Project-I

EE619A

03/Mar/2025

---

1 Design Specifications

You are tasked with designing a backend for a mixed-signal IC as shown in Fig.1. Only the signals relevant to the backend design are shown here.

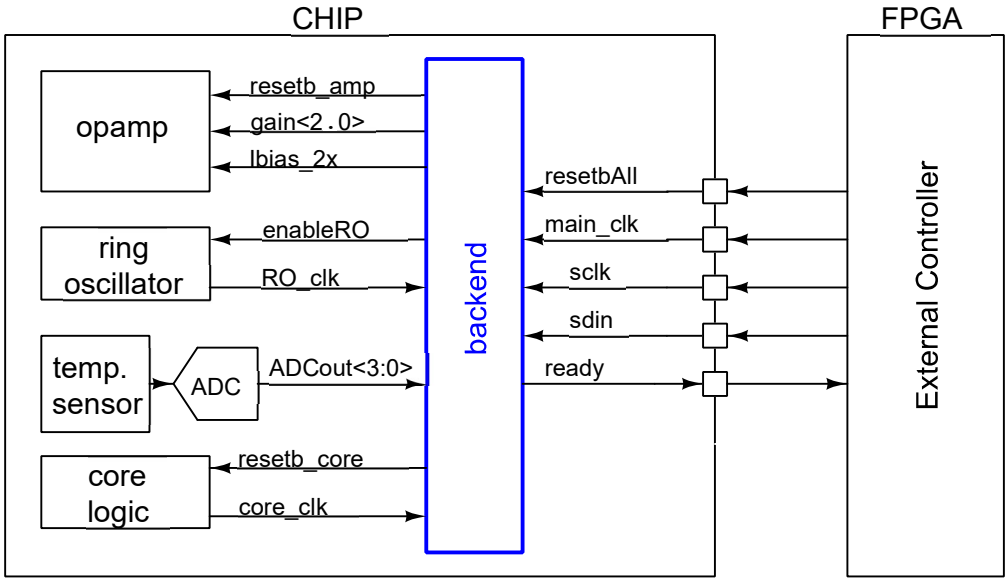

Figure 1: Overall block diagram.

The chip consists of an opamp, a ring oscillator, a temperature sensor, an ADC, a core logic and a backend. The backend has to ensure that all these analog modules are initialized correctly, and the chip is ready for use. The backend has to execute a start-up sequence for this, the details of which are given in Section 1.2. The chip will be interfaced with an external controller (most likely an FPGA) to control the start-up sequence. The outputs from the temperture sensor + ADC can be used to determine the temperature of the chip allowing to determine if the chip is in fast or slow conditions. If the chip is under slow conditions the bias current for the opamp needs to be doubled by setting Ibias_2x=1. Under the slow conditions, the core logic will also need to be operated at a slower clock. The algorithm for doing this is also explained in Section 1.2.

1.1 Inputs and Outputs

The input and output pins of the backend are shown in Fig.2, along with the nets that will be connected to these pins at the top-level.

Inputs

- **i_resetbALL** : Active low reset for the whole chip. When i_resetbALL = 0, the backend and other analog modules will be reset. When i_resetbALL goes from 0 to 1, startup sequence is initiated.
- **i_clk** : Main clock provided to the backend. Frequency: 500MHz.
- **i_sclk** : Clock provided for serial communication with the backend. Used to synchronize the programming data coming in through the i_sdin pin. i_sclk remains 1 when not in use. i_sclk, toggles like a clock when data is being sent from FPGA to the chip through i_sdin.

- **i_sdin** : Serial data input pin. FPGA will send 1 bit data at a time to the chip, to configure the gain of the opamp.
- **i_RO_clk** : The ring oscillator output. From simulations across PVT, the frequency of the ring oscillator can be anywhere within a range of 500MHz to 1.5GHz. i_RO_clk remains 0, when o_enableRO is 0.
- **i_ADCout<3:0>** : The temperature sensor output digitized using an ADC. The ADC output will range between 0 to 15.

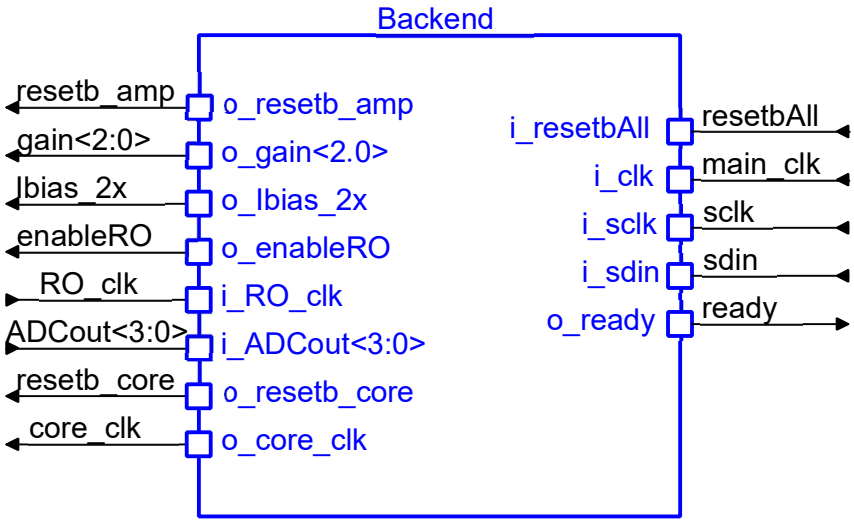

Figure 2: Input output configuration of the backend.

Outputs

- **o_ready** : When o_ready is 1, it indicates that the start-up sequence has been completed successfully and the amplifier and the core logic within the chip are ready to be used.
- **o_resetb_amp** : Active low reset for the opamp.
- **o_gain<2:0>** : Three bit value that controls the gain of the opamp.
- **Ibias_2x** : Controls the bias current in the opamp. If Ibias_2x is 1, the bias current in the opamp is doubled.
- **o_enableRO** : Active high enable signal for the ring oscillator.
- **o_resetb_core** : Active low reset for the logic core.
- **o_core_clk**: Clock to the logic core.

1.2 Backend funcionality - start-up sequence

Please design the backend so that it executes the following start-up sequence.

1. When i_resetbAll = 0, all outputs of the backend should be pulled to 0. All internal registers should be either set orreset.
2. When i_resetbAll becomes 1, the start up sequence is initiated. The following instructions have to be executed in thegiven order.
3. Wait for the serial data from i_sdin. Read the data in. Details of this serial communication is explained in the Section 1.3.
4. Based on the serial data received, set the value of o_gain (refer Section 1.3).
5. Set o_enableRO=1.

6. Wait for five i_clk cycles.
7. The ADC output is to be filtered through a 4-tap moving average filter as shown in Fig. 4. While this is expectedto be operating at all times, it's outputs when i_resetbAll=0 is not relevant. Implement the filter as a part of the backend. If the ADCavg<=12, set o_Ibias_2x=0 and o_core_clock=i_clk. If the ADCavg> 12, set o_Ibias_2x=1 and o_core_clock=i_clk/4 (frequency division). The state of the o_core_clk prior to this does not matter.
8. Set o_resetb_amp=1 and o_resetb_core=1.
9. Wait for five clock cycles.
10. Set o_ready=1.
11. Keep monitoring the ADCavg signal. If ADCavg goes above 12, set o_Ibias_2x=1 and o_core_clock=i_clk/4. IfADCavg goes below 8, set o_Ibias_2x=0 and o_core_clock=i_clk. If $8 \leq$ADCavg≤ 12 retain the values of o_Ibias_2x and o_code_clk as is. All other output values are to be held as is till the next falling edge of i_resetbAll.

1.3 Serial data communication

To configure the gain of the amplifier, overall 3-bit information is required. Instead of dedicating three pins for these, we send the data serially, one bit at a time. This technique is often used to reduce the number of pins required on the chip.

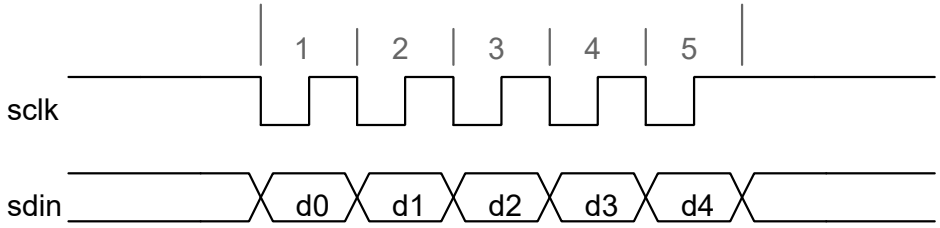

Figure 3: Serial data transmission using sclk and sdin lines.

The external controller will send the five programming bits after i_resetbAll is deasserted. The data transmission will be as shown in Fig.3. The FPGA will sent the data at the falling edge of sclk. The backend has to receive these at the rising edge of sclk. The data corresponding to the first sclk cycle is d0, second sclk cycle is d1 and so on. You can use a shift register configuration to read the sdin. d0 to d4 are related to the amplifier gains as follows.

- o_gain<2:0> : {d2, d3, d4}
- d0 and d1 are extra bits that may be used in the future. Not relevant to the current design.

Please note the order. o_gain<2> is d2, o_gain<1> is d3 and so on. The sclk frequency is expected to be lower than 10MHz. The exact frequency of sclk is not known at the time of design.

1.4 Moving average filter for the ADC output

The implementation of the 4-tap moving average filter is shown in fig. 4. z^{-1} referes to one clock cycle delay. The filter ensures that momentary glitchy outputs in the sensor does not affect our decision making. You may use any of architecture for the adder and the multipler. If you are able to optimize the filter implementation in terms of reducing the number of bits required for each operation you are free to do so, as long as the final transfer function from ADCout to ADCavg remains the same as shown in fig. 4.

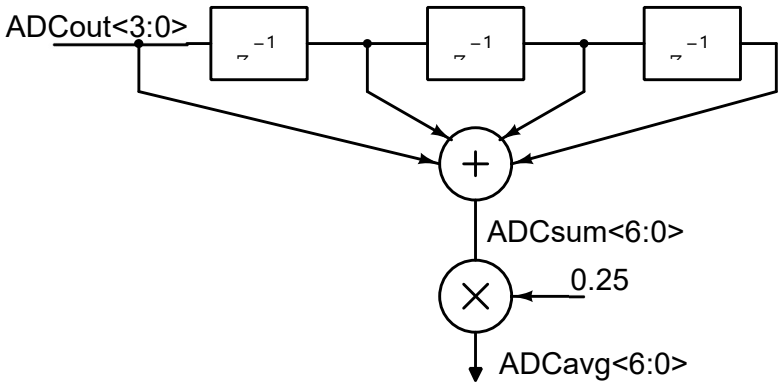

Figure 4: Serial data transmission using sclk and sdin lines.