USB 3.0-based Super Speed Data Acquisition Systems

Team:

Sathyasri S
BL.EN.U4ECE18138

Shariq Akhtar P.P.
BL.EN.U4ECE18140

Talluri Sayanth Kishore
BL.EN.U4ECE18157

Yash Rajesh Umale
BL.EN.U4ECE18182

Project Guides:

Dr. Navin Kumar
HoD, Dept of ECE, ASEB

Ms. R Srividhya
Head, DSAS, DSD, PDMG

Mr. Srinidhi M
Engr SF, DSAS, DSD, PDMG

Mr. Om Sai Ayyappa Reddy
Sci/Engr-SD, DSAS, DSD, PDMG

Agenda

- **Introduction and Motivation:** Requirement for Super Speed Data Acquisition
- **Objectives:** Deploy a firmware to implement Slave FIFO
- **Design and Analysis:** Configuring GPIF II, Implementation Logic and FX3 Firmware Application Structure
- **Results and Discussion:** I/O matrix configuration, Slave FIFO interface state machine, Read-write operations, firmware application
- **Conclusion:** A firmware built to implement the 2 – bit Slave FIFO mechanism, and to verify bulk – IN and loopback transfers across an FPGA

Introduction

- The process of sampling signals that measure real-world physical conditions and converting the resulting samples into digital numeric values [1] that can be manipulated by a computer or sensors is known as **data acquisition**.

- Data acquisition systems, or DAQ, are critical in product testing. With the invention and development of data acquisition systems (DAQs) capable of collecting data from a wide range of sensors.

- **DAQ is classified into two types:**
  - Analog Data Acquisition Systems
  - Digital Data Acquisition Systems

Introduction

- Space exploration (unmanned/ manual) involve logging data using **magnetic tape recorders.**
  - Limited data gathering and storage capabilities
  - Mechanical systems prone to failure

- **Solid-state technology** was found as an alternative for these problems.
  - Information stored in **binary** [1] can only be altered by **voltage changes** in transistors
  - Highly reliable, durable and compact.
  - Availability of multiple interfaces for **telecommand** and **telemetry**.

Introduction (Contd.)

- Solid state recorders (**SSRs**) are developed to [2]
  - Store data during imaging or continuous **recording**, and then retrieve during ground station visibility (**playback**)
  - Monitor during remote sensing and experimental missions
  - Store data and health parameters from multiple flight payloads

- The project focusses on
  - developing a robust **end – to – end system** to **evaluate** the flight model of SSRs
  - ensuring **SuperSpeed data recording and playback** with implementation of compression algorithms among other features using a custom board simulator.

Objectives

Main Objective:

To build and deploy a **system image/ firmware** which implements super-speed data transfers between **USB**, **GPIF** and **SPI interfaces** using the **2 – bit Slave FIFO mechanism**, and to verify **bulk – IN** and **loopback transfers** across an external master (FPGA or ASIC).

Objectives

Sub Objectives:

- Configuration of the **GPIF state machine** and I/O matrix.
- Provide **DMA transfer** of data for record and playback within the FX3.
- Testing the GPIF state machine with timing diagrams, streamer module and USB Control Utility.
- Testing **Bulk-IN data transfers** via the FX3 to the FPGA or USB host.
- Verifying the **loopback transfer** of data.

Methodology

- The **Cypress FX3 SuperSpeed Explorer Kit** offers a one-package solution for the integration of all aforementioned interfaces. It supports **SuperSpeed** data transfers over **USB 3.0** to other peripherals (**GPIF**, **SPI**, **UART**) within the kit.

- The kit uses the **AHB architecture** for internal DMA and system data transfers, along with the GPIF and SPI interfaces for our application.

- The project focusses on developing a firmware for the FX3 to facilitate data transfer between peripherals. The adjoint FPGA will be programmed to run tests on the SSR over multiple interfaces to gauge its reliability.

Literature Survey

[1] Y. J. Qian and K. Cui, **"Design of high speed CCD data acquisition system based on FPGA and USB3.0,"** *2015 International Conference on Information and Communication Technology Convergence (ICTC)*

- Investigates building a framework for a **test bench** that is inexpensive, scalable, open source, and easily implemented.
- Reviews modern protocols to rely on **USB to provide the physical link** and data protocol to interface an **FPGA to a PC**.
- Each **state** is programmed to perform certain **actions**, and the conditions are checked on each GPIF II clock edge.
- The FX3 device has an **internal DMA fabric** that connects the GPIF interface to the internal system memory.

Literature Survey

[2] Y. Gong and F. Yu, **"Design of high-speed real-time sensor image processing based on FPGA and DDR3,"** *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*

- FPGAs play a very important role in **sensor data acquisition** and **processing** with its rich interface resources and excellent parallel processing ability.
- Color image processing by a traditional PC cannot achieve the maximum video display frame rate due to high speed real-time requirements.
- Observations show that the FPGA-based video pipeline processing and DDR3 frame cache application can **increase the video frame rate** and **improve the transmission stability** of video data.

Literature Survey

[3] Tian, Fenxian et al. **"Design and Implementation of USB3.0 Data Transmission System based on FPGA."** *3rd International Conference on Computer Engineering, Information Science & Application Technology, 2019*

- Emphasis on a system using an FPGA as the core control chip, due to which the FX3 realizes high speed data transmission of the USB 3.0 **synchronous Slave FIFO** mode through the **GPIF** pins.
- The FX3 SDK is built and shipped along with the microcontroller to ease the design and development of the firmware.
- The GPIF II Designer, Eclipse IDE, **ThreadX RTOS**, USB Control Manager and other design tools are third-party software wrapped in Cypress APIs to enable native integration with the hardware.
- A USB device can implement one or more functions, and the capabilities of the device are reported to the host through a set of data blocks called **descriptors**.

Literature Survey

[4] C. K. Adithya Rangan, K. Aravinda Holla, V. Kulkarni, A. Kumar and A. Patil, **"Data Rate Based Performance Analysis and Optimization of Bulk OUT Transactions in USB 3.0 SuperSpeed Protocol,"** *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*

- Data rate based performance is used to verify if the design and implementation of USB 3.0 conforms to the protocol standards and system requirements.
- This paper presents an elaborate description of various facets through which data rate based performance analysis is achieved for USB 3.0 SuperSpeed **Bulk OUT** transactions.
- Deals with the characterization and statistical variation of causative factors for different scenarios to obtain **minimum**, **maximum** and **ideal data rate** thus leading to optimum performance.
- Challenges faced include synchronization issues or wrong identification, resulting in misleading results.

Literature Survey

[5] Minyeol Seo et al. **"An Effective Design of Master-Slave Operating System Architecture for Multiprocessor Embedded Systems,"**
Advances in Computer Systems Architecture, 12th Asia-Pacific Conference, ACSAC, 2007

- Explores the problem of designing an effective **master-slave** operating system architecture for multiprocessors.
- Identifies and addresses design issues that have significant impact on the **functionality** and **performance** of the master-slave approach.
- Investigates three major issues that are closely related to the architecture and performance of the master-slave approach:
  - structural design of a master-slave system
  - functional design of remote invocation mechanism
  - inherent architectural problem of the master-slave approach

Literature Survey

[6] J. G. V. Leañez et al., **"High-Speed Data Acquisition System for GNSS Applications,"** *2020 Argentine Conference on Electronics (CAE), 2020*

- Presents the design and implementation of a versatile high speed data acquisition system for **GNSS signals** capable of transferring large amounts of data to a PC.
- **Global Navigation Satellite System** (GNSS): a constellation of satellites providing signals from space that transmit positioning and timing data to GNSS receivers.
- Implemented on a **Xilinx Virtex UltraScale+ FPGA** with a high-throughput data path through the on-board DDR4 SDRAMs, based on which a ring buffer is designed to provide a 2 GB buffering capacity.
- In addition, a simple **differencing algorithm** with threshold detection is integrated into the back end for dynamic frame rate operation.

Literature Survey

Summarising the papers

Ref. No.	Observations	Problems
[1]	Built a framework for a **test bench** that is inexpensive, scalable, open source, and easily implemented.	The design goal of 3.2 Gbit/s is restricted by Slave FIFO mode (32 bit,100MHz max clock frequency).
[2]	**Increased video frame rate** and **improved transmission stability** of video data with FPGAs.	Incessant writing and reading can cause pointers to turn around and overlap, resulting in the loss of writing data or the invalidation of reading data.
[3]	Obtained a system capable of interfacing an FPGA with a host PC using SuperSpeed (USB 3.0)	The design part of the DDR3 controller and the USB3.0 interface in this system are the key points of difficulty.

Literature Survey

Summarising the papers

Ref. No.	Observations	Problems
[4]	Description of various facets through which data rate based performance analysis is achieved for USB 3.0 SuperSpeed **Bulk OUT** transactions.	Synchronization issues or wrong identification, resulting in misleading results.
[5]	Identifies and addresses design issues that have significant impact on the **functionality** and **performance** of the master-slave approach.	Tasks involving a significant portion of kernel mode operations can make the master a critical bottleneck; especially with multiprocessors.
[6]	A versatile high speed data acquisition system for **GNSS signals** capable of transferring large amounts of data to a PC was designed.	Absence of a 1553 Bus Controller, relying solely on orthogonally polarized antennas with polarization deficiencies.

Design and Analysis

- **Configuring the GPIF – II I/O Matrix**

  The **GPIF II Designer** is used for designing the **IO matrix**, which is responsible for assigning the **mapping of pins** as per the **slave FIFO** interface requirements.

- A **state machine** is designed using the tool in order to define the sequence of commands and execution, while ensuring conditional **state transitions based on signals** from an external processor.

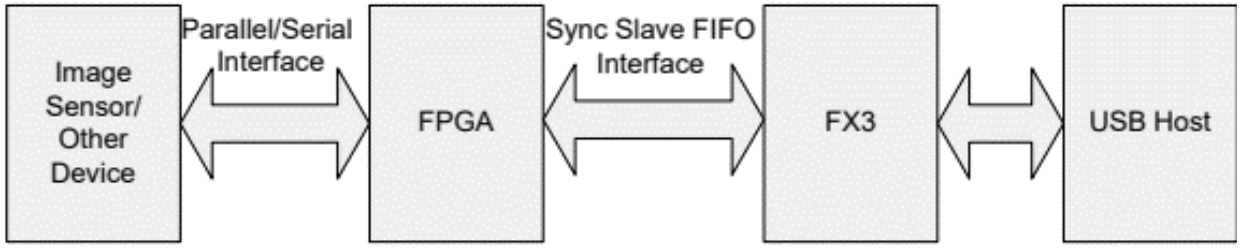

Fig. 1: Slave FIFO for transactions between the FX3 and an external master

Design and Analysis (Contd.)

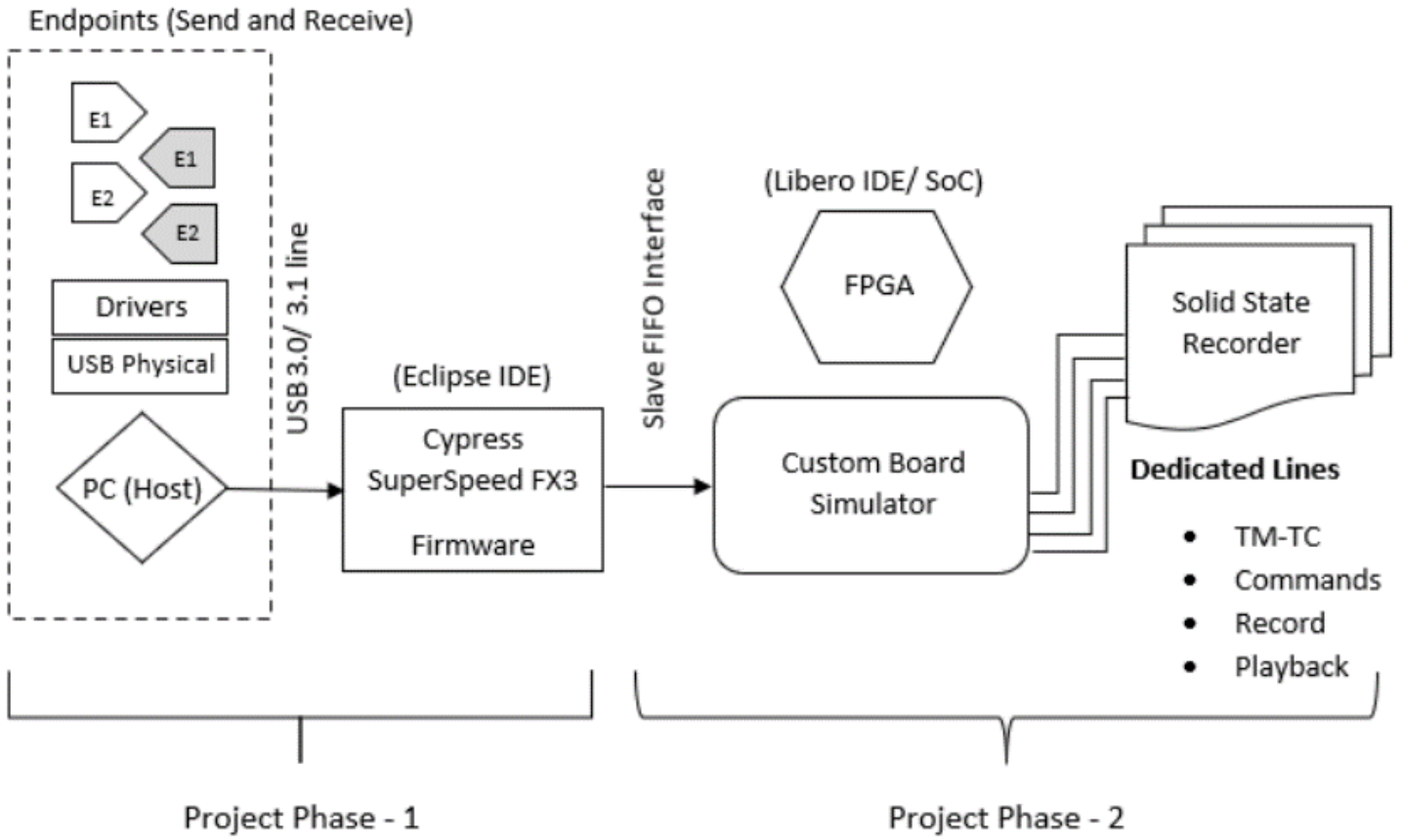

Fig. 2: General proposed architecture for the project

Design and Analysis

Configuring the I/O Matrix

The interface definition window and I/O matrix helps define the electrical interface in terms of:

- **data** and **address buses**
- the number and **direction of control signals**
- the interface clocking parameters

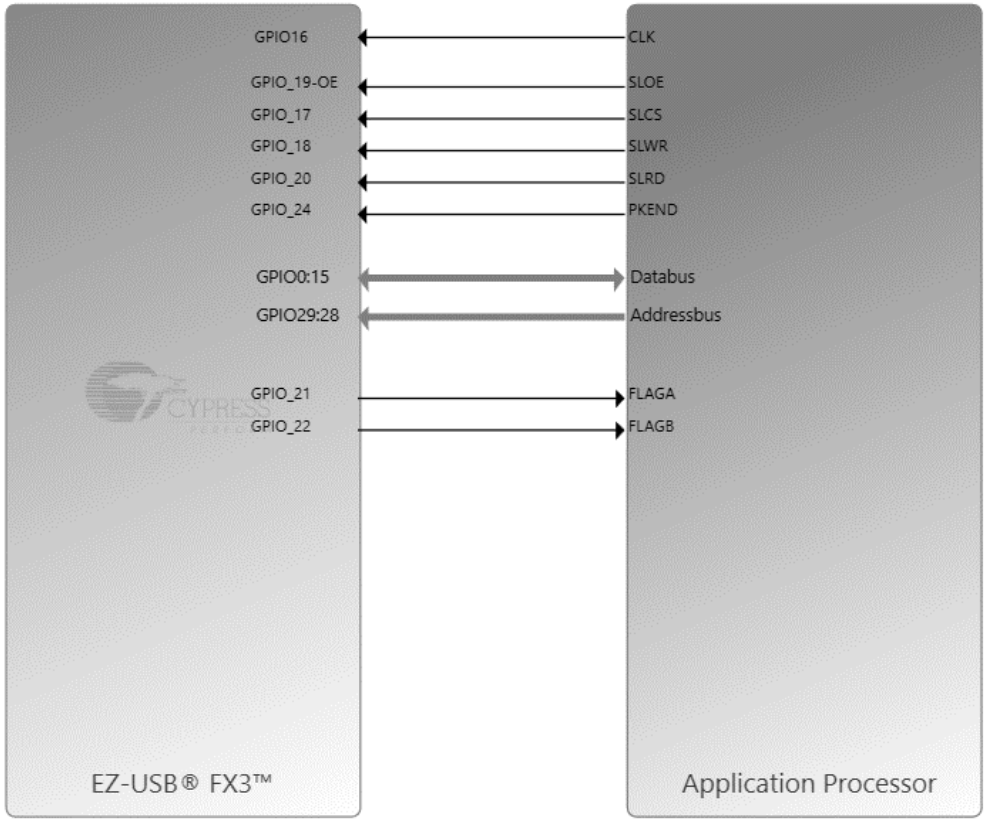

Fig. 3: I/O Matrix Configuration

Design and Analysis

Configuring the I/O Matrix

The I/O matrix helps in configuring

- FX3 peripherals such as I2C, UART, SPI
- Data **bus width**
- Endianness
- **Flags** to indicate the **status of the thread** in terms of current buffer conditions
- **Pin mappings** as per the desired interface

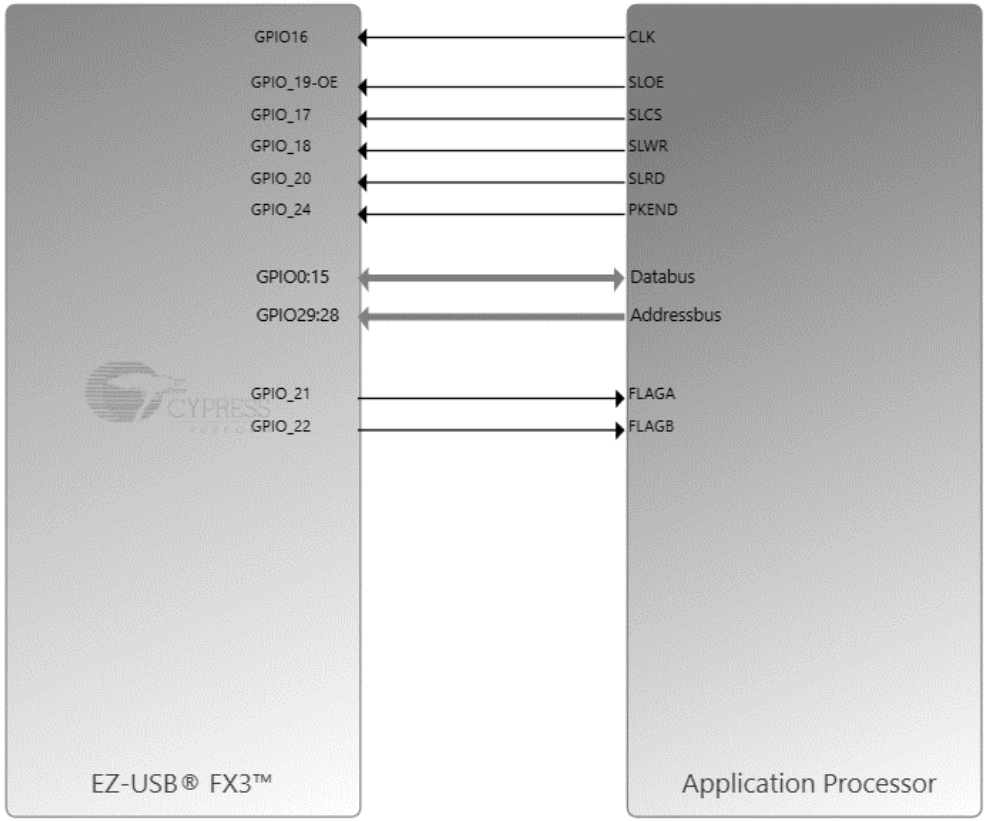

Fig. 3: I/O Matrix Configuration

Design and Analysis

Programming the GPIF II State Machine

- The GPIF II port provides a parallel interface with maximum of **32 bidirectional data lines**. They can be **time multiplexed** into address lines.

- The GPIF II Designer allows the user to
  - Create an **I/O matrix** for mapping pins to signals
  - Defining state machines for specific interfaces
  - **Generate a C header** file encoding all of the GPIF definitions

- The header file can be used to integrate the GPIF functionality into the slave FIFO application firmware.

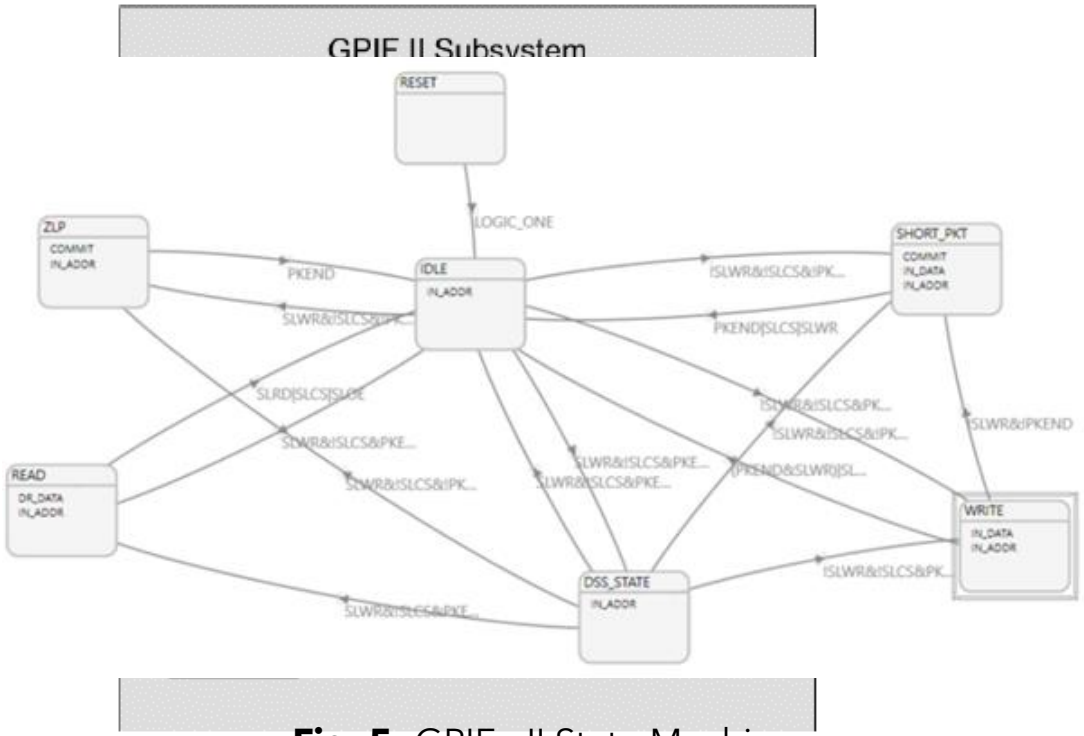

Fig. 5: GPIF - II State Machine

Fig. 4: GPIF - II Subsystem

Design and Analysis

Logic Diagram for Synchronous Slave FIFO:

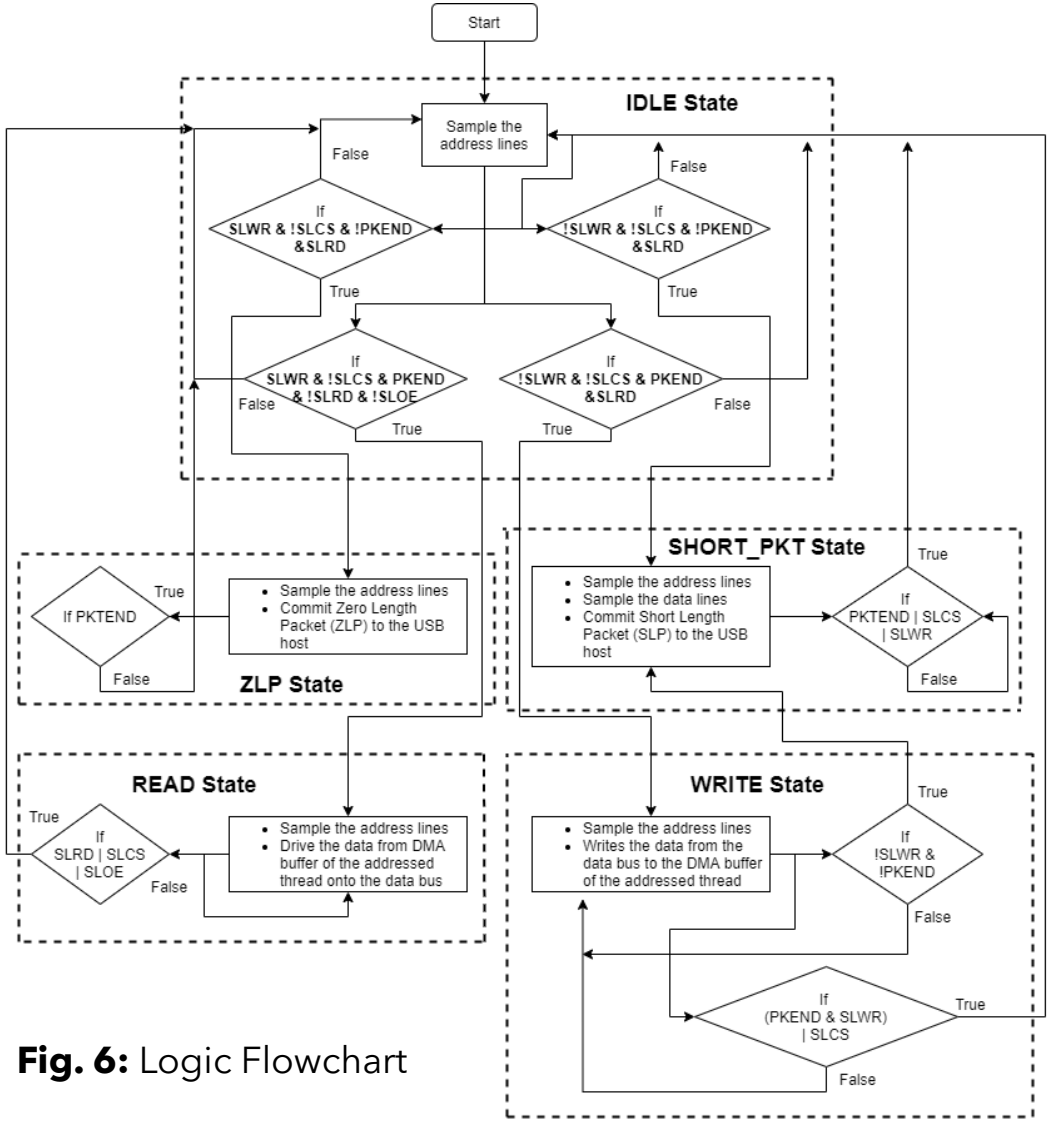

Fig. 6: Logic Flowchart

In order to configure the GPIF II interface, the following initial states need to be defined:

- **Interface type:** Slave
- **Communication type:** Synchronous
- **Endianness:** Little Endian
- **Data bus width:** 8 bit, 16 bit or 32 bit
- Address or data bus multiplexed
- **2 address pins** for selecting among 4 available threads
- **Active low signals** for **Read, Write, Chip Select** and **Output Enable.**

Design and Analysis

Implementation Logic

- The project consists of an **FX3 Development Kit** interconnected with an **evaluation kit (FPGA)** using a **Samtec to FMC interconnect board**.
- The interconnect board mates with the Samtec connector on the FX3 and the FMC connector on the FPGA.
- For execution and testing of the slave FIFO interface, the following components are included in the firmware.

Design and Analysis

Implementation Logic

Streaming (IN) data transfer

The FPGA does one-directional transfers: continuously writes data to FX3 over synchronous slave FIFO.

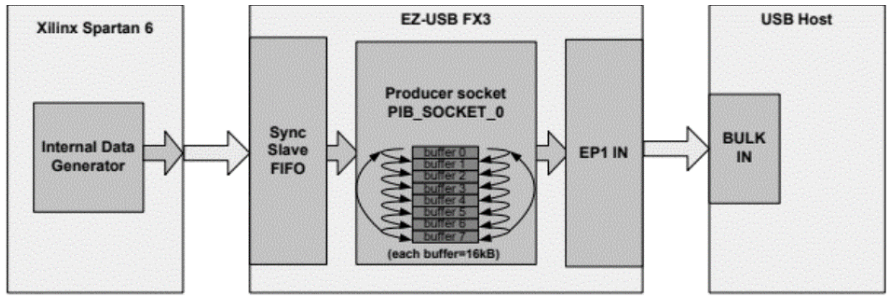

Fig. 7: Stream IN Transfer

Streaming (OUT) data transfer

The FPGA does one-directional transfers: continuously reads data from FX3 over synchronous slave FIFO.

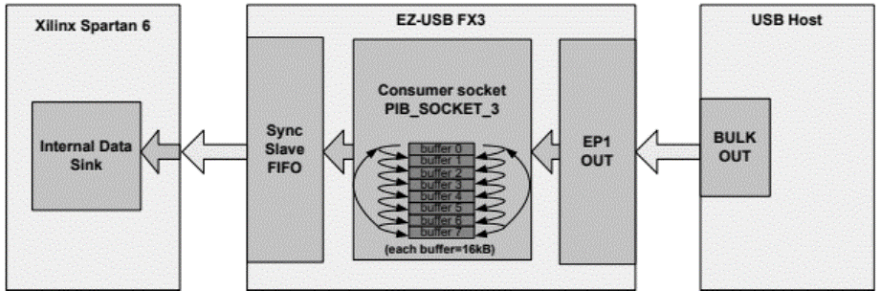

Fig. 8: Stream OUT Transfer

Design and Analysis

FX3 Firmware Application Structure

- All FX3 firmware applications consist of two parts:
  - **Initialization code** – Usually common for all applications
  - **Application code** – Code specific to the Slave FIFO application

- The typical file structure for the firmware application would be as follows:
  - A **header file** containing the **GPIF II descriptors** for 16-bit and 32-bit slave FIFO interface
  - A C program containing **USB descriptors**
  - A **header file** containing defines used in the main application. Several important constants essential for setting the initial conditions for execution are defined in this file.
  - A C program containing the **main application logic for synchronous Slave FIFO**.

Design and Analysis

Initialization Code

As part of the device initialization:

- The CPU clock is setup.
- **Vectored Interrupt Controller** (VIC) is initialized.
- Device caches are enabled (**8 kB data cache** and **8 kB instruction cache**)
- I/O's are configured based on the header files included.
- The inherent **ThreadX RTOS** is invoked, and the OS timer is initialized before transferring the control over to the RTOS scheduler.

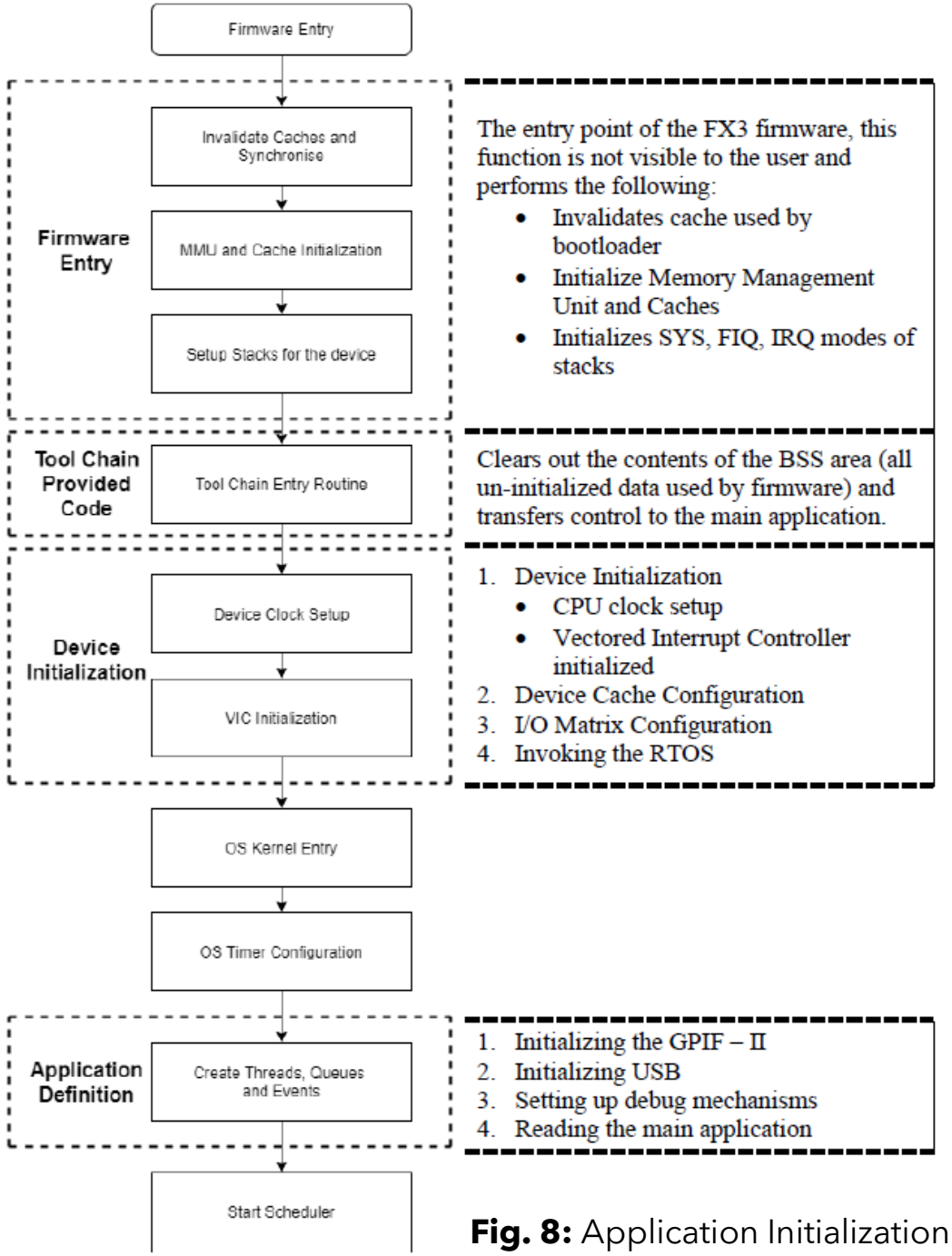

Fig. 8: Application Initialization

Design and Analysis

Application Code

The Slave FIFO application code comprises of three parts:

- The **application thread** function to perform application – specific initialization operations.
- The **application start** function to set up endpoints and DMA channels required for USB – GPIF data transfers.
- The **application stop** function to free up DMA channels and disable USB endpoints when a USB reset or disconnect is detected.

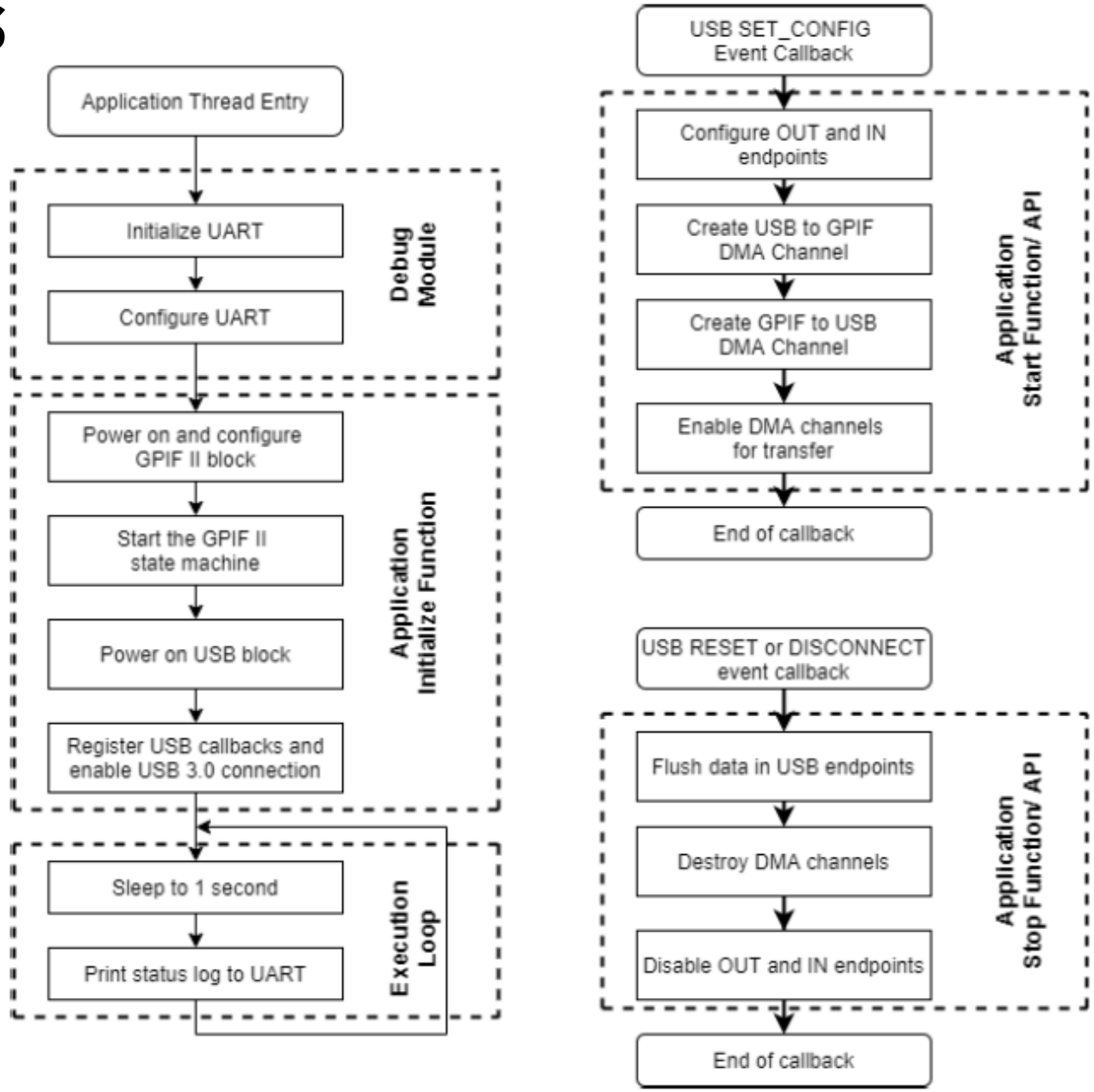

Fig. 9: Thread Entry and USB Events

Design and Analysis

Application Thread

Initializing the debug mechanism

The debug module uses the **UART** to output debug messages. The UART

- needs to be configured before the debug mechanism is initialized.
- needs to be initialized with the requisite parameters.
- has the **transfer size** set to infinite

The debug module is hence initialized, with two main parameters: the destination for the debug prints (UART socket) and the **verbosity** of the bug.

Initializing the main slave FIFO application

The application initialization consists of the following steps:

- **GPIF II Initialization**
- **USB Initialization**
  1. Initializing the USB stack
  2. Registering call-backs
  3. Defining USB descriptors

Results and Discussions

- USB was chosen as the high-speed data link between the FPGA and PC to leverage the simplicity of a VCP (virtual communication port).
- The **FX3 SuperSpeed Explorer Kit** integrates all of the different operations that interface with the SSR (telemetry, telecommand, etc.) onto a single unit, coupling the advantage of USB 3.0's high speed data transfer mechanism.
- As a precursor to testing solid state recorders, we first implement the Slave FIFO interface to the GPIF pins of the FX3 and configure the microcontroller as explained previously.

Results and Discussions (Contd.)

To summarize the work and results for phase 1 of the project:

- The GPIF II Designer is used to configure the I/O matrix
- The state machine for slave FIFO interface is designed and commissioned for integration with the FX3 firmware
- Read-write operations are observed with the configured GPIF interface as functions of time and input control signals
- A comprehensive firmware application is designed to handle USB/ GPIF callbacks and events, along with DMA configurations, descriptor definitions, thread and process handling along with setting up debug mechanisms.

Results and Discussions

Slave FIFO - Read Sequence

The external master drives the two-bit address on the ADDR lines and asserts the read or write strobes.

As observed:

- The FIFO address is stable and **SLCS** (chip select) is asserted
- **SLOE** (output enable) is asserted, and it signals the FX3 to drive the data bus.
- **SLRD** (read) is asserted.

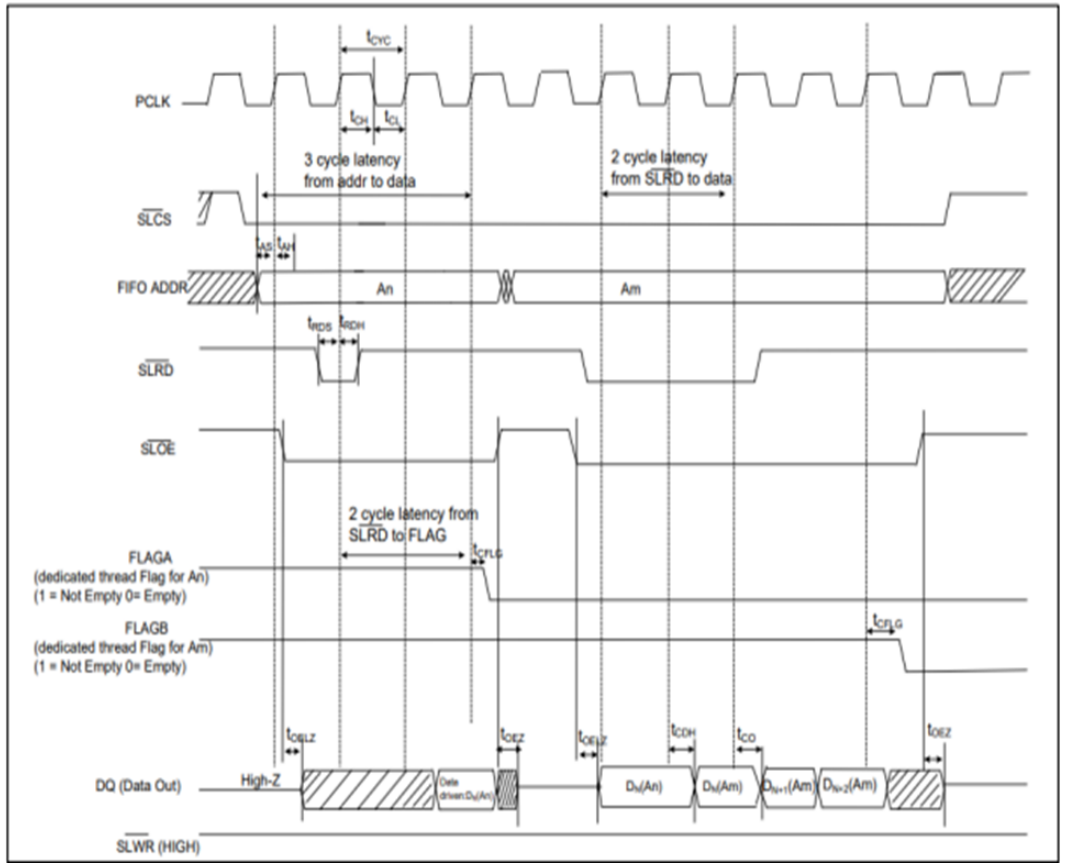

Fig. 10: Read Sequence – Timing Diagram

Results and Discussions

Slave FIFO - Write Sequence

- FIFO address is stable and the signal **SLCS** (chip select) is asserted.
- External master/ peripheral outputs the data onto the data bus.
- **SLWR** (write) is asserted.
- While **SLWR** is asserted, data is written to the FIFO. On the rising edge of the clock, the FIFO pointer is incremented.
- The FIFO flag is updated after a slight delay from the rising edge of the clock.

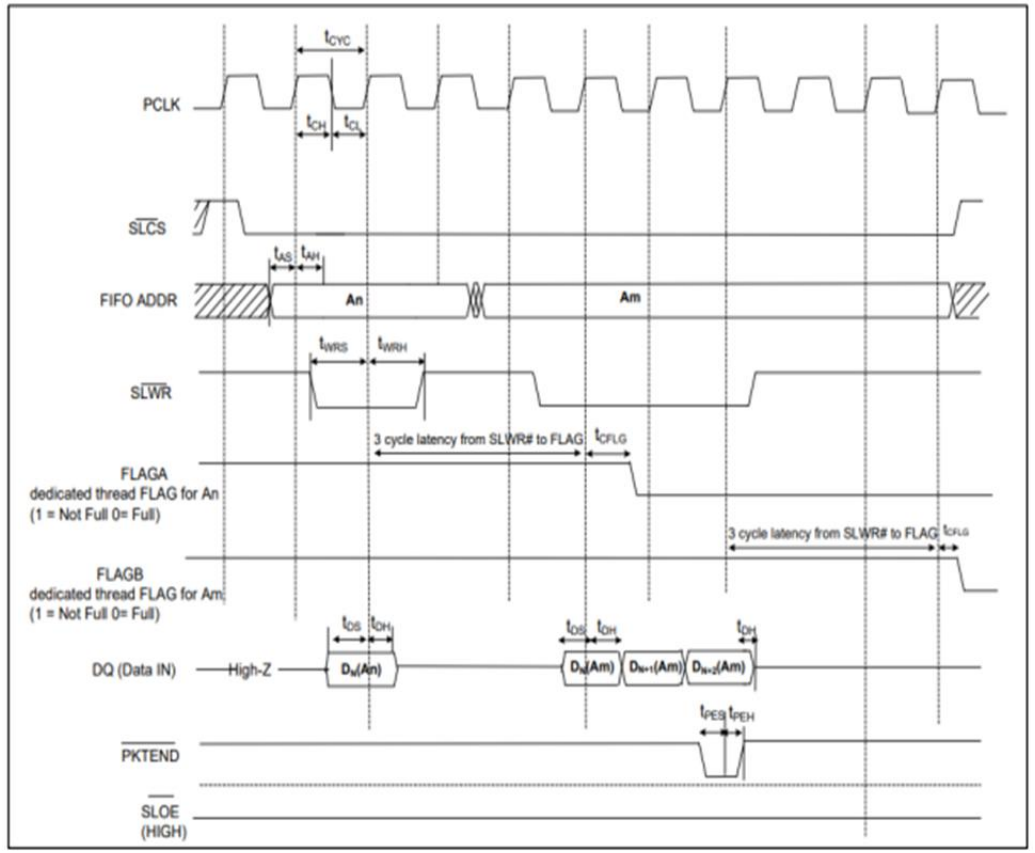

Fig. 11: Write Sequence – Timing Diagram

Results and Discussions

Streaming Transfers onto the FX3

- The SDK provides a couple of useful tools to track and gauge the extent of data transfer within the microcontroller.
- The FX3 firmware application is developed to facilitate data transfer between peripherals.
- The **USB Control Centre Utility**, along with the **C++ Streamer tool** can be used to quantitatively measure the average throughput observed at the USB and GPIF endpoints.

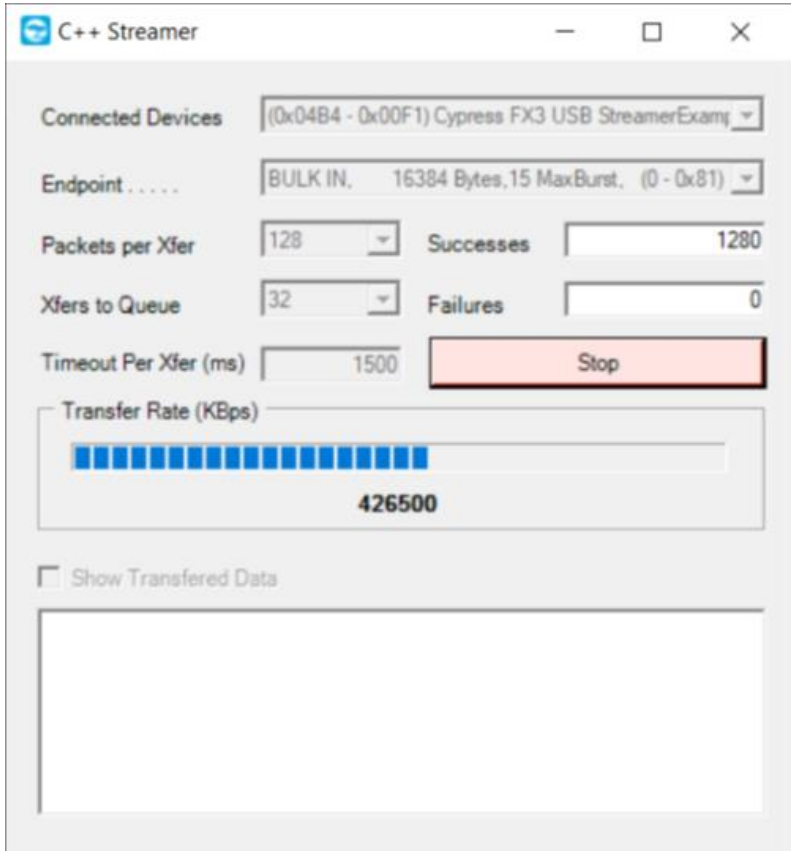

Fig. 12: Cypress USB Streamer Application

Conclusion and Future Enhancements

- A **system image/ firmware** built to implement the **2 – bit Slave FIFO mechanism**, and to verify **bulk – IN** and **loopback transfers** across an external master (FPGA).
  In order to achieve the above:
  - The **GPIF state machine** and I/O matrix were configured
  - **Bulk-IN data transfers** via the FX3 to the FPGA and USB host were executed
  - **Loopback transfer** of data occurred successfully
- A number of faults and challenges including IDE mismanagement, obsolete APIs, timeout errors, streamer and GPIF definition conflicts were encountered.

Future Enhancements

Task	Tentative Completion
Implementing a 1553 Bus Controller and building software to target the same	January – March 2022
Generating and receiving command packets for 1553 BC, pulse command and simulator FPGA to forward over an SPI interface	February – March 2022
Configuring the FPGA using VHDL to accept commands from the USB host via FX3 and run tests on the SSR	March 2022
Implementing compression and decompression algorithms to handle image and video data from SSR on the fly via FPGA	

Future Enhancements (Contd.)

Task	Tentative Completion
Implementing the command receiver to receive and interpret commands over SPI to the FPGA	April 2022
Defining and executing appropriate record/ playback protocols for data retrieval from SSR	
(Optional) – Use FPGA compute resources to perform video analysis and image processing	
Redesign FX3 firmware and FPGA configurations to facilitate real-time video streaming to FPGA/ from SSR	April – May 2022
Designing an application using Visual Studio to issue telecommand, telemetry and toggle compression for incoming or outgoing data	

Project Extension – Phase II

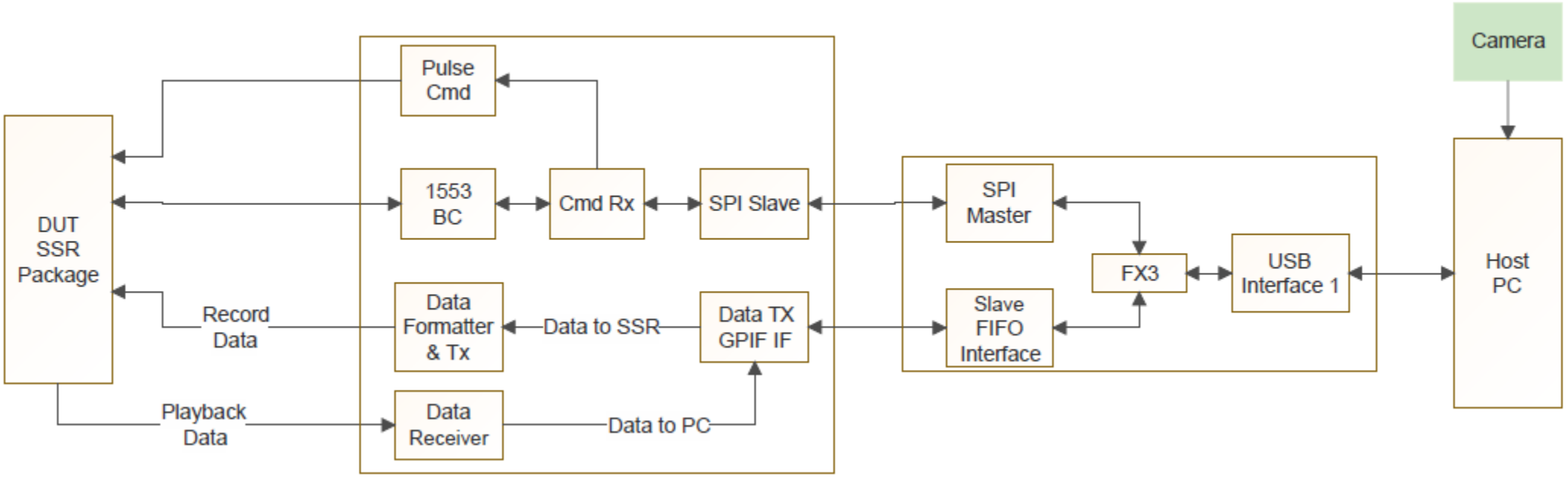

Fig. 13: Rough schematic to depict the project extension for phase II.

References

[1] Y. J. Qian and K. Cui, "Design of high speed CCD data acquisition system based on FPGA and USB3.0," 2015 International Conference on Information and Communication Technology Convergence (ICTC)

[2] Y. Gong and F. Yu, "Design of high-speed real-time sensor image processing based on FPGA and DDR3," 2017 3rd IEEE International Conference on Computer and Communications (ICCC)

[3] Fenxian Tian, Juan Li, Xinxin Sun and Jun Yang. Design and implementation of USB3.0 Data Transmission System based on FPGA. 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019), Advances in Computer Science Research, 2019.

References

[4] C. K. A. Rangan, K. A. Holla, V. Kulkarni, A. Kumar and A. Patil, "Data rate based performance analysis and optimization of bulk OUT transactions in USB 3.0 SuperSpeed protocol," 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2017, pp. 114-119, doi: 10.1109/ICATCCT.2017.8389117.

[5] Minyeol Seo et al. "An Effective Design of Master-Slave Operating System Architecture for Multiprocessor Embedded Systems," *Advances in Computer Systems Architecture, 12th Asia-Pacific Conference, ACSAC, 2007*

[6] J. G. V. Leañez et al., "High-Speed Data Acquisition System for GNSS Applications," 2020 Argentine Conference on Electronics (CAE), 2020, pp. 14-19, doi: 10.1109/CAE48787.2020.9046375.

Thank You