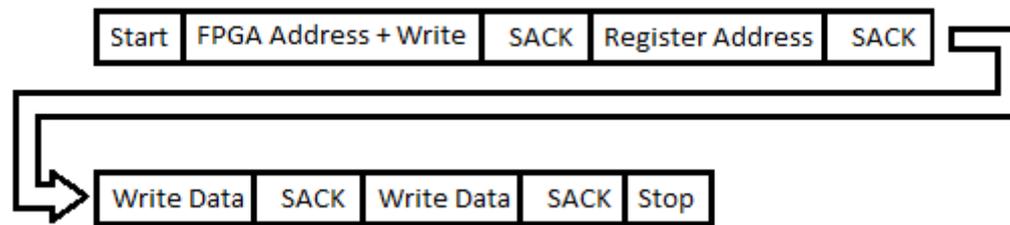
---

- ▶ Shift register disabled, but *shift\_counter* is not
  - ▶ Counts with each byte transfer up to however many are expected and returns to master instruction state
- ▶ Test Pulse and HV Settings are written over the I<sup>2</sup>C bus, and after the final byte is received, the bytes are concatenated and stored
- ▶ DAC Settings require one more step

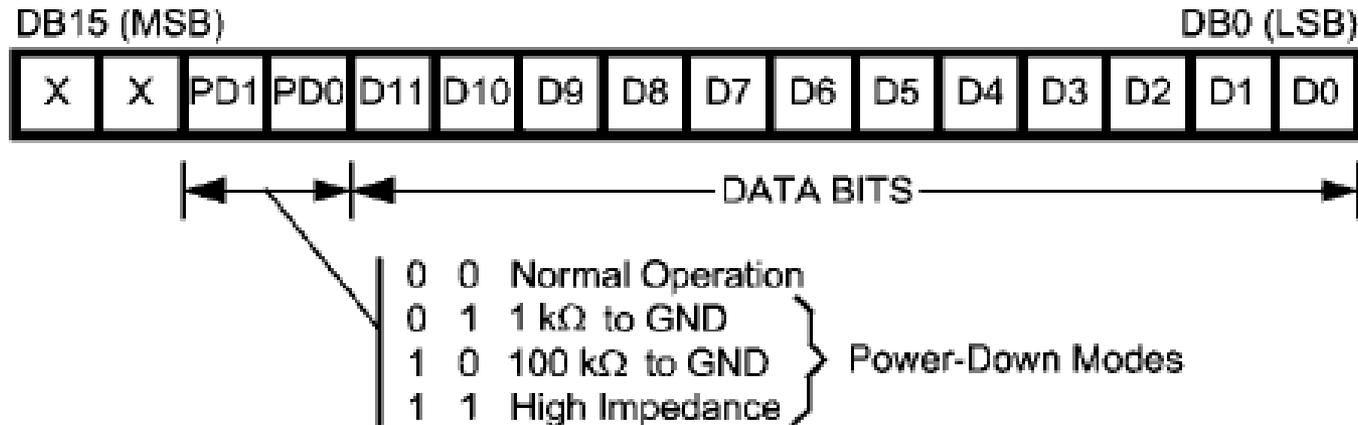


# Example Master Write operation

---



# DAC Discriminator Threshold Settings



Don't-Care Bit Value	DAC to be Addressed
0b00	DAC 1
0b01	DAC 2
0b10	DAC 3
0b11	DAC 4



# Read-Write Data

---

- ▶ When addressing a read-write enabled system, I<sup>2</sup>C master can read or write by performing appropriate bus actions
- ▶ Test Pulse Settings
  - ▶ Reads and writes three bytes at a time
- ▶ DAC Settings
  - ▶ Reads 8 bytes at a time
  - ▶ Write 2 bytes at a time
- ▶ HV Settings
  - ▶ Reads and writes one byte at a time



# Read-Write Data

---

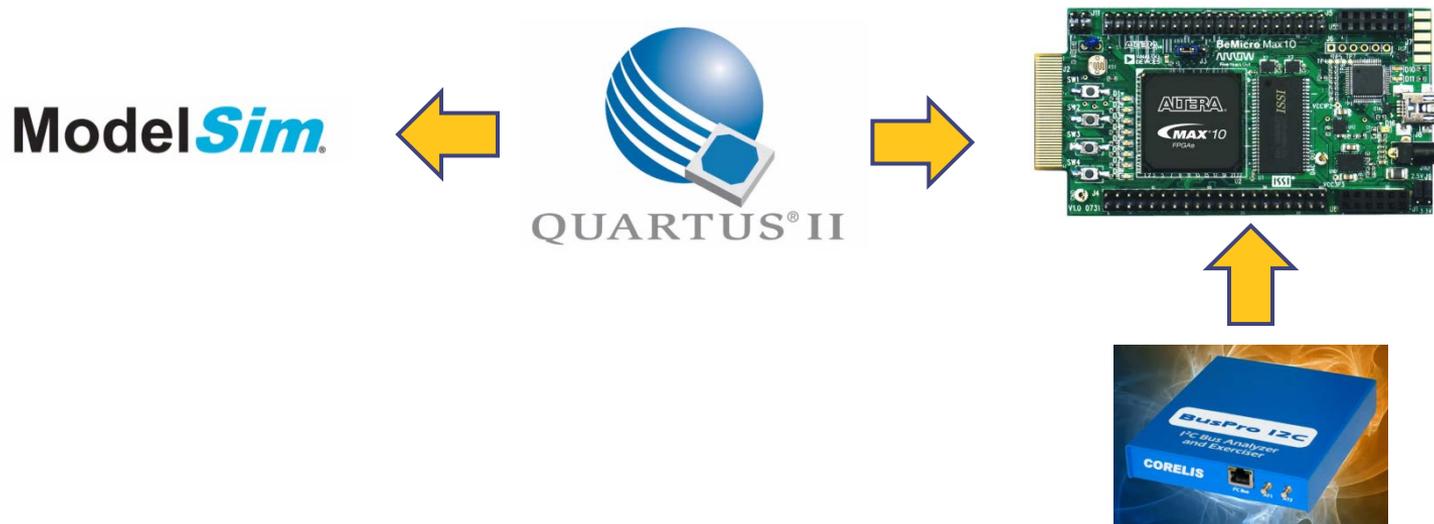
- ▶ Read and Write operations **cannot be interleaved**
- ▶ When reading or writing data to Test Pulse, DAC, or HV register, all operations must be completed fully
- ▶ Attempting to perform pieces of both operations will result in:
  - ▶ Incomplete read data
  - ▶ Undesirable writes to registers
- ▶ If this does happen, a system reset will revert system to safe state



# Testing and Prototyping

---

- ▶ Code written in VHDL, compiled with Quartus II
- ▶ Simulated in MentorGraphics ModelSim
- ▶ Synthesized and programmed onto BeMicro Max 10 development board
  - ▶ Tested with Corelis BusPro-I and I<sup>2</sup>C Exerciser Software



# I<sup>2</sup>C Exerciser Software

The screenshot displays the I<sup>2</sup>C Exerciser software interface, which is divided into three main sections:

- Monitor - (Untitled):** A table showing I<sup>2</sup>C transaction details. The table has columns for Line, Marker, Type, Location, Addr (Hex), AddrType, R/W, NAK, Error, KHz, I/O 1, I/O 2, Data Byte (Hex), and Time (us). The current line is 190, showing a Data Target transaction at address 66 with data byte BB.
- Debugger - (Untitled)\*:** A window for sending and receiving data. The Send section shows Addr: 66, AddrType: 7-Bit, Run: Single, and a Passed status. The Receive section shows Addr: 66, AddrType: 7-Bit, Bytes: 21, and a Passed status. A list of received bytes is shown: 01 CC CC DD DD EE EE FF FF 11 11 22 22 33 33 44 44 AA AA BB BB.
- Oscilloscope:** A waveform showing the SDA (Data) and SCL (Clock) signals. The SDA signal is shown as a series of data bytes: 01, CC, CC, DD, DD, EE, EE, FF, FF, 11, 11, 22, 22, 33, 33, 44, 44, AA, AA, BB, BB. The SCL signal is a regular clock signal. The scale is set to 50 us/div and the line is set to 190.

Short pulses occur during control hand-off for acknowledgement signals

# Special Conditions

---

- ▶ **When too many byte reads occur within a process:**
  - ▶ The interface responds by sending out the appropriate number of data bytes followed by zero bytes for excess reads
- ▶ **When too few byte reads occur within a process:**
  - ▶ The interface will remain in current state until all bytes have been read out. Since bytes are left in shift register, no data is lost by delaying reads.
- ▶ **When FIFO is full:**
  - ▶ Counter bit reads FF twice before decrementing
- ▶ **All other function occurs as intended**



# Important Considerations

---

- ▶ What will be our flight FPGA?
- ▶ Form Factor and Power Considerations
- ▶ Ensuring proprietary VHDL core compatibility
- ▶ How fast can we reliably drive our serial communication
  - ▶ 1 MHz? 5 MHz?





Questions?

