CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Digital Datapath for an Image Acquisition System

A graduate project submitted in partial fulfillment of the requirements

For the degree of Master of Science in

Electrical Engineering

By

Bartłomiej Kowalski

May 2014

The graduate project of Bartłomiej Kowalski is approved:

_____	_____
Dr. Ronald W. Mehler	Date
_____	_____
Dr. Somnath Chattopadhyay	Date
_____	_____
Dr. Nagi El Naga, Chair	Date

California State University, Northridge

DEDICATION

This graduate project is dedicated to my father, Antoni.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Digital Datapath for an Image Acquisition System

By

Bartłomiej Kowalski

Master of Science in Electrical Engineering

The Digital Datapath for an Image Acquisition System (IAS) describes the architecture of an FPGA-based system designed to capture, process, and transmit image data generated by an array of photo-detectors. Due to the substantial number of photo-detectors in the array, two of the most significant challenges and focuses of this project are to capture incoming image data simultaneously, and to organize it into a format which lends itself for serial transmission to the final destination. To accomplish these tasks, a memory mapping and data processing algorithm is implemented over two FPGA stages. The necessity for the IAS originates from the lack of hardware support between a long-range image targeting system, and a data processing application which reconstructs an image from the indistinguishable captured data. The objective of this graduate project is to design a system which meets the set of specified requirements, and to simulate the logic which controls the IAS datapath.

CHAPTER 1

THE IMAGE ACQUISITION SYSTEM

1.1 Application and Objective

The Image Acquisition System proposed in this graduate project is a sub-system of a larger and more complex imaging mechanism, one specifically intended for long-range image acquisition via aircraft. The complete system is comprised of three separate parts. First, an image targeting system transmits a laser beam at the area of interest. Next, the IAS captures the reflected speckle pattern on a 32 x 32 photo-detector array and transmits the raw speckle data to the final destination – a data processing application. And finally, the data processing application utilizes an image regeneration algorithm to produce an observable representation of the targeted area of interest from the captured speckle data. The image targeting system and data processing application are beyond the scope of this project.

The fundamental application of the IAS is to support the long-range imaging system by methodically acquiring samples of speckle pattern data from extensive distances. Succinctly, laser speckle photography techniques rely on collecting the reflected diffusion of a coherent laser, in order to extrapolate certain physical properties of the imaged medium [1]. Traditionally, this method of imaging has been carried out at the microscopic level, in the fields of molecular biology and materials engineering. In these applications, laser speckle images are processed by advanced computer programs to provide meaningful information about the molecular structure, on a level unattainable by most advanced magnification hardware.

In effect, the imaging system under discussion extends laser speckle photography techniques to a grander scale, over extremely large distances. In this application, the targeted objects are undetectable by the naked eye, binoculars, or even conventional telescopic photography hardware. Moreover, the airborne environment in which the system operates impedes image regeneration to the point that a single speckle pattern sample does not contain the sufficient amount of information which is required by the digital signal processing algorithm. Consequently, due to very precise sampling constraints, and considerable physical distance between the system host and the targeted objects, an off-the-shelf solution is entirely inadequate.

Clearly, this unique long-range imaging application requires an Image Acquisition System which is specifically designed to provide speckle pattern data in a manner which is harmonious with both the image targeting system, and data processing application. Thus, the objective of the IAS, and this graduate project, is very clear – to design a system which conforms to the specific set of requirements defined by the underlying imaging system.

1.2 Introduction to the Image Acquisition System

To carry out its function, the Image Acquisition System uses a stage of analog to digital converters (ADC) to sample the photo-detector array 18 times per millisecond. The ADCs sample the photo-detector array at a frequency of 12 MHz, and generate a digital 12-bit representation of each photo-detector's input. The IAS captures all ADC data streams in parallel, and prepares image data for the data processing application during the

last two stages of the image acquisition cycle. The final output is a high speed serial protocol transmission to the data processing application.

The imaging system cyclically targets points of interest at a frequency of 1 kHz – these are referred to as image acquisition cycles. During one image acquisition cycle, the IAS digitizes the photo-detector array, captures, processes, and then transmits all speckle pattern data to the data processing application.

1.3 Project Outline

1.3.1 Image Acquisition System Architecture

The IAS employs 128 Texas Instruments ADS5281 analog to digital converters, 128 Texas Instruments SN65LVDS386 differential line receivers, and four Xilinx Virtex-6 FPGAs (Package XC6VLX75T- FF484). Throughout this document, the four Virtex-6 FPGAs are referred to as FPGAs[0-3]. FPGA[0] is designated as the Main Controller.

The Image Acquisition System's photo-detector array is constructed from 4 x 4 photo-detector array modules. Sixty-four modules make up the entire 32 x 32 photo-detector array. The design of the 4 x 4 module is outside the scope of the digital datapath. With this level of abstraction, the datapath begins with ADS5281 configuration and control signals, followed by the capture and processing of 1,024 serial image data streams originating from the 128 analog to digital converters.

The IAS architecture is divided into three stages. Chapter 3 discusses the first stage, in which all ADS5281 ICs are powered up, initialized, and configured to sample and digitize the photo-detector array. FPGA[0] generates all ADS5281 control signals,

including the 12 MHz Sample Clock and 1 kHz Sample Pulse. Only the ADCs and FPGA[0] take part in the first stage.

Chapter 4 discusses the second stage. This is where serial image data generated by the ADS5281s is captured by FPGAs[0-3]. All four FPGAs are necessary to capture the substantial amount of image data produced by the 32 x 32 photo-detector array. Each FPGA receives image data generated by a 16 x 16 array which accounts for a quarter of the 32 x 32 array. Figure 1-1 illustrates how the photo-detector array is divided among the four FPGAs.

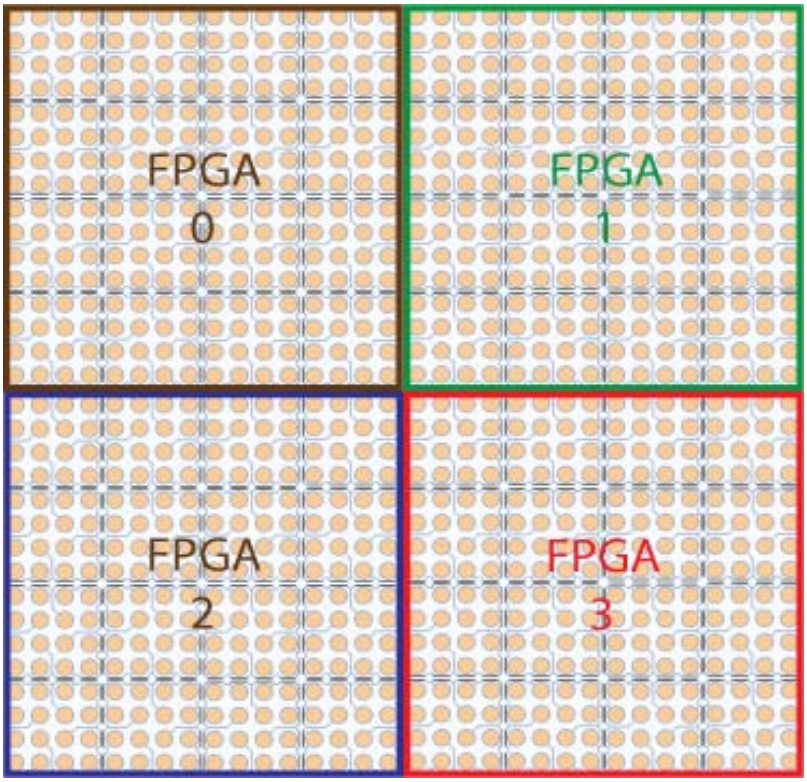

Figure 1-1. 32 x 32 Photo-detector Array Divided Into Four 16 x 16 Arrays

Each ADC device is assigned one SN65LVDS386 differential line receiver which generates single-ended signals from the ADC's LVDS outputs. The SN65LVDS386 devices and FPGAs[0-3] are part of the second stage. Second stage data processing is required to arrange collected image data into a logical order before transmission to the third stage. Data transfer from the second stage to the third stage is achieved using three Aurora 8b/10b serial protocol channels.

The Aurora 8b/10b protocol was developed by Xilinx as an open standard available for implementation to anyone without restrictions. Aurora 8b/10b is a "scalable, lightweight, link-layer protocol," designed for high-speed serial transmissions from point-to-point [2]. This simple packet-based protocol enables FPGAs[1-3] to transmit acquired image data to FPGA[0] with great speed and minimum I/O resources.

Chapter 5 explains the third stage of the IAS. In this last stage, image data from the entire 32 x 32 array is brought together in FPGA[0]. The collected image data is processed into the final output format and transmitted to the data processing application via a PCI Express protocol channel.

PCI Express is a high performance I/O bus used to interconnect devices in communication platforms and applications such as "mobile, desktop, workstation, server, and embedded computing" [3]. The protocol runs on most computer motherboards to provide chip-to-chip interconnection, as well as board-to-board interconnection when connecting to external devices, or in this application, the IAS. Like Aurora, PCI Express is a serial point-to-point, packet-based protocol, however it is equipped with more sophisticated features, such as error handling [3]. PCI Express is used in the IAS because

of its ease of integration with most computing machines, and ability to meet the IAS' data transmission requirements.

To verify the IAS design, the VHDL circuit descriptions required to configure and control all IAS hardware have been written, integrated together, and simulated. In the first stage, this includes VHDL modules to power-up, initialize, configure, and monitor the ADS5281 devices. In the second stage, data capture, data processing, Block RAM, and Aurora 8b/10b TX modules make up the bulk of the digital datapath. In the third stage, Aurora 8b/10b RX, Data Processing, and PCI Express modules are the main components.

The correctness of the IAS design is confirmed by the simulations at the end of each chapter. In Chapter 3, the simulations prove the IAS's compatibility with the ADS5281 analog to digital converter. In Chapters 4 and 5, the simulations prove that image data is processed correctly, and that it passes through the datapath's high-speed serial channels as expected.

1.3.2 Image Acquisition System Block Diagram

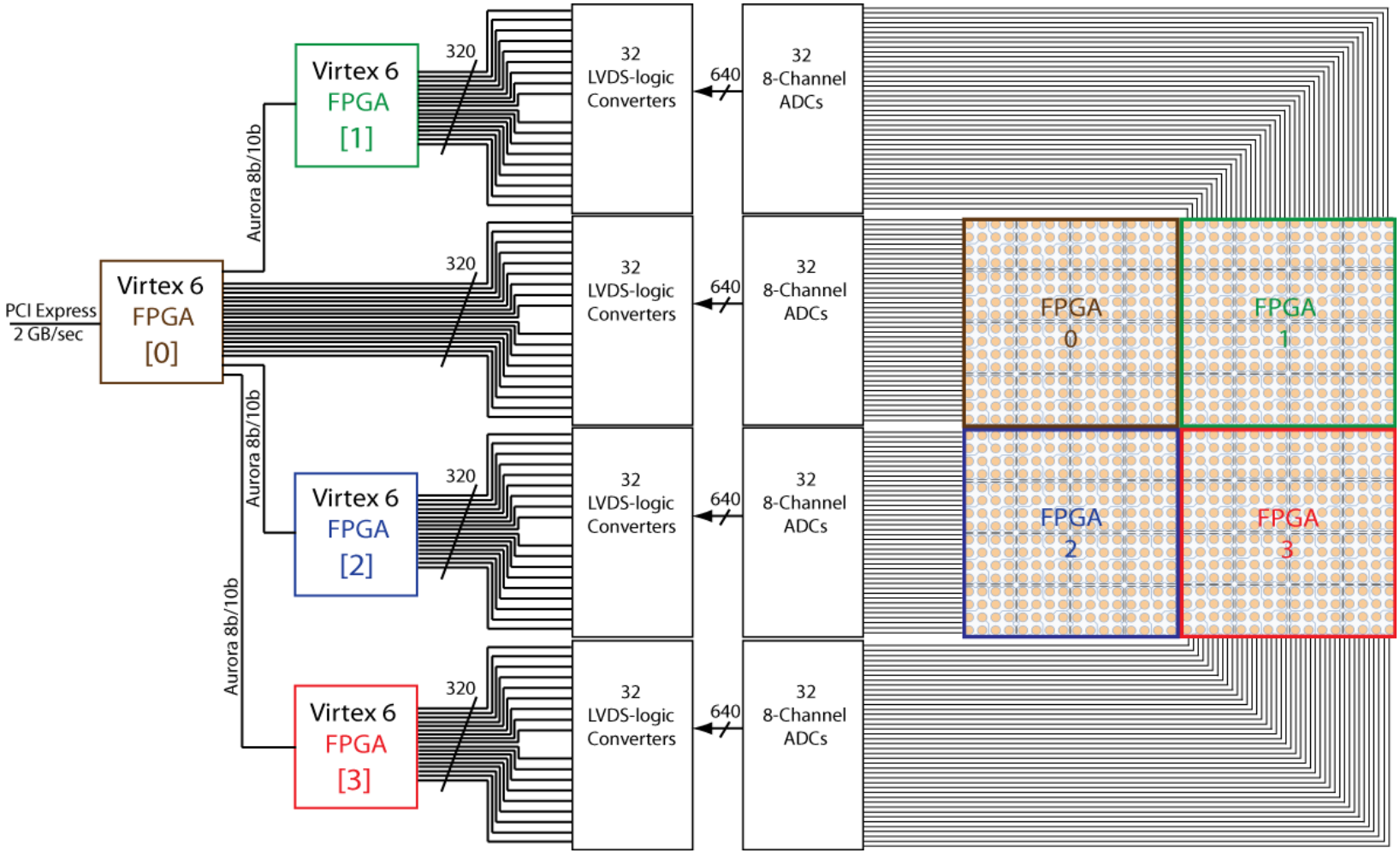

Figure 1-2. Image Acquisition System Architecture Block Diagram

CHAPTER 2

SYSTEM SPECIFICATIONS AND REQUIREMENTS

2.1 Specifications and Requirements

The specifications and requirements for the Image Acquisition System are as follows:

1. The dimensions of the photo-detector array shall be 32 x 32 photo-detectors, spaced 5 mm apart.
2. The photo-detector array shall be constructed using 4 x 4 photo-detector array modules.
3. A 1 kHz Sample Pulse signal shall initiate the capture of 18 samples of the 32 x 32 photo-detector array.
4. All photo-detectors shall be sampled at a frequency of 12 MHz, with 12 bit resolution.
5. The 32 x 32 photo-detector array shall be sampled, digitalized and transmitted to the final destination before the next Sample Pulse period.
6. Data from the 32 x 32 photo-detector array shall be multiplexed into a single data stream and transmitted to the final destination at 2 GB/s.

2.2 Specification Analysis

This section describes, illustrates, and analyzes each IAS specification to assist in understanding the design requirements.

1. The dimensions of the photo-detector array shall be 32 x 32 photo-detectors, spaced 5 mm apart.

A representation of the 32 x 32 array of photo-detectors is presented in Figure 2-1. There are a total of 1,024 photo-detectors in the array. Spacing the photo-detectors 5 mm apart results in a surface area of approximately 160 mm x 160 mm.

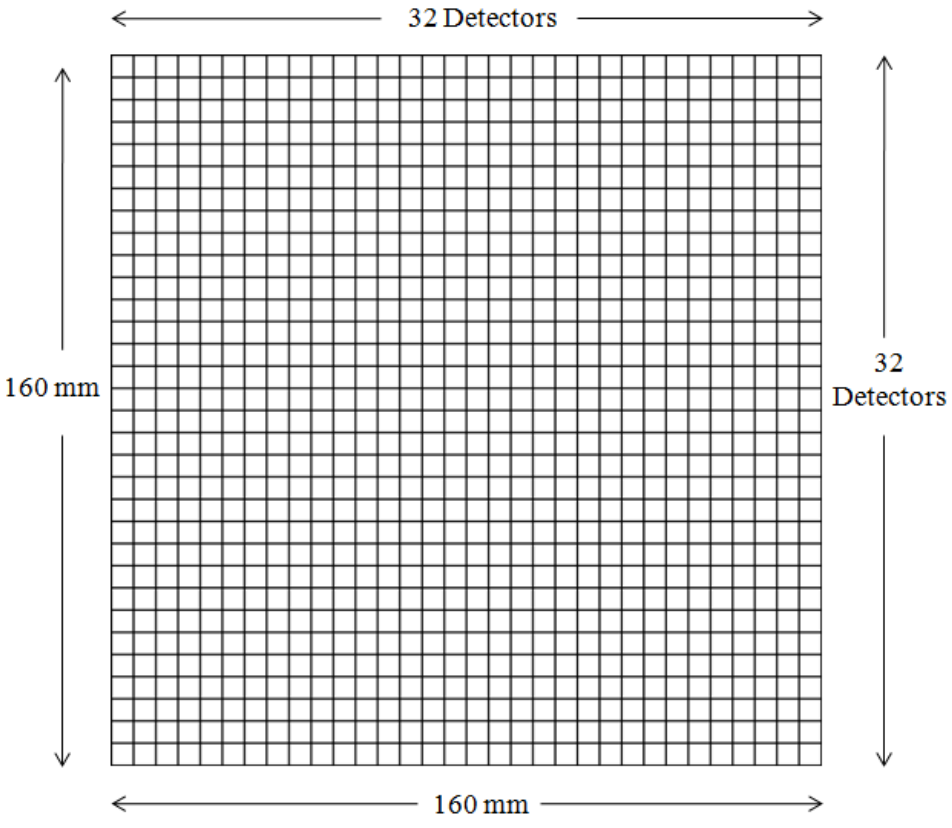

Figure 2-1. Array of 32 x 32 Photo-detectors Spaced 5 mm Apart

2. The photo-detector array shall be constructed using 4 x 4 photo-detector array modules.

The second specification states that the 32 x 32 photo-detector array shown in Figure 2-1 is to be constructed using 4 x 4 photo-detector modules. Sixty-four 4 x 4 modules are required for the entire 32 x 32 array. The 16 photo-detectors in each 4 x 4 module are denoted P0 through P15; note that naming begins in the upper left corner and progress to the right and down. 5 mm spacing (between the centers of each photo-detector) means that each module will have an approximate dimension of 20 mm x 20 mm. Figure 2-2 illustrates this concept.

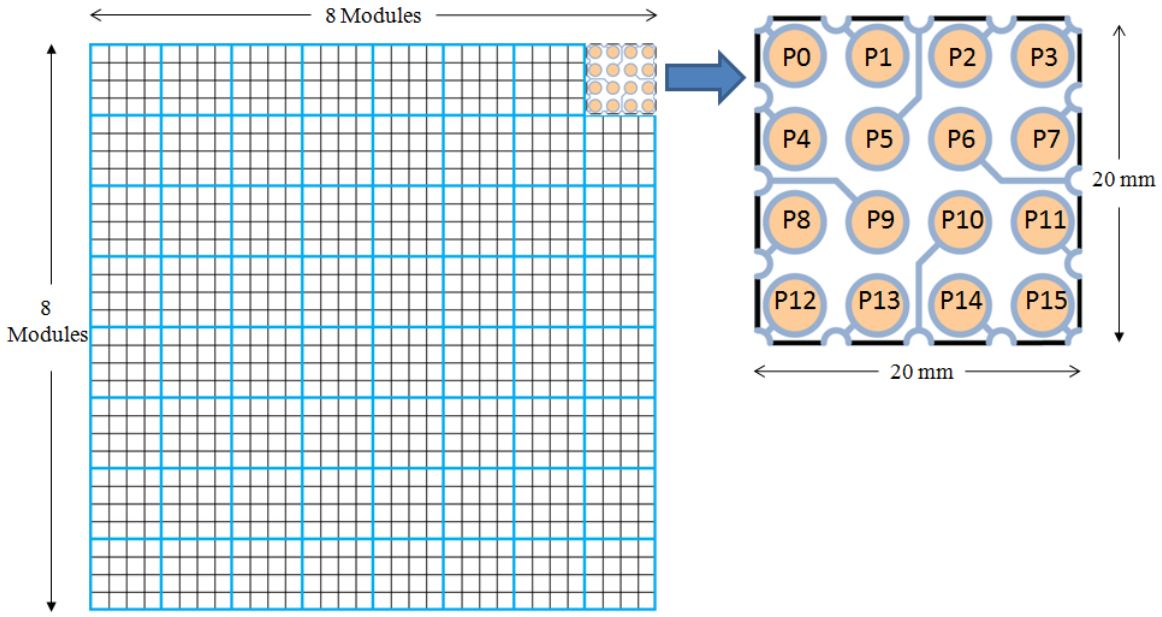

Figure 2-2. Nominal 4 x 4 Photo-detector Array Module

The design of each module is expected to employ two PCB layers. The top layer contains the 16 photo-detectors and an integrated conditioning circuit (photo-detector pre-amp) which converts the incoming light intensity (current) to a differential voltage

signal. The second layer consists of two ADS5281 8-channel analog to digital converters which sample the voltage signals from the 16 photo-detectors. The nominal construction of each 4 x 4 module is illustrated in Figure 2-3.

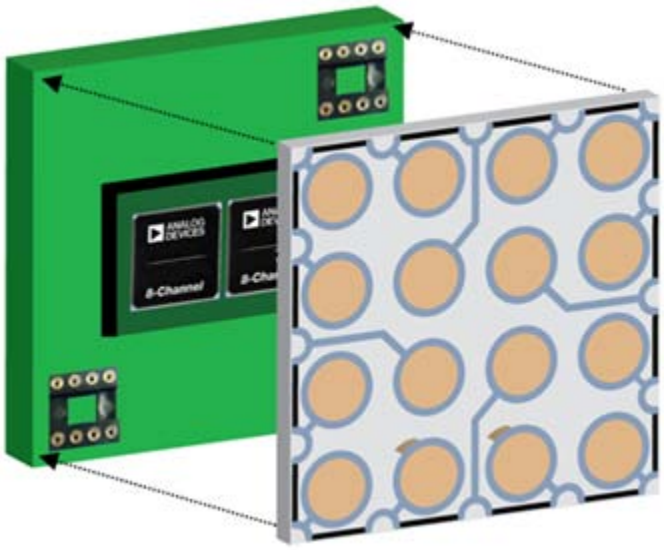

Figure 2-3. 4 x 4 Module Consisting of Two PCB Layers

3. A 1 kHz Sample Pulse signal shall initiate the capture of 18 samples of the 32 x 32 photo-detector array.
4. All photo-detectors shall be sampled at a frequency of 12 MHz, with 12 bit resolution.

The requirements for sampling and data acquisition state that the system shall initiate sampling on a 1 kHz Sample Pulse signal, or every 1 ms. Once sampling has been initiated, the Image Acquisition System shall collect 18 sample frames at a frequency of 12 MHz.

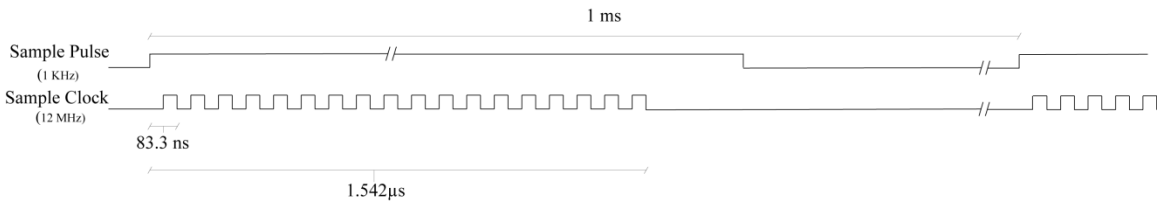

Figure 2-4. 1 kHz Sample Pulse Initiates 18 12 MHz Sample Frames

Figure 2-4 illustrates the timing relationship between the 1 ms Sample Pulse and 18 sample frames. Each Sample Pulse rising edge initiates the capture of 18 sample frames taken at 12 MHz, with 12-bit resolution; sample frames are acquired on the Sample Clock rising edge. The total time to sample 18 frames, $T_{18\ Frames}$, is 1.542 µs.

5. The 32 x 32 photo-detector array shall be sampled, digitalized and transmitted to the final destination before the next Sample Pulse period.
6. Data from the 32 x 32 photo-detector array shall be multiplexed into a single data stream and transmitted to the final destination at 2 GB/s.

During one image acquisition cycle, each of the 1,024 photo-detectors generates eighteen 12-bit samples. The total amount of data generated during one image acquisition cycle is calculated below.

$$\underbrace{\left(\underbrace{(32 * 32)}_{\substack{\text{total} \\ \text{photodetectors}}} * 12\right)}_{\substack{\text{bits in} \\ \text{one frame}}} * \underbrace{18}_{\substack{\text{number} \\ \text{of frames}}} = 221{,}184 \text{ bits/sample}$$

The IAS must output a total of 221,184 bits before the next image acquisition cycle. The last requirement for a single data stream output at 2 GB/s reveals the amount of time the Image Acquisition System will spend transmitting 18 sample frames worth of image data.

Data transmission time, $T_{Transmit\ Data}$, is calculated below.

$$T_{Transmit\ Data} = \frac{\overbrace{221{,}184}^{bits/sample}}{8 * \underbrace{2{,}147{,}483{,}648}_{2GBytes/sec}} = 12.875\ \mu s$$

$bits/byte$

The time required by the IAS to serially transmit one sample's worth of data is 12.875 µs. To determine how much time the IAS has to process the collected data, $T_{Process\ Data}$ is calculated.

$$\begin{aligned} T_{Process\ Data} &= \text{Sample Pulse Period} - T_{18\ Frames} - T_{Transmit\ Data} \\ &= 1\text{ms} - 1.542\mu\text{s} - 12.875\mu\text{s} \\ &= 0.985583\ \text{ms} \end{aligned}$$

With 1 ms between sample pulses, the Image Acquisition System has 0.986 milliseconds to process the collected data.

CHAPTER 3

ADS5281 - ANALOG TO DIGITAL CONVERTER

3.1 Introduction

In the Image Acquisition System, each photo-detector converts the incoming light intensity to a voltage signal with the help of additional signal conditioning circuitry. The resulting continuous analog voltage signal must be sampled at 12 MHz by an ADC which produces a 12-bit value that is digitally equivalent to the sampled voltage level.

In addition to these functional requirements, the suitable analog to digital converter must also fulfill certain physical requirements, as revealed through analysis of specification two. Since each 4 x 4 module has an approximate surface area of 20 mm x 20 mm, the ADC selected for the IAS must also fit within these dimensions.

After much research and careful consideration, the analog to digital converter chosen to sample and digitalize the incoming light signal is the Texas Instruments ADS5281. With 8 channels of simultaneous 12-bit sampling, and an available 9 mm x 9 mm QFN package [4], the ADS5281 was chosen because it satisfies both form and function requirements for the IAS.

This chapter discusses the use of Texas Instruments ADS5281, 8-channel analog to digital converters, in the Image Acquisition System. In addition to providing insight into the device's features, the VHDL circuit description which generates the necessary ADS5281 power-up, initialization, and configuration sequences is simulated and presented at the end of this chapter. A block diagram of the ADS5281 is provided in Figure 3-1 [4].

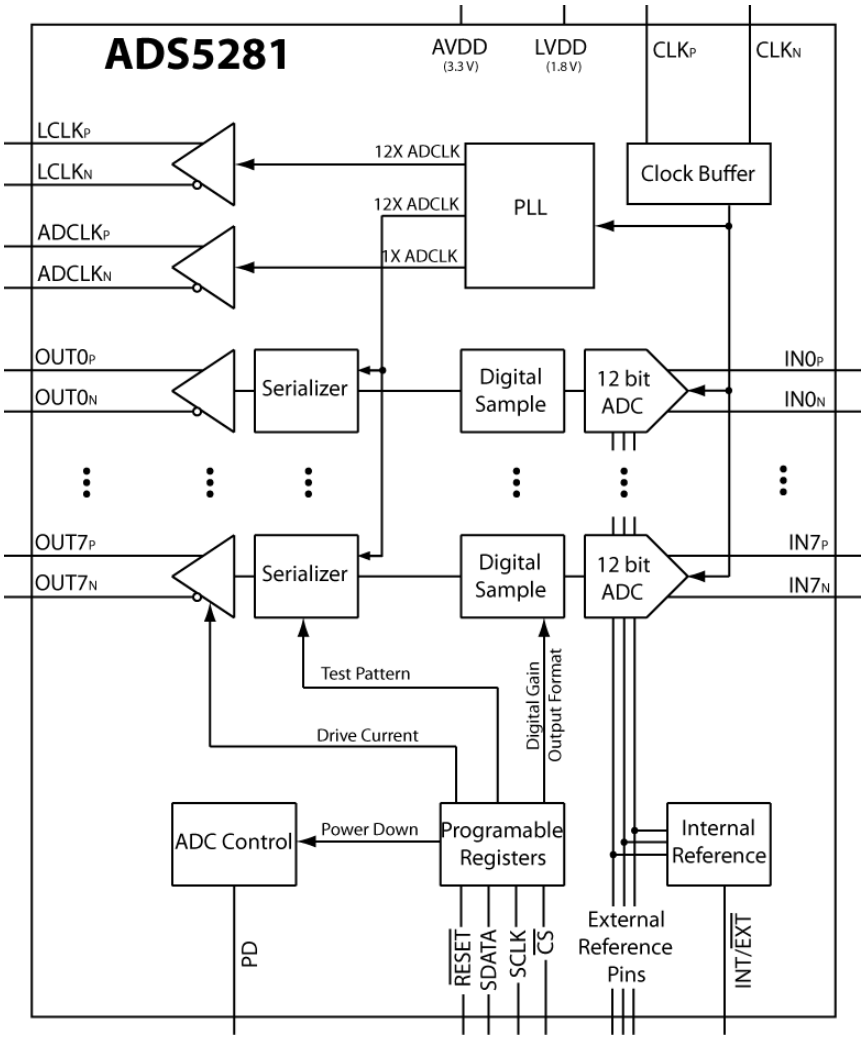

Figure 3-1. ADS5281 Block Diagram

3.2 ADS5281 Digital to Analog Converter

The high-performance, low-power, octal channel ADS5281 offers a variety of programmable features which truly provide an exceptional level of system integration while satisfying the critical ADC requirements [4]. Some of the features most significant to the Image Acquisition System are presented in Table 3-1.

Table 3-1. Summary of ADS5281 Features

Feature	**Description**	**Applicability to Image Acquisition System**
12-Bit Resolution	sample output is 12-bits wide	12-bit resolution satisfies specifications
Octal-Channel	8 channel simultaneously sampling	8 ADCs per ADS5281; 2 per module; 8,192 total
50MSPS	ADS5281 operates from 10 MHz to 50 MHz	ADS5281 Sample Clock is 12 MHz
Low Power	3.3 Volt analog supply, 1.8 Volt digital supply 64mW/channel power dissipation	Each ADS5281 requires: 3.3V AVDD and 1.8V LVDD Each ADS5281 dissipates 512mW of power
Programmable LVDS Output Formats	data output in straight offset, binary, or two's complement	Flexible data output formats allow for different design solutions
	LSB or MSB first	Data output is straight binary, MSB first; LCLK is 90° phase shifted
	LCLK (Bit Clock) Phase	
	Double Data Rate (DDR) or Single Data Rate (SDR)	Single Data Rate reduces complexity of data capture for FPGAs
Programmable Input Clock	single-ended or differential sample clock input	Single-ended sample clock significantly reduces clock tree and I/O requirement for FPGAs

Table 3-1 (continued). Summary of ADS5281 Features

Feature	Description	Applicability to Image Acquisition System
Internal/External Reference	INT/EXT pin controls whether ADS5281 will use internal or external voltage reference	ADS5281s use internal voltage reference. It is expected that all ADS5281 ICs will function using a common external reference voltage
Programmable Power Down Modes	two global power down modes: partial and complete power down PD pin used for power down control	ADS5281s will utilize partial power down mode to reduce power consumption
LVDS Test Patterns	ADS5281 can output a variety of test patterns which replace the normal ADC data output	LVDS test patterns should be used during development to test and verify the data capture interface

The remainder of this chapter describes how the ADS5281 devices interface with the rest of the IAS including: the various clock signals supplied to, and generated by the ADS5281; timing considerations for sampling, reset, power-up and power-down; the serial interface used to initialize and configure the ADS5281 devices; and LVDS-to-logic conversion of all ADS5281 outputs using SN65LVDS386 differential line receivers.

3.3 Photo-detector – ADS5281 Interface

The Image Acquisition System requires a total of 128 8-channel ADS5281 devices to sample and digitize the 1,024 photo-detectors which make up the complete array. From the IAS block diagram in Figure 1-2, it is evident that 32 ADS5281 devices are used to digitize image data for each FPGA.

The interface between the photo-detectors and ADS5281 integrated circuit chips is illustrated in Figure 3-2. Although the photo-detector array is comprised of 4 x 4 modules, each containing two ADCs, it is helpful to depict the photo-detectors in 16 x 16 arrays and their 32 ADC components combined into one block. This is fitting because each FPGA (and appropriate 16 x 16 photo-detector array) operates independently of each other when acquiring image data. Nevertheless, Figure 3-2 preserves the presence of 16 4 x 4 modules – they are labeled M0 through M15. The pair of ADCs which are connected to each module are also identified.

Each photo-detector is physically connected to a distinct ADC channel via differential signal input pins $IN_{[0\text{-}7]P}$ and $IN_{[0\text{-}7]N}$.[1] Photo-detectors P0 to P7 connect to inputs IN_0 to IN_7 of one ADS5281, while photo-detectors P8 to P15 connect to inputs IN_0 to IN_7 of a second ADS5281. This is somewhat exemplified through the 512-bit IN_BUS in Figure 3-2. To ensure that image data is correctly presented to the data processing application, it is essential that the IN_BUS bit-map ordering in Figure 3-3 be maintained for each 16 x 16 array.

[1] Reference Figure 2-1.

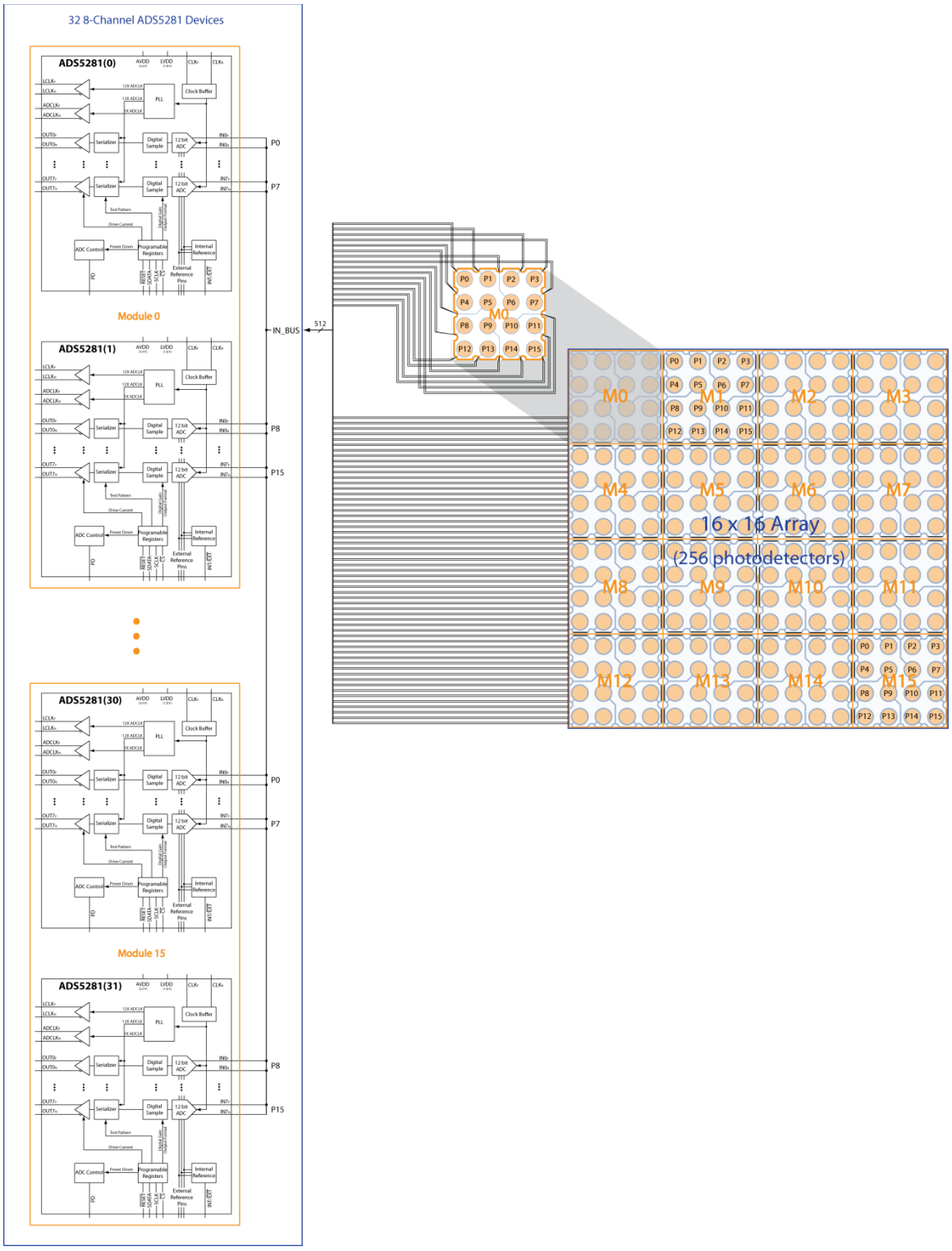

Figure 3-2. Photo-detector – ADS5281 Interface

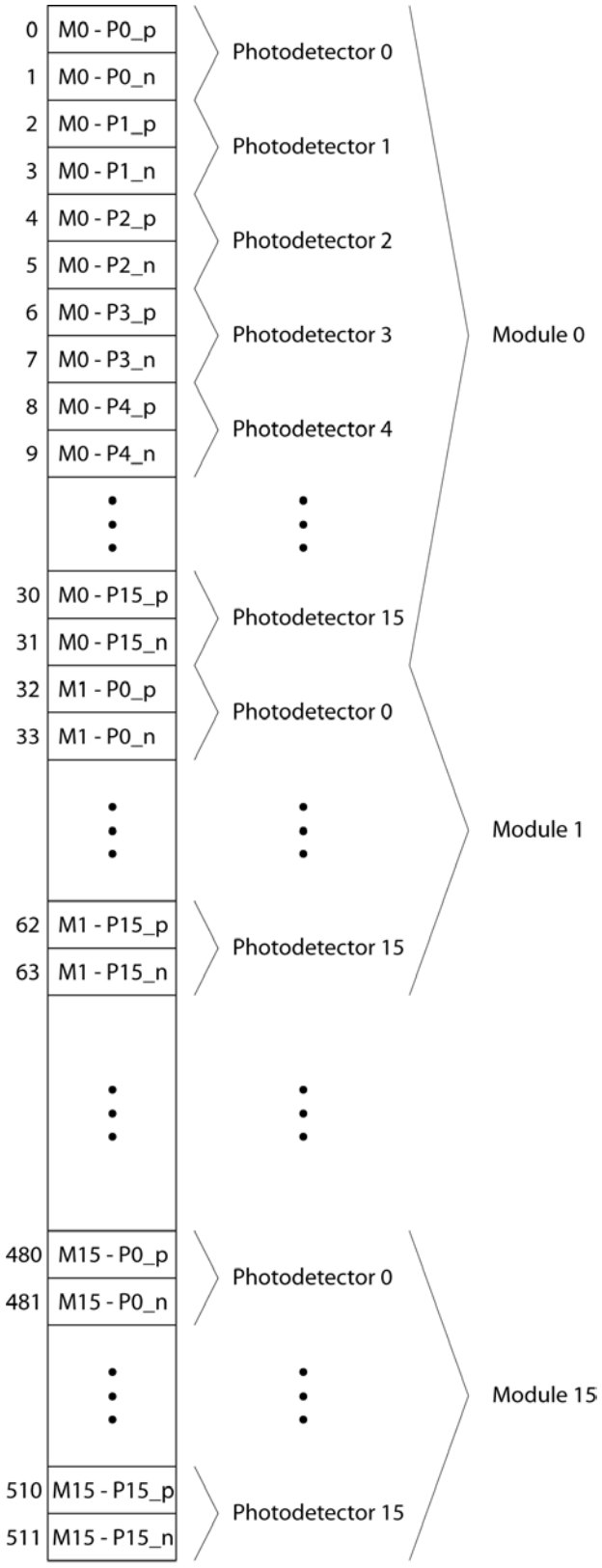

Figure 3-3. Photo-detector – ADS5281 Interface IN_BUS Bit-Ma

3.4 ADS5281 Sample Clock

The ADS5281 Sample Clock signal is generated by FPGA[0], and applied to the CLKP input pin of every ADS5281. As illustrated in Figure 2-4, the Sample Clock signal is used to capture 18 sample frames of image data. However, in order to avoid a clock disruption delay, the Sample Clock is never interrupted.[2] Consequently, the Sample Clock signal shown in Figure 2-4 is a mask of the actual Sample Clock signal applied to the ADCs.

All eight channels of each ADS5281 operate on the Sample Clock input; that is, each Sample Clock rising edge triggers the internal sample and hold circuits of all 8 channels. To ensure that the performance and timing for all 8 channels is identical, each ADC utilizes an internal clock buffer tree network to generate the eight individual sampling clocks for each channel [4]. Since the clock tree functions by matching the clock paths from the input source point to each sampling circuit, an aperture delay – from rising edge of Sample Clock to the actual instant of sampling – is introduced. Nevertheless, the carefully matched clock tree ensures that the aperture delay and jitter are the same for all eight channels [4].

The ADS5281 can be made to operate either in CMOS single-ended (1-wire) clock mode, or differential (2-wire) clock mode. Since sampling is at a relatively low frequency (below 30 MHz), there is no significant advantage of using a differential input clock [4]. Moreover, an additional clock line considerably increases the number of I/O needed per FPGA. The ADS5281 devices are programmed to operate using a single-ended input

[2] Reference section 3.8.

clock which is applied to the CLKP pin; the CLKN pin is kept at 0 VDC (logic level '0'). The CLK inputs are shown at the top of the ADS5281 block diagram in Figure 3-1.

3.5 ADS5281 Data & Frame Clock

Following the sample and hold circuit, data enters the Digital Sample block (in Figure 3-1) where it is assembled according to the LVDS data output format specified during ADS5281 configuration.[3] All ADS5281 ICs are configured for straight offset binary data output, with most significant bit (MSB) first, and 90 degree phase shift between data and data clock. After the Digital Sample block, formatted ADC data passes through a Serializer which generates the final ADC output according to the data rate mode of operation. To reduce the design complexity of data capture at the second stage, ADC data output is set to single data rate (SDR) mode. Keeping in mind that all ADC output are converted to single-ended signals, Figure 3-4 illustrates the SDR interface and timing characteristics [4].

To serialize digital data and facilitate data capture, each ADS5281 employs a phase-locked loop (PLL) circuit which generates a 12x Data Clock and 1x Frame Clock [4]. These signals are illustrated in Figure 3-4. In single data rate mode, the Data Clock has a frequency of 144 MHz, and is output in differential mode on the ADC's LCLKP and LCLKN pins. The Data Clock frequency is calculated below.

$$\text{Data Clock} = \frac{(\text{ADC Resolution}) \text{ X } (\text{Sample Rate})}{(\text{Wire Interface})} = \frac{(12 \text{ bits}) \text{ X } (12 \text{ MHz})}{(1 \text{ SDR})} = \mathbf{144MHz}$$

[3] Reference section 3.13.

The Frame Clock is output in different mode on the ADC's ADCLKP and ADCLKN pins. The Frame Clock has a frequency equal to the 12 MHz Sample Clock, however a propagation delay exists between the two signals.[4] According to Figure 3-4, the rising edge of the Frame Clock indicates the beginning of a new 12-bit sample output. Furthermore, Figure 3-4 shows that Frame Clock and Data are in phase, while Data and Data Clock are phase shifted by 90°.

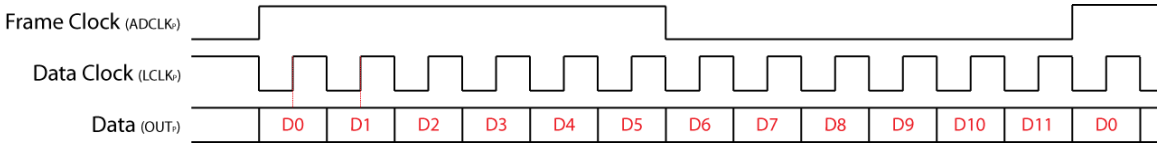

Figure 3-4. Single Data Rate Interface and Timing

To capture data, FPGAs[0-3] use the rising edge of the Frame Clock (ADCLKP) to trigger sampling of Data (OUTP) on the next twelve rising edges of the Data Clock (LCLKP) [4].[5]

3.6 ADS5281 Sample-to-Data Delay

Sample-to-data delay is defined as the time from first Sample Clock rising edge to first serial data bit output. Sample-to-data is a one-time delay, after which serial data on all 8 ADC channels is continuous. This is to say, that after an initial sample-to-data time, each subsequent rising edge of the Frame Clock denotes the beginning of a 12-bit sample.

[4] Reference section 3.6.

[5] Reference section 4.3.

The sample-to-data delay then depends on two ADS5281 characteristics – data latency and clock propagation delay. Data latency is defined as the time from first Sample Clock rising edge to the time first sample data is ready for output. Clock propagation delay is defined as the time from Sample Clock rising edge to Frame Clock (ADCLK) rising edge. Sample-to-data delay is illustrated near the bottom of the ADS5281 Timing Diagram in Figure 3-5; its equation is given below.

$$T_{\text{Sample-to-Data}} = \text{Data Latency} + \text{Clock Propagation Delay}$$

The ADS5281 also has an aperture delay which is present to ensure that sampling on all eight channels occurs simultaneously [4].[6] Aperture delay is the time from Sample Clock rising edge, to the actual time of sampling. It is identified in Figure 3-5 as the short time before each analog input sampling point. The existence of an aperture delay does not, however, influence sample-to-data time, since aperture delay is part of data latency. Similarly, the 12 MHz sampling frequency is unaffected since this delay is present in every sampling instance.

According to the ADS5281 datasheet, the ADC suffers a data latency of 12 Sample Clock cycles, and an average and maximum clock propagation delay of 12 ns and 14.1 ns, respectively [4].[7] Since the sampling frequency is 12 MHz, the Sample Clock has a period (and cycle) of 83.3 ns. As a result, ADS5281 data latency is 1 µs (12 x 83.3 ns). The maximum sample-to-data delay is calculated next.

$$T_{\text{Sample-to-Data}} = 1\mu s + 14.1ns = 1.014\mu s$$

[6] Aperture delay is 1.5 ns to 4.5 ns.

[7] Clock propagation delay is between 10 ns and 14.1 ns.

For the IAS, the existence of a sample-to-data delay means that data capture is slightly postponed. What is more, LVDS-to-logic conversion via the SN65LVDS386 delays data acquisition further. The total time for image data to reach FPGAs[0-3] is calculated later in this chapter.

3.7 ADS5281 Timing Diagram

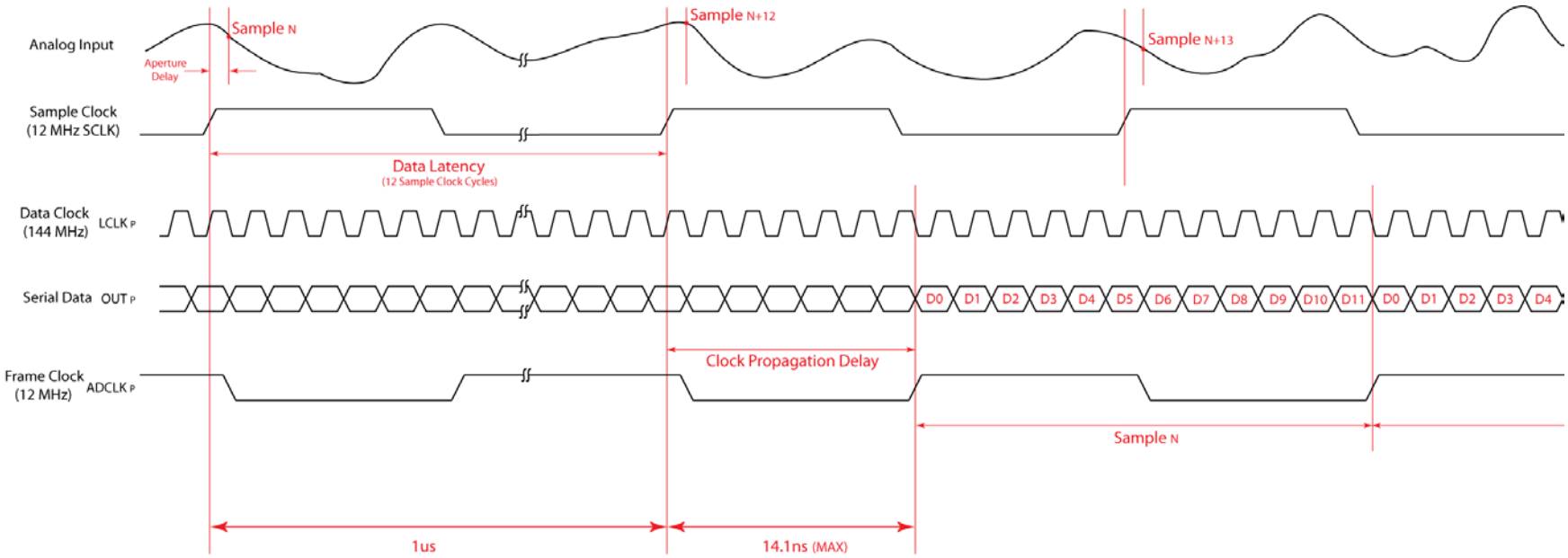

Figure 3-5. ADS5281 Timing Diagram

3.8 ADS5281 Power Down & Wake-Up

In an effort to reduce the Image Acquisition System's power consumption, all ADS5281 devices are powered down after image data is captured by FPGAs[0-3]. The ADS5281 offers two power down modes – complete power down and partial power down – each possesses a unique wake-up time (T_{Wake}). Wake-up time is the time it takes the ADS5281 to being producing valid data after coming out of power down mode, or if the Sample Clock is interrupted. Table 3-2 summarizes the ADS5281 wake-up time characteristics [4].

Table 3-2. ADS5281 Power Down and Wake-Up Time

Power Down Mode	**T_{WAKE} (µs)**
Complete Power Down	50
Partial Power Down	2
Sample Clock Interruption	40

The ADS5281 is configured for partial power down mode by writing to the power down modes configuration register.[8] In this configuration, all ADCs will require 2 µs to completely wake up and begin producing valid data. However, if the input Sample Clock is stopped and restarted, the ADS5281 will experience a 40 µs delay. For this reason, once the Sample Clock is started it is never interrupted.

The ADS5281 is powered down by asserting the external power down (PD) pin. The ADC is fully powered down 1 µs after the rising edge of the PD signal. The PD signal remains asserted until 2 µs before the next Sample Pulse. Figure 3-6 illustrates the timing characteristics of the power down signal [4].

[8] Reference section 3.13.

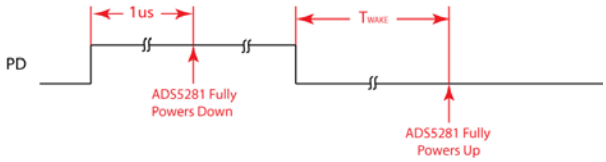

Figure 3-6. ADS5281 Power Down (PD) Timing

3.8.1 ADS5281: Initial Power-Up Timing Diagram

The recommended power-up and reset sequence for the ADS5281 is illustrated in Figure 3-7; power-up timing constraints are presented in Table 3-3 [4]. The ADC devices are powered up when 3.3 Volts and 1.8 Volts are applied to the AVDD and LVDD inputs, respectively. The execution of the power-up sequence is the responsibility of FPGA[0], and ends when the ADS5281s are ready for data conversion [4]. The power-up sequence takes approximately 20 ms.

Table 3-3. Power-Up Timing Constraints

Timing Parameter	Timing Constraint
T1	10 µs - 50 ms
T2	> 10 ms
T3	> 100 ns
T4	> 100 ns
T5	> 10 ms
T6	> 100 µs

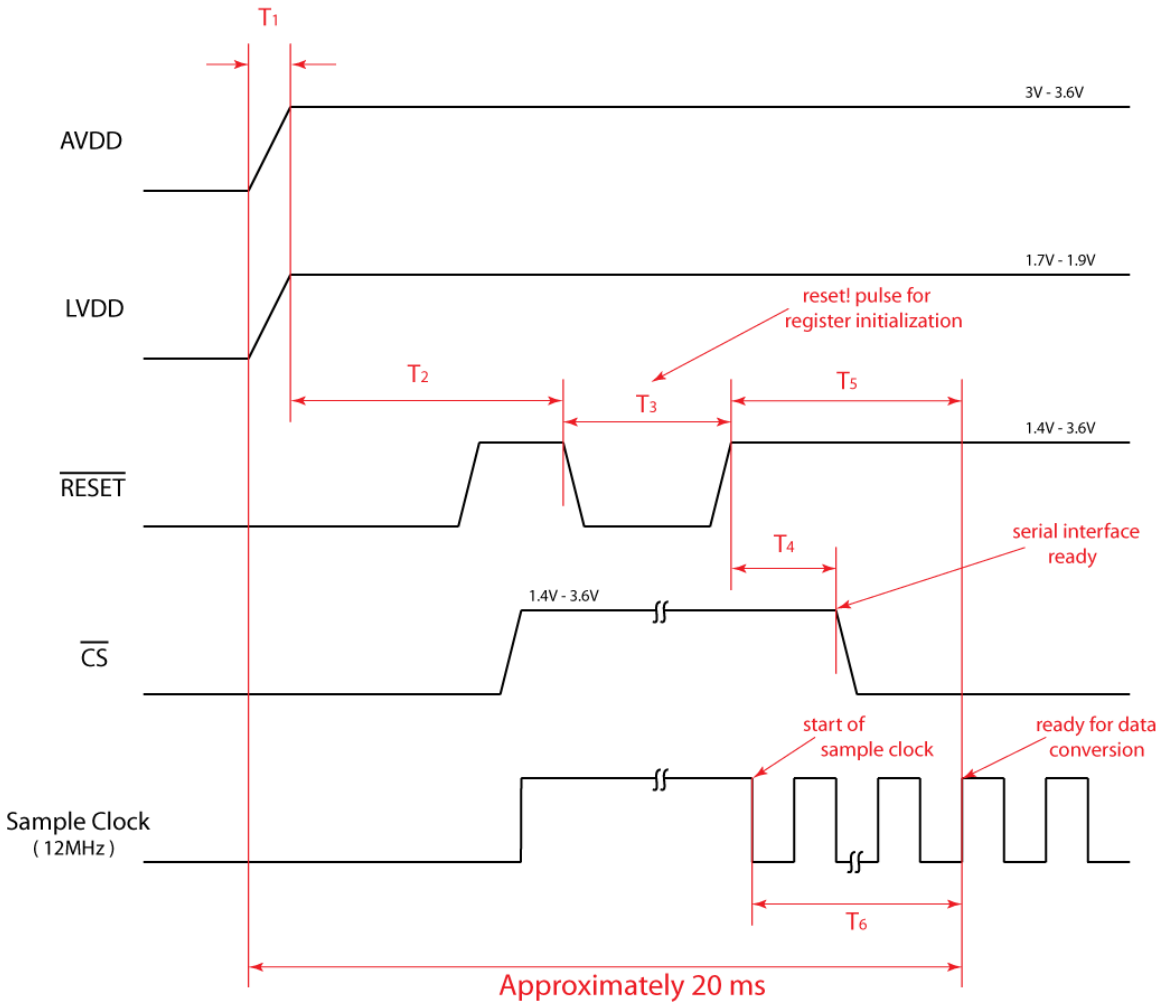

Figure 3-7. Initial Power-Up Timing Diagram

3.9 Initial Power-Up Simulation

The simulation waveform in Figure 3-8 was generated using the iSIM simulator available with the Xilinx ISE Design Suite 13.3 software. The VHDL hardware design is implemented in FPGA[0] to perform the initial power-up sequence illustrated in Figure 3-7; the timing constraints in Table 3-3 are of course maintained. The VHDL block diagram for the hardware design is presented in Figure 3-14.

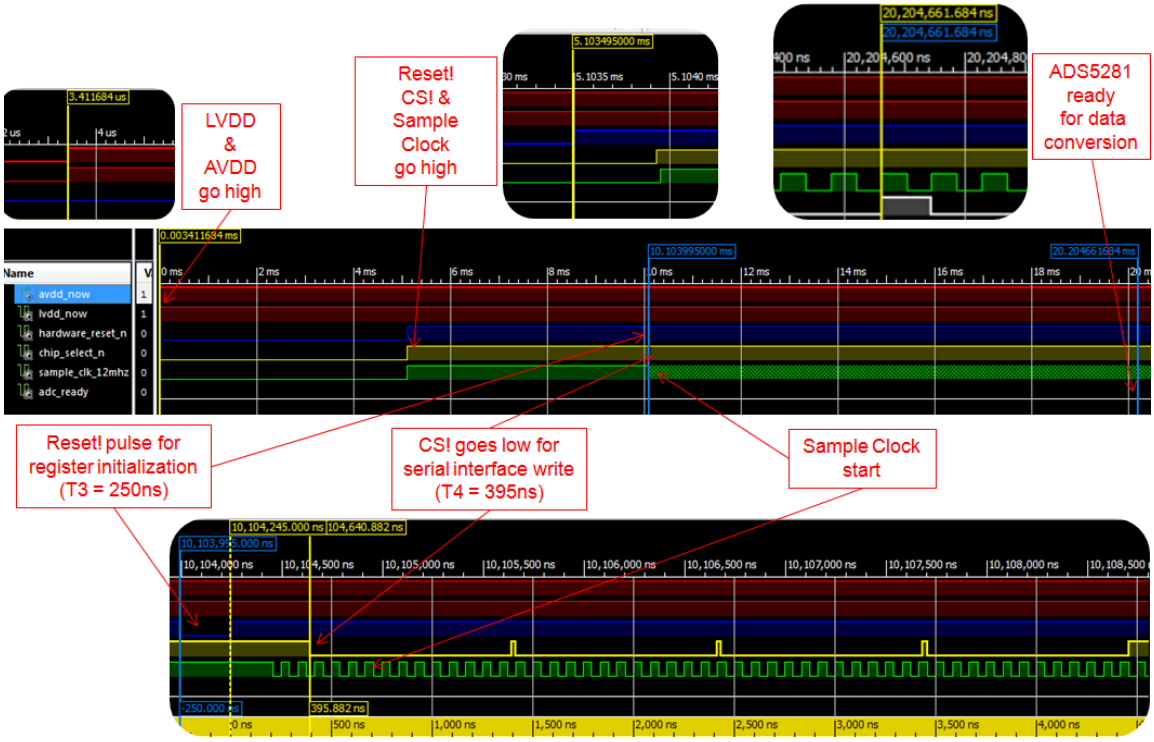

Figure 3-8. Initial Power-Up Simulation

3.10 ADS5281 Reset and Register Initialization

After power-up, the ADS5281's internal registers must be initialized to their respective default values. Register initialization can be done in one of two ways: [4]

1. Hardware reset – by applying a low-going pulse on the $\overline{\text{RESET}}$ pin.
2. Software reset – through the serial interface by setting the RST bit high.

During initial power-up of the IAS, register initialization is carried out by FPGA[0] via the external hardware reset pin. In Figure 3-7, the low-going reset pulse is identified near the center of the image, and shown constrained by T3. After all registers have been initialized to their default values, the initialization registers in Table 3-4 are written into with the appropriate register data. Writing the specified data to these two registers is necessary in order to reconfigure the ADS5281s for ac-coupled analog input; this process must happen after every hardware or software reset operation [4].

Table 3-4. ADS5281 Initialization Registers

Initialization Register	Register Address	Register Data
Initialization Register 2	0x01	0x0010
Initialization Register 5	0xE2	0x00C0

ADS5281 initialization (and configuration) registers are accessed through the serial interface which is described in the next section. In Figure 3-7, the falling edge of the $\overline{\text{CS}}$ signal marks the beginning of the serial interface register write procedure to all ADS5281 devices, simultaneous. During this procedure, FPGA[0] writes to the two initialization registers listed in Table 3-4, immediately followed by the configuration registers described in section 3.13.

3.11 ADS5281 Serial Interface

The serial interface information in this section, and the next, is derived from the Texas Instruments ADS5281 Datasheet [4]. Access to the ADS5281 initialization and configuration registers is achieved through the serial interface formed by three ADS5281 pins:

- $\overline{CS}$ – active low chip select. Falling and rising edges indicate the start and end of a new serial sequence, respectively.
- SCLK – serial interface clock. The serial interface can work with SCLK frequencies in the range 10 Hz – 20 MHz. For the IAS, the interface operates using a 12 MHz SCLK.
- SDATA – serial interface data. Each SDATA word contains 24-bits; excess bits are ignored. As illustrated in Figure 3-9, the first 8-bits contain the register address, and the remaining 16-bit form the register data.

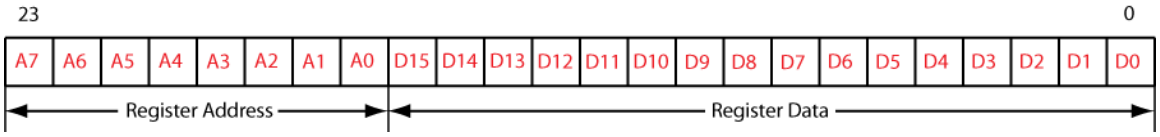

Figure 3-9. ADS5281 Serial Interface SDATA Word

When writing to the serial interface, the sequence is as follows.

1. $\overline{CS}$ is brought low, enabling a serial shift of bits into the ADS5281.
2. SDATA is latched every rising edge of SCLK.
3. SDATA is loaded into the addressed register on every 24th SCLK rising edge.

4. $\overline{CS}$ is brought high, indicating the end of a serial sequence. Multiple 24-bit SDATA words can be loaded within a single active chip select pulse.[9]

The serial interface timing diagram is presented in Figure 3-10. Serial interface timing constraints are listed in Table 3-5.

[9] Each ADS5281 register is written to within one active $\overline{CS}$ pulse. Reference Figure 3-11.

3.12 ADS528 Serial Interface Timing

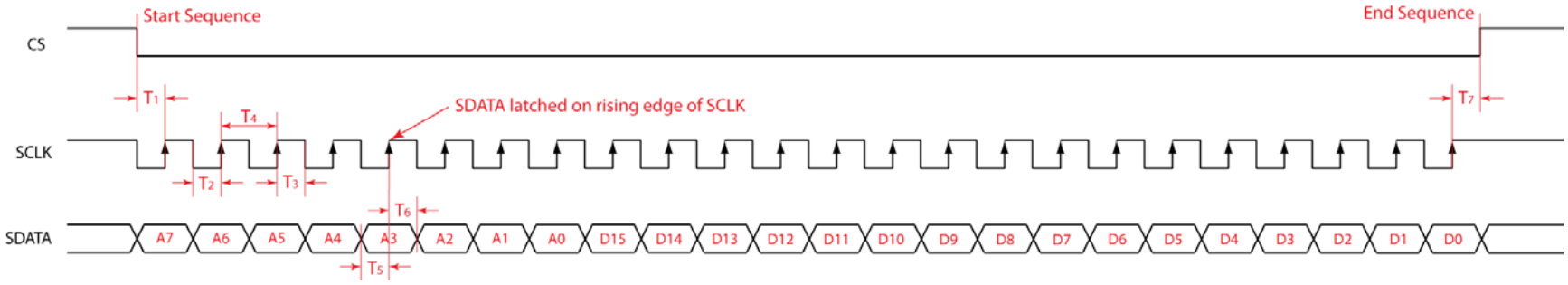

Figure 3-10. Serial Interface Timing Diagram

Table 3-5. Serial Interface Timing Constraints

Parameter	Description	Minimum
T1	CS fall to SCLK rise	8 ns
T2	SCLK low time	20 ns
T3	SCLK high time	20 ns
T4	SCLK period	50 ns
T5	SDATA setup time	5 ns
T6	SDATA hold time	5 ns
T7	last SCLK rise to CS rise	8 ns

3.13 ADS5281 Configuration Registers

To configure the ADS5281 for application in the IAS, FPGA[0] must write into the configuration registers specified in Table 3-6 through the serial interface. This occurs immediately following reset and register initialization. The power-up sequence up to this point is as follows:

1. Power is applied to ADS5281 devices – AVDD and LVDD signals go high.
2. Hardware reset operation – low-going pulse on external $\overline{\text{RESET}}$ pin to initialize registers to their default values.
3. Register initialization – overwrite initialization registers 2 and 5 (through the serial interface) to configure ADS5281s for ac-coupled analog input.

The ADS5281s can now be configured for the appropriate mode of operation through the serial interface. By writing to the Power Down Modes and Data Output Format registers in Table 3-6, the device is configured to utilize some of the key features identified in Table 3-1.

Table 3-6. ADS5281 Configuration Registers

Configuration Register	Description	Register Address	Register Data
Self-clearing Software Reset	resets all internal registers to default values	0x00	0x0001
Power Down Modes	global partial power down mode, PD pin configured for partial power down mode	0x0F	0x0500
Data Output Format	binary, single data rate (SDR), MSB first, LCLK phase 90°	0x46	0x8218

Data written to the Power Down Modes register instructs the ADS5281 to enter partial power down mode when the PD pin is asserted. The Data Output Format register configures the ADC for binary output in single data rate, with most significant bit first. Additionally, to allow the Data Clock to be used for data capture, Data Clock and Data outputs are phase shifted by 90 degrees [4]. These are the only two configuration registers written to during the initial power-up sequence.[10]

The Self-clearing Software Reset register is not written to during the initial power-up sequence. It is only included in Table 3-6 because the IAS is capable of writing to it as an alternate way to initialize the ADS5281 registers. By setting the RST bit (LSB) high, the ADS5281 initializes its internal registers to the respective default values, and then self-resets the RST bit low. In this case, the external $\overline{\text{RESET}}$ pin must remain high. As stated in section 3.10, a software reset operation must be followed by writing to the initialization registers in Table 3-4, and then the two configuration registers described in this section. The software reset operation is not used during normal system operation.

[10] Reference Figure 3-11.

In the future, the IAS may access additional configuration registers not listed in Table 3-6. Among these are the LVDS Test Pattern registers, and the Clock, Reference, and Data Output Modes register. The LVDS Test Pattern registers may be used during testing to ensure that data capture is accurate and complete. The Clock, Reference, and Data Output Modes register will likely become necessary for the IAS's operation, since the ADS5281s will most certainly use external reference voltages to improve gain matching across the 128 devices [4].

3.14 Register Initialization and Configuration Simulation

The waveform presented in Figure 3-11 verifies that the appropriate initialization and configuration registers are written to through the serial interface. The top waveform shows the entire initialization and configuration sequence; the middle and lower waveforms are zoomed-in views. The chip select signal in the upper-most waveform corresponds to the chip select signal near the bottom of Figure 3-8. The chip select falling edge marks the beginning of serial interface write into the two initialization registers from Table 3-4, and two configuration registers described in the previous section.

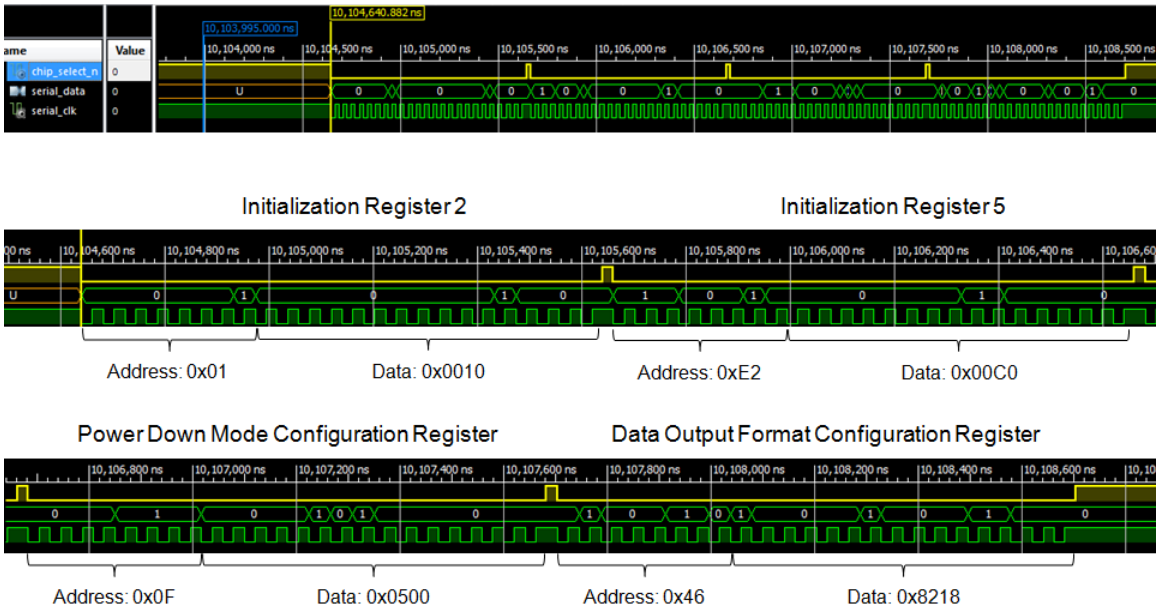

Figure 3-11. Register Initialization and Configuration Simulation

3.15 ADS5281 – FPAG[0] Interface

3.15.1 Control Signals

All 128 ADS5281 devices are initialized, configured, and managed by control signals generated by three modules existing in FPGA[0]. These three modules are identified in Figure 3-13 as MASTER CONTROL, ADS5281 POWER-UP, and ADS5281 INITIALIZE. The signals generated by these three modules are common to all ADS5281 devices.

Each module is responsible for a specific set of control signals. The ADS5281 POWER-UP module starts the IAS by carrying out the recommended initial power-up and reset sequence in Figure 3-7. Immediately following the reset cycle, the ADS5281 INITIALIZE module is activated to initialize the internal ADS5281 registers and configure the devices via the serial interface. The simulation waveforms in Figures 3-8 and 3-11 present the operations performed by the ADS5281 POWER-UP and ADS5281 INITIALIZE modules, respectively. The MASTER CONTROL module is at all times operational, however with respect to the ADS5281 devices, it is only responsible for generating the 12 MHz Sample Clock, and Power Down signal.

The three modules mentioned above makeup the ADS5281 – FPGA[0] interface. The modules responsible for capturing data in FPGAs[0-3] contain additional signals which are connected to the ADS5281 devices, but which are not part of the ADS5281 – FPGA[0] Interface. These modules are included in Figure 3-13 for completeness, but they are part of the second stage and will be discussed in the next chapter.

3.15.2 Differential ADS5281 Outputs

All ADS5281 outputs are low voltage differential signals which must be converted to single-ended logic signals prior to FPGA input. Each ADS5281 generates 8 differential data signals, and two differential clock signals. Since all ADS5281 outputs must pass through the LVDS-to-logic converter stage, the propagation delay of the SN65LVDS386 device does not affect signal synchronization and in turn data capture. What should be considered, however, is channel-to-channel output skew and part-to-part skew of the converter. Specifically, the converter must comply with skew requirements, support a 144 MHz data rate, and contain at least 10 simultaneous channels. The 16-channel Texas Instruments SN65LVDS386 differential line receiver meets all these requirements, and is selected for the IAS.

Each ADS5281 is designated one SN65LVDS386 device; only 10 of the 16 channels are utilized. A total of 128 SN65LVDS386's are required for the IAS. Since part-to-part skew for the SN65LVDS386 is less than 1 ns, [5] an acceptable alternative may be to utilize all 16 channels of only 20 LVDS-to-logic converters (20 devices x 16 channels = 320 differential signals). This alternative is recognized, but reject for the IAS in order to eliminate the possibility of signal synchronization errors, and simplify implementation and development in the future. The chosen configuration allows each SN65LVDS386 converter to process signals generated by a single ADS5281, thus operating on signals from only one clock domain.

The functional block diagram for the SN65LVDS386 is illustrated in Figure 3-12. Enable port pins ENA, ENB, ENC, and END each enable four channels; only ports A, B, and C are enabled for the IAS. The device operates from a single 3.3 Volt power supply

applied to six separate inputs [5]. In Figure 3-13, image data on the OUT_BUS, and data clock signals on the LCLK_BUS and ADCLK_BUS, are converted to single-ended signals in the subsequent block of 32 SN65LVDS386 devices. ADS5281 data outputs are routed to SN65LVDS386 ports A and B; ADS5281 data and frame clocks are routed to channels 1 and 2, of port C.

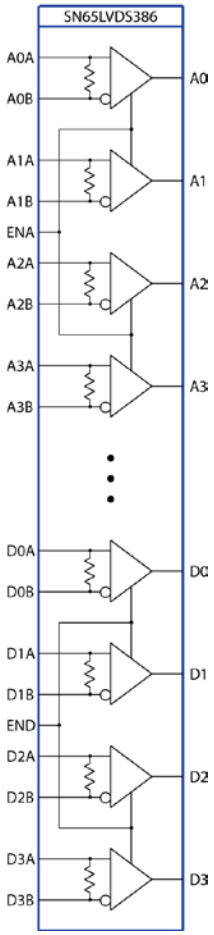

Figure 3-12. SN65LVDS386 Differential Line Receiver Functional Block Diagram

According to its datasheet, the SN65LVDS386 introduces approximately 2.6 ns of propagation delay, 100 ps of output skew, and less than 1 ns of part-to-part skew [5]. To verify that these characteristics will not compromise image data, the worst case scenario

must be examined. The ADS5281 Data Clock (LCLK) is the fastest signal passing through the SN65LVDS386 converter, and therefore represents the worst case. At 144 MHz, the LCLK has a period of 6.9 ns, and pulse width of 3.45 ns. Consequently, skew close to, or greater than 3.45 ns may result in sampling data before the signal is stable. Since output skew and part-to-part skew add up to less than 1.1 ns, the SN65LVDS386 satisfies the requirements for LVDS-to-logic conversion.

The time it takes image data to arrive at the FPGA inputs, $T_{\text{Data to FPGA}}$, can now be calculated. This time constant takes into account the time to sample 18 frames, the one-time sample-to-data delay, and the propagation delay characteristics of the SN65LVDS386.

$$\begin{aligned} T_{\text{Data to FPGA}} &= T_{\text{18 Frames}} + T_{\text{Sample-to-Data}} + \text{SN65LVDS386}_{\text{Propagation Delay}} \\ &= 1.5\mu s + 1.014\mu s + 2.6ns \\ &= 2.517\mu s \end{aligned}$$

Accordingly, the time that the IAS has to process the collected data ($T_{\text{Process Data}}$) is recalculated.

$$\begin{aligned} T_{\text{Process Data}} &= \text{Sample Pulse Period} - T_{\text{Data to FPGA}} - T_{\text{Transmit Data}} \\ &= 1ms - 2.517\mu s - 12.875\mu s \\ &= 0.984608\,ms \end{aligned}$$

The Image Acquisition System has .985 milliseconds to process all 18 sample frames of data before the next sample pulse arrives and a new set of data is captured for transmission to the data processing application.

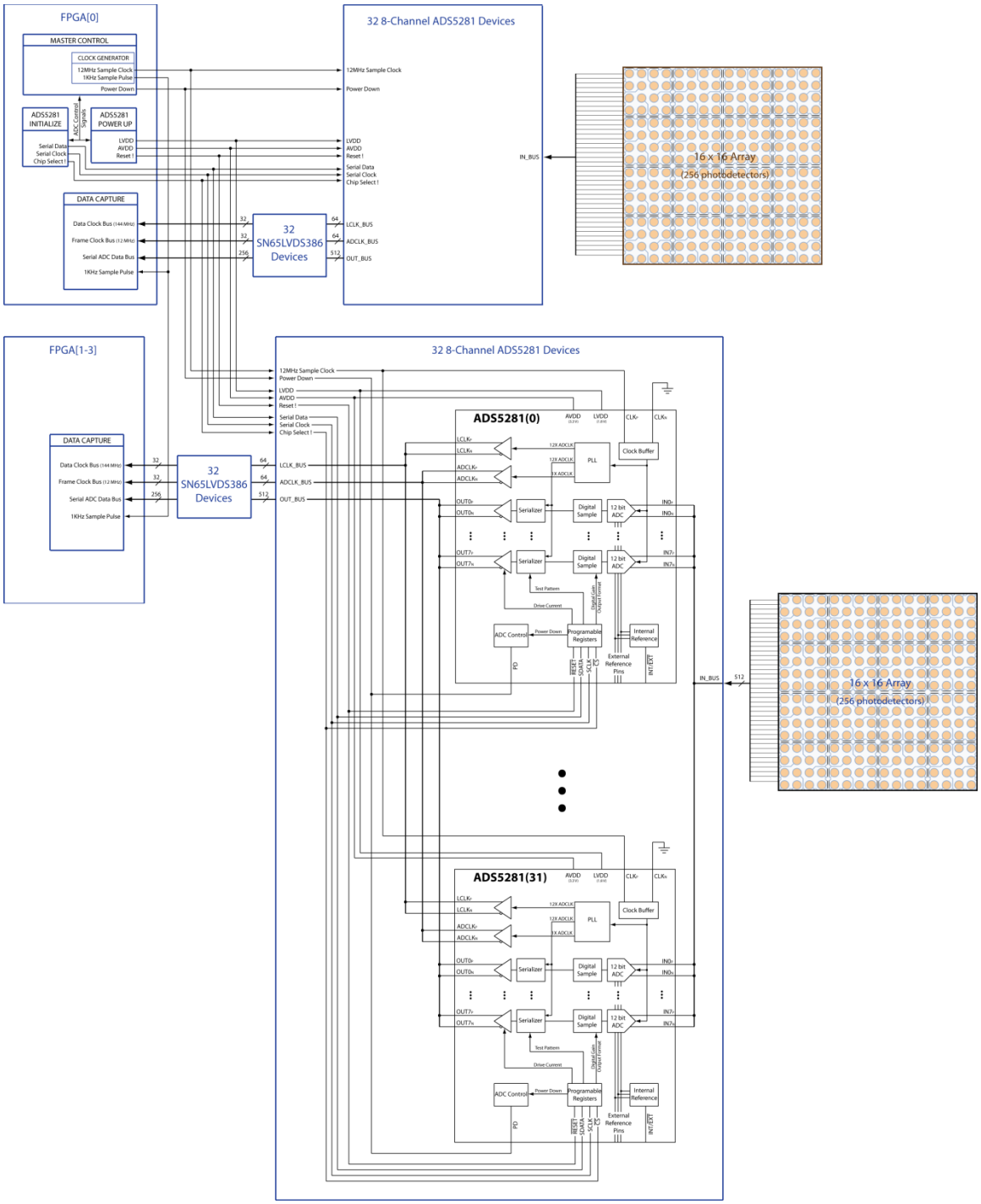

Figure 3-13. ADS5281 – FPGA[0] Interface

3.16 ADS5281 – FPGA[0] Interface Modules

The ADS5281 – FPGA[0] interface is made up three VHDL modules – MASTER CONTROL, ADS5281 POWER UP, and ADS5281 INITIALIZE. These modules are illustrated in Figure 3-14. In Figure 3-13, these control signals are simply labeled, ADC Control Signals. Each module contains one finite state machine which uses the identified control signals to communicate with the other ADS5281 – FPGA[0] interface modules. The three modules work together to ready the ADS5281 devices, and then maintain and monitor their function. The interactions between these modules and the hardware required for each module to perform its essential functions, is the subject of this section.

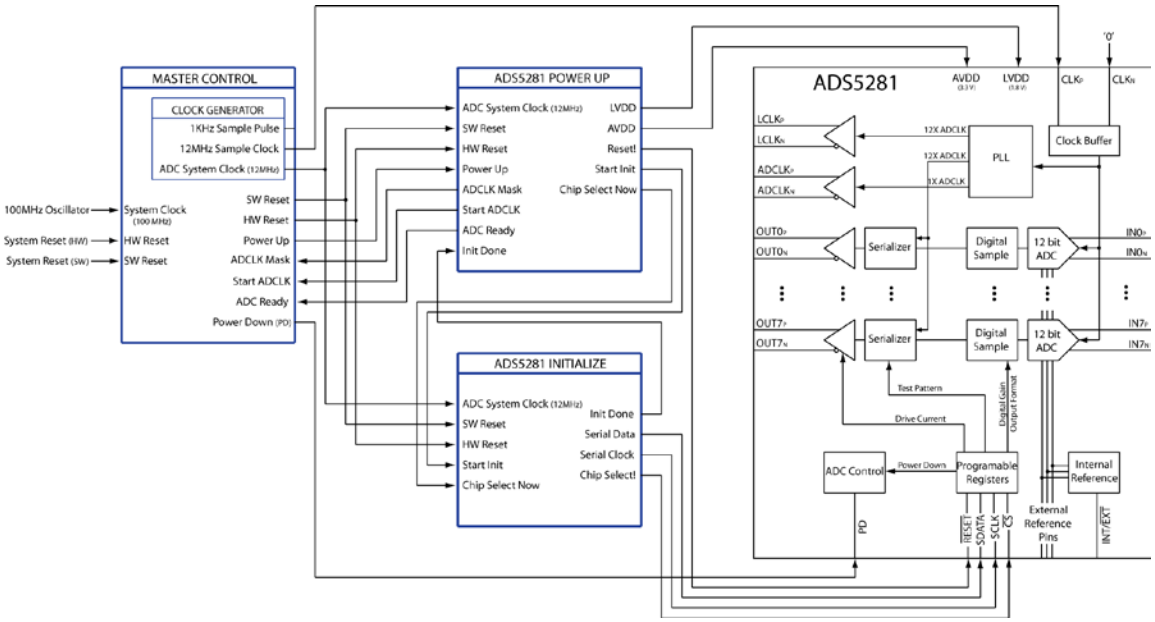

Figure 3-14. ADS5281 – FPGA[0] Interface Modules

3.16.1 MASTER CONTROL Module

The MASTER CONTROL module is undoubtedly the principal synchronizing component in the IAS. Internally, it is made up of four major hardware elements – a Master Control finite state machine, a Clock Generator IP Core, a Sample Clock Generator process, and a Sample Pulse Generator process. In the ADS5281 – FPGA[0] interface, only the Clock Generator, Sample Clock Generator process, and parts of the Master Control state machine are relevant. Only the sections applicable to the ADS5281 devices are discussed in this chapter.

The Master Control finite state machine starts as soon as all FPGA clock signals (generated by the Clock Generator) are stable. As its first task, it initiates the ADS5281 power-up and initialization sequence by asserting the Power Up signal in Figure 3-14. The Master Control state machine then goes on to cyclically keep track of ADS5281 sample latency, monitor ADS5281 sample frame outputs, and generate the ADS5281 Power Down signal. Lastly, the Master Control state machine is also responsible for generating the hardware and software reset signals, HW Reset and SW Reset, to the other ADS5281 – FPGA[0] interface modules whenever they are appropriate, or required by the IAS.[11] The Master Control finite state machine is illustrated in Figure 3-15.

The Clock Generator hardware is a Xilinx Core Generator IP Core, created using the LogiCORE Cocking Wizard Graphical User Interface (GUI) available in the ISE design environment. The Clock Generator supplies the IAS with the essential 12 MHz clock frequency, among others which will be discussed later. The 100 MHz System Clock signal in Figure 3-14 supplies the primary input clock frequency which is used to

[11] The Master Control state machine is also responsible for initiating second stage data capture.

generate all other clock frequencies in the system.[12] The 12 MHz clock is used by the Sample Clock Generator process to produce the Sample Clock signal which is applied to all ADS5281 devices. The 12 MHz clock also doubles as the ADC System Clock signal which is used to synchronize the ADS5281 – FPGA[0] interface modules.

The last two hardware components in the MASTER CONTROL module are the Sample Clock Generator and Sample Pulse Generator processes. From the ADS5281 initial power-up timing diagram in Figure 3-7, it is evident that the ADCs require a specific Sample Clock sequence in order to properly power-up. To fulfill this requirement, the Sample Clock Generator process masks the 12 MHz clock signal generated by the Clock Generator hardware. The Sample Pulse Generator process also modifies the 12 MHz clock, however the resulting 1 kHz Sample Pulse signal is only required by second stage data capture modules. In Figure 3-14, signals Start ADCLK and ADCLK Mask are used by the Sample Clock Generator process, while the ADC Ready signal is used by the Sample Pulse Generator process.

It is appropriate to mention here that because both of these processes operate on the 12 MHz clock, the ADS5281 clock propagation delay is not violated by the IAS.[13] This is true because the Sample Pulse Generator process triggers on the falling edge of the 12 MHz Sample Clock, while the Sample Clock Generator process triggers on the rising

[12] The System Clock signal originates from an external oscillator; this part has not been specified.

[13] Reference section 3.6.

edge. This phase shift allows for a maximum of 41.66 ns (83.33 ns ÷ 2) for the ADS5281 clock propagation delay. This is much more than the specified maximum of 14.1 ns.[14]

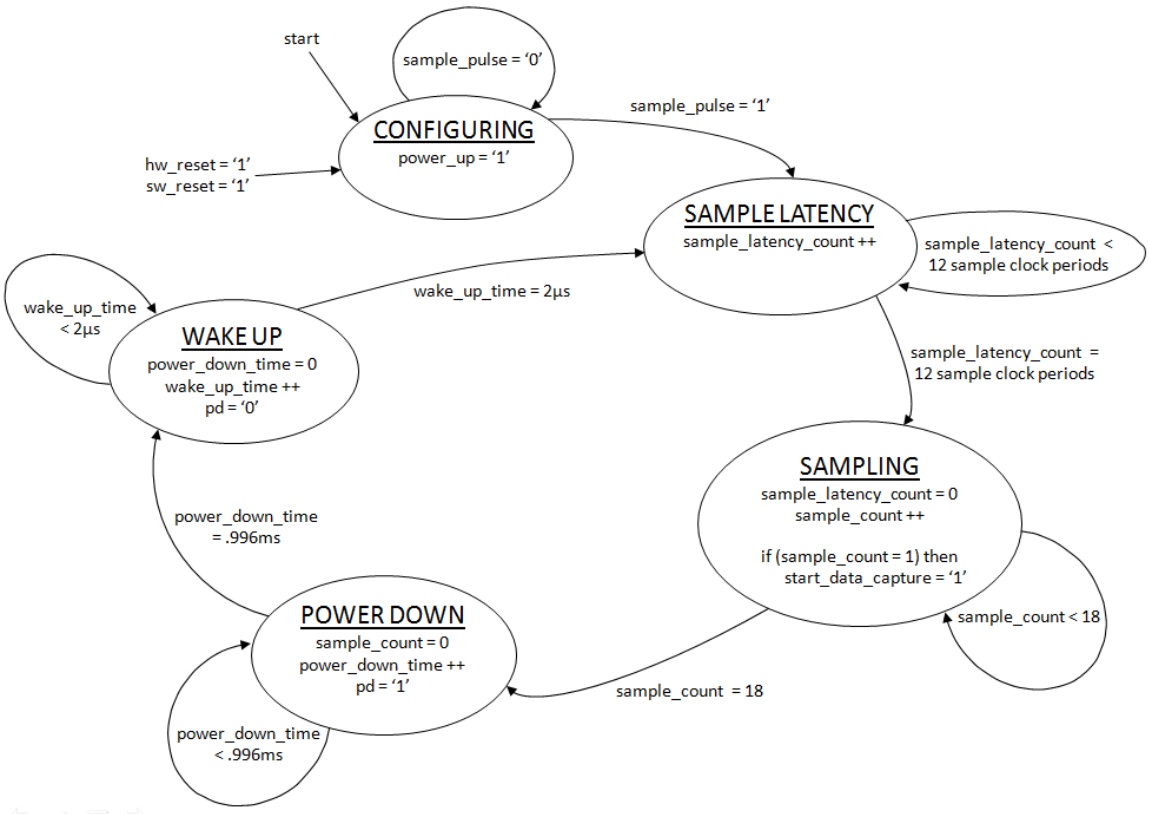

Figure 3-15. Master Control Finite State Machine

3.16.2 ADS5281 POWER UP Module

Once the IAS is turned on, the ADS5281 POWER UP module is activated to execute the ADS5281 initial power-up sequence illustrated in Figure 3-7. The power_up signal in Figure 3-15 is used to enable the Power-Up Sequence finite state machine which makes up this module's hardware. The ADS5281 POWER UP module is responsible for three

[14] Reference Figure 4-6.

ADC control signals which are applied to all ADS5281 devices. In Figure 3-14, these signals are Reset!, AVDD, and LVDD.

Due to the nature of digital signals, the digital datapath does not have a means of generating the actual analog and digital supply voltages, 3.3V and 1.8V, respectively. Consequently, the analog supply voltage signal (AVDD), and the digital supply voltage signal (LVDD) simply specify the moment these voltages should be applied to all 128 ADS5281 devices.

The Power-Up Sequence finite state machine works in conjunction with both the MASTER CONTROL and ADS5281 INITIALIZE modules. As soon as the Power-Up Sequence state machine is started, both supply voltages are asserted and the initial power-up sequence gets underway. When the power-up sequence calls for control signals beyond the command of the ADS5281 POWER UP module, the Power-Up Sequence state machine manipulates signals start_init, chip_select, start_adclk, adclk_mask, and adc_ready. In Figure 3-14, these signals are Start Init, Chip Select Now, Start ADCLK, ADCLK Mask, and ADC Ready. The Power-Up Sequence state machine is illustrated in Figure 3-16.

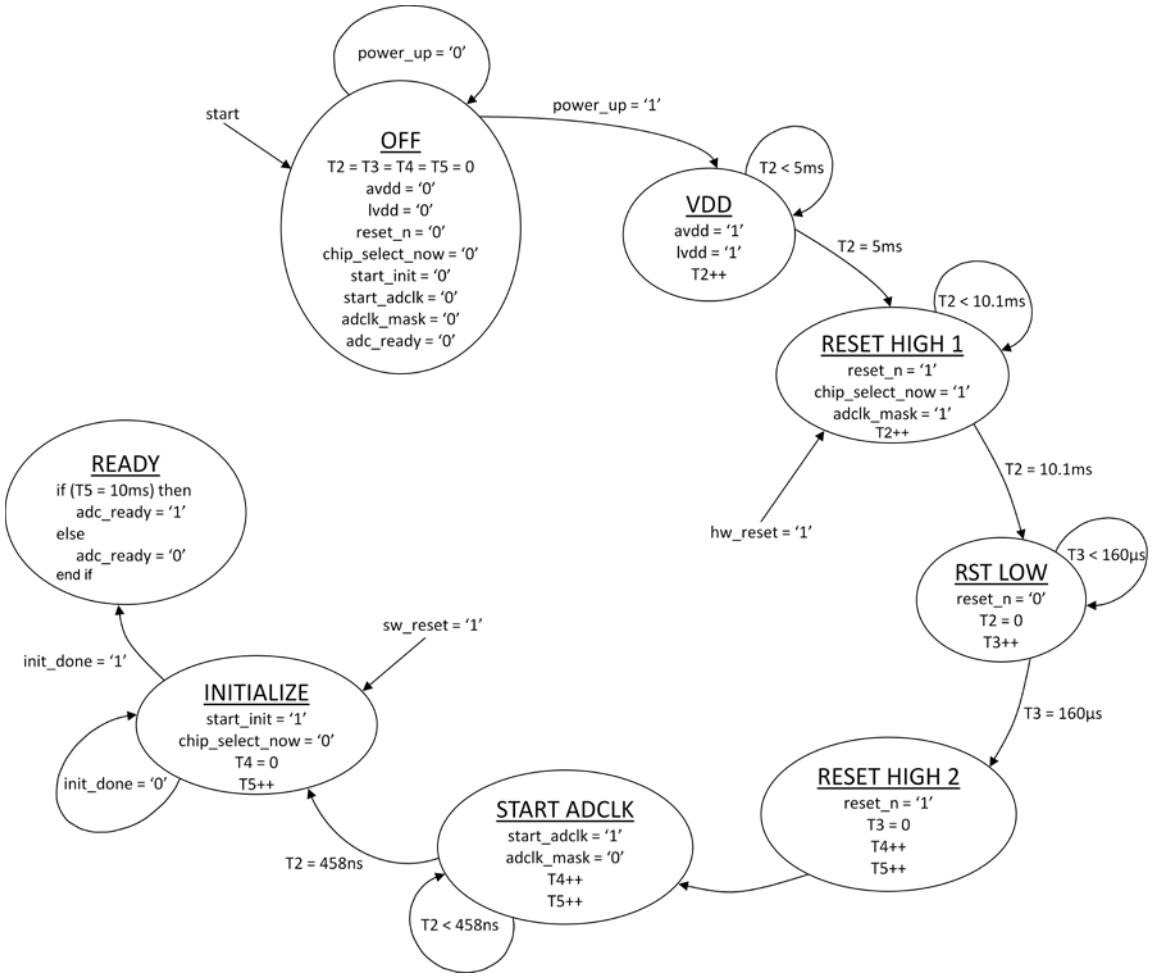

Figure 3-16. Power-Up Sequence Finite State Machine

3.16.3 ADS5281 INITIALIZE Module

The ADS5281 INITIALIZE module hardware consists of a Serial Interface finite state machine and five initialization and configuration registers. The state machine is started by the Start Init signal show in Figure 3-14 (start_init in Figure 3-17). This event triggers two internal processes which control serial interface signals Chip Select!, Serial Clock, and Serial Data. Upon completing register initialization and configuration, the ADS5281 INITIALIZE module asserts the Init Done signal in Figure 3-14 (init_done in

Figure 3-17) to notify the Power-Up Sequence state machine that initialization is complete, and that the ADS5281 power-up sequence can continue. The Serial Interface finite state machine is illustrated in Figure 3-17.

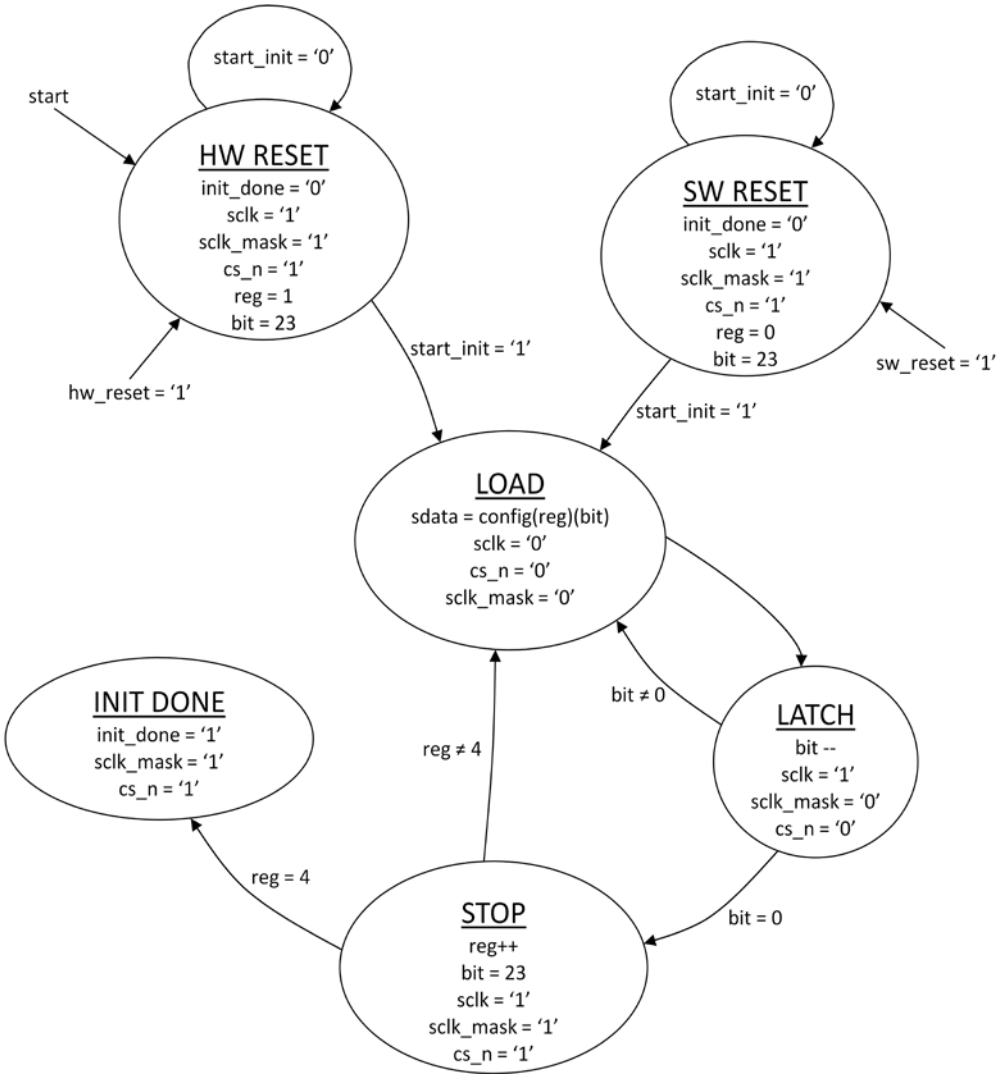

Figure 3-17. ADS5281 Serial Interface Finite State Machine

CHAPTER 4

SECOND STAGE DATA CAPTURE & AURORA 8B/10B TX

4.1 Introduction

As a result of the processes which take place in the first stage, all ADS5281 devices are programmed to continuously produce valid image data synchronized with the constantly running Sample Clock. In the second stage, the IAS gets underway collecting that data by executing three key functions: data capture, data processing, and Aurora 8b/10b transmission. All four Virtex-6 FPGAs are active in the second stage by replicating these tasks.[15]

The second stage commences when FPGA[0] generates a rising edge transition on the 1 kHz Sample Pulse signal. This event triggers DATA CAPTURE modules in FPGAs[0-3] to capture the 18 subsequent 12-bit data samples, on 256 serial ADS5281 channels. The 32 ADS5281s connected to each FPGA also generate 32 Data Clocks and 32 Frame Clocks, for a total of 320 signals.[16]

To accomplish data capture, image data from the ADCs is de-serialized, padded with four zeros to form 16-bit image data words, and then stored on Block RAM memories. Since Virtex-6 FPGAs do not support Block RAMs with more than two data inputs, image data cannot be stored immediately on a single memory. Consequently, serial data arriving simultaneously on 256 ADC channels, is stored on 256 independent 18 x 16-bit word BRAMs.

[15] Aurora 8b/10b transmission is not required by FPGA[0] since all data is ultimately collect there.

[16] LVDS-to-logic conversion via SN65LVDS386s is assumed.

Data processing begins as soon as the first image data words are written into their appropriate BRAMs. Since each FPGA stores a quarter of the entire image on 256 separate memories, image data must be assembled into a predefined order prior to transmission to the third stage. The ordering of image data words at this stage is inconsequential, so long as it is understood and maintained. Nevertheless, since ordering is required, the standard left-to-right and top-to-bottom format is generated, one sample frame at a time. This ordering is referred to as the second stage output format.

The last function carried out in the second stage is the data transfer of all portions of the photo-detector array image to one central location, in FPGA[0]. This is accomplished by establishing three independent Aurora 8b/10b serial protocol channels. Image data acquired by FPGA[0] is processed into the second stage output format and remains in FPGA[0], while image data acquired and processed by FPGAs[1-3] is transmitted to FPGA[0]. The Aurora 8b/10b interface separates the second and third IAS stages. Transmission is part of the second stage, whereas reception is part of the third stage and is discussed in the next chapter.

4.2 Second Stage Architecture

The three tasks performed in the second stage directly correlate to three principal components existing in FPGAs[0-3]. In the Second Stage Block Diagram in Figure 4-1, these components are DATA CAPTURE, DATA PROCESSING, and AURORA TX. The 256 Block RAMs which are used to temporarily store second stage image data are considered part of the DATA CAPTURE module, however they are separately grouped into 256 BRAM blocks, as was similarly done for the ADS5281s and SN65LVDS386s.

The DATA CAPTURE modules are responsible for three functions which allow the IAS to store incoming image data to Block RAMs. First, they determine when ADC data is valid using the 1 kHz Sample Pulse signal. This is a seemingly simple task, however it is complicated by the ADS5281 latencies and delays discussed in Chapter 3. Second, since ADC data arrives concurrently on 256 serial inputs, each DATA CAPTURE module must parallelize that data before it can be written to memory. This task is convoluted by the large number of inputs and the fact that each ADS5281 device generates a unique Data and Frame Clock, and thus operates in its own clock domain. Lastly, each DATA CAPTURE module generates write control signals for 256 Block RAMs at the appropriate time.

The DATA PROCESSING modules perform two critical tasks for each FPGA. First, they produce the second stage output format by reading from the 256 BRAMs and multiplexing out the desired image word. Second, they buffer the resulting output in a 16-bit word FIFO for Aurora 8b/10b transmission.

The AURORA TX modules in FPGAs[1-3] each employ the reference design provided with the Aurora 8b/10b IP Core to establish and implement the Aurora 8b/10b protocol channel with FPGA[0]. Each Aurora 8b/10b channel uses a two byte wide user data interface, allowing 16-bit image words (which are queued in the Aurora TX FIFO) to remain intact during transmission to FPGA[0].

4.2.1 Second Stage Block Diagram

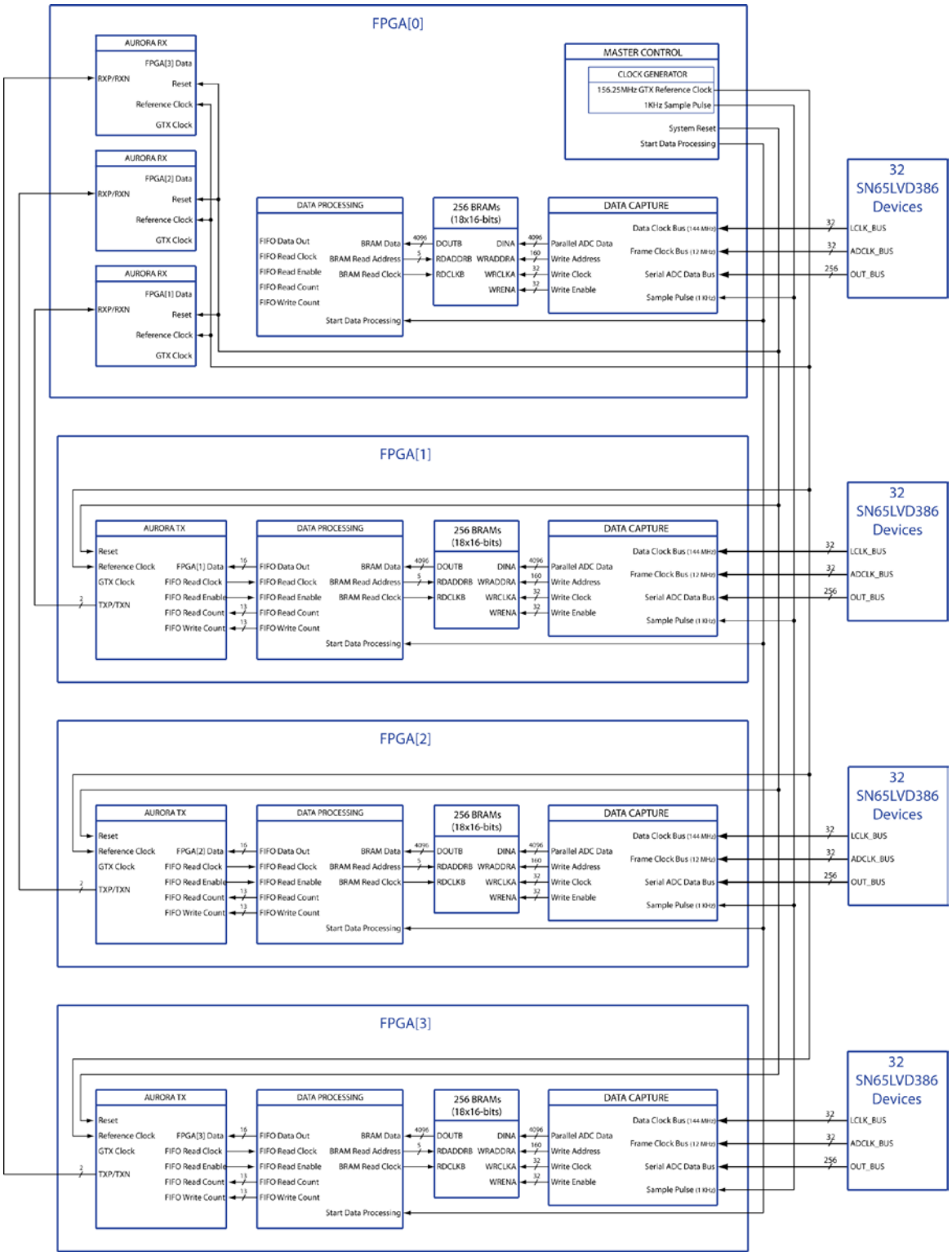

Figure 4-1. Second Stage Block Diagram

4.3 Data Capture

Second stage data capture is the acquisition of image data generated by the 32 x 32 photo-detector array during one Sample Pulse period. As is illustrated in Figure 4-1, FPGAs[0-3] each contain DATA CAPTURE modules which receive ADS5281 digital data on the 256-bit Serial ADC Data Bus input. Since all ADS5281 devices operate independently of each other, each DATA CAPTURE module receives two additional clock busses to synchronize data; these are the 144 MHz Data Clock Bus and 12 MHz Frame Clock Bus. Figure 4-3 presents the bit-mapping which must be maintained for the three busses. Lastly, to determine when image data is valid, FPGAs[0-3] use the 1 kHz Sample Pulse signal generated by FPGA[0].

Each DATA CAPTURE module is made up of two types of (VHDL) modules. The first is a Serial-to-Parallel module which handles parallelization of serial ADC data.[17] Thirty-two Serial-to-Parallel modules are necessary because each ADS5281 devices operates in its own clock domain, and consequently data capture must be handled independently for each ADC. The BRAM Write Control modules generate all the appropriate write control signals for the 256 Block RAMs which follow.

[17] ADS5281 outputs pass through a stage of SN65LVDS386 differential line receivers before entering the DATA CAPTURE modules.

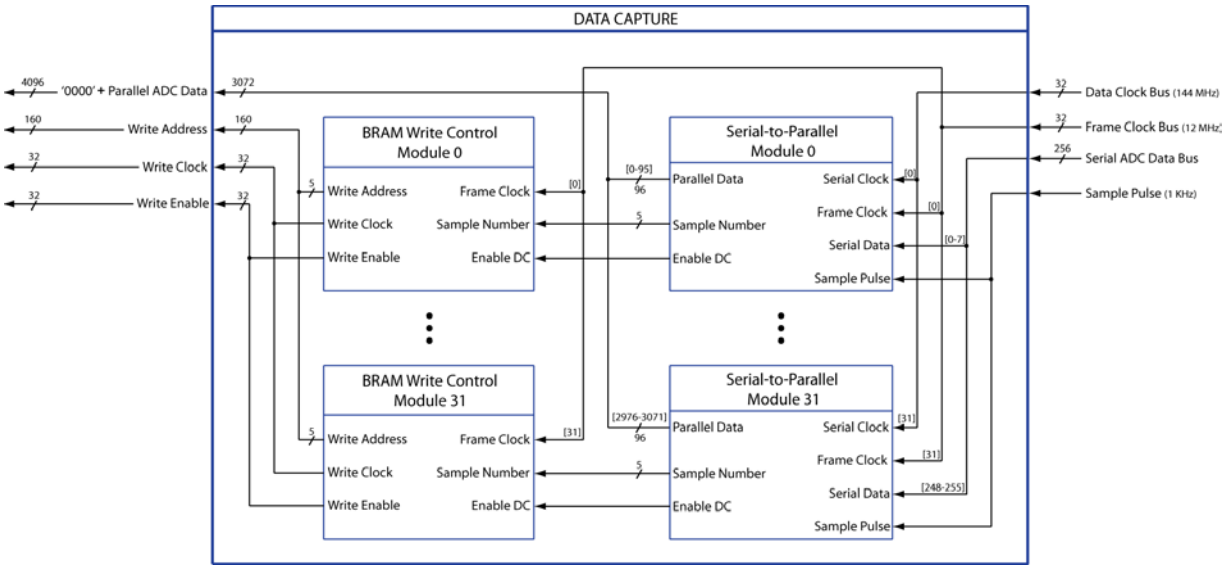

Figure 4-2. Data Capture Block Diagram

Image data enters the Data Capture modules via 256 serial inputs and exits in the form of 256 parallel 12-bit words. The resulting output is the 3,072-bit Parallel ADC Data bus, shown in Figure 3-2. Prior to storage in BRAMs, four zeros are inserted in the most significant bit (MSB) locations of each 12-bit image word. This zero-padding is necessary because Virtex-6 BRAM memories must have a word length which is a byte integer. Nevertheless, this is somewhat convenient because the Aurora 8b/10b user interface also has a width which must be a multiple of 8 bits. This means that image words stored in BRAM memories can remain in one piece during Aurora 8b/10b transmission.

Although this approach suffers from additional overhead, the added transmission time is insignificant to the amount of time the IAS has to process image data. Furthermore, the alternative of breaking up image words for BRAM storage and Aurora 8b/10b transmission severely complicates the IAS design and verification. Consequently, each

ADS5281 channel, and incidentally each photo-detector, is allocated one 18-word BRAM memory. This memory structure results in 256 independent BRAMs which are depicted as one BRAM block in Figure 4-1.

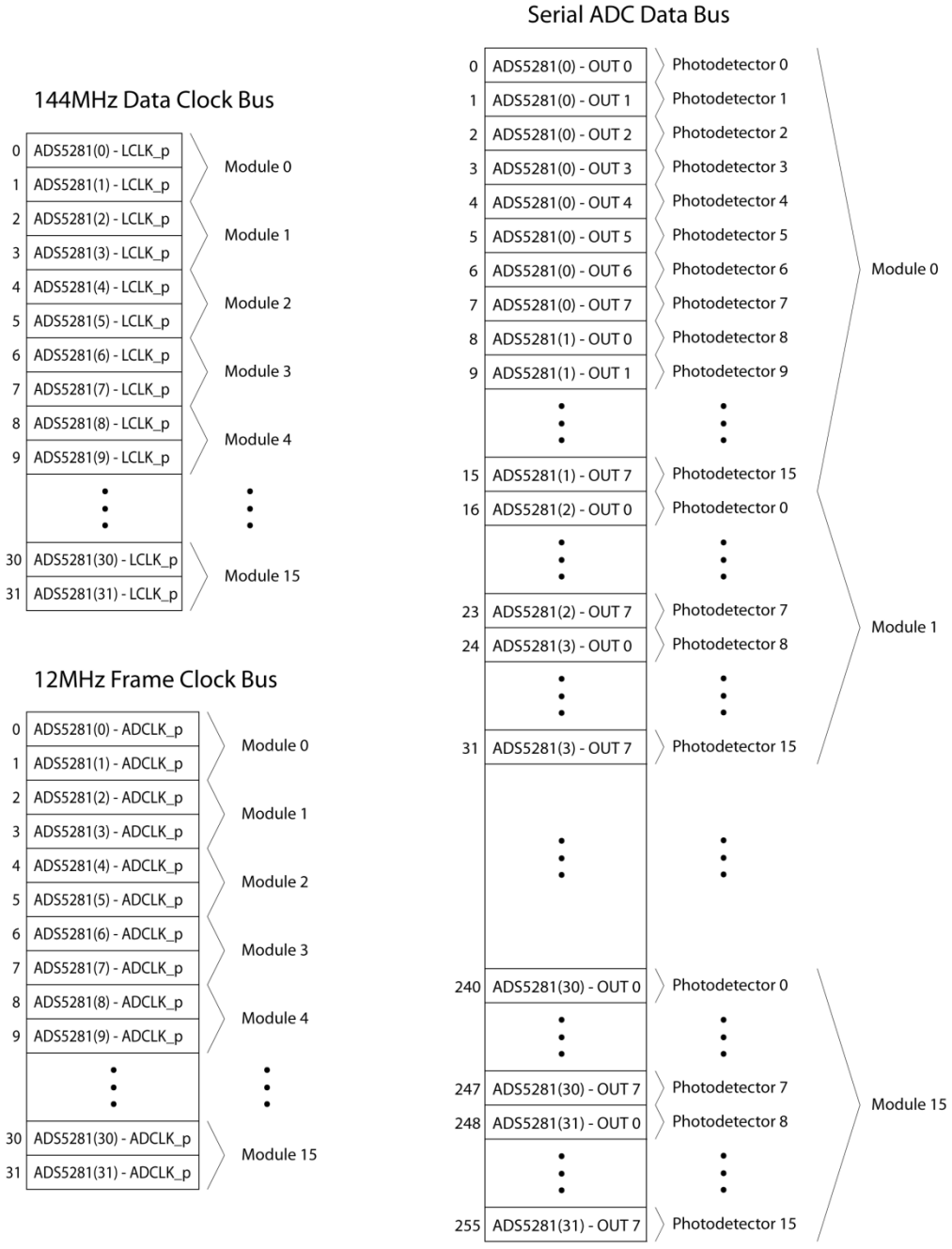

Figure 4-3. ADS5281 Data Clock Bus, Frame Clock Bus, and Serial Data Bus Bit-Maps

4.3.1 Serial-to-Parallel Conversion

There are 32 Serial-to-Parallel modules in each DATA CAPTURE module. Each Serial-to-Parallel module is responsible for making parallel, the 8 serial outputs of one ADS5281 device. Each Serial-to-Parallel module employs a finite state machine which helps it determine when ADC image data is valid. The Serial-to-Parallel finite state machine in Figure 3-4 is synchronized with the 12 MHz ADS5281 Frame Clock, and is started by the Sample Pulse rising edge.

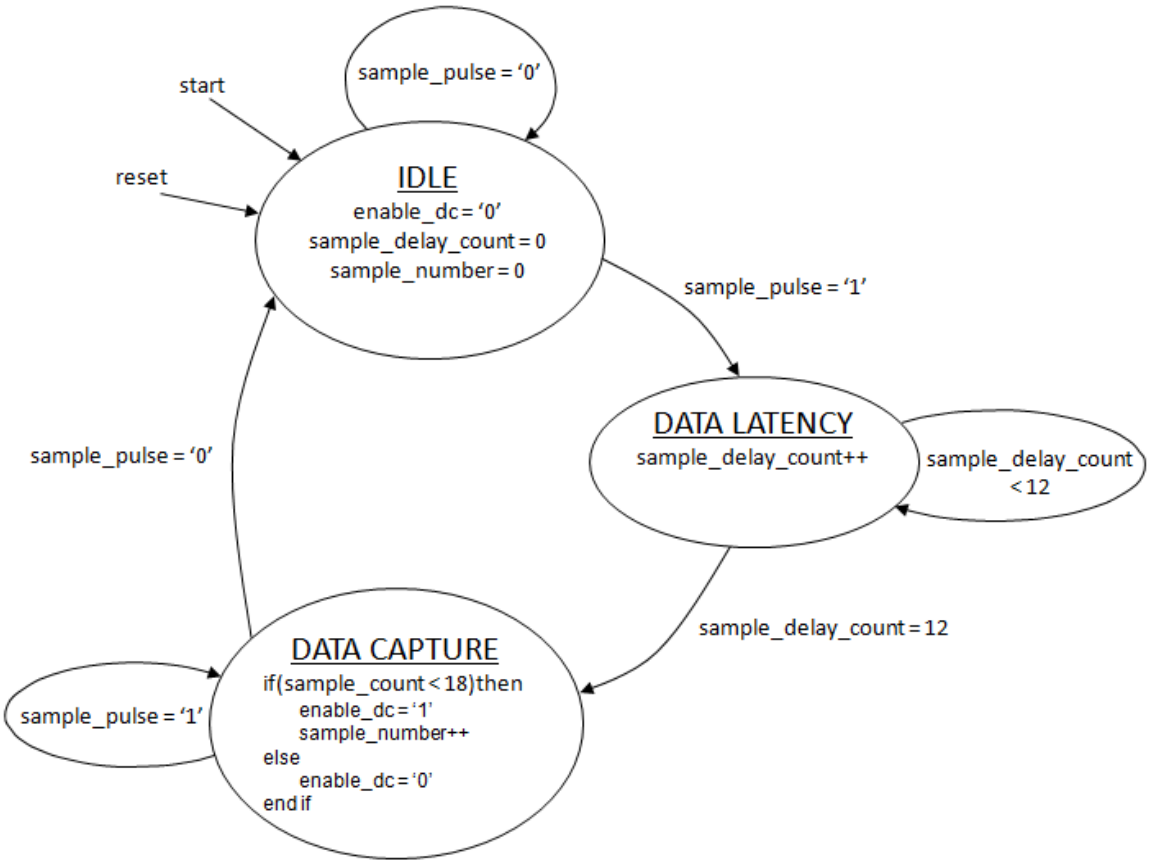

Figure 4-4. Serial-to-Parallel Finite State Machine

To ensure accurate data capture, the Serial-to-Parallel state machines must abide by the clock latency and sample delay constraints described in Chapter 3. It is understood that a

12 Sample Clock data latency, and maximum 14.1 ns clock propagation delay exist before the first serial data bit appears on the ADS5281 outputs. To accommodate this, the Serial-to-Parallel state machines enter the DATA LATENCY state on the rising edge of the sample_pulse signal. The ADS5281 clock propagation delay is satisfied by the fact that each Serial-to-Parallel module operates on the ADS5281 Frame Clock. Furthermore, it is guaranteed that the Serial-to-Parallel state machines take into account the correct Frame Clock edges, because the Sample Pulse and Sample Clock signals are phase shifted by 90 degrees. This is explained in greater detail in section 4.3.3.

Once valid image data is present at the ADC outputs, the Serial-to-Parallel state machines enable eight 12-bit shift registers which make up the remainder of the Serial-to-Parallel module's hardware. ADC data is programmed to arrive MSB first, so data capture hardware left-shifts incoming image data using the 144 MHz Serial Clock signal. Figure 4-5 illustrates the essential function of the Serial-to-Parallel modules.

Lastly, in preparation for storing parallel image data to Block RAMs, the Serial-to-Parallel state machines each generate a 5-bit sample_number signal; 5 bits are required because 18 sample frames are captured. As indicated in the Serial-to-Parallel state machine in Figure 4-4, the sample_number signal increments from 0 to 17 during each Sample Pulse. These signals serve as the BRAM write address for subsequent BRAM Write Control modules.

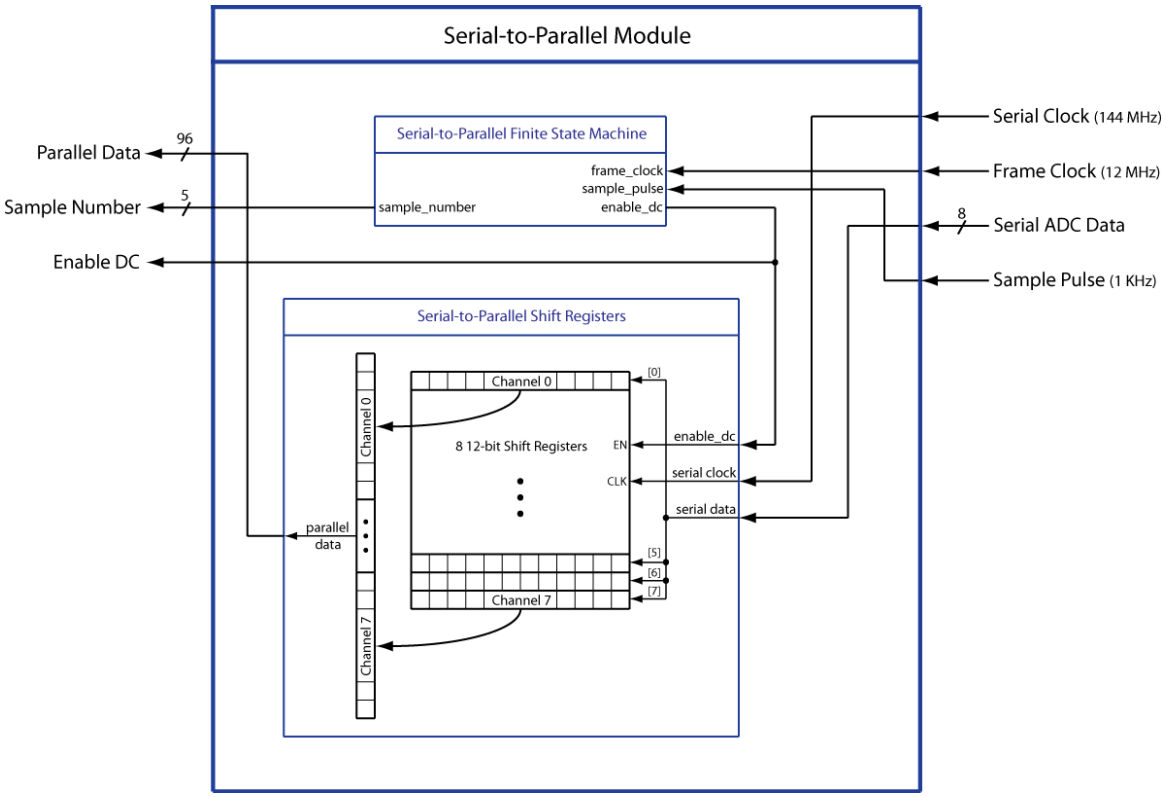

Figure 4-5. Serial-to-Parallel Block Diagram

4.3.2 BRAM Write Control

There are 32 BRAM Write Control modules in each DATA CAPTURE module. Each BRAM Write Control module correlates to one Serial-to-Parallel module, and is responsible for generating write control signals to store image data produced by one ADS5281 device. Since each ADC channel is stored on a separate BRAM, every BRAM Write Control module is associated with eight Block RAMs.

The data capture block diagram in Figure 4-2, shows two important details about the BRAM Write Control module. First, each BRAM Write Control module controls three BRAM write interface signals which are bussed together for output. Second, parallel ADC data bypasses the BRAM Write Control modules altogether. This design decision

puts all timing responsibilities under the control of the Serial-to-Parallel finite state machines. Therefore, the function of the BRAM Write Control modules is simply to relay the BRAM Write Address, Write Enable signals, and unmask the BRAM Write Clock. When writing image data to Block RAMs, the first sample frame is stored in BRAM location 0; the last sample frame is stored in BRAM location 17.[18]

4.3.3 Data Capture Timing

Second stage data capture is best understood by examining the timing diagram in Figure 4-6. The diagram illustrates the entire data capture cycle, beginning with the Sample Pulse rising edge, and ending with the last (18^{th}) sample frame being written to Block RAMs.[19] Three key points to data capture are discussed in this section. The first is the manner in which the IAS handles ADS5281 sample latency and clock propagation delay to correctly capture valid image data. Second, is a closer look at the timing of parallelization of serial ADC data. And third, the timing of BRAM write port signals to store captured samples.

As previously mentioned, the Serial-to-Parallel state machine is primarily responsible for handling the ADS5281 sample latency and clock propagation delay. However, this task is truly made possible by a combination of factors, most notably that the Sample Pulse Generator produces the Sample Pulse at 90 degrees phase shift to the Sample Clock;[20] this effect is identified at the top of Figure 4-6. The phase shift helps the IAS to account for ADS5281 sample latency (of 12 Sample Clocks), by allowing enough time for Serial-

[18] Reference Figure 4-8.

[19] Only one ADS5281 serial data channel is illustrated.

[20] Reference section 3.15.

to-Parallel modules to catch the correct Sample Clock edges. Sample latency is endured at the beginning of every Sample Pulse, between the first and twelfth valid Sample Clock edges.

ADS5281 clock propagation delay is the time between Sample Clock and Frame Clock rising edges. This time delay is identified in Figure 4-6 immediately following the twelfth valid Sample Clock.[21] To ensure that the clock propagation delay is not violated, the IAS relies on two design details. First, the Serial-to-Parallel state machine runs on the ADS5281 Frame Clock rather than the Sample Clock. Although this logical decision seems to solve the clock propagation delay problem on its own, upon closer examination of the point in time when the Sample Pulse goes high, it becomes evident that without the phase shift, the Serial-to-Parallel state machine could easily capture a non-valid Sample Clock edge. This would occur if the Sample Clock rising edge happens on or less than 14.1 ns before the Sample Pulse rising edge. In that circumstance, DATA CAPTURE modules would unintentionally collect image data corresponding to the Sample Clock edge from before the Sample Pulse period. This becomes of growing concern when clock tree size, clock skew, and clock jitter are taken into consideration.

Once sample latency and clock propagation delay are fulfilled, serial ADC data begins streaming out and must be made parallel immediately. The precise moment that serial data for the first valid sample is present at the ADS5281 outputs is identified on the Serial Data signal in Figure 4-6. It is at this time, that the Serial-to-Parallel state machines enable their appropriate 12-bit shift registers and begin counting received samples (see Figure 4-4). These two events are identified in the data capture timing diagram on the

[21] According to ADS5281 device specifications, [4] this unit of time is not longer than 14.1 ns.

Enable DC and Sample Number signals, respectively. Since these signals are the products of Serial-to-Parallel state machines, they both occur on the rising edge of the Frame Clock.

Assertion of the Enable DC signal authorizes Serial-to-Parallel registers to begin shifting in Serial Data bits on every Data Clock rising edge. Upon shifting in the twelfth Serial Data bit, all twelve data bits are registered as the Parallel Data signal. This moment in time is identified in Figure 4-6 with the "1st sample parallel data available" marker. It is important to understand that ADS5281 serial data is continuous and transient. That is to say, it must be captured when it is available, or it is lost forever. In fact, the IAS does not support re-transmission at any phase, including after data parallelization. Consequently, parallel image data must be stored in BRAMs as soon as it is generated by Serial-to-Parallel modules.

Parallel data is written to Block RAMs with the support of BRAM Write Control modules. Although parallel image data bypasses these modules completely, the Enable DC signal notifies them that parallel image data will arrive before the next Frame Clock rising edge. As a result, BRAM Write Control modules assert the BRAM Write Enable signal on the next negative Frame Clock edge. To complete data capture, BRAM Write Control modules unmask the BRAM Write Clock and generate the BRAM Write Address; this is illustrated in the last two signals in Figure 4-6.

In the data capture example provided in Figure 4-6, parallel image data 0x001 is stored in BRAM memory location "00000," followed by parallel image data 0x002 in memory location "00001," and ending with parallel image data 0x012 (eighteen) in BRAM

memory location “10001” (seventeen). These three BRAM write events are identified at the bottom of the data capture timing diagram. After the eighteenth sample is stored, the BRAM Write Control module deasserts BRAM Write Enable, masks the BRAM Write Clock, and resets the BRAM Write Address to “00000,” in preparation for the next image acquisition cycle.

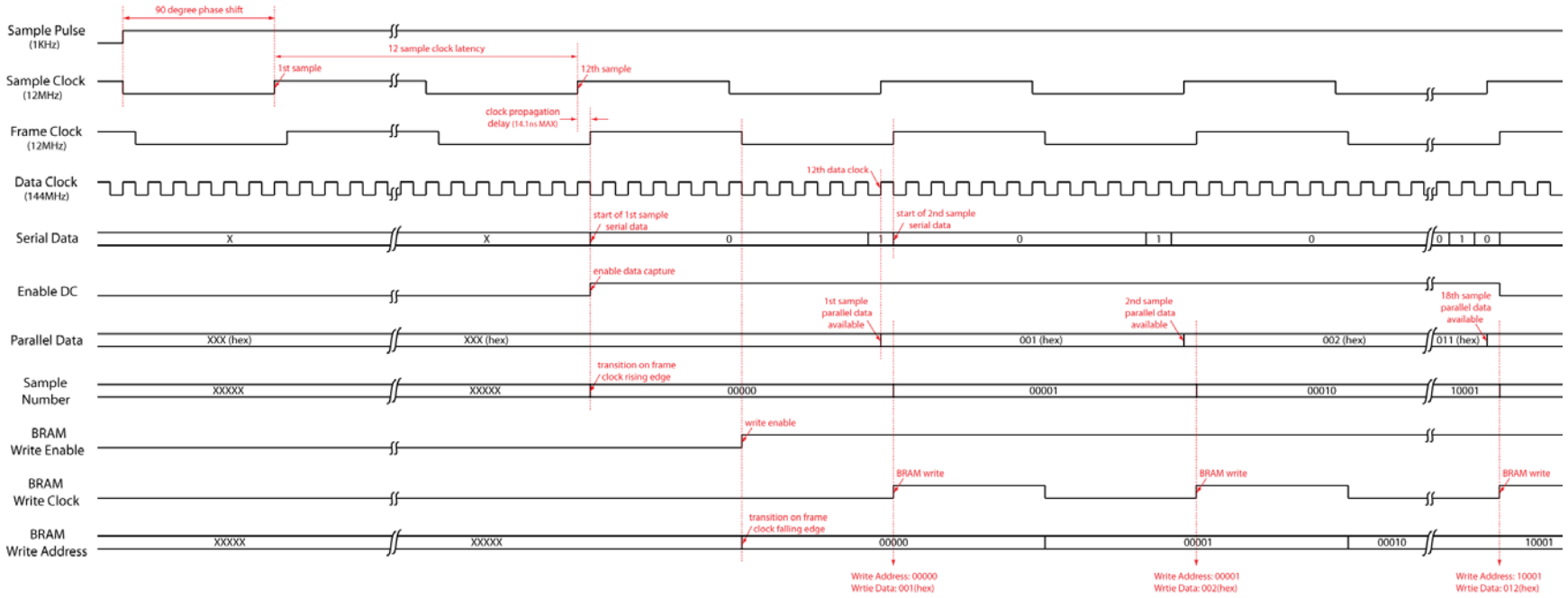

Figure 4-6. Second Stage Data Capture Timing Diagram

4.4 Second Stage Memory Structure

4.4.1 Overview

The 256 independent BRAMs which comprise the second stage memory are recognized as a distributed memory structure. This design decision comes from the fundamental requirement that image data from all 1,024 photo-detectors must be captured simultaneously. It is further emphasized by the incapability of Virtex-6 BRAM memories to support more than two data inputs. What is more, to facilitate data processing without delay, Block RAMs must be configured for separate read and write ports – leaving only one input for writing. All these restrictions lead to a second stage memory architecture which employs 256 discrete Block RAMs. This concept is best visualized in Figure 4-7. From this illustration, it is evident that the adopted memory mapping methodology uses 256 Block RAMs to capture second stage image data from 256 photo-detectors.

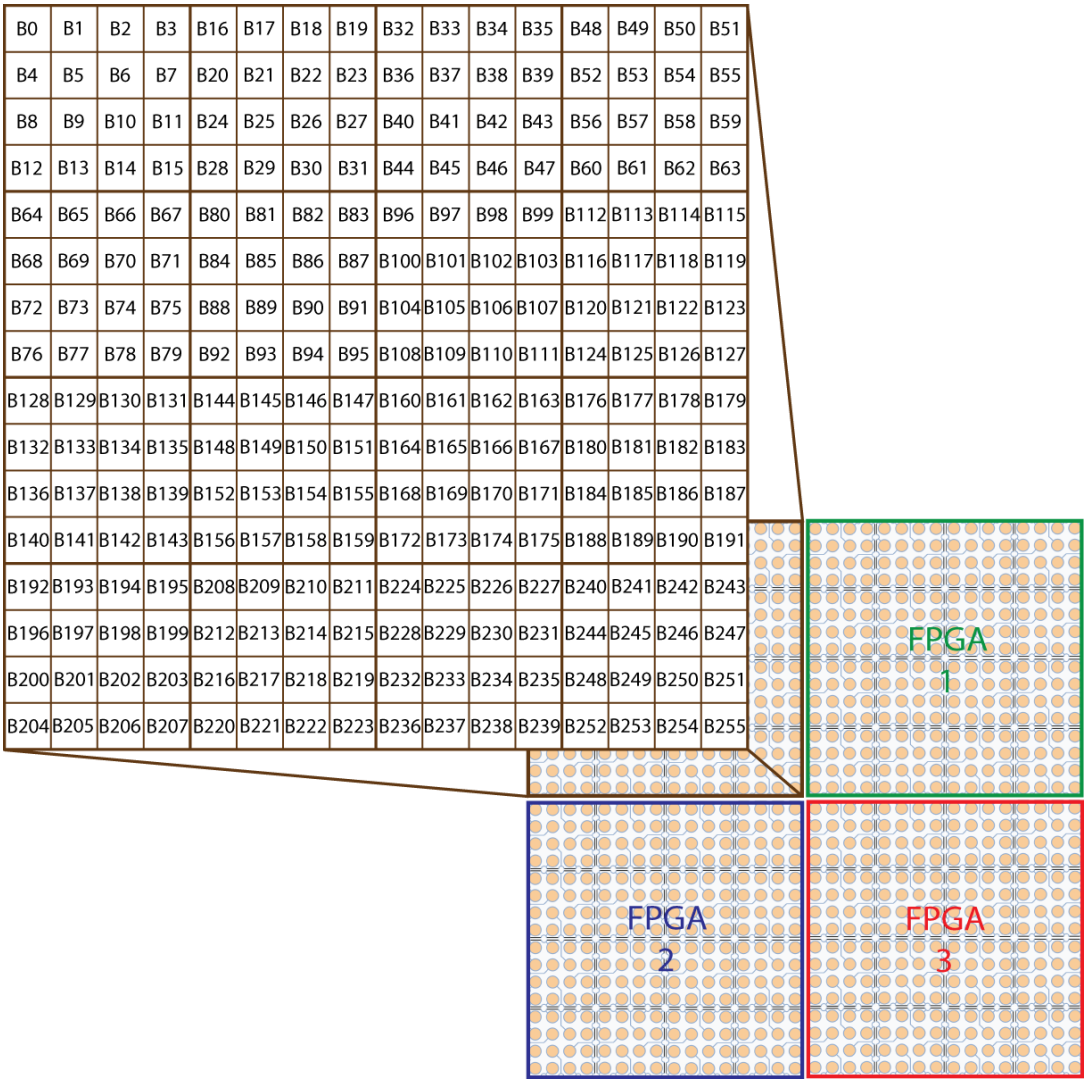

Figure 4-7. Second Stage BRAM Memory Map

4.4.2 Second Stage BRAM Memory

The capacity requirement for second stage BRAMs is determined by the inability of the IAS to re-transmit image data at any point in the datapath. This ultimately means that each BRAM is only responsible for storing image data captured during one sample pulse. In other words, image data captured during one sample pulse is overwritten in the next sample pulse. For this reason, second stage BRAMs are only required to store eighteen

12-bit sample words. Notwithstanding, Virtex-6 Block RAMs must possess a word width which is a byte integer, so each BRAM actually stores eighteen 16-bit words, or 288 bits. The total amount of image data stored by each FPGA in the second stage is therefore 73,728 bits, or 9,216 bytes (256 BRAMs x 288 bits).

Figure 4-8 illustrates one of the 256 second stage BRAMs. Image words are stored in the 12 least significant bits of each BRAM memory location; the remaining four most significant bits are zeros. Image data captured during sample frame 0 (the first sample) is written to BRAM location 0, followed by sample frame 1 in memory location 1, and henceforth. The last (seventeenth) sample frame is written to BRAM location 17.

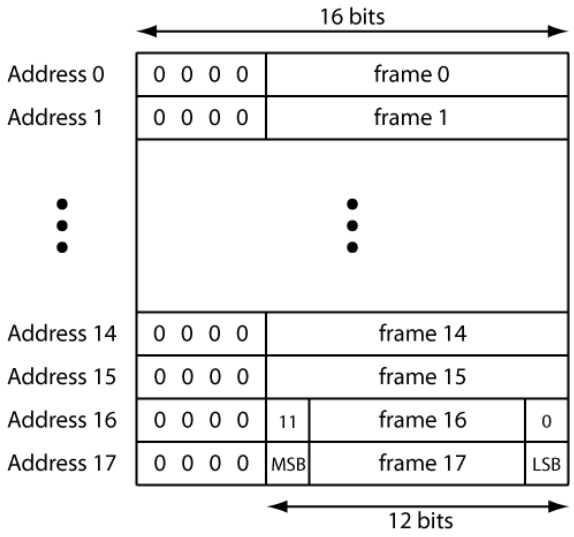

Figure 4-8. Second Stage Block RAM

4.4.3 Second Stage Memory Organization

By now it should be clear that each of the four FPGAs captures image data produced by 256 photo-detectors, and digitized by 32 ADS5281 analog to digital converters. To capture serial ADC data, the IAS must synchronize data acquisition with frame and data clocks provided by each ADS5281 device. Since each ADS5281 supports eight photo-detectors, each BRAM Write Control Module controls eight Block RAMs. The block diagram in Figure 4-9 shows how the DATA CAPTURE module's output signals are distributed to the 256 BRAMs.

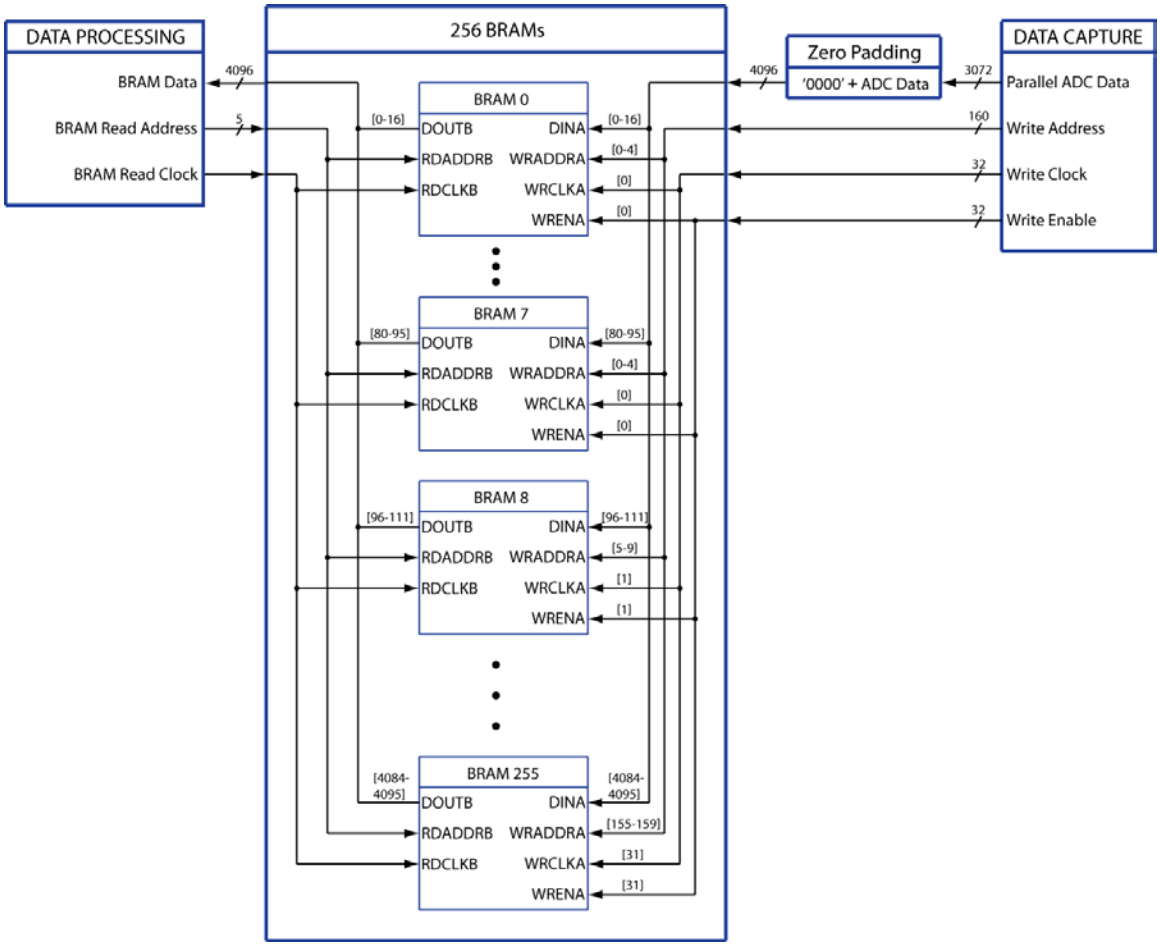

Figure 4-9. Second Stage BRAM Block Diagram

4.5 Second Stage Data Processing

4.5.1 Overview

Once image data is stored on BRAMs in FPGAs[0-3], it must be collected in one central location before being transmitted to the data processing application. In the IAS, this central location is FPGA[0], and the means of data transfer is the Aurora 8b/10b serial protocol. Since second stage image data is distributed among 256 BRAMs, the IAS must provide a mechanism to read and organize that data into a predefined format. This is the objective of second stage data processing.

As it turns out, however, the IAS requirements and specifications make no mention of the desired format in which image data is presented to the data processing application. This ambiguity leaves the system designer with the freedom to choose the most logical and effective format for the final image data output. All methods are valid so long as there is no data loss, so the question becomes, which approach makes life easier further down the datapath? The selected second stage output format is described in the next section. In order to generate this format, image data is processed and reorganized as it makes its way through the IAS. Accordingly, FPGAs[0-3] each contain a DATA PROCESSING module which assist in preparing image data for final output in the third stage.

The DATA PROCESSING modules contain three internal components which together organize image data into the second stage output format. One of these is a BRAM Read Control module which generates the BRAM read port signals required to access stored image data. Second is the Image Data Multiplexing module which contains multiplexing hardware to select the desired image word. And last is a first in first out queue, Aurora

TX FIFO, which buffers the multiplexed output for the AURORA TX module which follows. The second stage data processing block diagram is illustrated in Figure 4-10.

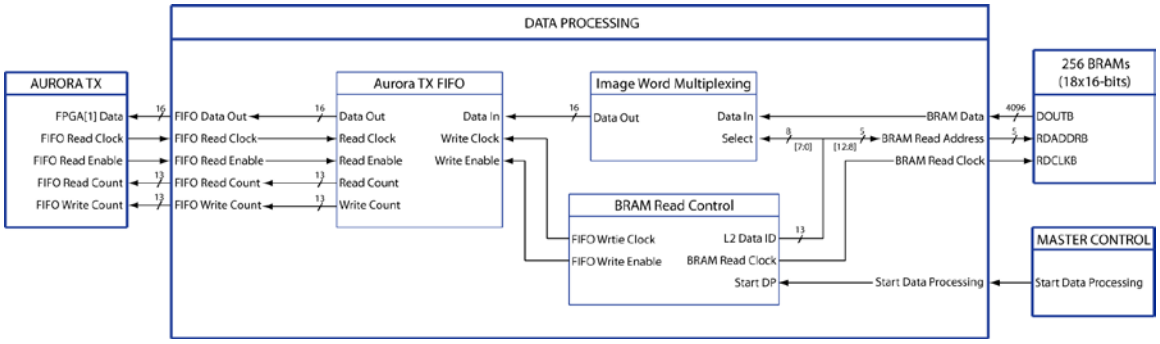

Figure 4-10. Second Stage Data Processing Block Diagram

4.5.2 Second Stage Output Format

Mimicking standard video output formats, image data is presented to the data processing application in the logical left-to-right, top-to-bottom order, one frame at a time. Following suit, DATA PROCESSING modules in FPGAs[0-3] work to organize image data in the same manner. Figure 4-11 illustrates this concept.

The pictured 16 x 16 photo-detector arrays in Figure 4-11 represent all 256 photo-detectors each FPGA is accountable for. The photo-detectors are labeled P0 through P15, in each 4 x 4 module. The 16 4 x 4 modules are labeled M0 through M15. It is important to remember that image data from each photo-detector is digitized with 12-bit resolution and stored in 16-bit BRAM memory locations in accordance with Figure 4-8. As soon as the first sample is written to BRAM memory locations 0, DATA PROCESSING modules start the process of producing the second stage output stream illustrated at the bottom of Figure 4-11. The first 16-bit data word transmitted corresponds to photo-detector P0, in module M0, in frame 0. The last 16-bit image data word transmitted corresponds to

photo-detector P15, in module M15, in frame 17. These data words are identified in the far right and far left, respectively, on the output stream.

Each row in Figure 4-11 contains 256 bits (16 photo-detectors x 16-bit words), each frame contains 4,096 bits (16 rows), and the total amount of image data processed in the second stage is 73,728 bits (18 frames).

4.5.3 BRAM Read Control Module

The BRAM Read Control module is considered the main component in the DATA PROCESSING block. It is responsible for three critical tasks which help bring about the second stage output format in Figure 4-11. As expected, one of these tasks is to generate the appropriate BRAM read port signals so that the correct sample frame is accessed from all memories. The most important of these signals (and the ones included in Figure 4-10) are the BRAM Read Address and BRAM Read Clock. Its second task is to multiplex out the desired 16-bit image data word from the 256 BRAM outputs. To accomplish this, the BRAM Read Control Module generates an 8-bit Select signal. The last function of this module is to manage Aurora TX FIFO write port signals, in order to buffer the output format stream for the subsequent Aurora 8b/10b TX component.

The hardware used to accomplish these tasks is the BRAM Read Control finite state machine illustrated in Figure 4-12. The state machine begins in the IDLE state and is initiated by the Start DP signal (this is the Start Data Processing signal in Figure 4-10). The assertion of the Start DP signal moves the state machine into the BRAM READ state where reading from Block RAMs is started by unmasking the BRAM Read Clock.

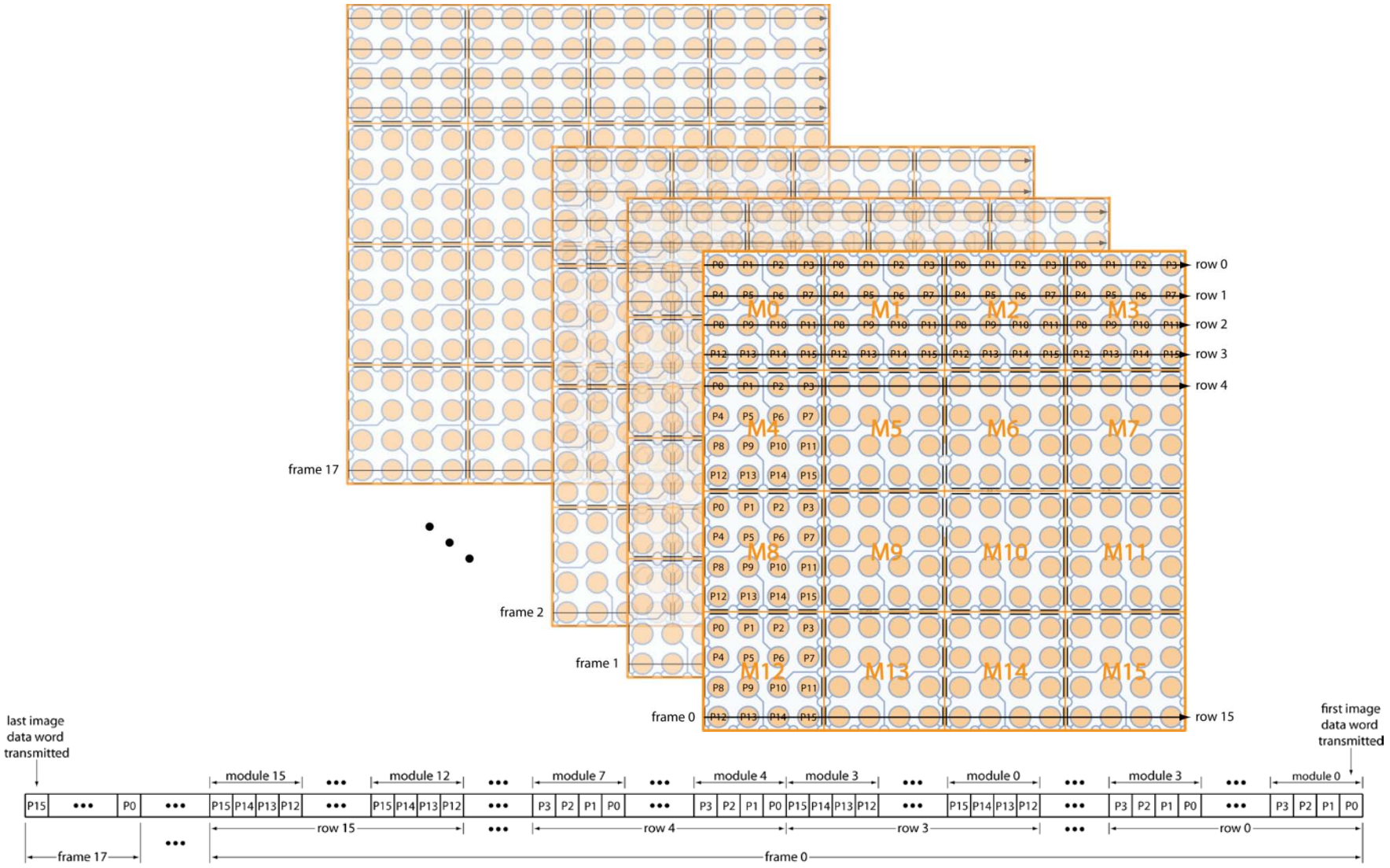

Figure 4-11. Second Stage Output Format

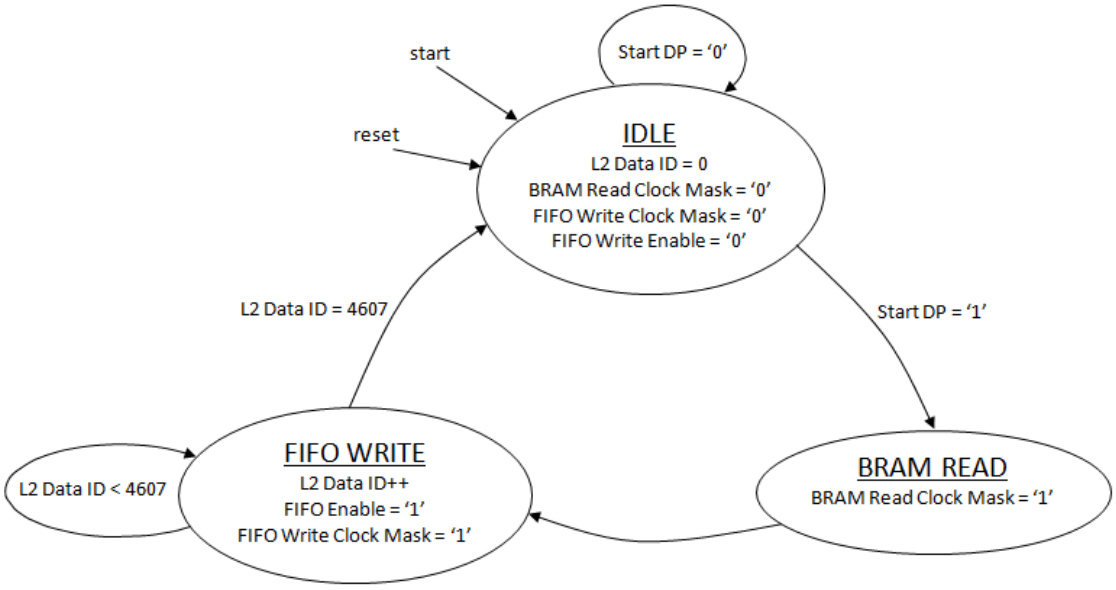

Figure 4-12. BRAM Read Control Finite State Machine

Once reading from BRAMs is in progress, the BRAM Read Control state machine enters the FIFO WRITE state where the multiplexed 16-bit BRAM outputs are written to the Aurora TX FIFO. To accomplish this, a 13-bit counter called the L2 Data ID is incremented until all image data captured in the second stage is read from BRAMs and buffered in the FIFO. The BRAM Read Address and Select signals in the Data Processing block diagram (Figure 4-10) are derived from the L2 Data ID signal.

L2 Data ID bits are mapped to the BRAM Read Address and Image Word Multiplexing Select signal according to Figure 4-13. Since the L2 Data ID is a simple binary counter, this bit mapping is specifically designed to work with the multiplexing hardware to generate the second stage output format. The upper 5 bits indicate which of the 18 sample frames is being accessed from all Block RAMs. The lower 8 bits select the appropriate image data word from modules M0 – M15 and photo-detectors P0 – P15.

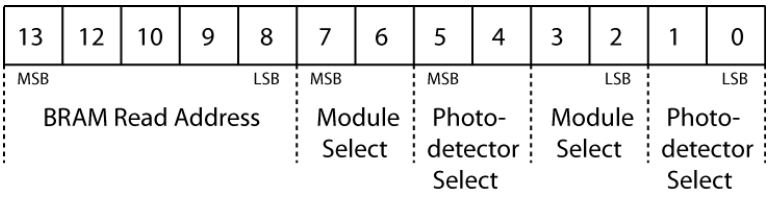

Figure 4-13. L2 Data ID Counter Bit Map

To access image data corresponding to frame 0, all 256 BRAMs read from memory location 0.[22] Once image data from frame 0 has been multiplexed into the second stage output format (and buffered in the FIFO), the BRAM Read Address advances to memory location 1. There are 256 photo-detectors so 8 bits are required to indexing through them all. There are 18 sample frames so 5 bits are required for the BRAM read address.

Once the L2 Data ID counter begins incrementing, the Aurora TX FIFO must be enabled and its write clock unmasked. In Figure 4-12, the two signals which correspond to these operations are FIFO Enable and FIFO Write Clock Mask – both signals go to logic level 1. The BRAM Read Control state machine remains in the FIFO WRITE state until all 4,608 image data words have been written to the Aurora TX FIFO.

[22] Reference Figure 4-8.

4.5.4 Second Stage Multiplexing

Second stage multiplexing happens in the Image Word Multiplexing module. This module uses the 8 least significant bits of the L2 Data ID to select one 16-bit image data word from the 256 BRAM outputs. Figure 4-14 illustrates the multiplexing hardware at the second stage.

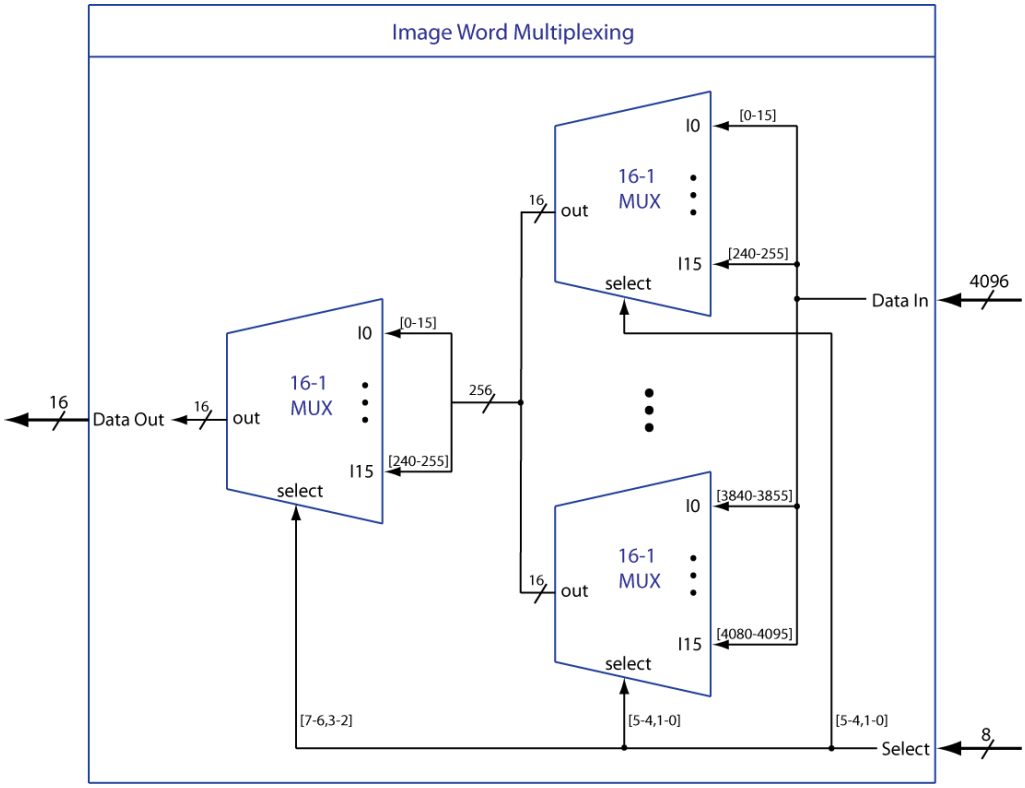

Figure 4-14. Second Stage Multiplexing Hardware

The 16-1 multiplexers in Figure 4-14 have been specifically designed for the Image Acquisition System. They are combinational VHDL circuits written in behavioral architecture, using one case statement. The output of the Image Word Multiplexing module is a 16-bit wide data stream which makes up the second stage output format.

4.5.5 Aurora TX FIFO

The purpose of the Aurora TX FIFO is to buffer the data stream generated by the Image Word Multiplexing module for Aurora 8b/10b transmission into the third stage. Each FPGA processes image data into its own Aurora TX FIFO.

The FIFO hardware is a Xilinx IP Core, created using the LogiCORE FIFO Generator Wizard available in the ISE design environment. A screenshot of the FIFO Generator summary page is provided in Figure 4-15.

Figure 4-15. Xilinx LogiCORE FIFO Generator Summary

The Aurora TX FIFO hardware is built from four 36K Block RAMs. The FIFO is capable of storing up to 8,191 16-bit words, but only 4,608 are transmitted during each Sample Pulse. The FIFO is cleared when both Read Enable and Write Enable signals (from Figure 4-10) are unasserted.

All FIFO write port signals are generated by the BRAM Read Control module.[23] All FIFO read port signals are generated by the AURORA TX module, which is discussed in the next section. FIFO read-write conflicts are avoided completely because FIFO read and write clocks are derived from the GTX Transceiver reference clock, and are the same (156.25 MHz) frequency. This means that a single image word written to the FIFO will sufficiently buffer the data and prevent collisions. The Image Acquisition System writes 1,536 data words into the FIFO before the first read operation.

Keeping in mind that data processing begins after the first BRAM memory location is written into, the total time for data processing at the second stage is calculated below.

$$T_{\text{Second Stage Data Processing}} = \frac{1}{12\ x\ 10^6} + \frac{1{,}536}{156.25\ x\ 10^6} = 9.914\ \mu s$$

It takes one 12 MHz Frame Clock period to write to the first BRAM memory location, followed by 1,536 FIFO writes at a frequency of 156.25 MHz. It should be noted that the FIFO write clock can operate up to 3 GHz without causing data corruption because the aggregate data rate for writing to BRAMs is 3.072 GHz. This is true because 256 BRAM memory locations are written to every 12 MHz. Furthermore, it is not required to allow 1,536 FIFO write operations before the first read; as explained earlier, one write operation is enough. These modifications should be made if the IAS must support a larger photo-detector array, or faster data processing.

With the second stage output format buffered in the Aurora TX FIFOs, second stage image data processing is complete and Aurora 8b/10b transmission can begin.

[23] Reference section 4.5.3.

4.6 Aurora 8b/10b Data Transfer

4.6.1 Aurora 8b/10b Introduction

The Aurora 8b/10b protocol was designed by Xilinx for pint to point link-layer data transfers over a high-speed serial channel. Data transfer is encapsulated in Protocol Data Units (PDU's) which are exchanged between the user application and the Aurora 8b/10b interface. PDU's are transferred between channel partners over the Aurora 8b/10b channel which is made up of one or more, full-duplex, Aurora 8b/10b lanes [2]. Figure 4-16 illustrates the data flow of PDU's over an Aurora 8b/10b channel; the Figure has been taken with written permission from Xilinx's Aurora 8b/10b Protocol Specifications.

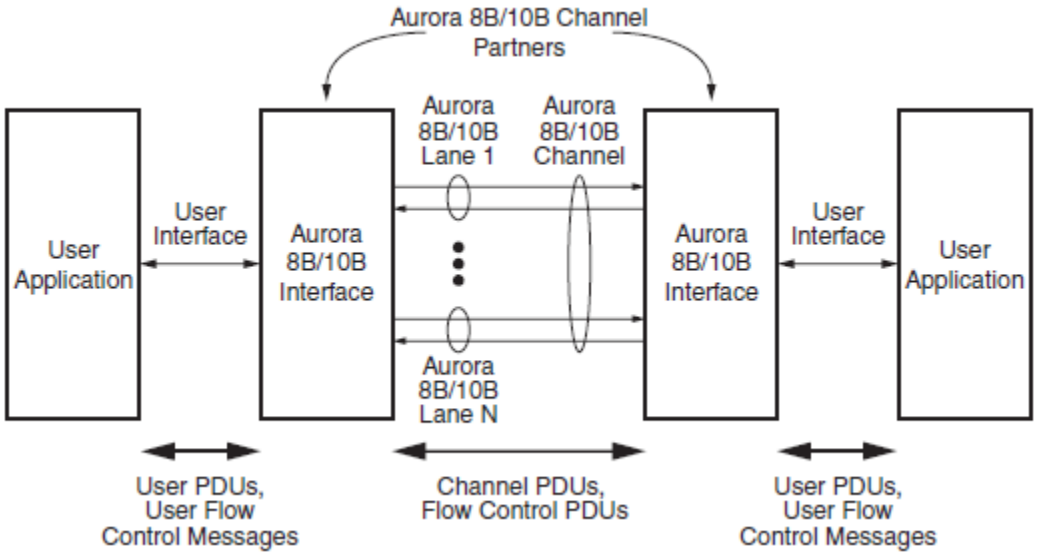

Figure 4-16. Aurora 8b/10b Channel

The Aurora 8b/10b protocol was selected for transferring image data from FPGAs[1-3] to FPGA[0] because of its high speed serial capabilities, ease of integration, and well-documented support. For the Image Acquisition System, each Aurora 8b/10b channel is configured for one 16-bit wide lane which serves as the user data interface. The selected

unencoded line rate of 3.125 Gb/s is generated by the Aurora 8b/10b Core hardware using a 156.25 MHz GTX Transceiver reference clock. Since FPGAs[1-3] are not required to receive any data, the channel is configured to transmit only, in Simplex operation. A Streaming interface is selected to simplify the complexity of the AURORA TX module's user interface, and four sideband signals are used to initialize each Aurora 8b/10b channel. The Aurora 8b/10b channel is configured using the LogiCORE Aurora 8b/10b Customizer which is accessible via the Xilinx ISE Core Generator Wizard. Figure 3-17 is a screenshot of the Aurora 8b/10b Customizer GUI used to configure each Aurora 8b/10b channel from the transmitting side.

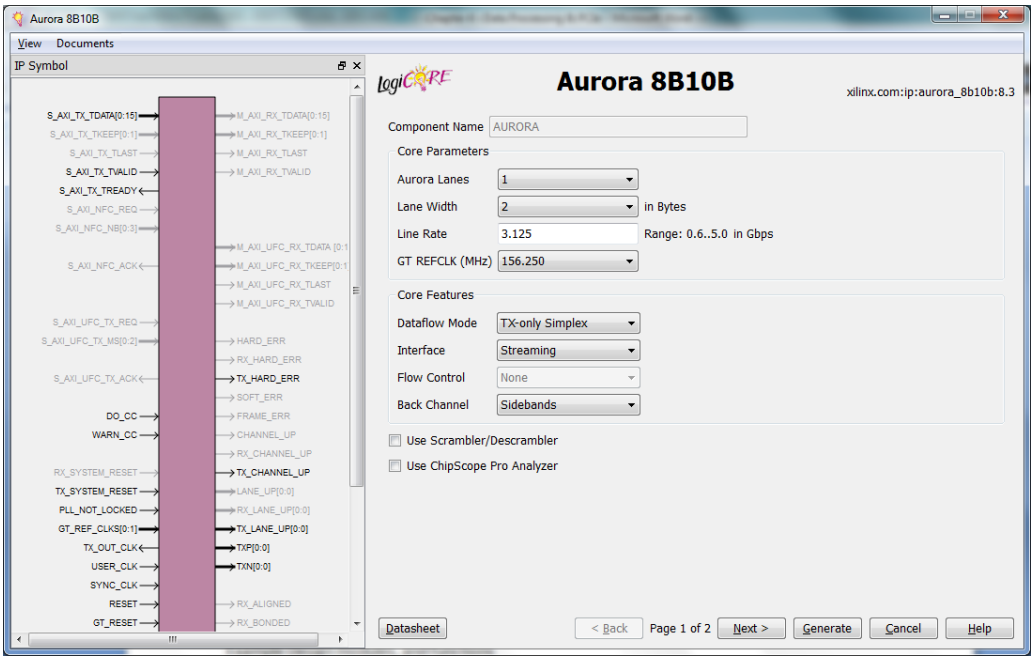

Figure 4-17. Xilinx LogiCORE Aurora 8b/10b Customizer for Aurora 8b/10b TX

The Aurora 8b/10b IP Core Customizer in Figure 4-17 illustrates how the Aurora 8b/10b channel is configured with the parameters described earlier. One Aurora Lane means that one Virtex-6 GTX Transceiver will be used. The Lane Width is set to 2 bytes because

image data words are 16-bits long. The unencoded bit rate at which image data is transmitted over the Aurora 8b/10b channel is set to 3.125 Gb/s. A 156.25 MHz GTX reference clock is selected; this clock frequency determines the Aurora 8b/10b line rate, and serves as the user clock for interfacing the IAS application with the Aurora 8b/10b interface. The Core also features the TX-only Simplex dataflow mode which limits the Aurora 8b/10b channel to unidirectional data transfers from AURORA TX modules in FPGAs[1-3], to the three AURORA RX modules in FPGA[0]. Simplex operation poses additional restrictions on how the Aurora 8b/10b channel is initialized, and disables Flow Control altogether. In Duplex dataflow mode, Aurora 8b/10b channel initialization occurs automatically since data flow is bidirectional. In Simplex mode, the Aurora 8b/10b channel must be initialized via timers or sideband signaling. The IAS carries out channel initialization using four Sideband signals. Lastly, the Aurora 8b/10b datapath interface is set to Streaming. This configuration uses a simpler word-based interface, and data valid signaling [6].

4.6.2 Aurora 8b/10b Data Transfer Calculations

The Aurora 8b/10b transmission rate includes approximately 20% of overhead as a result of 8b/10b encoding. The aggregate data rate for Aurora 8b/10b transmission is 2.5 Gb/s, or 312.5 MB/s.

$$\begin{aligned}\text{Aurora 8b/10b Aggregate Data Rate} &= (.8 * \text{ line rate}) * \text{Aurora lanes} \\ &= (.8 * 3.125\ \text{Gb/s}) * 1\ \text{lane} \\ &= 2.5\ \text{Gbits/second}\end{aligned}$$

To calculate the amount of time for Aurora 8b/10b transmission, the amount of image data transmitted is divided by the data transmission rate. The time required to read image data words from the Aurora TX FIFO is not included in this calculation because this operation is concurrent with Aurora 8b/10b transmission.

$$T_{\text{Aurora Transfer}} = \frac{\overbrace{9{,}216}^{Bytes}}{\underbrace{312.5\ x\ 10^6}_{MBytes/sec}} = 29.491\ \mu s$$

Since FPGAs[1-3] transfer image data in parallel, the total time for Aurora 8b/10b data transmission is 29.491 µs.

In Chapter 2, the total time for the IAS to process collected image data was calculated to 984.608 µs. Since second stage data processing takes 9.914 µs,[24] and Aurora 8b/10b transmission takes 29.491 µs, data processing in the third stage of the IAS must take no longer than 945.203 µs.

[24] Reference section 3.5.5.

4.6.3 AURORA TX Module

The AURORA TX modules in Figures 4-1 and 4-10 are modified implementations of the Aurora 8b/10b reference design, which is provided by Xilinx when the Aurora 8b/10b IP Core is generated. Some of the key VHDL modules delivered with the reference design are a Clock module, Clock Compensation module, Reset Logic module, Frame Generator module, and of course the Aurora 8b/10b Simplex Core. The Clock module generates the Aurora 8b/10b application user clock (from the GTX Transceiver reference clock) which is used to interface the IAS with the Aurora 8b/10b Core. The Clock Compensation module maintains synchronization between Aurora 8b/10b channel partners by inserting specific sequences during idle transmission. The Reset module is responsible for reset input debouncing, but performs the required reset logic for channel initialization as well. The Frame Generator module is used by the reference design to produce random data which is transmitted and later verified to ensure proper channel functionality.[25]

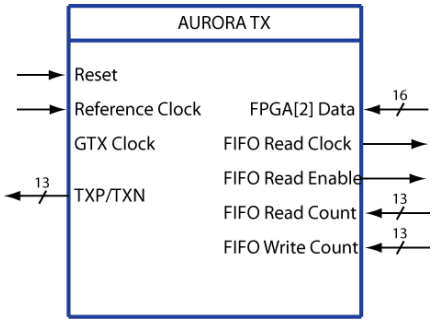

Figure 4-18. Second Stage Aurora TX Module

To implement the Aurora 8b/10b reference design in the IAS, all of the modules and signals from the reference design are kept unchanged with the exception of the Frame

[25] Information regarding the Aurora 8b/10b TX reference example design and its modules has been obtained from the provided VHDL source code.

Generator module. Naturally, this module is modified to read from the Aurora TX FIFO once it has been sufficiently filled. The black box diagram of the AURORA TX module is provided in Figures 4-18, for reference. The FIFO Write Count signal is monitored to determine if the FIFO is sufficiently full. Upon that event, the module enables the FIFO read port and unmasks the FIFO Read Clock signal to commence Aurora 8b/10b transmission. The GTX Clock signal is included as a reminder that the Aurora 8b/10b protocol piggybacks on the Virtex-6 GTX Transceivers, and that an on-board high-speed oscillator is required for each FPGA. The output of the AURORA TX module is a high speed serial data stream on the differential signal pair TXP and TXN.

4.7 Second Stage Simulation

In Chapter 3, the simulations for the ADS5281 power-up and initialization sequences are presented as proof that the digital datapath correctly carries out those essential functions. Although hardware implementation is not carried out on any ADS5281 devices, the simulations clearly demonstrate that the IAS complies with the ADC's requirements for functional configuration and timing, and conforms to the related interfaces (i.e. serial interface, power supply interface, analog signal input interface, and power-down input).

That being said, the absence of actual ADS5281 components limits how the IAS can be simulated once the ADCs should be producing digital data. To overcome this predicament, the ADS5281 device (and more specifically, its output interface) is modeled in VHDL. The resulting ADS5281 Chip Model module is instantiated 128 times in the IAS design. For practical purposes, throughout this document, there is no distinction between the ADS5281 Chip Model and the ADS5281 device illustrated in Figure 3-1.

4.7.1 ADS5281 Chip Model Module

Understandably, the ADS5281 Chip Model module exists only to simulate the ADC outputs, thus allowing testing of "image" data capture, and in turn the rest of the datapath design. The module accepts a 12 MHz Sample Clock, and an asynchronous Reset signal. In return, it produces a 12 MHz Frame Clock (ADCLK), 144 MHz Data Clock (LCLK), and 8 distinct serial data outputs (OUT[0-7]). Since the SN65LVDS386 is only required for hardware implementation, the ADCLK, LCLK, and OUT[0-7] signals are single ended and connect directly to DATA CAPTURE modules on FPGAs[0-3].

The ADS5281 Chip Model contains a Xilinx Clock Generator IP Core which uses the Sample Clock input to generate its Data and Frame Clocks. The Clock Generator hardware conveniently produces a Clocks Locked signal which is used to signal the start of valid ADC data. The module is designed to produce different, and constantly changing, serial data for each output channel. This is necessary to make sense of captured data and to verify the correctness of the design. All the ADS5281 Chip Model outputs uphold the single data rate interface and timing constraints described in section 3.5.

The behavior of the 8 serial data outputs for the ADS5281 Chip Model is described next. Upon starting valid data (frame 0) OUT[0] = 0x0, OUT[1] = 0x1, and so on ending with OUT[7] = 0x7. In the next Frame Clock cycle (frame 1), OUT[0] = 0x1, OUT[1] = 0x2, and so on ending with OUT[7] = 0x8. This continues until the eighteenth Frame Clock cycle (frame 17), when OUT[0] = 0x11, OUT[1] = 0x12, and so on ending with OUT[7] = 0x18. The second stage data capture timing diagram in Figure 4-6 illustrates this precise output sequence for channel 1 (OUT[1]) of one of the ADCs.

4.7.2 Second Stage Data Capture Simulation

The simulation waveform in Figure 4-19 is the continuation of the simulation waveforms presented in Chapter 3. It shows the start of second stage data capture, and includes all the vital data capture signals from Figure 4-6. The ads_ready signal at the bottom of the image indicates that the ADS5281 Chip Model module has locked on to the 12 MHz ads_sample_clock signal, and is generating stable ADCLK and FCLK clocks. These two clock signals are the 32-bit d_clk_bus and f_clk_bus signals. The clock propagation delay discussed in section 3.6, can be observed in this simulation as well. This is the slight skew between the ads_sample_clock and the f_clk_bus signals; it occurs because of the Clock Generator hardware in the ADS5281 Chip Model.

The sample_pulse signal at the top of the figure goes high soon after the ADS5281 Chip Model is ready. Note that the ads_sample_clock and sample_pulse signals are phase shifted by 90 degrees, and that the 12 Sample Clock data latency is satisfied before the first sample frame is captured. Sample frames are identified in the simulation with the id_sample_count signal; this corresponds to the Sample Number signal in Figure 4-6, however the count starts at 1 in simulation. The enormous serial and parallel data buses which contain "image" data from all 128 ADS5281 Chip Model modules are not shown in the simulation because deciphering them is absolutely senseless. The second stage output format simulation in Figure 4-20 is better suited to show accurate data processing in the second stage.

Data capture is initiated by the enable_dc signal which goes high when serial data for the first sample frame begins. The wr_en_a signal goes high during the middle of the first serial sample (on the falling edge of the f_clk_bus signal). In Figure 4-6, this is the

BRAM Write Enable signal. The wr_clk_a signal is unmasked after the twelfth serial data bit of the first sample frame is received. In Figure 4-6, this occurs on the BRAM Write Clock signal. Lastly, the wr_address_a signal is incremented from 0 to 17 to store image data to the correct (frame) memory locations on second stage Block RAMs. The wr_address_a is represented by the BRAM Write Address signal in Figure 4-6.

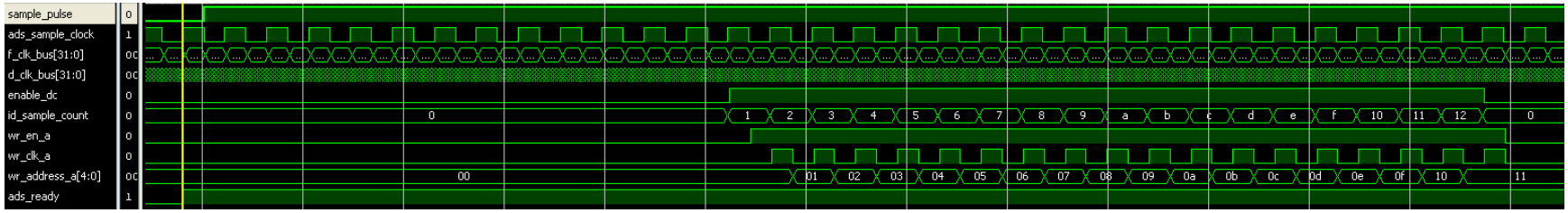

Figure 4-19. Second Stage Data Capture Simulation

4.7.3 Second Stage Output Format Simulation

Once data capture in FPGAs[0-3] is complete, the IAS begins processing image data into the second stage output format. To speed up simulation and facilitate comprehension of the waveform output, only two of the 128 ADS5281 Chip Model modules are generated for the second stage output format simulation. These two ADS5281 modules are associated with module 0 (M0) in Figure 4-11. This means that for each sample frame, only M0 "image" data is shown. In Figure 4-20, this is identified by the 18 (frame) data bursts on the 15-bit fifo_data_in signal. Note that these correlate with 18 BRAM reads, presented on the rd_address_b signal. The data capture waveform signals from Figure 4-19 are included (at the top) for reference.

Referencing back to Figure 4-11, and keeping in mind that photo-detectors P0–P7 correspond to one ADS5281 Chip Model, and P8–P15 correspond to another ADS5281 Chip Model, Figure 4-21 takes a closer look at these frame data bursts. Since module M0 is part of rows 0 though 3, each frame data burst in turn has three sub-bursts which contain "image" data from four photo-detectors. In Figure 4-21, "image" data bursts for frames 0 through 2 are shown on the fifo_data_in signal. Figure 4-22 continues zooming-in on "image" data, and proves that the second stage output format is indeed produced by the IAS. The 4-bit apd_sel and module_sel signals are used to multiplex out the desired "image" data word from 256 second stage BRAMs. These are the L2 Data ID, Photo-detector Select, and Module Select signals from Figure 4-13. The fifo_wr_en signal identifies the beginning of the second stage output format data stream. As expected, the first "image" data words are 0x0, 0x1, 0x2, and 0x3; this is row 0 of fame 0. The second row begins with data word 0x4. The photo-detectors in the third row are not

pictured, but be aware that they produced by the second ADS5281 Chip Model module, so "image" data words begin with 0x0, just like for row 0; row 4 is like row 2.

The end of the second stage output format is shown in Figure 3-23. As was explained earlier, for each sample frame, the ADS5281 Chip Model modules increment the output by one. This feature allows for a high degree of certainty that the second stage output format is correctly generated by the IAS, by simply examining the last couple of rows of the last frame. Figure 4-23 shows rows 14 and 15, of frame 17. As expected, the first "image" data word of row 14 is 0x11 (0x0 + 17). The first "image" data word of row 15 is 0x15 (0x4 + 17).

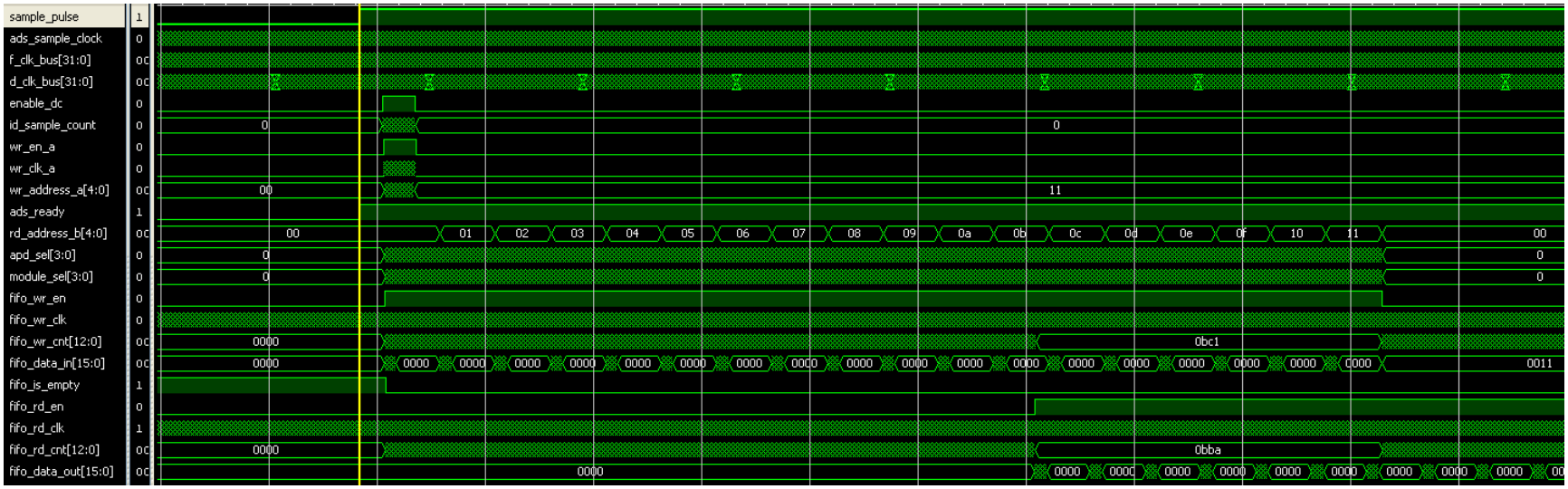

Figure 4-20. Second Stage Output Format Simulation (18 frames)

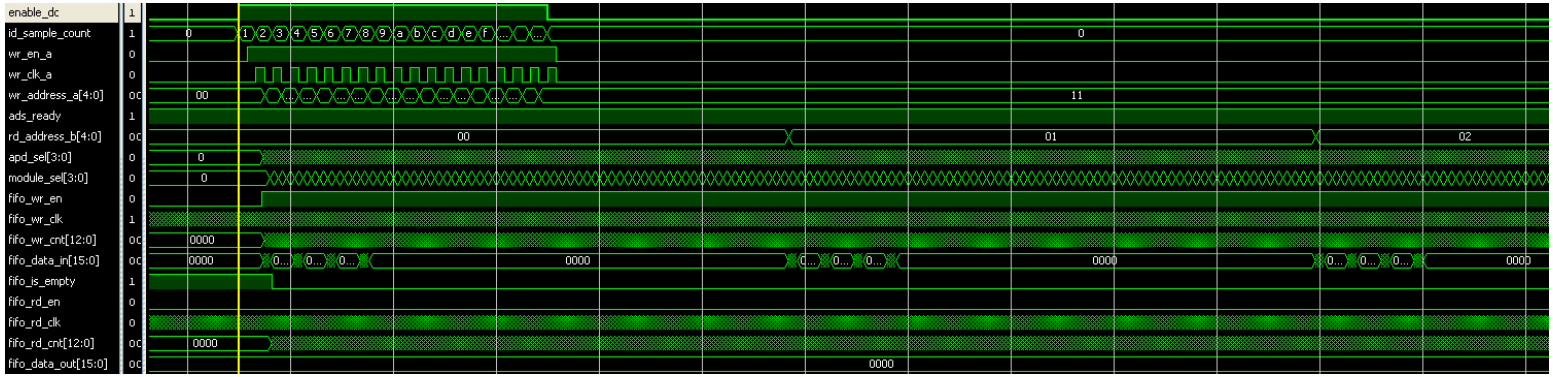

Figure 4-21. Second Stage Output Format Simulation ("image" data bursts)

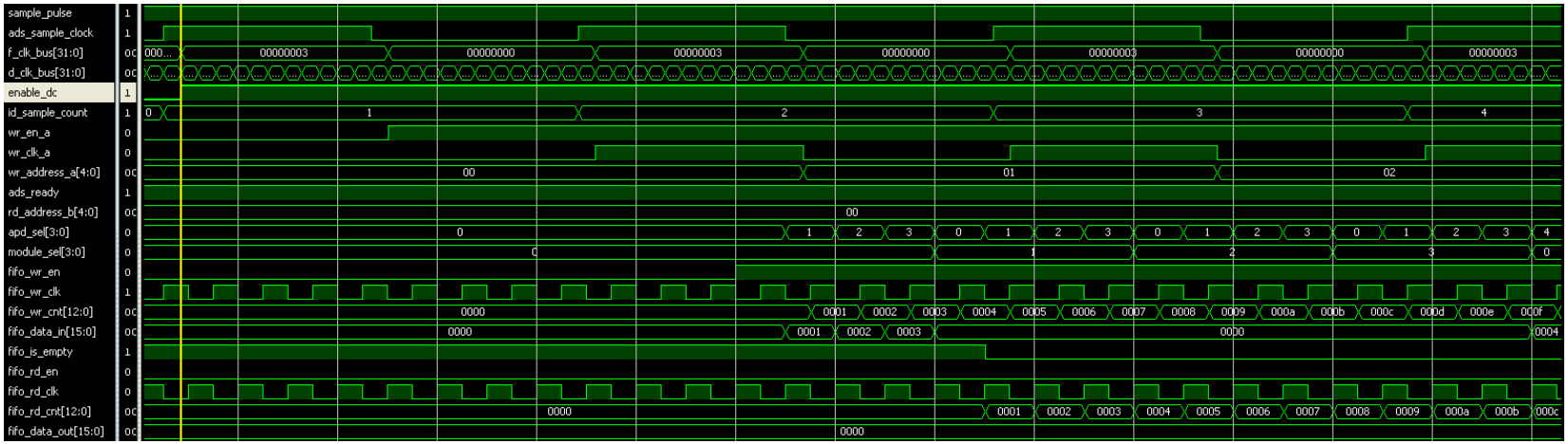

Figure 4-22. Second Stage Output Format Simulation (rows 0 and 1 of frame 0)

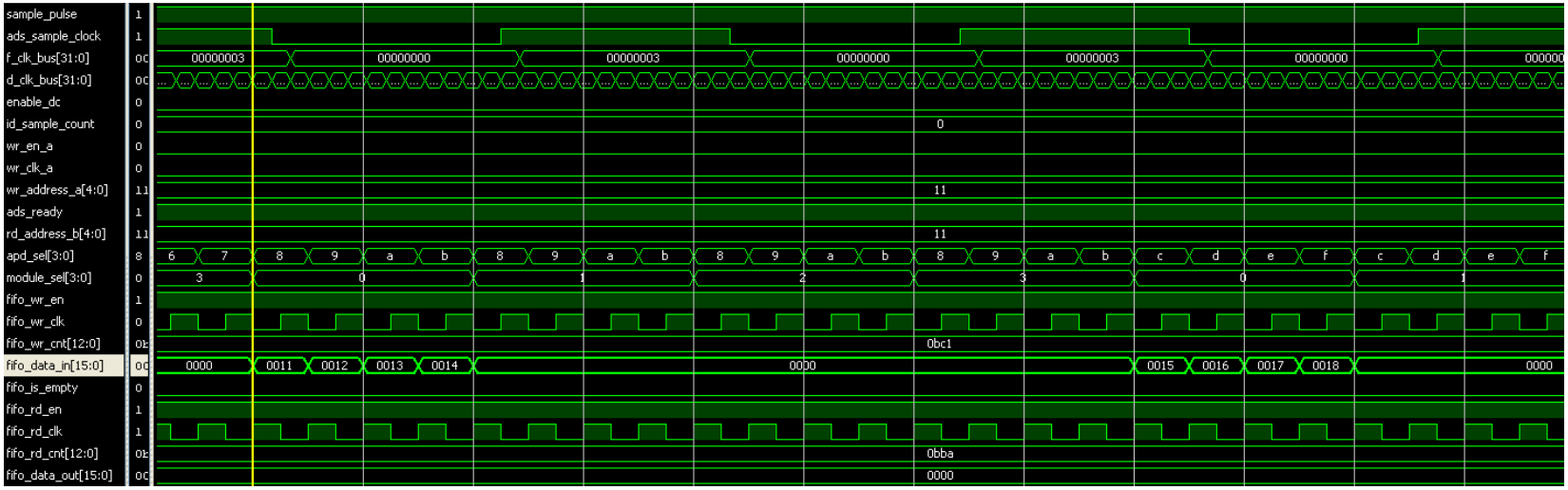

Figure 4-23. Second Stage Output Format Simulation (rows 14 and 15 of frame 1)

CHAPTER 5

THIRD STAGE DATA PROCESSING & PCI EXPRESS

5.1 Introduction

In the third and final stage of the Image Acquisition System, all 18 frames of image data generated by the 32 x 32 photo-detector array are output to the data processing application on a PCI Express (PCIe) protocol channel. Only FPGA[0], the Master Controller, is used to accomplish this task via three important steps. First, image data from FPGAs[0-3] is stored on 4 Block RAM memories. Second, data processing hardware multiplexes stored image data into the final output format, and stores the output-ready data stream in the PCIe TX BRAM . And finally, FPGA[0] transmits image data from the current image acquisition cycle on a PCI Express channel, at a minimum 2 GB/s.

Three modules are responsible for the third stage workload, these are FPGA[1-3] AURORA RX, THIRD STAGE DATA PROCESSING, and PCI EXPRESS TX. A block diagram of the third stage is provided in Figure 5-1. FPGA[1-3] AURORA RX modules receive Aurora 8b/10b protocol transmissions from the second stage. The THIRD STAGE DATA PROCESSING module handles image data storage and final output format multiplexing. The PCI EXPRESS TX module is responsible for establishing the PCI Express x8 lane channel with the data processing application, and transmitting image data from the last 1 kHz Sample Pulse period to the data processing application.

5.1.1 *Third Stage Block Diagram*

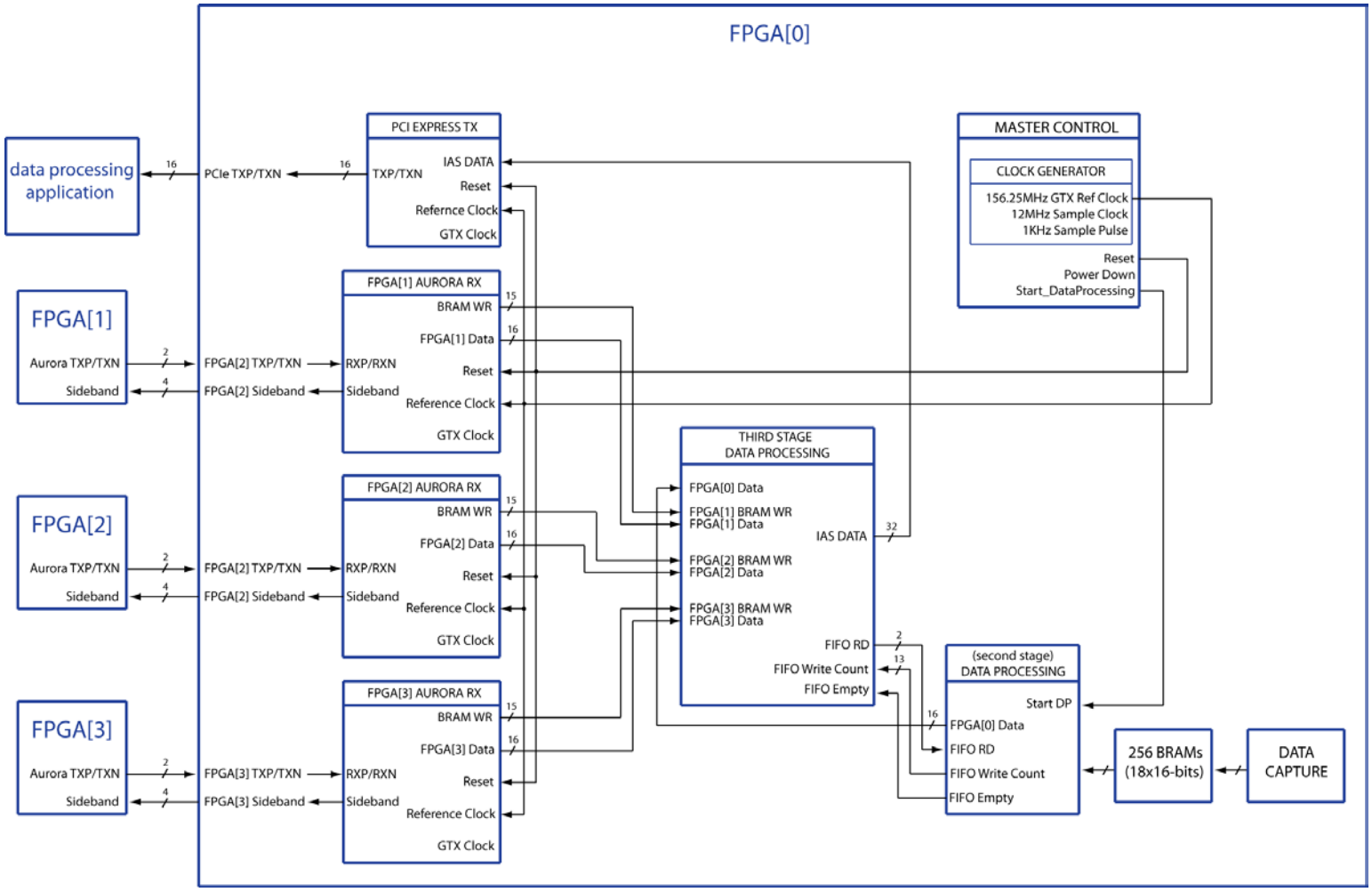

Figure 5-1. Third Stage Block Diagram

5.2 Aurora 8b/10b RX

Image data which is captured and processed by FPGAs[1-3] in the second stage is received in the third stage via three Aurora 8b/10b protocol channels. In Figure 5-1, these channels are the FPGA[1-3] TXP/TXN signals. To establish each Aurora 8b/10b channel, FPGA[0] instantiates three Aurora 8b/10b RX modules. These modules must configure the Aurora 8b/10b channel with the same parameters and features as their channel partner on FPGAs[1-3]. Figure 5-2 shows a screen shot of the Xilinx LogiCORE Aurora 8b/10b Customizer for the Aurora 8b/10b RX modules.

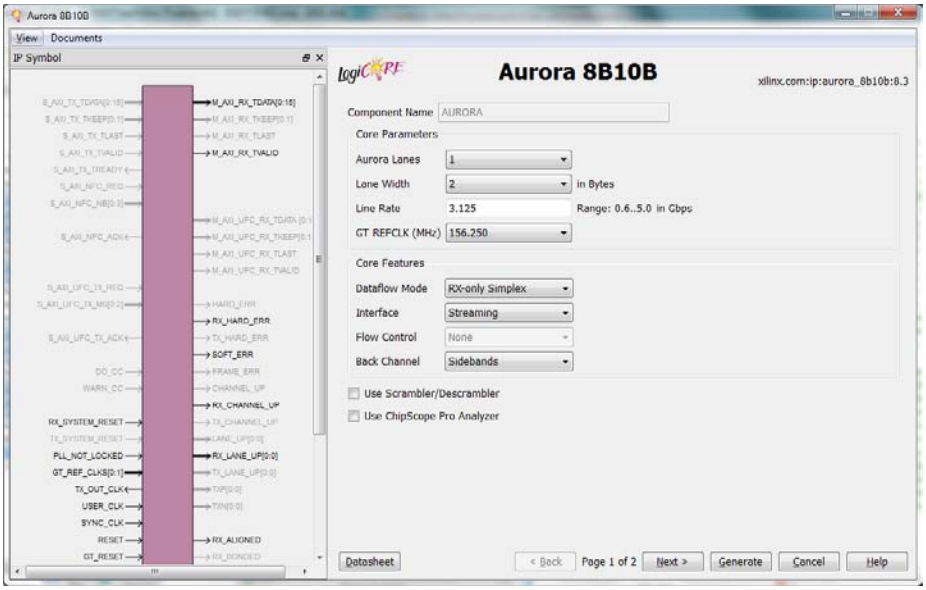

Figure 5-2. Xilinx LogiCORE Aurora 8b/10b Customizer for Aurora 8b/10b RX

For a description of the Aurora 8b/10b Core parameters and features, reference section 4.6. With every Aurora 8b/10b RX Core generation, Xilinx provides the same reference example design as for the Aurora 8b/10b TX Core. The reference example design is used as a guide for connecting AURORA TX and RX modules together.

5.2.1 FPGA[1-3] AURORA RX Modules

Similar to their TX counterparts, Aurora 8b/10b RX modules on FPGA[0] utilize the RX portion of the example design to establish the Aurora 8b/10b channel. The modules included on the receiving side of the reference example design are analogous to the modules on the transmit side. These include a Clock module, Reset Logic module, Frame Checker module, and the Aurora 8b/10b RX Core.

In the reference example design, the Clock module generates a phase-locked loop (PLL) user clock; the IAS application uses this clock signal to pass image data words to the Aurora 8b/10b RX interface. The Reset Logic module ensures that the proper reset sequence takes place when the Aurora 8b/10b channel is being initialized. The Frame Checker module verifies that the transmitted PDUs match those which are received.

In the Image Acquisition System, the FPGA[1-3] AURORA RX modules on FPGA[0] incorporate a modified version of the receiving section of Aurora 8b/10b reference example design. In particular, all the necessary modules and interconnecting signals are left unchanged, with the exception of the Frame Checker module. To adapt the Frame Checker module to the IAS, the module is redesigned to generate BRAM write port signals when incoming image data is valid. In Figure 5-1, the Aurora 8b/10b RX modules are FPGA[1-3] AURORA RX. For reference, Figure 5-3 shows the Aurora 8b/10b RX module for FPGA[1].

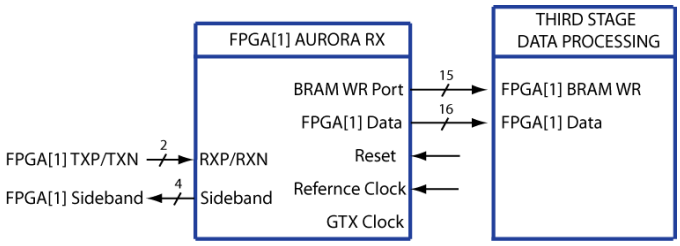

Figure 5-3. Aurora 8b/10b RX Module for FPGA[1]

Each Aurora 8b/10b serial data stream is received by FPGA[1-3] AURORA RX modules on the RXP/RXN input. This differential signal is transmitted at a line rate of 3.125 Gb/s, and carries all image data acquired by one second stage FPGA, during one image acquisition cycle. The Sideband signals are used to communicate the initialization status between channel partners. In Simplex dataflow mode, the Aurora 8b/10b RX module sends four sideband signals (Aligned, Bonded, Verify, and Reset) to the Aurora 8b/10b TX module [6].

The output of each FPGA[1-3] AURORA RX module is a 16-bit data stream (FPGA[1-3] Data), and three BRAM write control signals (combined into BRAM WR Port). The FPGA[1-3] Data signal contains image data from the second stage, and is received in the second stage output format which is described in section 4.5.2. The BRAM WR Port signal contains the control signals required to store image data in Block RAMs inside the THIRD STAGE DATA PROCESSING module. The task of recognizing valid data, and producing the appropriate BRAM write control signals is the responsibility of the modified Frame Checker module.

5.3 Third Stage Data Processing

Third stage data processing ultimately has one objective - to format and buffer image data for PCI Express transmission. The final ordering of image data words is referred to as the final output format. The final output format is illustrated in Figure 5-10, but can literally be described as left-to-right, top-to-bottom, one sample frame at a time (starting with frame 0). Three module groups operate within the THIRD STAGE DATA PROCESSING module in Figure 5-1. The third stage data processing block diagram in Figure 5-4 reveals these modules.

Understandably, data processing must wait until image data from FPGAs[1-3] arrives in FPGA[0]. Four BRAM memories are used to store image data captured by each FPGA. In Figure 5-4, these memories are FPGA[0-3] Data BRAM. The FINAL FORMAT MULTIPLEXING module uses one state machine and an L3 Data ID counter to execute the logic which results in the 16-bit wide final output format data stream. And last, the PCI EXPRESS TX BRAM stores (and sources) image data for PCIe transmission. To simplify testing and verification, 16-bit image data words on the Final Output Format signal are padded with zeros in the most significant bits (MSBs) before they are stored in the PCI EXPRESS TX BRAM module. In this way, image data words remain intact during PCIe transmission.

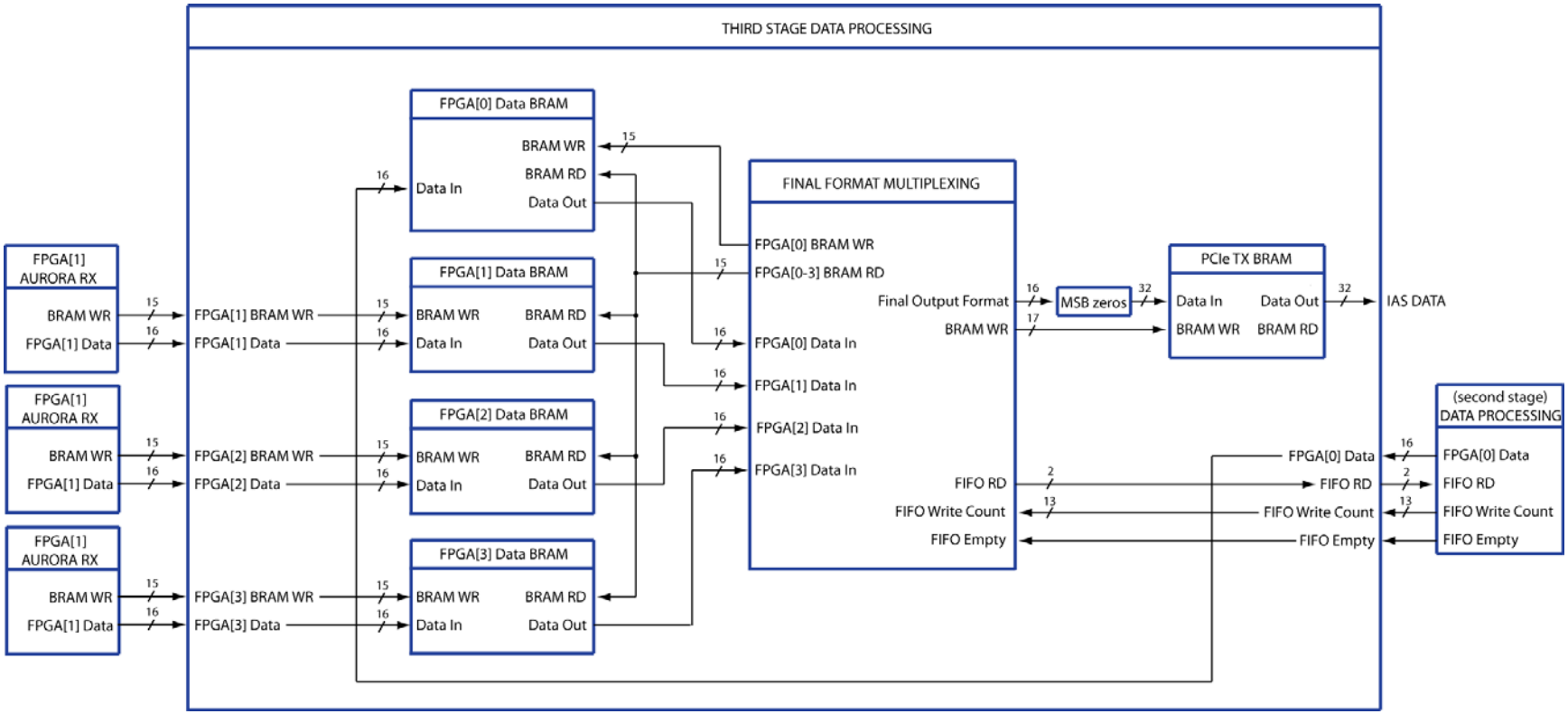

Figure 5-4. Third Stage Data Processing Block Diagram

5.3.1 Third Stage BRAMs

Four Block RAM memories are instantiated in the THIRD STAGE DATA PROCESSING module; in Figure 5-4 these are FPGA[0-3] Data BRAM. The purpose of these memories is to temporary store image data from the three FPGA[1-3] AURORA RX modules, and one (second stage) DATA PROCESSING module. The BRAMs are simple dual port RAMs created using Xilinx's LogiCORE Block Memory Generator Wizard. Each BRAM has the capacity to store 4,608 16-bit words, and requires thirteen address bits to address the entire memory space. Figure 5-5 shows the LogiCORE Block Memory Wizard configuration for third stage BRAMs.

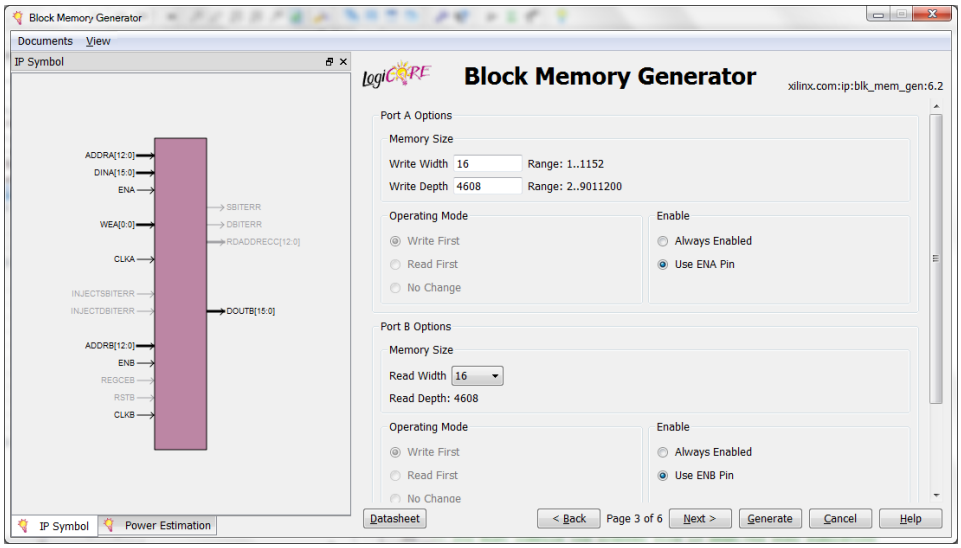

Figure 5-5. Xilinx LogiCORE Block Memory Generator

The data stored in each BRAM represents the total amount of image data captured by each FPGA during one sample pulse. Image data arrives at the BRAM write port in the second stage output format, and is stored in sequential BRAMs memory locations as is illustrated in Figure 5-6.

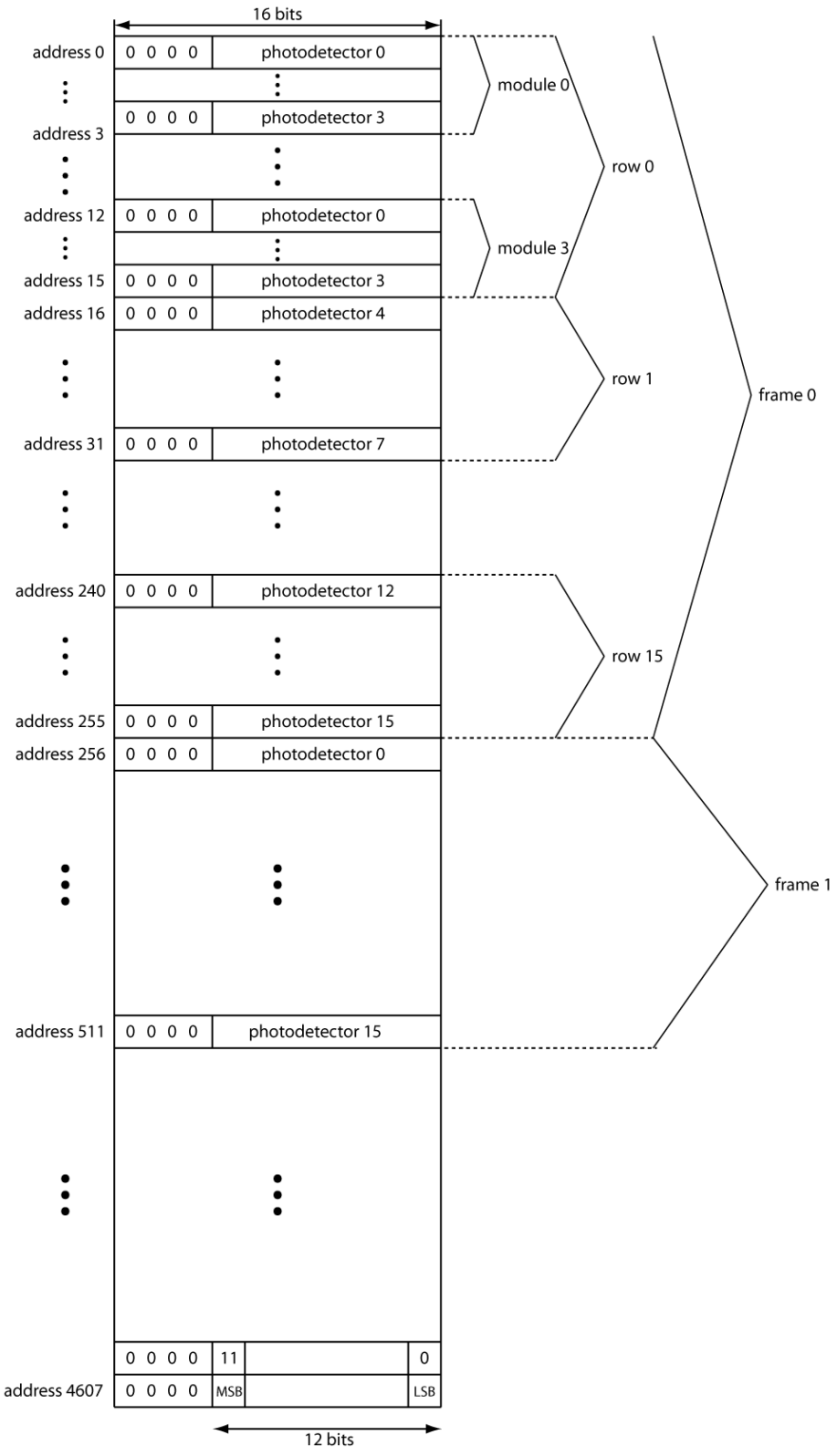

Figure 5-6. Third Stage Block RAM Memory

5.3.2 FPGA[0-3] Data BRAMs

For image data captured by FPGAs[1-3], the IAS utilizes FPGA[1-3] AURORA RX modules to generate the write port signals for FPGA[1-3] Data BRAMs. This is a very practical approach, because FPGA[1-3] AURORA RX modules are already responsible for determining when incoming data is valid. The write port signals required by each BRAM are a write clock, write enable, and a 13-bit address.[26] In Figure 5-4, these signals are grouped together into the 15-bit wide FPGA[1-3] BRAM WR signals, and originate from FPGA[1-3] AURORA RX modules.

For image data captured by FPGA[0], the FINAL FORMAT MULTIPLEXING module is responsible for getting image data from the (second stage) DATA PROCESSING module to the FPGA[0] Data BRAM. To accomplish this, the FINAL FORMAT MULTIPLEXING module generates read port signals for the second stage Aurora TX FIFO (existing in the second stage DATA PROCESSING module), and write port signals to the FPGA[0] Data BRAM. Only two signals are required to read from the FIFO – a write clock and a write enable. In Figure 5-4, these signals are grouped together into the FIFO RD signal. The FPGA[0-3] Data BRAM write port consists of a write clock, a write enable, and a 13-bit write address. Note that in Figure 5-4, these three signals are grouped into the 15-bit BRAM WR signals, and originate from the FINAL FORMAT MULTIPLEXING module for the FPGA[0] Data BRAM.

Once all four FPGA[0-3] Data BRAMs have been filled, the Image Acquisition System has successfully centralized image data for the 32 x 32 photo-detector array in FPGA[0]. Each BRAM contains 18 sample frames of image data from a quarter of the photo-

[26] Reference Figure 5-5.

detector array. Since image data PDU's received on the Aurora 8b/10b channel are stored within one user clock, writing to FPGA[0-3] Data BRAMs adds no additional time to the IAS datapath.

FPGA[0-3] Data BRAM modules are read and multiplexed into the final output format by the FINAL FORMAT MULTIPLEXING module. Read port signals for FPGA[0-3] Data BRAMs include a read clock, read enable, and 13-bit read address. In Figure 5-4, these are the 15-bit wide BRAM RD signals.

5.3.3 Third Stage Data Processing State Machine

To coordinate third stage data processing, the FINAL FORMAT MULTIPLEXING module relies on the finite state machine in Figure 5-7. This Mealy/Moore hybrid state machine is responsible for controlling write port signals to the FPGA[0] Data BRAM, read port signals to the FPGA[0-3] Data BRAMs, read port signals to the FPGA[0] Aurora TX FIFO, and write port signals to the PCIe TX BRAM. Read and write ports for the FPGA[0-3] Data BRAMs require a clock, an enable, and a 13-bit address. The FIFO read port only requires a clock and an enable signal. The PCIe TX BRAM write port requires a write clock, write enable, and 15-bit write address. For clarity, in Figure 5-4, read and write port signals for each module are grouped together.

The third stage data processing state machine begins in the IDLE state. FPGA[0-3] Data BRAM read and write ports are disabled, FPGA[0] Aurora TX FIFO read port is disabled, PCIe TX BRAM write port is disabled, and the L3 Data ID counter is set to zero. The state machine waits until the FPGA[0] Aurora TX FIFO is no longer empty before advancing to the FPGA[0] FIFO WRITE state.

The purpose of the FPGA[0] FIFO WRITE state is to sufficiently fill the FPGA[0] Aurora TX FIFO, before starting to read from it. Section 4.5.5 explains that only one FIFO write operation is enough to prevent read/write conflicts. Nevertheless, the IAS permits 1,536 writes in order to guarantee that the state machine does not attempt to process data too early. Once 1,536 image data words are written, the state machine enables the FPGA[0] Aurora TX FIFO read port, and proceeds to the next state.

The FPGA[0] FIFO READ/FPGA[0] BRAM WRITE state provides the necessary logic to write the contents of the FPGA[0] Aurora TX FIFO into the FPGA[0] Data BRAM. In this state, FIFO read port signals are generated on the rising edge of the user clock, and BRAM write port signals (including a registered write address) are generated on the falling edge of the user clock. Once the FPGA[0] Aurora TX FIFO is empty, the state machine enables the FPGA[0-3] Data BRAM read ports, and advances to the final state. This transition marks the end of image acquisition, and beginning of final output format multiplexing.

The final state in the third stage data processing state machine is FPGA[0-3] BRAM READ/PCIe TX BRAM WRITE. As its name suggests, this state is responsible for read port signals to the four FPGA[0-3] Data BRAM modules, and write port signals (including a registered write address) to the PCIe TX BRAM module. Read port signals are asserted on the positive edge, while write port signals are asserted on the negative edge. The multiplexing hardware used to generate the final output format operates between these two clock edges. The FPGA[0-3] BRAM READ/PCIe TX BRAM WRITE state also maintains a 15-bit binary counter called L3 Data ID. As was similarly

done in the second stage, the L3 Data ID counter is parsed into the appropriate BRAM addresses and MUX select lines.

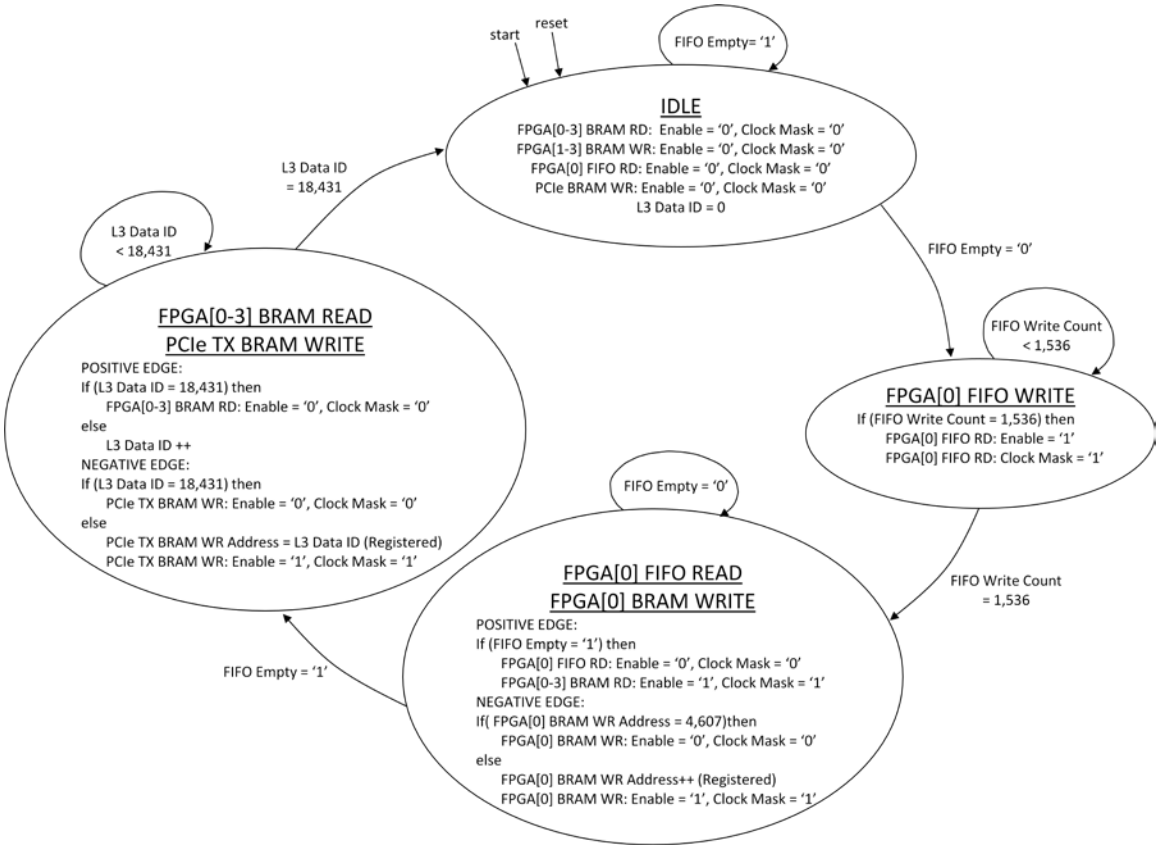

Figure 5-7. Third Stage Data Processing State Machine

5.3.4 L3 Data ID

The L3 Data ID is 15-bit binary counter kept by the third stage data processing state machine. L3 Data ID has three important purposes, although it functions as a simple binary counter. The counter is designed to work with the IAS hardware to multiplex out a 16-bit data stream containing the final output format for all 18 frames of a 32 x 32 photo-detector array. Recall that each 16-bit image data word contains the 12-bit sample from one ADC channel, from 1 of the 18 captured sample frames.

Fifteen bits are required for the L3 Data ID counter because there is a total of 18,432 image data words in the final output format (18,432 in binary is 100 1000 0000 0000). This is the minimum number of bits required to address the memory space for all IAS data. Another way to look at it, multiplexing takes place between four BRAMs (2 select bits), each having 13 address bits. It is appropriate to mention here that the L3 Data ID counter is registered in the last state of the third stage data processing state machine, and the resulting signal serves as the PCIe TX BRAM write address.

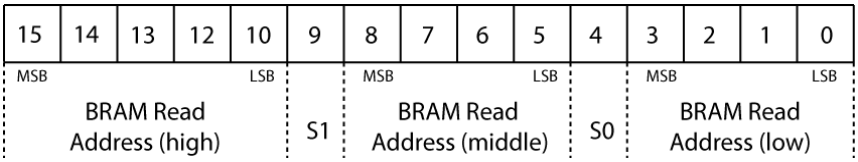

Figure 5-8. L3 Data ID Counter Bit Map

The bit map for the L3 Data ID counter is provided in Figure 5-8. The BRAM Read Address is analogous with the 13-bit read address that is part of the FPGA[0-3] BRAM RD signal in Figure 5-4. Bits 4 (S0) and 9 (S1) are multiplexer select bits. Once the L3 Data ID counter reaches 18,432, the final output format data stream is complete.

5.3.5 Third Stage Multiplexing

Third stage multiplexing is a very simple process because of the data processing work performed in the second stage. The hardware too, is incredibly simple – only a single 4-1 multiplexer is required. Figure 5-9 illustrates the essential multiplexing hardware in the FINAL FORMAT MULTIPLEXING module from Figure 5-4.

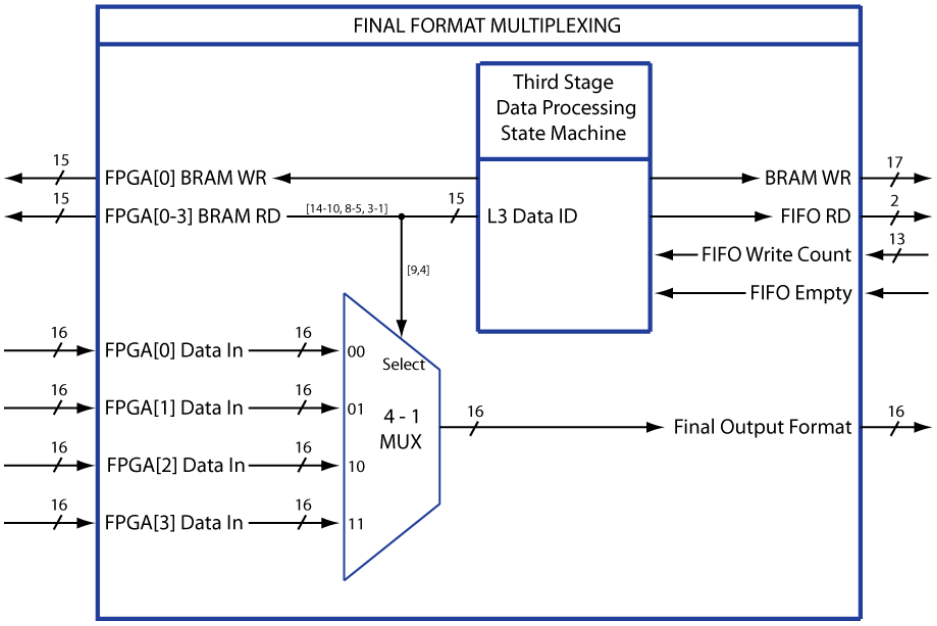

Figure 5-9. Final Format Multiplexing module

The Third Stage Data Processing State Machine block corresponds to the state machine in Figure 5-7. This component provides all the BRAM and FIFO read and write port logic, as well as the L3 Data ID signal. Two L3 Data ID bits (9 and 4) are used for the 4-1 MUX Select signal. The simple binary counter and 4-1 MUX design described here is possible because image data is pre-processed into rows and frames in the FPGA[0-3] Data BRAM modules. This memory mapping lends itself nicely for generating the final output format.

5.3.6 Final Output Format

The final output format for the Image Acquisition System is left-to-right, top-to-bottom, one sample fames at a time starting with frame 0. Figure 5-10 illustrates this ordering, and how it applies to each FPGA's photo-detector array space. The data stream representation near the bottom of the figure identifies data transfer units as FPGA[0-3] rows. For example, the first image data unit transmitted is F[0]r0 – this represents row 0 of FPGA[0]. The second image data unit transmitted is row 0 of FPGA[1]. After FPGA[0] and FPGA[1] transmit row 15, the multiplexing hardware switches to select rows 0 through 15 of FPGA[2] and FPGA[3].

To get a better understanding of how all this relates to the L3 Data ID and the multiplexing hardware, a closer look at Figure 5-6 is required. According to this figure, FPGA[0-3] Data BRAMs store row 0 in memory space 0-15; row 1 in memory space 16-31, and so on. This means that each row occupies a memory space which is addressed by 4 bits. What is more, from Figure 5-6 it is evident that sample frames are stored in successive chunks of 256 memory locations. This means that each frame is addressable with 8 bits.

This understanding of the third stage BRAM memory space lead to the L3 Data ID bit map in Figure 5-8. The S0 bit (bit 4) multiplexes between FPGAs 0 and 1 (or 2 and 3) after one row has been read out. The S1 bit (bit 9) is located after 8 BRAM Memory Address bits, so multiplexing between top and bottom portions of the image takes place after one complete frame has been addressed.

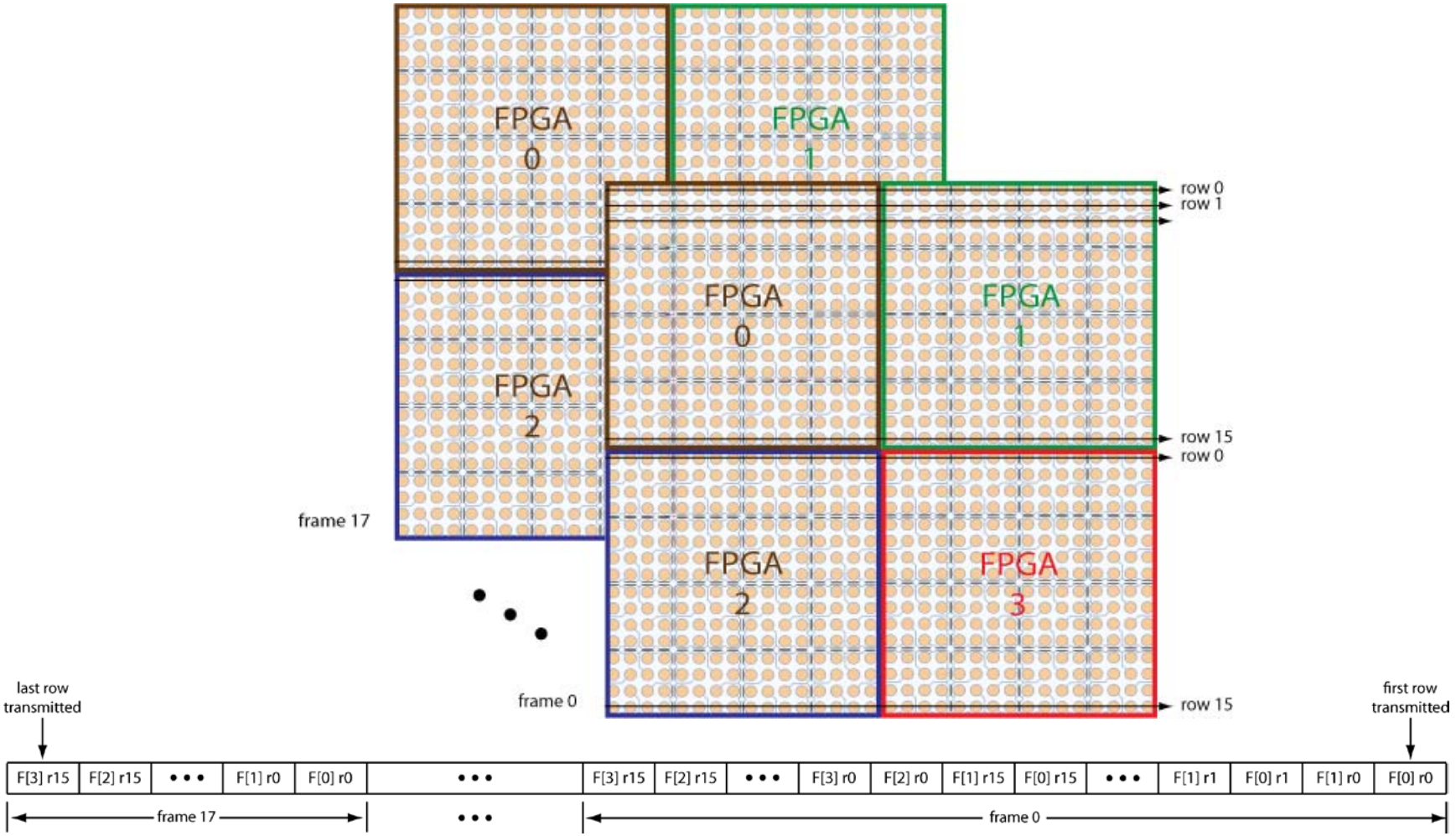

Figure 5-10. IAS Final Output Format

5.3.7 PCIe TX BRAM

The PCIe TX BRAM is required to buffer the final output format data stream, and source image data for PCI Express transmission. The BRAM is a simple dual port Block RAM which was generated using Xilinx's LogiCORE Block Memory Generator. The BRAM read and write ports are 32-bits wide to conform to the PCI Express Root Port example design which is discussed in the next section. The PCIe TX BRAM has the capacity to store all 18,432 image data words generated during one image acquisition cycle. Both the read and write ports require a clock, a port enable, and a15-bit address. Figure 5-11 shows a screenshot of the PCIe TX BRAM configuration in the LogiCORE Block Memory Generator.

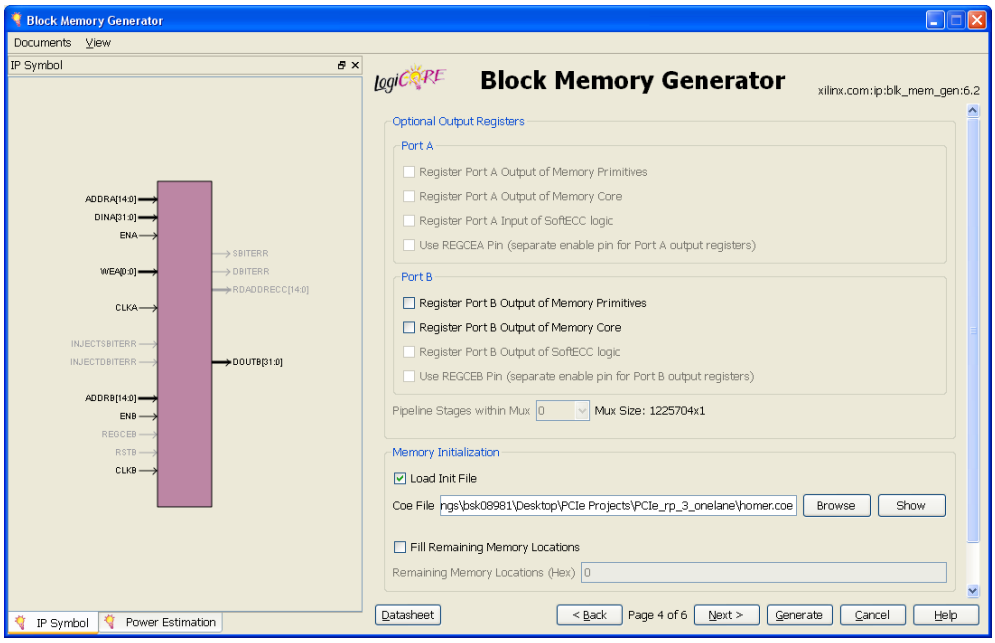

Figure 5-11. Xilinx LogiCORE Block Memory Generator Settings for PCIe TX BRAM

The PCIe TX BRAM module is intended to be the interfacing component between the generated final output format and the PCI EXPRESS TX example design. Consequently, the write port signal to the PCIe TX BRAM are controlled in the FINAL FORMAT MULTIPLEXING module, and the read port signals are controlled by the PIO Master design in the PCI EXPRESS TX module.

The amount of time required for third stage data processing is only dependent upon the amount of time it takes to write image data to the PCIe TX BRAM. Reading form the Aurora TX FIFO (for FPGA[0] Data BRAM) takes place during second stage Aurora 8b/10b transmission. Furthermore, the PCIe TX BRAM is written to during the same clock cycle as FPGA[0-3] Data BRAMs are read from. A phase-locked loop 156.25 MHz GTX reference clock signal serves as both the read and write clock. The third stage data processing time ($T_{\text{Third Stage Data Processing}}$) is calculated next.

$$T_{\text{Third Stage Data Processing}} = \frac{1}{156.25\ x\ 10^6} + \frac{18{,}432}{156.25\ x\ 10^6} = 117.971\ \mu s$$

It takes 117.971 µs to write 18,432 image data words into the PCIe TX BRAM. A one clock delay is included in the calculation to account for reading from FPGA[0-3] Data BRAMs. In all, third stage data processing takes 117.971 µs.

5.4 PCI Express

The PCI Express protocol was selected for transmitting the final IAS output because of its high-speed serial capabilities and ease of integration with the data processing application. According to the Image Acquisition System specifications, the final output stream must be transmitted at a minimum 2 GB/s. The PCI Express (Gen1) protocol specifies a data rate of 2.5 Gb/s, per PCI Express lane [8]. However, like Aurora 8b/10b, the protocol suffers from approximately 20% overhead, primarily from 8b/10b encoding.

$$\text{Aggregate PCIe lane rate} = \underbrace{0.80}_{\text{aggregate \%}} \; x \; \overbrace{2.5\ Gbps}^{\text{PCIe lane data rate}} = 2 \text{ Gbps}$$

The result is an aggregate PCIe lane rate of 2 Gb/s. With this information, the calculation below shows that the IAS must implement, at minimum, an 8-lane PCIe link between itself and the data processing application.

$$\text{Number of PCIe lanes} = \frac{\overbrace{16\ Gbps}^{\text{2GBps IAS requirement}}}{\underbrace{2\ Gbps}_{\text{PCIe lane rate}}} = 8 \text{ lanes}$$

A PCIe link is a point-to-point connection between two PCIe ports. The protocol utilizes dual-simplex links, which means each lane consists of a differential signal pairs for one RX signal and one TX signal [7].

At this point, two comments regarding PCIe in the IAS should be made. First, due to VHDL code size limitations and licensing restrictions on the Xilinx iSIM simulator, PCIe transmission to the data processing application is limited to one lane, and must be separated from the rest of the IAS design. In Figure 5-1, this division can be

characterized by removing the PCI EXPRESS TX module from the block diagram, and making that module a stand-alone project. Nevertheless, the IAS is designed to facilitate integration between the two projects with only a few modifications, and is discussed under that assumption.

Second, it should be noted that the data processing application, to which all image data is transmitted to, is never specified in the requirements. This permits a wide range of testing configuration, including the Root Port configuration example design which is provided when the Virtex-6 IP Core for PCI Express is generated [7]. The IAS utilizes this pre-existing example design to establish its PCIe protocol channel, and transmit image data to the symbolic data processing application.

For Virtex-6 PCIe IP Core configuration settings and instructions for implementation, please reference Chapter 4 of the Virtex-6 FPGA Integrated Block for PCIe User Guide [7].

5.4.1 PCIe Root Port Example Design

To aid in development of applications, Xilinx provides two example designs with the generation of the Virtex-6 FPGA Integrated Block for PCI Express – a Root Port example design, and an Endpoint example design. The PCI Express Root Port example design is used for this project. The details of the Root Port example designs are beyond the scope of this project, particularly because only transmission is required. The design hierarchy, taken from the top-level file, board.vhd, is provided in Figure 5-12.

In summation, the PCIe Root Port example design instantiates a Root Port design which sources data, and an Endpoint design which receives and verifies data. Among the

numerous modules existing in each design, are instances of even more designs. On the Endpoint side, a Programmed I/O (PIO) Slave design is implemented to service memory read and write operations by the Root Port application. On the Root Port side, a Configurator design configures the Endpoint design, and its downstream PIO Slave design. The Root Port design also includes a PIO Master design which initiates all traffic across the established PCIe x1 link [7]. The PCIe TX BRAM module from Figure 5-4 is implemented within the PIO Master design.

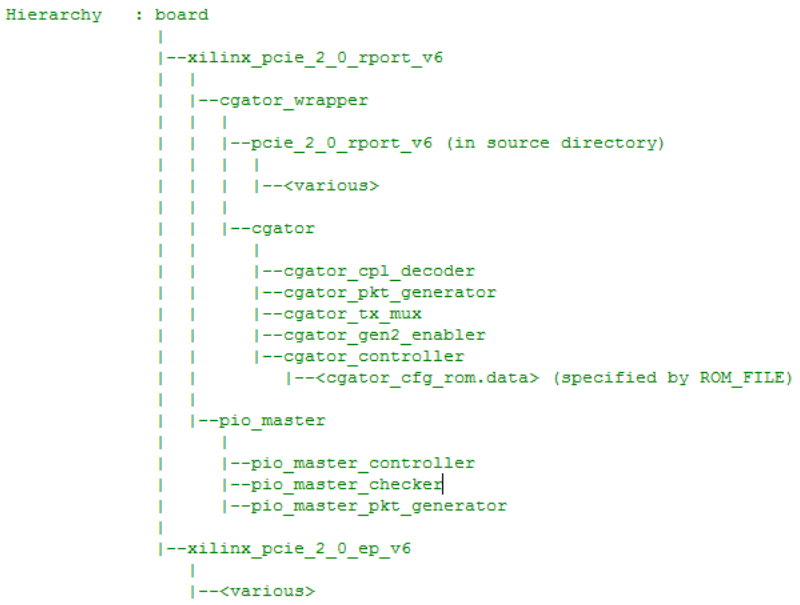

Figure 5-12. Design Hierarchy for the Virtex-6 FPGA Root Port Example Design

5.4.2 PCI EXPRESS TX Module

For consistency with the IAS, the Virtex-6 Integrated Block for PCI Express Root Port example design is referred to as the PCI EXPRESS TX module. As it is provided, the PCI EXPRESS TX module uses the PIO Master design to transmit 32-bit data words to four different memory spaces on the PIO Slave design connected to the Endpoint

design.[27] The PIO Master design then goes on to read from all four memories, and completes the write/read/verify cycle.[28]

Three significant modifications had to be made to the PIO Master design to accommodate the PCI EXPRESS TX module to the IAS. First, the PCIe TX BRAM module was implemented as the only data source for transmission. Second, the state machine which addressed four different memory spaces was modified to address only one memory space. And lastly, the state machine which controlled the write/read/verify cycle was changed to produce read port signals to the PCIe TX BRAM, and to only perform write operations on the Endpoint design.

Although the Endpoint design in the PCI EXPRESS TX module is beyond the scope of this project, several modifications to its PIO Slave design had to be made as well to verify proper functionality of PCI EXPRESS TX module. Most notably, the memory space targeted by the PIO Master in the Root Port design had to be expanded to store image data from the entire 32 x 32 array, and the memory addressing mechanism had to be modified to support 15-bit wide address.

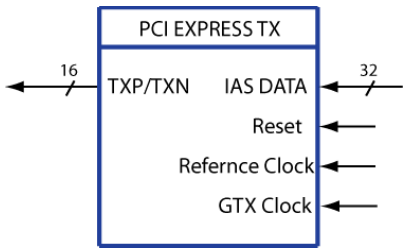

Figure 5-13. PCI EXPRESS TX module

[27] Virtex-6 Integrated Block for PCI Express Root Port Example Design (pio_master.vhd).

[28] Virtex-6 Integrated Block for PCI Express Root Port Example Design (pio_Master Controller.vhd).

The PCI EXPRESS TX module in Figure 5-13 is exceedingly oversimplified due to its isolation from the rest of the IAS. The 32-bit IAS DATA signal represents the image data source signal which delivers final output format data words for PCIe transmission. The 16-bit TXP/TXN output represents the outgoing serial data stream generated by the 8-lane PCIe link. Recall that a PCIe link consists of dual simplex links, however, since the IAS is not designed to receive data from the data processing application, only the outgoing (TXP/TXN) signals are included in Figures 5-1 and 5-13.

5.5 Third Stage Simulation

5.5.1 Final Output Format Simulation

The simulation waveform in Figure 5-14 demonstrates the IAS's ability to produce the final output format. This data stream is buffered on the fpga_0_l3_fifo_wr_data signal shown under the "Final Output Format (FIFO)" divider. Note that there are 36 "image" data bursts on this signal because only module 0 (M0) outputs are generated for each FPGA. Each data burst corresponds to rows 0 through 3 for FPGAs[0-1] and FPGAs[2-3], so two bursts are needed for each of the 18 sample frames. The space between bursts are rows 4 through 15. The four FPGA[0-3] Data BRAM outputs which are multiplexed to produce the final output format are signals fpga_[0-3]_l3_bram_rd_data. In Figures 5-4 and 5-9, these signals correspond to FPGA[0-3] Data In inputs.

The simulation also identifies the three Aurora 8b/10b RX channels through which image data for the 32 x 32 photo-detector array is centralized in FPGA[0]. These signals are fpga_[1-3]_aurora_rx_data and fpga_[1-3]_aurora_rx_valid.

When the simulation output is zoomed-in on one of the 36 data bursts from Figure 5-14, 8 sub-bursts are shown. There are 8 of these sub-bursts because they correspond to image data for rows 0 through 3 for FPGA[0] and FPGA[1], in frame 0. Figure 5-15 shows these sub-bursts; reference Figure 4-10 for a better grasp of this concept. Zooming-in further still, Figure 5-16 shows the start of the final output format data stream. As expected the first image data words for row 0, of frame 0, are 0x0 through 0x3. This is shown twice because these are outputs from FPGA[0] and FPGA[1]. The beginning of the second row is also shown, starting with image data word 0x4.

The start of row 0 in frame 17, the last frame, is shown in Figure 5-17. Recalling how the ADS5281 Chip Model generates outputs, the expected output of 0x11 (17) through 0x14 (20) is observed on the final output format data stream signal. Lastly, Figure 5-18 shows that the last L3 Data ID count is 0x47FF (18,431); this is of course the number of image data words in the IAS's final output format.

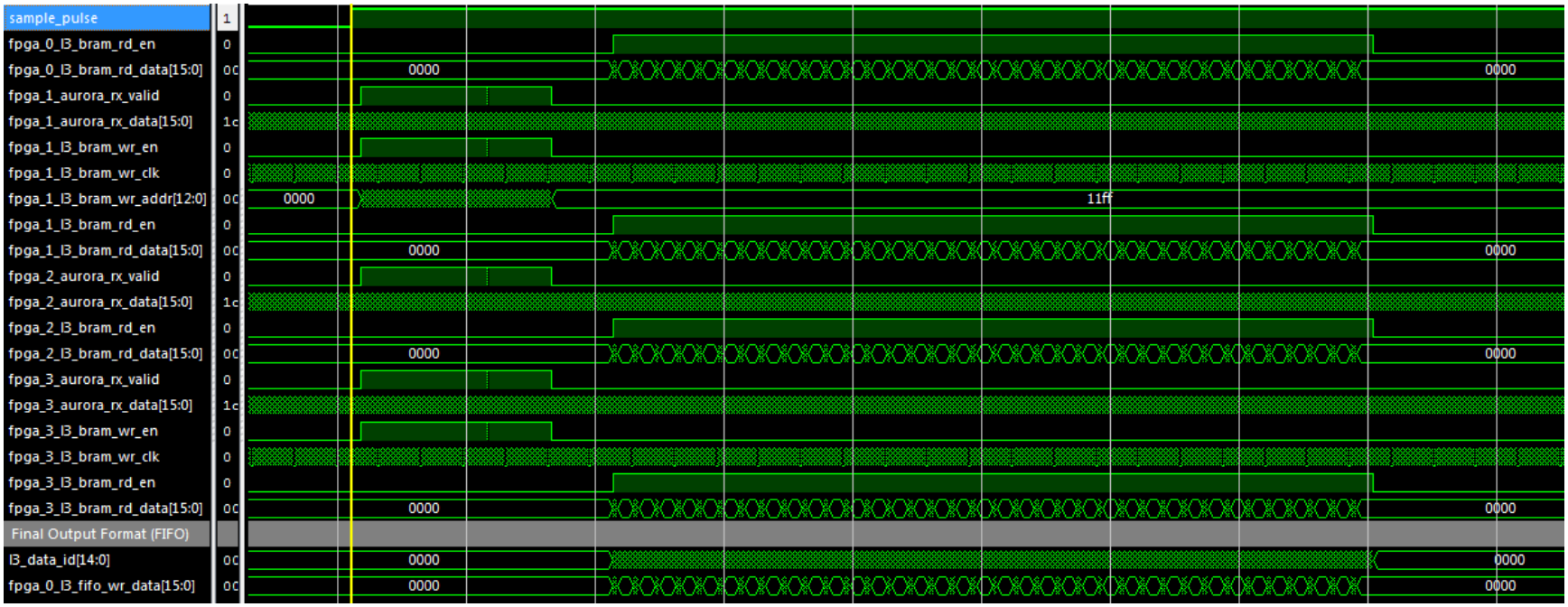

Figure 5-14. Final Output Format Simulation

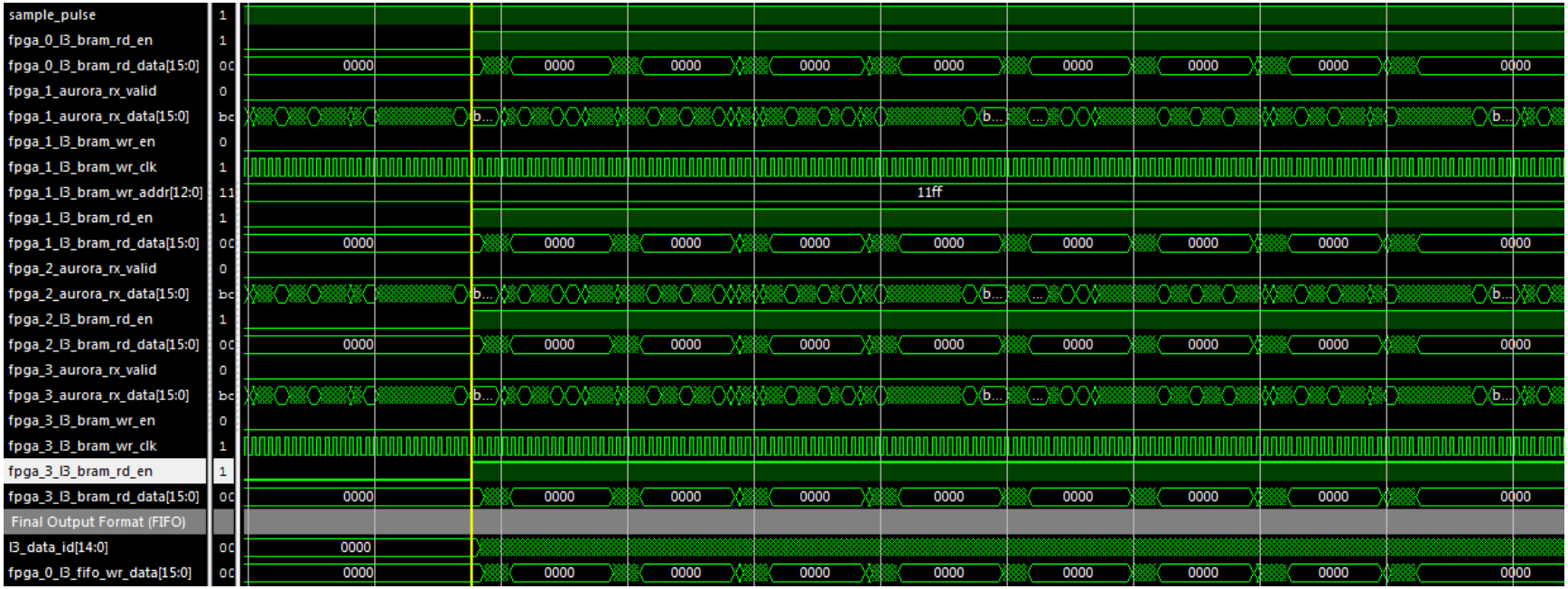

Figure 5-15. Final Output Format Simulation (rows 0-3 for FPGAs[0-1] of frame 0)

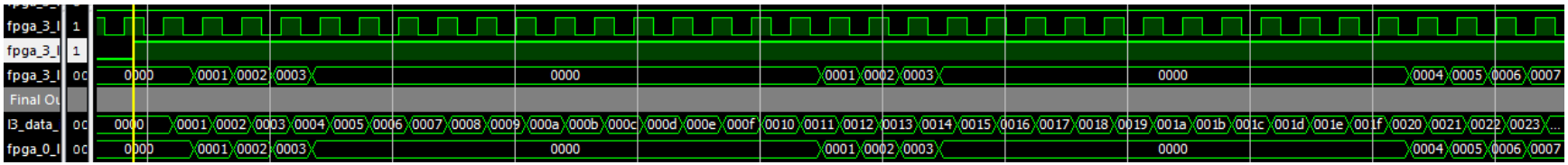

Figure 5-16. Final Output Format Simulation (row 0 and start of row 1 of frame 0)

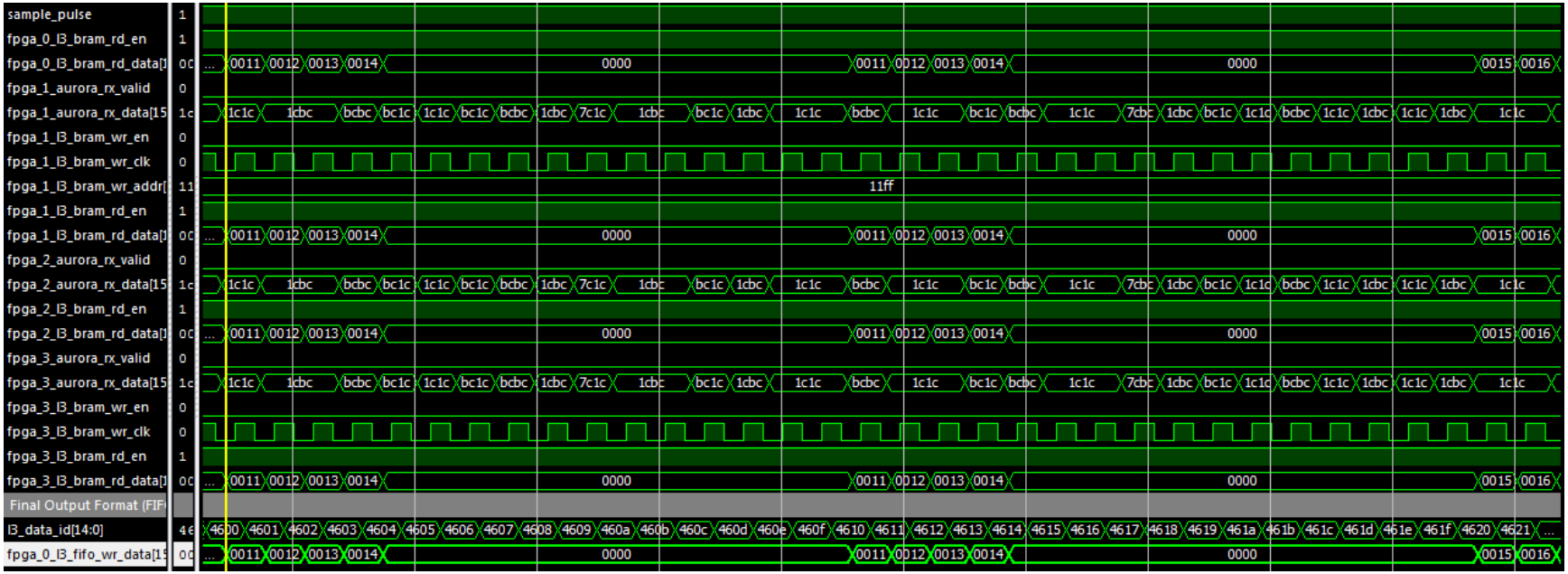

Figure 5-17. Final Output Format Simulation (row 0 and start of row 1 of frame 17)

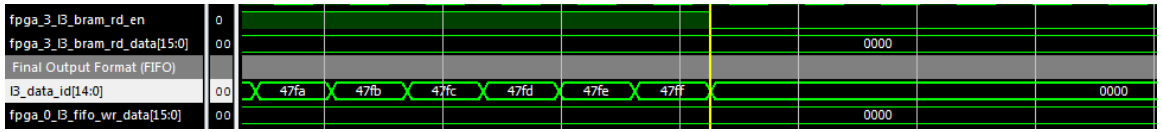

Figure 5-18. Final Output Format Simulation (last L3 Data ID count)

CHAPTER 6

CONCLUSION

The Image Acquisition System described in this graduate project is an implementation-ready design which fulfills all the specifications and requirements given in Chapter 2. The following timing calculation verifies that the IAS carries out all the required functionality well within the specified 1 ms time-frame. It takes the Image Acquisition System a total of 173.229 µs to complete one image acquisition cycle.

Table 6-1. Total Time for IAS

IAS Time Constants	Time (µs)
$T_{\text{Data to FPGA}}$	2.517
$T_{\text{Second Stage Data Processing}}$	9.914
$T_{\text{Aurora Transfer}}$	29.491
$T_{\text{Aurora Transfer Latency}}$	0.333
$T_{\text{Third Stage Data Processing}}$	117.965
$T_{\text{Transmit Data}}$	12.875
$T_{\text{PCIe Transfer Latency}}$	0.134
T_{Total}	173.229

The seven IAS time constants included in this calculation have been calculated throughout the report. $T_{\text{Data to FPGA}}$ is the time it takes to digitize photo-detector image data. The considerations for this time constant are discussed in Chapters 2 and 3. $T_{\text{Second Stage Data Processing}}$ is the time it takes FPGAs[0-3] to complete the first stage of data process. $T_{\text{Aurora Transfer}}$ is the time required for FPGAs[1-3] to transmit their image data to FPGA[0] via Aurora 8b/10b protocol channels. Both of these time constants are explained in greater detail in Chapter 4. One subject not discussed in Chapter 4, however, is the expected latency of data transmission on the Aurora 8b/10b channel.

According to the Aurora 8b/10b Product Guide, there is a maximum latency of 52 user clock cycles due to pipeline stages in the GTX Transceiver and Aurora protocol engine [2]. In the IAS, the interface between Aurora 8b/10b (and PCIe) and data processing modules is synchronized with the 156.25 MHz GTX Transceiver reference clock. Accordingly, the Aurora 8b/10b transfer latency, $T_{\text{Aurora Transfer Latency}}$, is 0.333 µs. In total, Aurora 8b/10b transmission takes 29.824 µs.

$$T_{\text{Aurora Tranfer Latency}} = \frac{52}{156.25\ x\ 10^{6}} = 0.333\ \mu s$$

Moving further down the digital datapath, $T_{\text{Third Stage Data Processing}}$ is the time it takes the IAS to process all image data captured during one Sample Pulse, into the final output format. Chapter 5 describes the mechanism responsible for this process. Notwithstanding, the total time the IAS spends processing image data into the final output format, is a sum of data processing at the second and third stages, as well as Aurora 8b/10b transmission time. The IAS spends a total of 157.703 µs (9.914 µs + 29.824 µs + 117.965 µs) processing image data generated during one image acquisition cycle.

The time required to transmit image data captured during one image acquisition cycle, is $T_{\text{Transmit Data}}$; this value was calculated in section 2.2.2. Similar to Aurora 8b/10b, the final PCIe transmission suffers from an additional TX and RX pipeline latency of 21 user clock cycles, or 0.134 µs [9].

$$T_{\text{PCIe Tranfer Latency}} = \frac{21}{156.25\ x\ 10^{6}} = 0.134\ \mu s$$

Consequently, PCI Express transmission to the data processing application requires 13.01 µs in total.

The IAS design described in this graduate project presents the VHDL hardware modules which are required to configure the ADS5281 devices, perform data processing at the second and third stages, and establishing the Aurora 8b/10b and PCIe protocol channels. The VHDL modules are written in the Xilinx ISE design environment, version 13.3. As previously described, the PCI Express portion of the design is kept separated from the rest of the IAS. The PCIe TX BRAM serves as the interfacing component between the two designs.

The use of zero padding throughout the design is implemented to simplify the two high-speed serial interfaces – Aurora 8b/10b and PCI Express. The intention was to successfully simulate a working design, and then work backwards to improve the efficiency by modifying the memory modules which supply data for Aurora 8b/10b and PCIe transmission. However, since the IAS accomplishes the image acquisition cycle in just 173.229 µs, these changes are not required. As it exists, the IAS produces 32-bit image data words to the data processing application; only the 12 least significant bits contain actual image data. It is understood that this is quite inefficient, but the data packing scheme was accepted since the IAS performs significantly faster than required.

The accomplishment of the IAS design was not without many difficulties and complications. Among these were the countless hours spent debugging, and a very significant setback due to computer memory limitations on the host PC, Xilinx iSIM simulator licensing restrictions, and excessive time spent waiting for simulation results.

In retrospect, PCI Express would not have been chosen as the final image data transfer protocol. The simpler and equally effective Aurora 8b/10b protocol would have been selected, particularly since Aurora 8b/10b had already been established and verified to be working. Altogether, the IAS design presented in this graduate project has been successfully simulated by the author, and has the capability for direct implementation in hardware.

WORKS CITED

[1] K. D. Kihm, "Laser Speckle Photography Technique," *Advances in Heat Transfer*, vol. 30, pp 255-311, 1997.

[2] Xilinx Incorporated, "Aurora 8B/10B Protocol Specification version 2.2," (SP002), April 19, 2010. www.xilinx.com

[3] R. Budruk, D. Anderson, and T. Shanley, *PCI Express System Architecture*. PC System Architecture Series. Mindshare Incorporated, J. Winkles, Ed. Boston: Addison-Wesley Developers Press, 2003, chapters 1-2.

[4] Texas Instruments Incorporated, "12-bit Octal-Channel ADC Family Up to 65MSPS," ADS5281 datasheet, (SBAS397H), December 2001 [revised March 2008]. www.ti.com

[5] Texas Instruments Incorporated, "High-Speed Differential Line Receivers," SN65LVDS386 datasheet, (SLLS394H), September 1999 [revised May 2007]. www.ti.com

[6] Xilinx Incorporated, "LogiCORE IP Aurora 8B/10B v8.1 User Guide," (UG766), April 24, 2012. www.xilinx.com

[7] Xilinx Incorporated, "Virtex-6 FPGA Integrated Block for PCI Express User Guide," (UG671), January 18, 2012. www.xilinx.com

[8] Xilinx Incorporated, "LogiCORE IP Virtex-6 FPGA Integrated Block v2.5 for PCI Express Product Specification", (DS800), January 18, 2012. www.xilinx.com

[9] Xilinx Incorporated, "Virtex-6 FPGA GTX Transceivers v2.6 User Guide," (UG366), July 27, 2011. www.xilinx.com

[10] A. Athavale and C. Christensen, "High-Speed Serial I/O Made Simple A Designers' Guide, with FPGA Applications," (PN0402399), Xilinx Incorporated, Connectivity Solutions Edition 1.0, 2005.

[11] ARM, "AMBA 4 AXI4-Stream Protocol Specification Version: 1.0," (ID030510), ARM, 2010. www.arm.com

[12] Xilinx Incorporated, "LogiCORE IP Aurora 8B/10B v8.2 Product Guide," (PG046), July 25, 2012. www.xilinx.com

[13] Xilinx Incorporated, "LogiCORE IP FIFO Generator v8.1 User Guide," (UG176), March 1, 2011. www.xilinx.com

[14] Xilinx Incorporated, "ML605 Hardware User Guide," (UG534 v1.6), July 18, 2011. www.xilinx.com