WEAPL01:

PRESENT AND FUTURE OF **HARMONY BUS**, A REAL-TIME HIGH SPEED BUS FOR DATA TRANSFER BETWEEN FPGA CORES

ACKNOWLEDMENT:

J. Serrano and OHWR team at CERN; J.Avila-Abellan, S. Blanch, A. Burgos, G. Cuní, D. Fernández, O. Matilla, M. Rodriguez and J. Salabert at ALBA Synchrotron; P. Sjöblom and A. Milan from MAX IV laboratory.

Basics of Harmony Bus

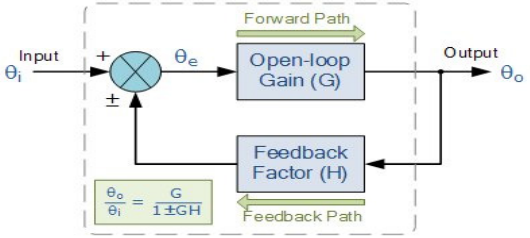
Extra Functionalities for Feedback System Designs

Examples of Implementation

What's Next

Introduction (1/2) - Feedback systems

Feedback loops take the system output into consideration, which enables the system to adjust its performance to meet a desired output response.



HW FPGA-Based — **MHz** µsecs ⟷ **KHz** msecs — **PC-Based**

Bandwidth

Advantages:
- Fast, Low cost, Reprogrammable.
- Massive parallel complex operation.
- Multiple I/O's.

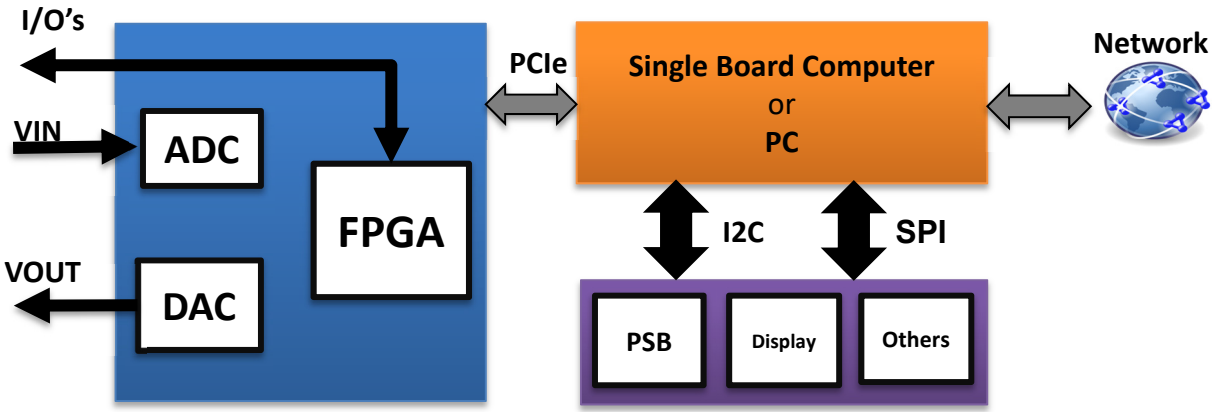
Disadvantages:
- Complex programming.
- Hard to integrate with full custom designs.
- Manage with high-level control software.

Advantages:
- Complex data processing.
- Data sharing.

Disadvantages:
- Real-time data processing.
- Higher latencies.

Introduction (2/2) - Framework for real-time data processing

- Requirements for new in-house developments:
  - Versatile.
  - Easy to adapt to scientist needs.
  - Multiple operation ranges from µsecs to msecs.
  - Real-time calculus.
  - Easy to adapt to any control-system.
- Final solution:



Birth of Harmony Bus (1/2)

- **Problem**
  Communication between FPGA and PC (SBC) via PCIe limits real-time data acquisition.

- **Solution ➔** Separate functionality

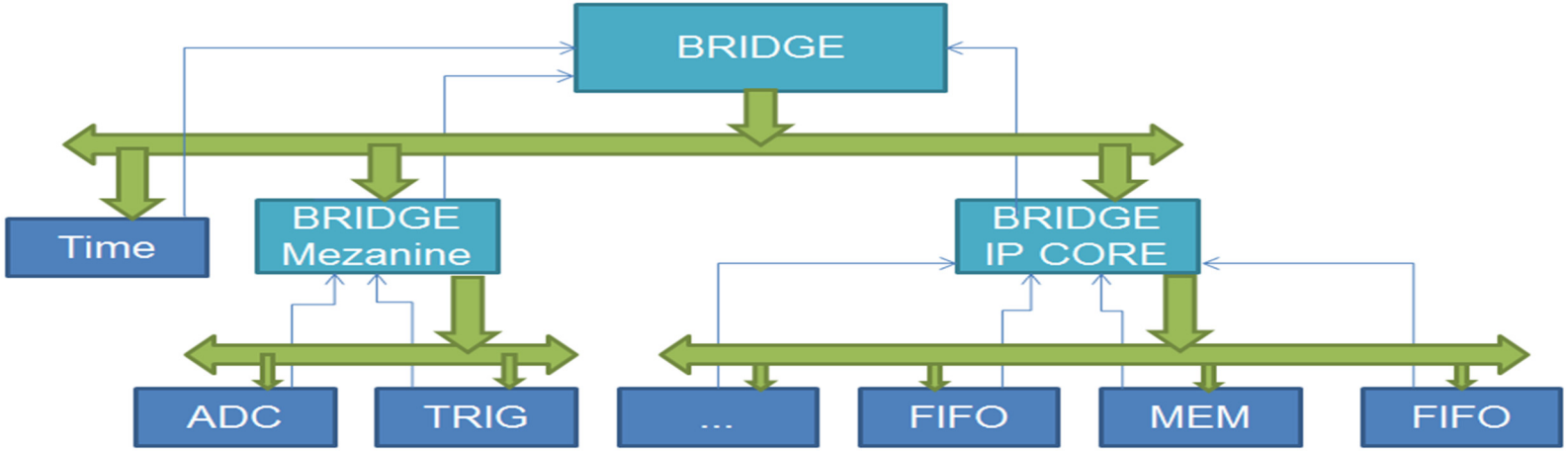
  FPGA: Cores with a concrete function that share data between them, to perform real-time acquisition.

  PC: Configuration, Diagnostics tools, data high level processing and data sharing.

  - Data sharing between PC and FPGA (via PCIe) ➔ Self Describing Bus (SDB)
    - Developed by CERN OHR group. http://www.ohwr.org/
    - Allows to enumerate the cores that are available in the current FPGA binary.
    - It is a self description structure that provides metadata about the logic blocks.
    - Each block is assigned to a virtual memory address and its contents is defined in the FPGA code (configuration parameters or data).

  - Data sharing between FPGA cores ➔ Harmony Bus (HB)

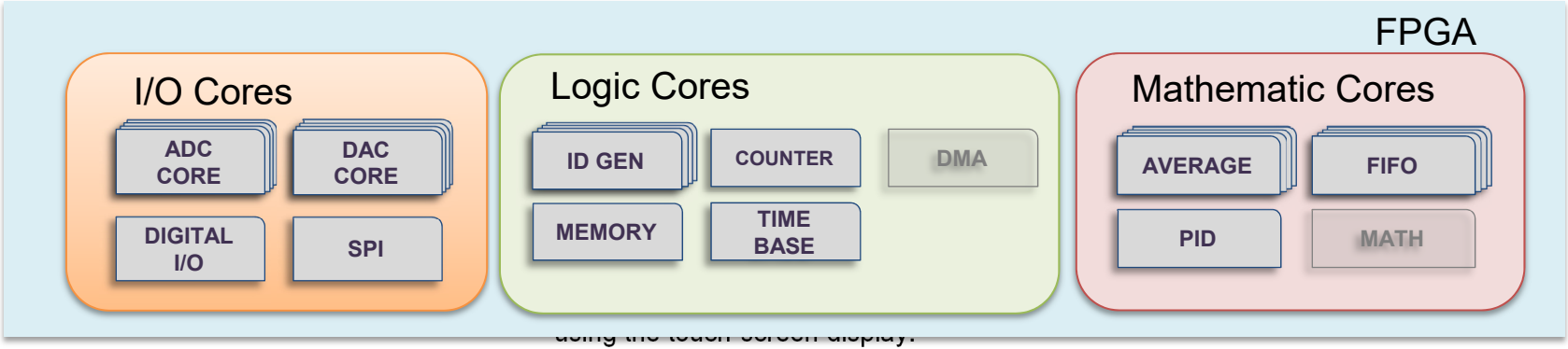
Harmony Bus (2/2)



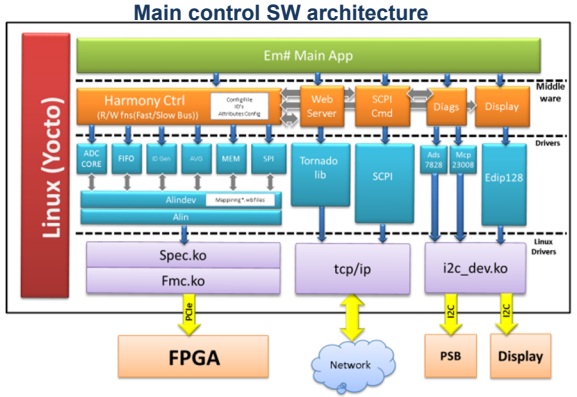
- Harmony is working at 125 MHz Clock.
- Bus lines:
  - 2x 64 bits (upstream; downstream): 32 bits DATA, 24 bits TimeStamp(in ns) & 8 bits Identification (ID).
  - 3 Control lines: DataValid, Reading, DataRequest.
- Data is shared among cores through Bridges, that raise messages with a Round-Robin algorithm.
- Ex.: with 1 bridge, and 2 ID; it is possible to send data at 62MS/s. Adding more ID's and Bridges reduces bandwidth and increase Jitter.

Extra Functionalities for Feedback System Designs

- Set of cores integrated in the FPGA sharing data through HB:

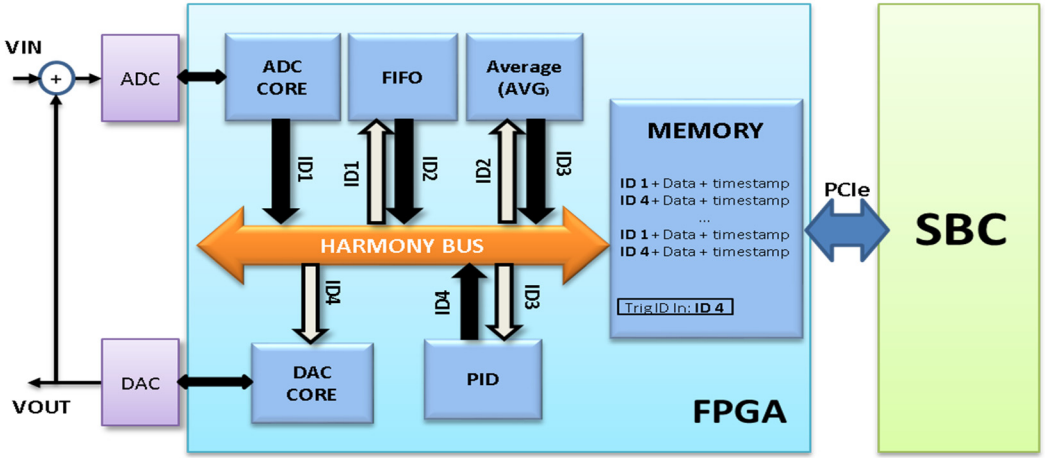


- High level control software running in PC.
  - Features complex embedded control software based on Linux OS.
  - Dynamic configuration of the FPGA cores to use for each application.
  - Correspondence between FPGA cores and control software blocks.
  - Easy user control:
    - Via telnet using the SCPI protocol.
    - Via ssh using the advanced control tools.
    - Via web using a Tornado webserver.
    - Local control is available through navigation menus using the touch-screen display.



Examples of Implementation (1/3) - Feedback System

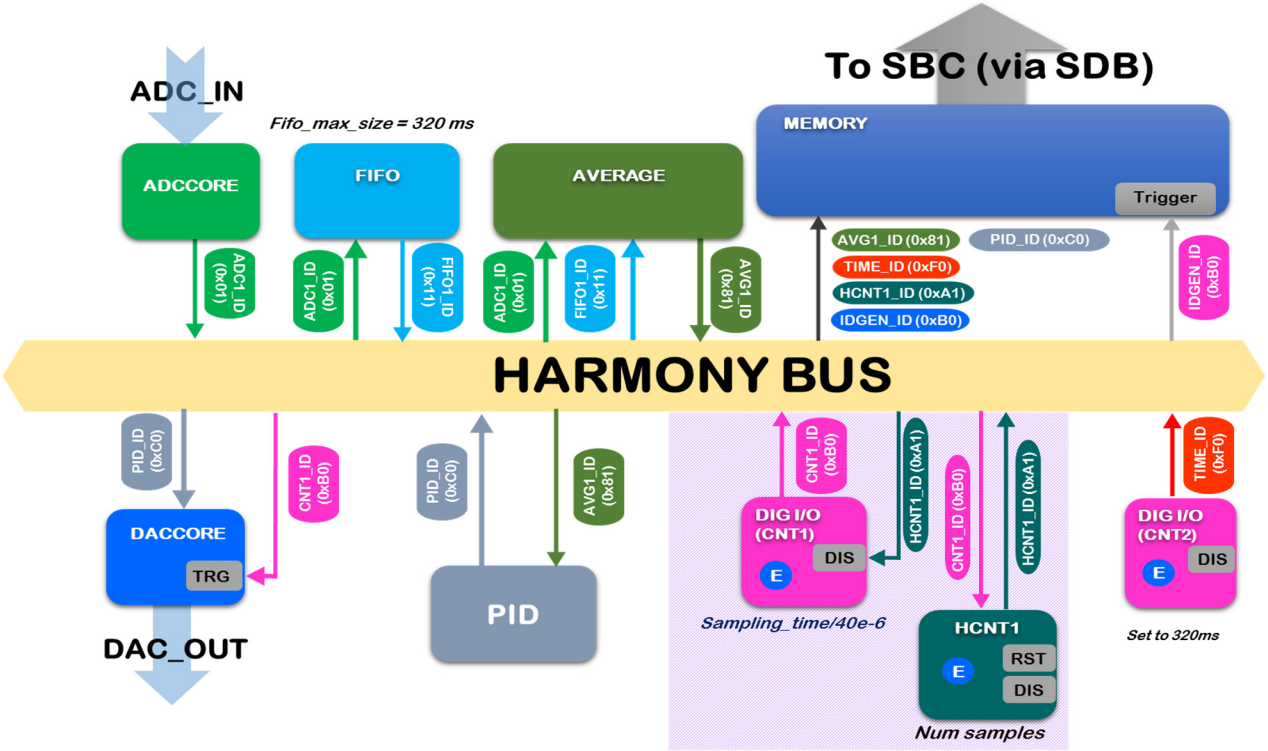
❖ Configuration of a feedback system by the high-level software:



- The PID core provides control over the feedback loop.
- Data acquired at 200KS/s with and oversampling range from 0 to 64.
- FIFO + AVERAGE core are used to generate a mobile average .
- FIFO size limited to 320ms.
- DACCORE output frequency is 100KHz.
- Memory stores data coming from all cores and this data can be used by the high level software in the PC or SBC to monitor and diagnose the status of the feedback system at any point.

Examples of Implementation (2/3) - Data flow

- Flow data diagram for a feedback system



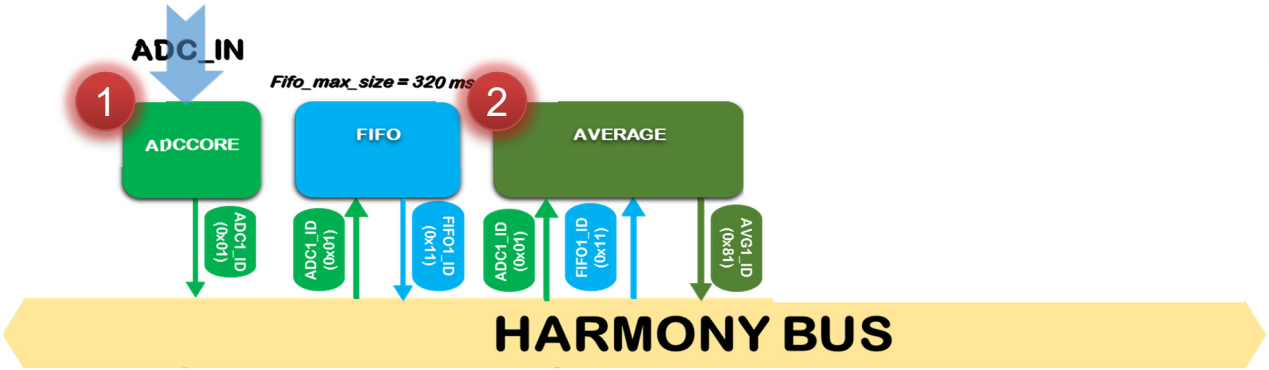
Examples of Implementation (2/3) - Data flow

- Flow data diagram for a feedback system

1 Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).

Examples of Implementation (2/3) - Data flow

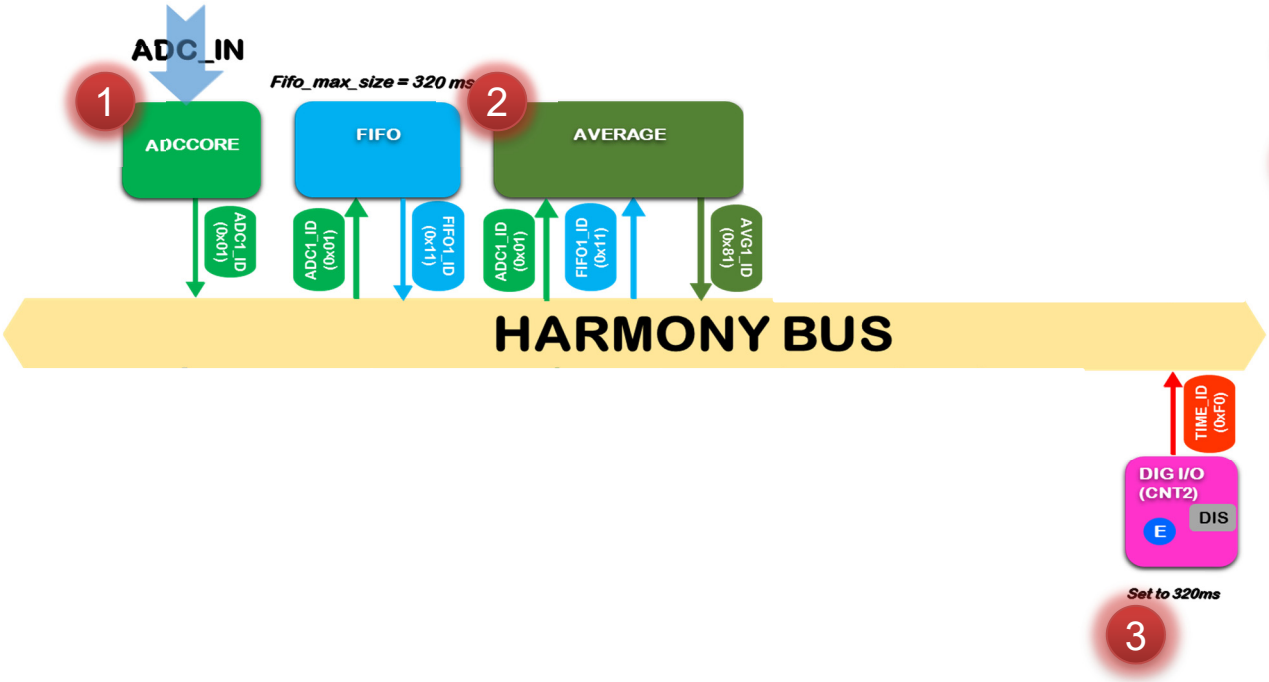
- Flow data diagram for a feedback system



1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).

Examples of Implementation (2/3) - Data flow

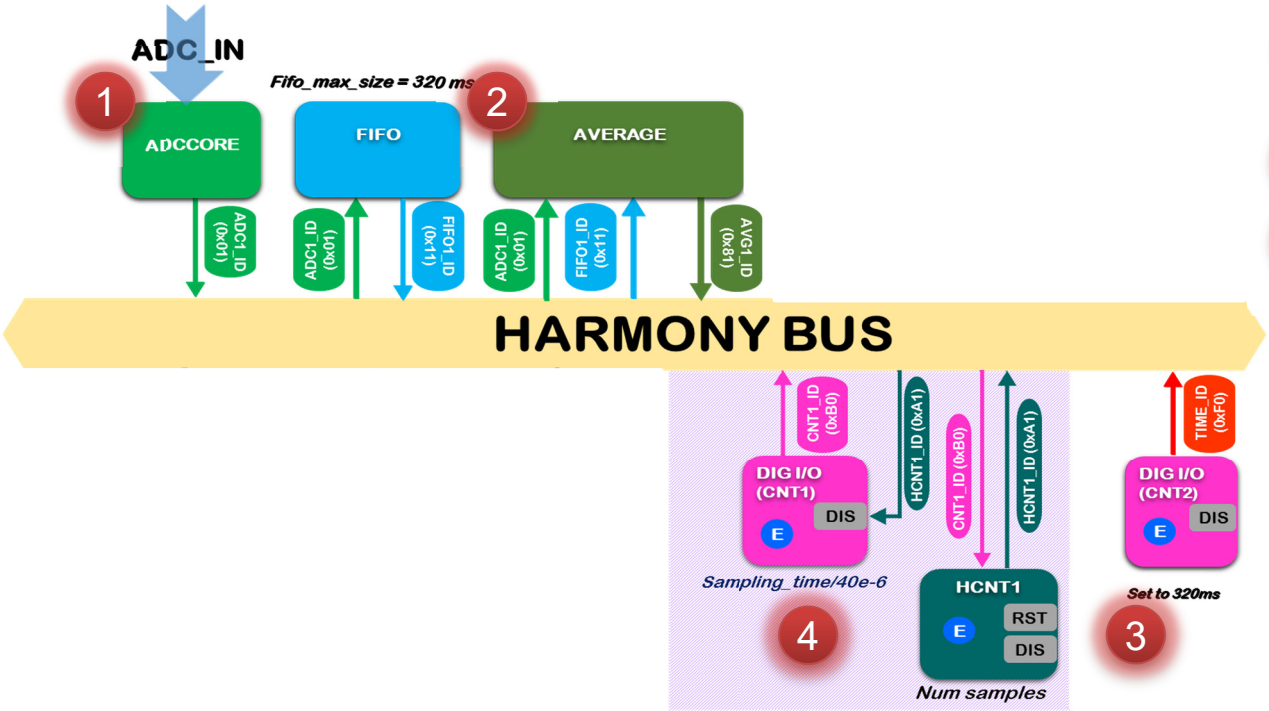
- Flow data diagram for a feedback system



1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).
3. DIGI/O CNT2 provides the time base signal (TIME_ID) for the timestamp.

Examples of Implementation (2/3) - Data flow

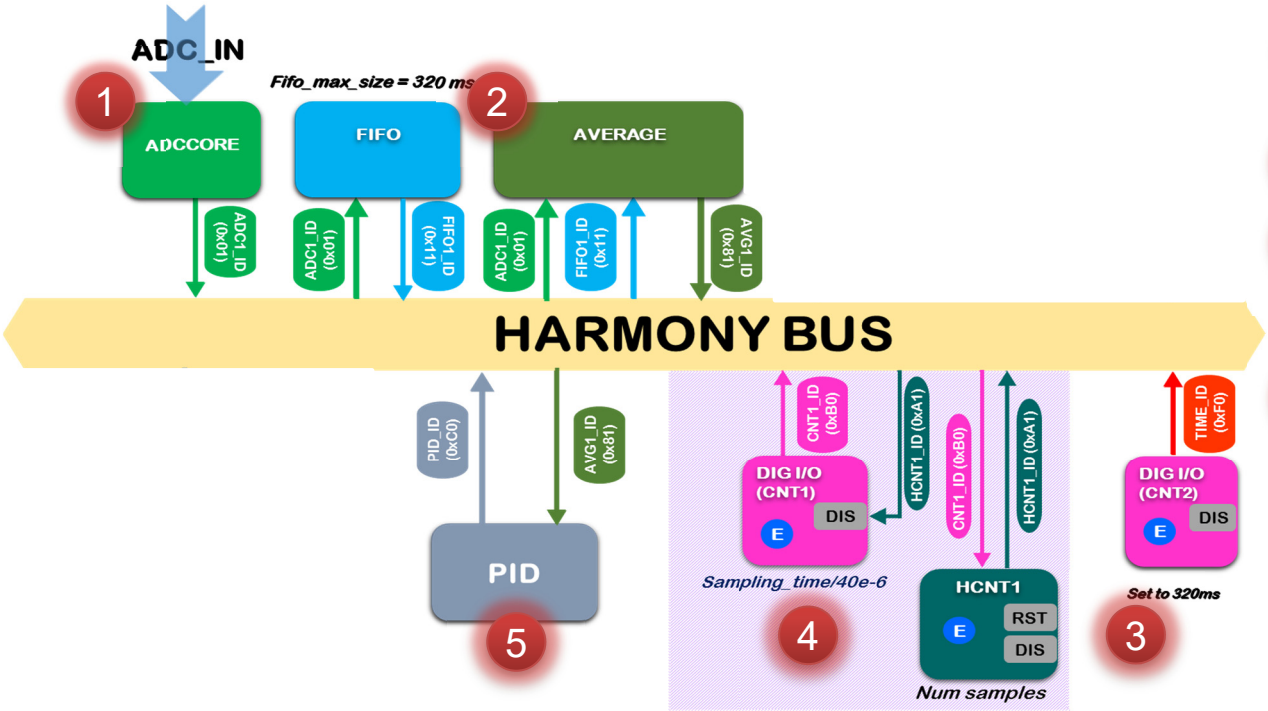
- Flow data diagram for a feedback system



1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).
3. DIGI/O CNT2 provides the time base signal (TIME_ID) for the timestamp.
4. DIGI/O CNT1 and HCNT1 used to configure the *sampling time* and the *number of samples* to acquire (CNT1_ID).

Examples of Implementation (2/3) - Data flow

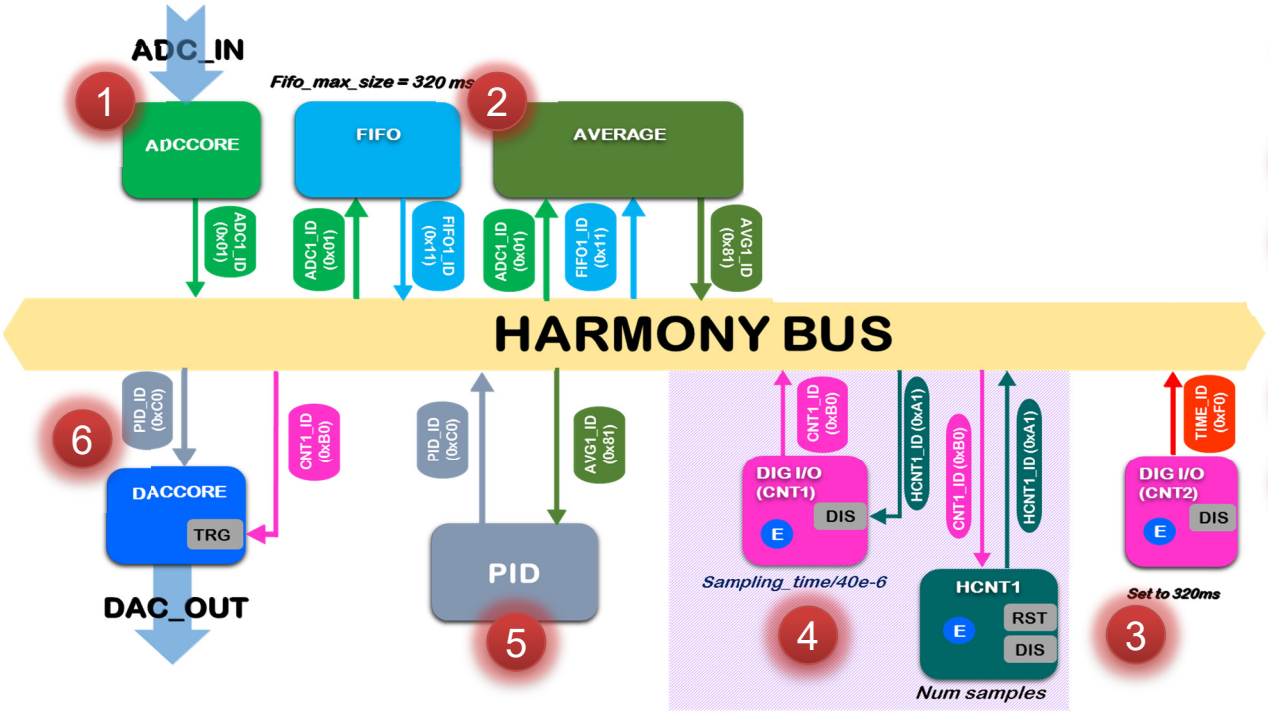
- Flow data diagram for a feedback system



1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).
3. DIGI/O CNT2 provides the time base signal (TIME_ID) for the timestamp.
4. DIGI/O CNT1 and HCNT1 used to configure the *sampling time* and the *number of samples* to acquire (CNT1_ID).
5. PID module generates the corrected signal PID_ID.

Examples of Implementation (2/3) - Data flow

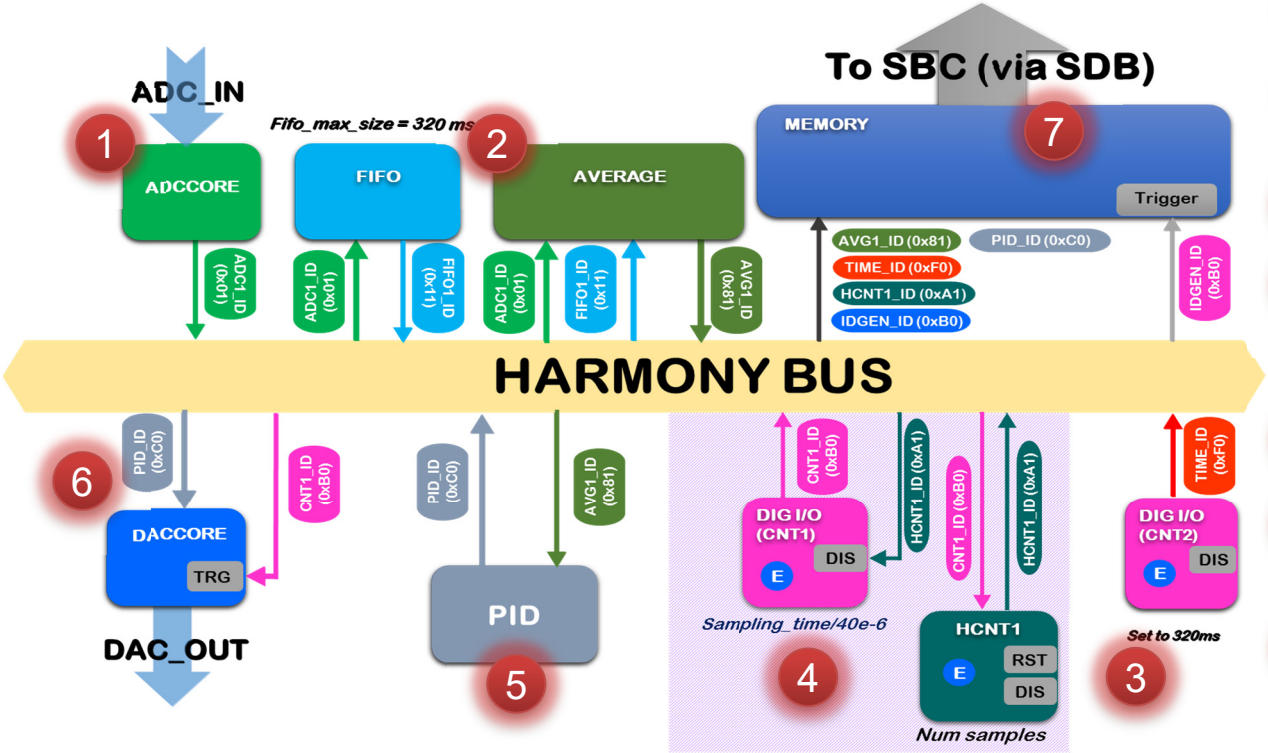
- Flow data diagram for a feedback system



1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).
3. DIGI/O CNT2 provides the time base signal (TIME_ID) for the timestamp.
4. DIGI/O CNT1 and HCNT1 used to configure the *sampling time* and the *number of samples* to acquire (CNT1_ID).
5. PID module generates the corrected signal PID_ID.
6. DACCORE module controls the DAC that converts the PID_ID every CNT1_ID.

Examples of Implementation (2/3) - Data flow

- Flow data diagram for a feedback system

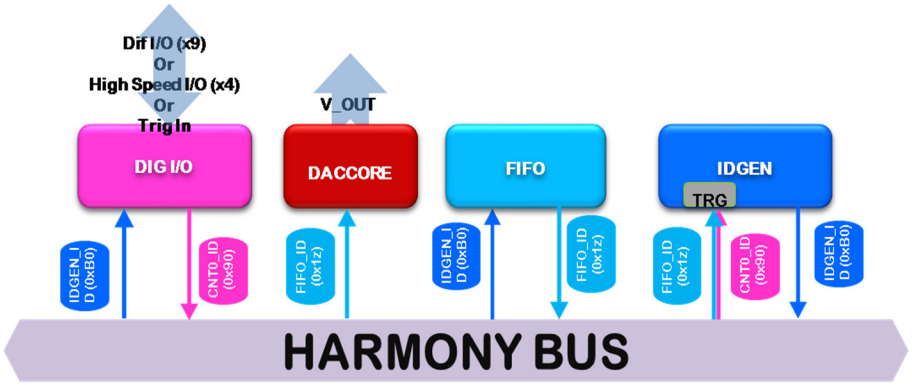


1. Data acquired by the ADC, is read by the ADCCORE block (ADC1_ID).
2. FIFO and AVERAGE are used to get the mobile average of the input (AVG1_ID).
3. DIGI/O CNT2 provides the time base signal (TIME_ID) for the timestamp.
4. DIGI/O CNT1 and HCNT1 used to configure the *sampling time* and the *number of samples* to acquire (CNT1_ID).
5. PID module generates the corrected signal PID_ID.
6. DACCORE module controls the DAC that converts the PID_ID every CNT1_ID.
7. All data sent is stored in the FPGA memory every trigger signal to be used for diagnostics or other applications.

Examples of Implementation (3/3) – Other possibilities

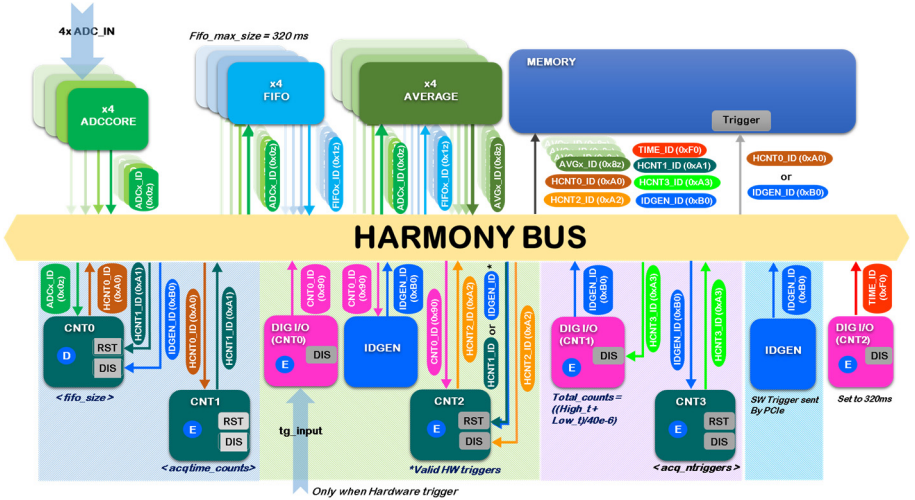
Analog/Digital function generator

- Use to synchronize or control other systems through the digital I/O or the analogue out.
- Can work in parallel with other functionalities.
- Generate fixed analogue values, flanks, pulse trains or any pattern previously stored in the FIFO.



ALBA Electrometer Em#

- It adds 4 Current Amplifiers and a Carrier Board to the base design to measure low noise currents.
- It acquires 18-bit samples from 100pA to 1mA ranges with real time-stamp.
- **TUAPL04:** J. Avila-Abellan et al., “Em# Electrometer Comes to Light”, this conference.



What's Next (1/2) - New Cores

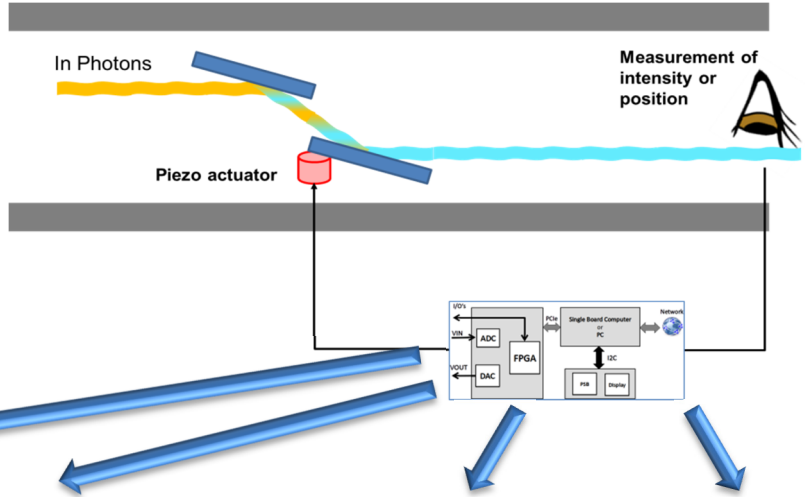
❖ Extend the FPGA cores library

- MATHematical core for real-time calculus in the FPGA.
- Direct Memory Acces (DMA) core to improve data read and processing speed.
- External HB synchronization. Core to interconnect different Harmony Buses.
- More ideas/cores to come.

What's Next (2/2) - Example of application

Monochromator Controller

- A closed loop using piezo actuator in the second crystal of a Monochromator.
- Automatic control of Intensity or position of the beam.

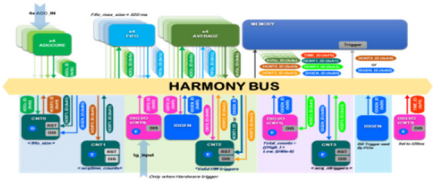


Different Configurations:

1- Piezo characterization using the function generator.

2- PID feedback plus diagnostics using the feedback system.

3- Read low noise currents.



…and more….

.....

Conclusions

Harmony Bus has demonstrated to be a powerful tool to implement in a very flexible way feedback systems where latencies are not required to be lower than tenths of µs.

Together with the use of Dynamic ID's, this new design lets:

- To achieve **multiple configurations** and functions in the same equipment, with just simple commands in the high level software.
- To cover different configurations without hardware changes needed, neither changes in the FPGA gateware.
- Provides tools and **data with timestamp to diagnose how the feedback system is working** and adjust it accordingly dynamically using commands.
- Provides **real-time diagnostics, data monitoring and high level data processing.**
- Operation frequency ranges up to MHz's.

Questions???....

...Many Thanks!!!!