INDIAN INSTITUTE OF TECHNOLOGY (ISM), DHANBAD

# Implementation of XADC using SPI Protocol on Zynq-7000 FPGA

**Project Report**

**Member 1:** Penmatsa Rethika Surya Sri

**Member 2:** Menda Srihari Naidu

**U**nder supervision of

**Prof. Rahul Bhattacharya**

**D**epartment Of Electronics Engineering

## Contents

## Abstract

The purpose of this project is to design, analyze and implement a complete data acquisition and serial communication system using the internal XADC module available in the Zynq-7000 FPGA. Instead of relying on external ADC hardware modules, the internal XADC was explored in depth and used to convert an analog signal into a digital value. After obtaining the digital samples from the XADC, they were transmitted serially using a custom-designed SPI interface implemented entirely in Verilog. Internal signals were monitored using the Integrated Logic Analyzer (ILA) to verify the timing relationships.

A major emphasis of this work is not just implementation, but the detailed understanding of timing parameters such as sampling time, acquisition time, ADC clock frequency, conversion latency and SPI clock derivation. Several misunderstandings were corrected during the learning process, and this report explains them thoroughly so that future students do not face similar confusion. This project ultimately demonstrates how real-world analog signals can be successfully captured, processed and transferred within an FPGA-based system.

## 1 Introduction

In digital electronics and FPGA-based systems, almost everything operates using binary numbers. However, most real-world signals — such as sound, temperature, voltage and sensor outputs — exist in analog form. Therefore, before any digital processing can be done, these analog signals must be converted into digital form using an Analog-to-Digital Converter (ADC).

The Zynq-7000 FPGA platform provides a powerful advantage by including an internal ADC known as the XADC. Since it is already built into the device, no additional external ADC hardware is required. This creates opportunities for compact design and faster data availability.

The main idea behind this project is to study the internal XADC thoroughly, understand how it samples and converts analog inputs, and then design logic to transmit the converted data serially using a custom SPI protocol. The SPI protocol was chosen because it is widely used, simple to implement and efficient for transferring digital data.

Rather than simply implementing the design, significant time was spent understanding every parameter and correcting misconceptions. This report describes each conceptual step in detail, moving from theory to implementation in a structured and educational manner.

## 2 Understanding the Internal XADC

The XADC block inside the Zynq-7000 is capable of measuring both internal FPGA voltages and external analog inputs. It produces a 16-bit digital output for every conversion.

In this project, one of the external VAUX channels (VAUXP0 / VAUXN0) was connected to a function generator that produced a sine wave. The XADC was configured to read this channel continuously.

### 2.1 System Clock and ADC Clock

The FPGA board operates with a system clock frequency of:

$$f_{sys} = 125\text{ MHz}$$

Meaning each clock cycle takes:

$$T_{sys} = \frac{1}{125\text{ MHz}} = 8\text{ ns}$$

However, the XADC has a limitation. It cannot operate at such a high frequency. According to specifications, the XADC clock must always remain below:

$$f_{adc} < 26\text{ MHz}$$

Therefore, the FPGA automatically divides the system clock internally. In our case, the clock used by the XADC becomes approximately:

$$f_{adc} \approx 25\text{ MHz}$$

Giving a clock period of:

$$T_{adc} = \frac{1}{25\text{ MHz}} = 40\text{ ns}$$

This clock is essential because all internal sampling operations are governed by it.

### 2.2 Clarifying Acquisition Time

Initially, while studying the parameters, an incorrect assumption was made. It was believed that the acquisition time (set to 4 cycles) directly determined sampling speed. The calculation was mistakenly done as:

$$\text{sampling rate} = \frac{25\text{ MHz}}{4}$$

However, this interpretation was wrong.

Acquisition time actually represents how long the XADC allows the analog signal to charge the internal sampling capacitor. When acquisition is set to 4 cycles, it simply means the signal is sampled for:

$$4 \times 40\text{ ns} = 160\text{ ns}$$

It does not define the frequency of conversion.

### 2.3 Conversion Rate

The true rate at which new samples are generated depends on the conversion rate. For our configuration, the XADC provides roughly:

$$961.5\text{ k samples/sec}$$

Which means the time for one full conversion is:

$$T_{conv} = \frac{1}{961.5k} \approx 1.04\ \mu s$$

Thus, the DRDY (data ready) signal goes high approximately every 1040 ns, informing that a new 16-bit result is available.

Understanding this distinction was crucial and completely changed how the SPI timing had to be designed.

## 3 Detailed Study of SPI Protocol

Before transmitting XADC data, a clear understanding of the SPI protocol was essential. SPI (Serial Peripheral Interface) is a synchronous communication method mainly used between microcontrollers and peripheral devices such as sensors, ADCs, DACs, memories and shift registers.

### 3.1 Basic Concept

SPI follows a master–slave structure. The master always controls communication, while the slaves respond. The protocol uses four primary lines:

- **SCLK** – Clock generated by the master
- **MOSI** – Master Out, Slave In
- **MISO** – Master In, Slave Out
- **CS / SS** – Chip Select (active low)

When the master wants to communicate with a slave, it pulls CS low. Data is then shifted bit-by-bit on every clock edge. SPI is full-duplex, meaning data can flow in both directions at the same time.

### 3.2 Internal Shift Register Operation

Both the master and slave contain shift registers. When a clock pulse occurs:

- One bit shifts out from MOSI,
- One bit shifts in from MISO,
- Data keeps circulating like a circular buffer.

This makes SPI extremely fast and efficient.

### 3.3 Daisy-Chain Architecture

In systems with many slaves, connecting separate CS lines becomes difficult. Daisy-chaining allows MOSI of one device to connect to MISO of the next. Only one chip-select is used, reducing required GPIO pins. Data flows through all devices sequentially.

### 3.4 SPI Modes

SPI timing depends on:

$$CPOL \quad \text{(clock polarity)}, \quad CPHA \quad \text{(clock phase)}$$

The four modes define when data is sampled and when it is shifted. This flexibility ensures compatibility with many peripheral devices.

### 3.5 Advantages

SPI provides many benefits:

- Low power consumption
- Very high speed
- Full-duplex transfer
- Simple hardware connections
- Flexible timing modes

### 3.6 Limitations

However, there are also drawbacks:

- Requires at least four wires
- No built-in addressing mechanism
- Does not support multi-master configuration

- More GPIO pins needed as number of slaves increases

Even with these limitations, SPI remains one of the most widely used communication protocols due to speed and simplicity.

## 4 Serial Transfer Using SPI Protocol

Once a 16-bit digital value is available from XADC, the next challenge is transmitting that data serially. SPI (Serial Peripheral Interface) is ideal because it uses:

- a serial clock (SCLK),
- a chip select signal (CS),
- and a single data line (MISO in our case).

The goal was to design all of these signals manually using Verilog code.

### 4.1 Deriving the SPI Clock

Since one XADC value is produced every 1040 ns and there are 16 bits per value, the SPI bit period must satisfy:

$$\frac{1040}{16} = 65 \text{ ns}$$

Thus, SPI clock frequency approximately becomes:

$$f_{spi} = \frac{1}{65 \text{ ns}} \approx 15.38 \text{ MHz}$$

However, our FPGA already runs at 125 MHz, whose period is 8 ns. Therefore, the SPI clock must be generated by using the system clock creatively.

By counting 8 system clock cycles, we get:

$$8 \times 8 = 64 \text{ ns}$$

which is very close to our requirement. To maintain timing accuracy, occasionally an additional cycle is inserted. This careful arrangement allowed the SPI timing to align closely with the XADC output speed.

### 4.2 Master Module Operation

The master detects every DRDY pulse and takes it as a trigger to start serial transmission. When data is ready:

1. CS goes low, indicating SPI transaction is starting.
2. SCLK pulses are generated.
3. Data is shifted over 16 cycles.

Although our current version keeps CS low longer than ideal, it still successfully transmits the data. Improvements can later restrict CS to only remain low during actual transfers.

### 4.3 SPI Interface (Slave Behavior)

On the receiving side inside FPGA, a 16-bit shift register holds the XADC output. Whenever a falling edge of SCLK occurs:

1. The most significant bit is sent onto MISO.
2. The register shifts left.
3. After 16 shifts, the register reloads with the next XADC value.

Future work can include gating data based on CS to create a fully compliant SPI peripheral.

## 5 Debugging Using ILA

Debugging FPGA hardware cannot be done with normal measuring instruments alone. The Integrated Logic Analyzer allows internal signals to be viewed directly.

For our function generator signal of 10 kHz, one complete cycle lasts:

$$100\,000 \text{ ns}$$

Since every sample is taken at system clock edges (every 8 ns), the required number of samples to observe one entire waveform becomes:

$$\frac{100\,000}{8} = 12\,500$$

Therefore, the ILA depth was selected as 32k. This ensured complete waveform visualization with additional margin.

Using ILA, we were able to observe:

- XADC data output,
- DRDY pulses,
- serial SCLK waveform,
- chip select transitions,
- shifting operations,
- and the final MISO output.

This confirmed that theoretical timing matched actual hardware execution.

## 6 Summary of Constraints

The clock pin was configured for 125 MHz operation. The VAUX pins were correctly mapped to accept analog input. GPIO pins controlled LEDs for visualization purposes, while the debugging probes were connected to the ILA module. Proper attention to pin mapping ensured stable and correct operation.

# 7 Extension Towards External ADC Interfacing

While this project mainly focused on understanding and utilizing the internal XADC of the Zynq-7000 FPGA, an external ADC based system was also studied conceptually to understand how real-world ADC peripherals interact with FPGA logic. This section outlines the approach and design considerations involved in interfacing an external ADC using the SPI protocol.

## 7.1 Motivation for Using an External ADC

Although the internal XADC provides sufficient resolution and ease of access, external ADCs are commonly used in practical systems due to the following reasons:

- Availability of higher resolution and programmable gain options
- Better isolation from internal FPGA noise
- Support for multiple input channels and higher accuracy
- Compatibility with industry-standard communication protocols

Studying an external ADC helps in understanding real-time data acquisition systems used in embedded and industrial applications.

## 7.2 External ADC Communication Process Using SPI

In an external ADC setup, the FPGA acts as an SPI master and controls the complete data acquisition process. The general sequence followed is:

1. Chip Select (CS) is asserted low to initiate communication.
2. Configuration or command bits are transmitted through MOSI.
3. The ADC performs sampling and conversion internally.
4. Converted digital data is shifted out serially through MISO.
5. CS is deasserted to end the transaction.

All timing parameters such as clock frequency, setup time, hold time and data alignment must strictly follow the ADC datasheet specifications.

## 7.3 Design Approach Adopted in This Work

In this project, a finite state machine (FSM) based SPI controller was designed in Verilog to understand the external ADC communication flow. The design focused on:

- Generating SPI clock using the system clock
- Controlling CS timing explicitly
- Serial shifting of command and data bits
- Synchronizing ADC data availability with SPI transactions

The emphasis was placed on understanding timing relationships and protocol behavior rather than achieving a finalized analog waveform at the output. This study forms a strong foundation for full external ADC integration in future implementations.

## 8 Results and Discussion

The system behaved exactly as predicted. The XADC continuously converted the incoming sine wave, producing smooth digital samples. The SPI transmission successfully serialized the 16-bit word for every conversion. Timing diagrams captured through ILA closely matched theoretical expectations.

An important takeaway was that understanding timing is far more important than coding alone. Small misinterpretations about acquisition time or clock division can lead to major design errors. By carefully analyzing each parameter, a reliable working design was achieved.

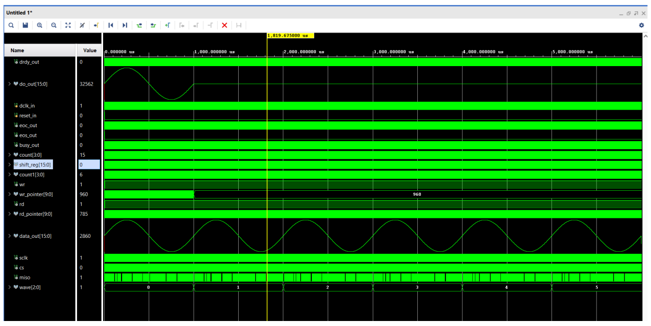

Figure 1: Generation of Sine Wave using RAM

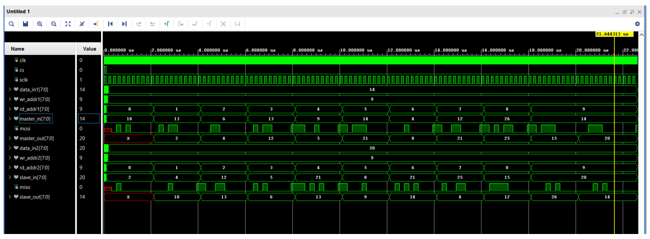

Figure 2: Simulation results of SPI protocol

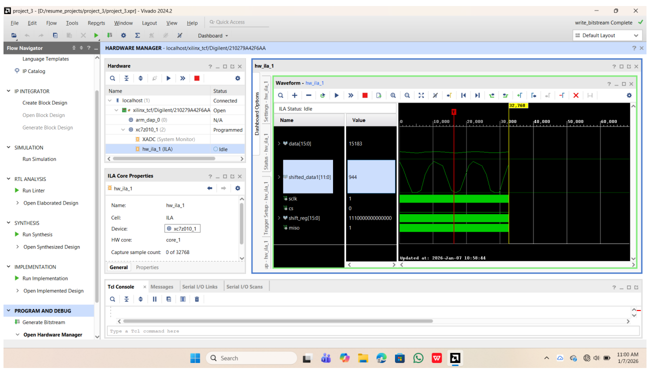

Figure 3: Output Wave Form on ILA

## 9 Conclusion

This project provided an in-depth learning experience about how analog signals interact with digital logic inside an FPGA. By working through each problem manually — from XADC configuration to SPI serial transfer — deeper clarity was gained regarding timing dependencies and internal architecture.

The project successfully proved that the internal XADC of Zynq-7000 can be effectively used without relying on any external ADC modules. Additionally, SPI communication implemented inside the FPGA demonstrates how digital data can be efficiently transmitted.

## 10 Future Scope

In the future, the design can be improved and extended in several directions. A microcontroller can be connected externally to read SPI data. A DAC can be added to reconstruct the signal. Filter modules and averaging systems may improve accuracy. GUI-based visualization tools can also be added for improved graphical representation.