EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN – AB DEPARTMENT

Design of a self test functionality for the Beam Loss Monitoring acquisition chain

Erik Verhagen

Abstract

Technical student work, 6 months, May – November 2006
Under supervision of ***Bernd Dehning*** - AB–BI

The CERN's next experiment facility, the Large Hadron Collider (LHC), is aimed to explore never reached energy levels in particle physics. As this energy grows, interactions of the beam with the experiment equipments could be destructive, particularly for the superconducting magnets. These devices are sensitive to beam losses, which must be accurately monitored to avoid magnet quenches to occur. To guarantee this accuracy during the planned LHC operating period of 15 years, a self test functionality for the beam loss monitoring acquisition chain is foreseen to be implemented, beside the measurement electronics. This report is giving a detailed description of this functionality, with focus on the measurement principles, the content of the design and the available features.

Geneva, Switzerland

November 10, 2006

Contents

1 Introduction

The aim of this project is to implement a self test functionality for the LHC Beam Loss Monitoring system, in order to ensure the good working of its acquisition chain.
This test sequence will take place between the experiments, when no beam is present in the vacuum tube.

This report offers a detailed description of this system and the information needed to make a proper use of it during the final implementation in the Beam Loss Monitoring system. An emphasis is given on useful information needed during the synthesis phase, such as global parameters, and input frequency considerations. Additionally, this document is aimed to give technical details on what is done during the 6 months Technical Student work, omitting intentionally the details on how it is done.

First, let's have a look at the specifications defined at the beginning of the internship period. The functionality must be an add-on to the existing measurement electronics, in order to perform an examination of the acquisition chain. To test whether the chain is ready or not should consist in a harmonic analysis, based on the data provided by the measurement electronics. The aim of this system is not to give a detailed value of the chain parameters, but a flag whether or not a beam can be injected.

The functionality should be capable of performing the analysis on a large number of channels, but for test purpose the initial configuration will be handling four channels. The target is the same as the measurement system, a large scale Stratix FPGA hosted on a so-called combiner card. To ease the comprehension of the design during the reading of this document, a complete RTL view of the implemented design is provided in Appendix A.

2 Measurement principles

2.1 Gain

The first parameter to measure in a harmonic analysis is the closed loop gain. The simplest way to achieve this measurement is to calculate the ratio between the peak to peak values of the acquisition chains output and the stimulating reference signal. These two values should therefore be extracted from the running sum inputs and the reference signal ADC's, and a division unit is needed to be implemented to perform the calculation.

The peak to peak extraction mechanism (**generic_peak_to_peak** components in the RTL view) is based on a successive value comparison, enabling thus the production of an accurate value every input signal period (sine wave) at the same point. This same point consideration will be useful for the phase measurement. The processing is the following : when the signal reaches a maximum, this value is stored in a register, and when a minimum is reached, a combinatory subtraction occurs between the two values. A peak to peak value is thus outputted instantaneously, and a *peak_to_peak_valid* flag is set to valid during one clock cycle, informing the cascaded blocks that a new value is available. The clock is given by a local clock tree, synchronized with the chosen running sum refreshing rate.

The gain is of course based on arbitrary values due to the successive data conversion steps. Nevertheless, a good precision is necessary on this values. A fixed point arithmetic representation is chosen to ensure this, with a modifiable number of digits after the coma. This number will be fixed after the first tests, when a better knowledge of the actual parameters will be defined concerning the acquisition chain (in term of signal latency for example)

2.2 Phase

The phase is the second characteristic of a physical device in a harmonic analysis. It is given by measuring the interval between the peak to peak value production of the reference signal and the chosen running sum. According to 2.1 a *peak_to_peak_valid* signal is notifying the presence of a new peak to peak value, which occurs every input signal period at the same point (eg. when a minimum is reached). This is giving the time reference.

In this design, the phase measurement breaks up in to successive operations. First, a period length information is extracted with a first counter (**ref_counter**) and an arbitrary clock. A number of clock ticks representing the 360 degrees of one period is thus given. Afterward, a second counter (referenced **channel_counter**) is counting the delay between both the reference and the channels *peak_to_peak_valid* signal, based on the same arbitrary clock. The ratio between this two numbers gives an information on the induced phase, in the form of a percentage of a whole period. To obtain a readable value of this ratio, it should by multiplied by 360 for a result in degrees, or by 2π for a result in radians. The strength of this measurement method is that it only requires the reuse of the division unit needed by the gain calculation.

Concerning the counters, a *reset_hold* input makes it possible to reset them asynchronously while keeping on the output the last value reached by the counting process. This is necessary because the phase delay and the gain operation sequence requires the phase division to occur far after the reset of the counters. For the channel data counter (**channel_counter**), this *reset_hold* signal is controlled by either the reference and the channel *peak_to_peak_valid* flag (through an **OR gate**). This is to count only the delay between the 2 input signals. The path of the channels *peak_to_peak_valid* flag corresponding to the channel number is selected through a one bit 4 to 1 multiplexer, controlled by the same address lines as the peak to peak values data.

2.3 Notes

In this section, we will discuss some important issues concerning the gain and the phase measurement. First, an important matter to keep in eye, is that the final destination of this system is the combiner card. For test purpose however, the system has been adapted to fit in the DAB64 FPGA, in append to the BLMTC designed by ***Christos Zamantzas***. This explains why the processing is done on 4 channels (see RTL view, Appendix A). The sequencing of the measurements is defined as following : first the gain and then the phase of the 4 channels cyclically.

Another important matter is the precision given by the division unit. To ease the magnitude comparison, a fixed point representation is chosen. This allows to simply compare a defined number of relevant most significant bits at the system's output in order to validate or not if the given values are inside the recommended boundaries. The number of decimals must thus be statically defined before the synthesis, as shown in the procedure described in 4.1. The values shown in this report (first test values) are achieved with a precision of 6 digits after the coma.

It is also important to notice that the measured values of gain and phase are concerning the whole acquisition chain. This implies of course the ionization chamber but also the high voltage control cable, the low current chamber output coax, the parameters induced by the measurement electronics and the running sums operation. A little phase is also induced by the filtering feature of the self test functionality. This must be considered if an accurate closed loop analysis is planned. In our case of a simple *valid / not valid* test, these offset values should only be included in the range boundaries calculation, for the final magnitude comparison.

The last point to mention is concerning the delay induced by the division operation. This operations are usually done in several clock cycles, which generates an output latency. An effort has been done in this design to reduce this latency and to make it constant (refer to the design specifications of the division unit in 3.2). This latency is also dependent on the signal chosen to clock the division unit. This frequency is by default equivalent to the refresh rate of the reference input, in order to keep the synchronization needed for debugging purpose. But if a faster result measurement is needed, this clock signal can also be tied to a global clock out of the rest of the design, such as a 40 MHz clock tree.

3 Design work

3.1 Control Unit

Like in the most data processing architectures, our system is made of an execution unit and a control unit. This section is giving a description of the control unit. It is based on a finite state machine of Moore's type, and every state is corresponding to an operation inside the execussion unit. This is the simplest way to achieves a good sequencing between the gain and phase measurements of the four (and more afterward) channels. Seven states are covered in the normal sequence for one channel, and 3 error states are present when calculation failures occur. The state transitions are triggered by both channel and reference *p2p_valid* signals, division unit control signals and counter overflow flags.

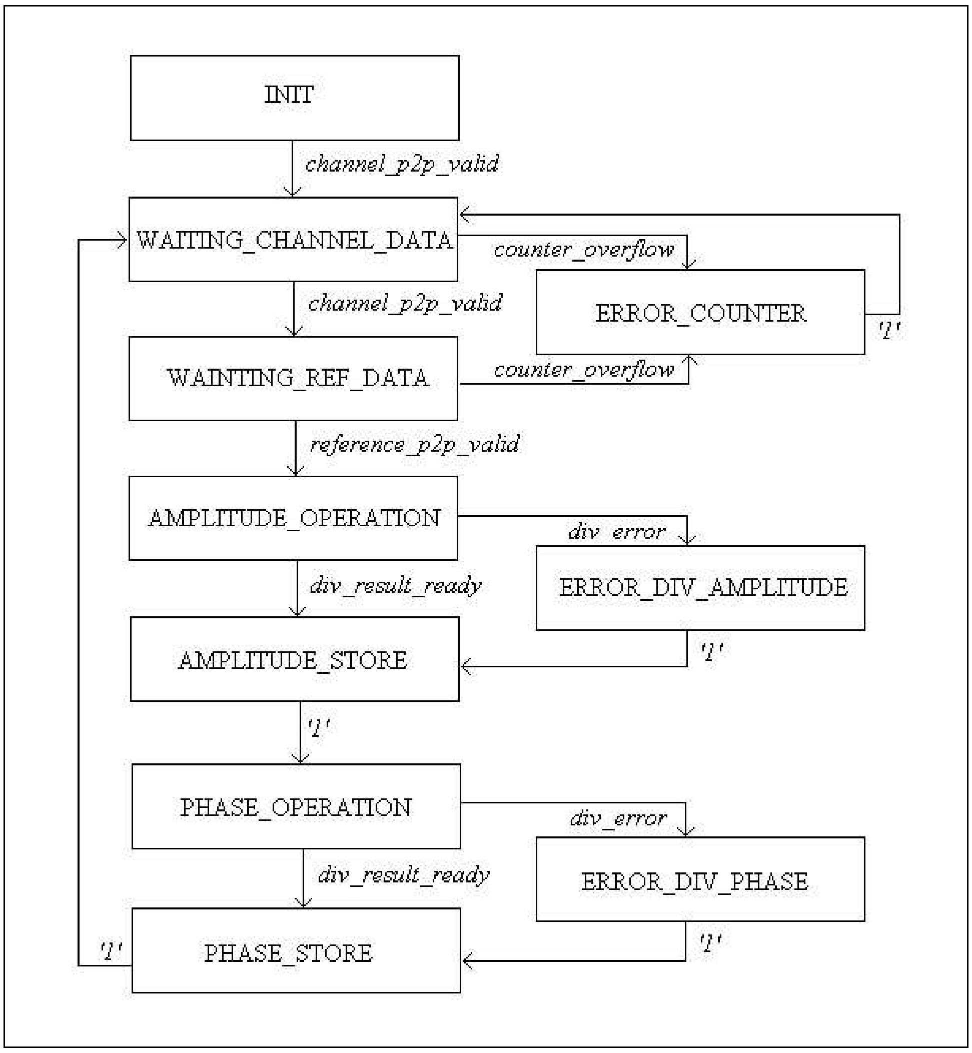


Figure 3.1: *State transitions diagram - Control unit*

3.2 Division Unit

This section describes the implementation of a home made, fixed point division unit to process the calculation of both the gain and the phase. Home made means that it is optimized to fit in our defined specifications. The first discussed constraint is the amount of logical elements occupation. Division units are usually quite big, and this has to be constrained to fit in the single FPGA test environment. The second issue is giving light on the latency needed to complete the calculation, which is also a typical issue of division units. Pipelined divisions often take a fixed amount of clock cycles to produce a result.

In order to ensure the lowest possible occupation of this logical block in the FPGA, a simple structure has been designed in structural VHDL. This is aimed to make a tiny but efficient design, regardless of the synthesis tool performance. A good comprehension of fixed point arithmetic is however needed to understand the design tricks used to achieve this work. The functioning is based on 2 shift registers (one for the partial dividend, the other for the quotient) and a combinational subtractor. The divider for its part is stored in a static locked register as the second operand of the subtractor. The principle is that both shift registers shift at every clock cycle, and the subtractor's borrow bit indicates if the divider fits in the dividend. If so, this bit is resulting '0', and its complement can be appended to the quotient shift register. In this case, the remainder has to be concatenated to the higher part of the first subtracters operand.

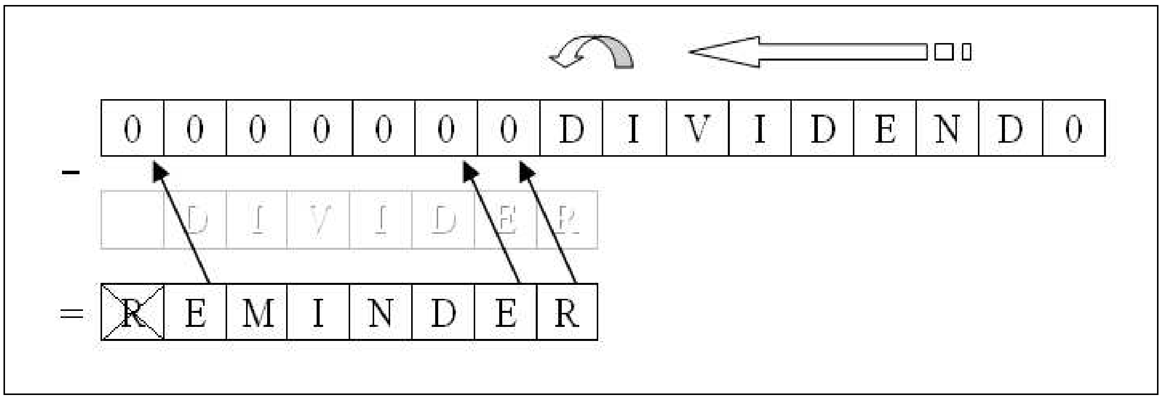


Figure 3.2: *Division registers - Succeeded subtraction*

In the other case, if a borrow bit is produced, its complement is also appended to the quotient register, but the remainder register (subtraction result) is invalidated and a shift on the dividend register is only operated. When synthesized, this division unit fills only around 1 percent of an Altera EP1S40 Stratix device, mainly with flip flops (due to the registers).

Now, concerning the latency needed to produce a result. The processing of all the sequential stages is pipelined, which implies that the latency is only a function of a fixed number of clock cycles. That is, it is ours to define this number, to ensure that a calculation produces its result before new data arrives from the peak to peak extractors.

Because the operation process is based on operand shifts, the number of consecutive clock cycles needed to provide a result depends on the operands size, and the number of significant bits after the comma. The dividend and result registers shift occurs every clock cycle, regardless whether or not the subtraction has succeeded. So the number of bits for the result is the same as the dividend size for an integer division. To perform a fixed point division, the dividend has to be shifted left again (multiply by 2) as long as the number of wanted bits after the comma is not reached. The rest of the processing is exactly the same as for an integer division.

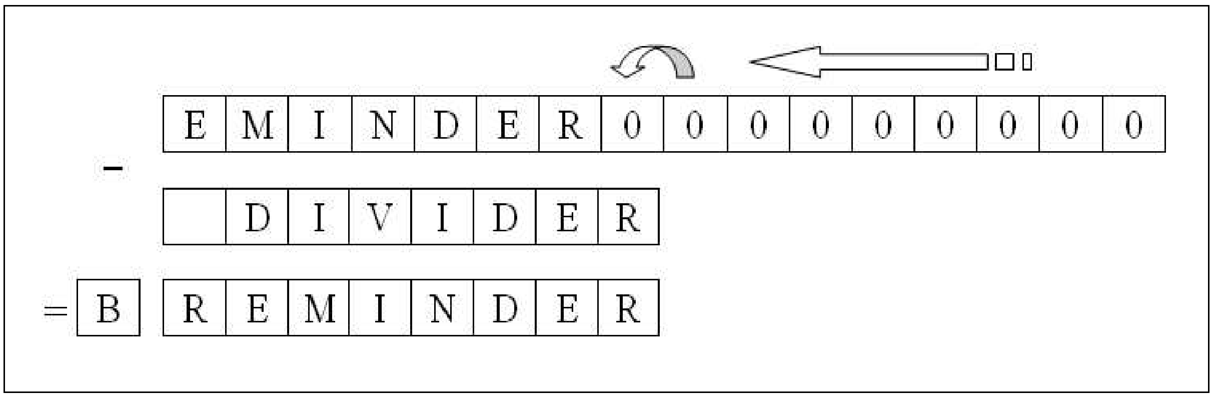


Figure 3.3: *Division registers - Fixed point part of the division*

For obvious synchronization reasons, two clock cycles are added at the begin and the end of the calculus, to load and store the operands and the result. So the total number of latency clock cycles is given by :

$$\textbf{Latency} = [2 + \text{dividend}'\textit{length} + \text{number decimals}] \ \textbf{clock cycles} \tag{3.1}$$

In our test configuration for example, the dividend is a 16 bits wide integer, and the fixed point precision is set to 6 bits. The global division latency of this operation is thus 24 clock cycles.

3.3 Filtering

As we saw in 2.1, the peak to peak value extraction is based on a successive instant value comparison. This means that it offers no noise immunity since no glitches are allowed. As the currents in the ionization chamber are close to the noise limit, this issue had to be fixed. The simplest way is to create digital filters. This section handles about the design of these filters.

One time more, the main point in the specifications is to lower the space occupied in the FPGA. The test configuration is aimed to process data from 4 channels, which means the same number of filter. For the final design much more input channels are foreseen. Digital filters are usually heavy sequential blocks consumers due to the large amount of shift register. Two main types exist, the Finite Impulse Response Filter (FIR), and the Infinite Impulse Response one (IIR). Because of the large sampling ratio between our signals ($\leq$1 Hz signal vs. 1 KHz sampling rate), the FIR filters would take a huge amount of registers. Thus, the choice has been made to implement IIR Filters. These filters usually offer good performances because they are generated from analog equivalents, and they need less registers due to their recursive structure. The drawbacks of designing this type of filter however are higher accuracy needed for data representation inside the structure, a good expertise in digital signal processing theory, and a perfect knowledge of the inner precision errors propagation to avoid the filter becoming unstable.

After some investigations, it seems that the best fitting filter in regard of the input signal and the offered hardware characteristics is a second order Butterworth equivalent. The methodology to design such a filter is to start from the analog filter and find the best way to approach its characteristics in the digital domain. The detailed calculation is given in Appendix B.1 (for details on b_1, b_2 and C coefficients), and the key steps are given below :

1. Analog 2^{nd} order Butterworth filter transfer function :

$$\boxed{H(s) = \frac{1}{1 + \sqrt{2}.\frac{s}{\omega_c} + (\frac{s}{\omega_c})^2}} \tag{3.2}$$

2. Bilinear transform :

$$\boxed{H(z) = \frac{z^{-2} + 2.z^{-1} + 1}{b_1.z^{-2} + b_2.z^{-1} + C} = \frac{Y(z)}{X(z)}} \tag{3.3}$$

3. Recursive relation ($[Z]^{-1}$ transform) :

$$\boxed{y(n) = \frac{x(n) + 2.x(n-1) + x(n-2) - b_1.y(n-1) - b_2.y(n-2)}{C}} \tag{3.4}$$

As we said before, a strict control on the inner data paths must be ensured, to avoid overflows or successive errors accumulation. Simulations has thus been performed in C. This step also had for aim to quantify the performance of our filter choice, namely cut off frequency matters, phase shifts and gain issues. The best coefficients precision have been found with this successive approximation, and their representation is therefore fixed to signed 16 bits words. The shift registers containing the data have been fixed to a width of 20 bits, which is enough to prevent overflows with the given signal dynamic.

The implementation of this filters in VHDL needs to include calculation-intensive functions in this general purpose FPGA. The presence of 112 nine-bit DSP blocks (cascaded MAC's) guided thus our choice to implement powerful fixed point arithmetic in the design. A dedicated VHDL *package* exists to achieve this, developed by the VHDL reflexion group (*vhdl.org* or *eda.org*) for the next **IEEE** revision of VHDL. It is called *fixed_pkg_c*. This *package* adds a new VHDL data types with a signed (sfixed) and an unsigned (ufixed) fixed point type, constrained from the MSB (positive, natural part) to a negative LSB (number of bits after the comma). All the native VHDL operators are also overloaded to enable efficient data processing with these new data types. Furthermore, this *package* is completely synthesizable with any **IEEE** compliant synthesis tools.

4 Features

4.1 Global constants

In order to ease the use of this test functionality, several useful features are added, enabling one to synthesize the design without knowing its content. One of these features is given by the interface of the design with the designer. This unique interface is provided in a single file, a *package*, defining all the parameters to be propagated to the different blocks of the design.

This propagation is done through generic parameters in all the functional blocks of the design. A top *entity* brings all these blocs together in a global floor plan, and makes links between them. One parameter defined in this package is, for example, the buses width of the interconnection links, which coincide with the input and the output width of the concerned devices. Here is a listing of the global generic parameters.

```
package selfTest_amplitude_phase_4_channels_package is -- global design parameters
  constant running_sum_number : natural := 7; -- RS to process data on
  constant channel_data_width : natural := 40; -- decoder output width
  constant reference_data_width : natural := 16; -- reference data width
  constant internal_data_width : natural := 16; -- width of the division input
  constant number_of_symetric_channels : natural := 4; -- number of channels
  constant output_number_decimals: natural := 6; -- number of decimals
```

The shown values of the parameters here are examples, used in the test configuration of the design. These can be changed easily before the synthesis, to fit adequately the requirements of the final configuration. A maximal flexibility is thus available, inside the design to fit with the required precision, but also for the external interfaces to the BLMTC design.

4.2 Filter characteristics

In this section we will discuss about the filtering characteristics, and point out interesting details regarding this feature. As we have seen in 3.3, the filters are designed with the help of a new library which will be included in the new **IEEE** release of VHDL. This library enables the use of a new data type for fixed point representation, needed for the filters implementation. The same global constants method described previously is used to constrain the new data type inside the filters. These constraints are also in the head package.

```
-- arithmetic representation considerations (fixed point) for the filtering
constant filter_internal_MSB : integer := 21; -- fixed point MSB for data representation
constant filter_internal_LSB : integer := -14; -- fixed point LSB for data representation
constant filter_coef_MSB : integer := 1; -- fixed point MSB for filter coefficients
constant filter_coef_LSB : integer := -18; -- fixed point LSB for filter coefficients
```

Additionally, filters are usually defined by their coefficients, which are calculated with the sampling frequency, the right generating polynomial coefficients (Butterworth, Chebyshev...) and the cut off frequency. Again, to ease the use of this design, an effort has been made to eclipse the design structure, and to offer a unique, easy to use interface. Thus, the only parameter to specify is the input frequency :

```
-- frequency applied to the design (for filter cut off calculation)
constant F : real := 0.153; -- input Frequency (in Hz)
```

All the further parameters and coefficients are calculated from the previously given constants, during the compilation phase of the synthesis operation. For example, the sampling period is gathered from the running sum number through an array of frequencies, and the filter cut off frequency is given by the double (one octave above) of the input frequency. The filter coefficients are thus calculated statically with the definition of a second order Butterworth equivalent.

```
-- LEAVE UNTOUCHED, inherited constants:
 type running_sum_table is array(0 to 11) of real;
 constant RS_updates : running_sum_table := (0.00004, 0.00004, 0.00004, 0.00004,
  0.00008, 0.00008, 0.00256, 0.00256, 0.08192, 0.08192, 0.65536, 0.65536);
 constant Ts : real := RS_updates(running_sum_number);

-- LEAVE UNTOUCHED, filter parameters
 constant wc : real := (2.0*MATH_PI) * (2.0*F); -- pulsation
 constant C : real := 1.0 + ((sqrt(2.0)*2.0) / (Ts*wc)) + (4.0 / ((Ts*wc)*(Ts*wc)));
 constant b1 : real := 2.0 - (8.0 / ((Ts*wc)*(Ts*wc)));-- Calculated coefficients
 constant b2 : real := 1.0 - ((sqrt(2.0)*2.0) / (Ts*wc)) + (4.0 / ((Ts*wc)*(Ts*wc)));
 -- normalized coefficients :
  constant na1 : sfixed (filter_coef_MSB downto filter_coef_LSB) := To_sfixed (1.0/C,
   filter_coef_MSB, filter_coef_LSB);
  constant na2 : sfixed (filter_coef_MSB downto filter_coef_LSB) := To_sfixed (2.0/C,
   filter_coef_MSB, filter_coef_LSB);
  constant na3 : sfixed (filter_coef_MSB downto filter_coef_LSB) := To_sfixed (1.0/C,
   filter_coef_MSB, filter_coef_LSB);
  constant nb1 : sfixed (filter_coef_MSB downto filter_coef_LSB) := To_sfixed (b1/C,
   filter_coef_MSB, filter_coef_LSB);
  constant nb2 : sfixed (filter_coef_MSB downto filter_coef_LSB) := To_sfixed (b2/C,
   filter_coef_MSB, filter_coef_LSB);
```

4.3 Synthesis

This self test functionality is aimed to be an add-on to the measurement electronics already existing in the FPGA. A focus should therefore be made on the different inputs and outputs of the device. The only interfaces to process the test are given by the running sum operations. Thus, the input is an complete array of 12 buses (one for every sum) to be connected to the measurement outputs, including the 6 *enable* signals. Only the relevant signals (corresponding to the chosen sum) will be propagated through the decoder. A user definable width bus is present, with a *Ref_valid* signal to notify when a new value is present. This signal will be used as synchronization signal for the whole functionality. A *test_clock* is also present and a *test_enable* to switch the test on, typically when no beam is present is the ring.

Concerning the outputs, no definitive configuration has for the moment been specified. Nevertheless, to realize the relevant tests to validate the good working of this functionality, 8 busses giving 4 gain and 4 phase values are considered as outputs. The basic gain and phase measurement functionality is thus validated. The action of decision whether or not the acquisition chain is valid has still to be discussed. This should output a single flag per channel to send to the global interlock system.

To notify errors during the calculation of the gain or the phase value, an error code generator has been included. The codes are the following :

1. ”100...0” for channel counter overflow
2. ”110...0” for reference counter overflow
3. ”111...0” for division error (division by 0 error)

Furthermore, the position of the error code in the output busses indicates the channel on which the error occurred and for which measurement (gain or phase).

To achieve a complete measurement cycle for the 4 consecutive channels, the control unit has to process its algorithm with a strict respect of the sequencing (data availability, control signals, etc.). Because of this dependency of each blocks with the others, the best way to test the good working of the functionality is to analyze this control unit sequence with a signal tap.

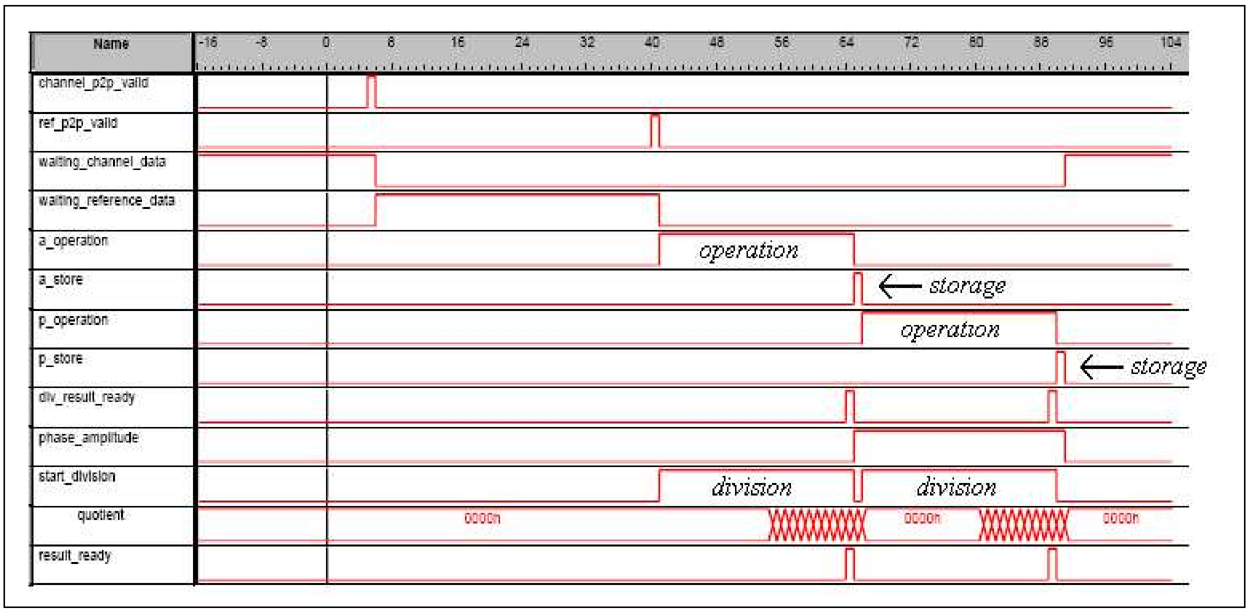


Figure 4.1: *Calculation sequence - Control unit*

An interesting point is to calculate the number of clock cycles needed to produce a couple of gain/phase values. This value is fixed and dependent of the number of relevant digits of the result. For the precision used in the test configuration (16 bits, 6 bits after the comma) the number of clock cycles needed is : $2 + 2 * (16 + 6) = 46$ clock cycles. The first ’2’ is the number of clock cycles to needed to store the gain and phase values in the output registers.

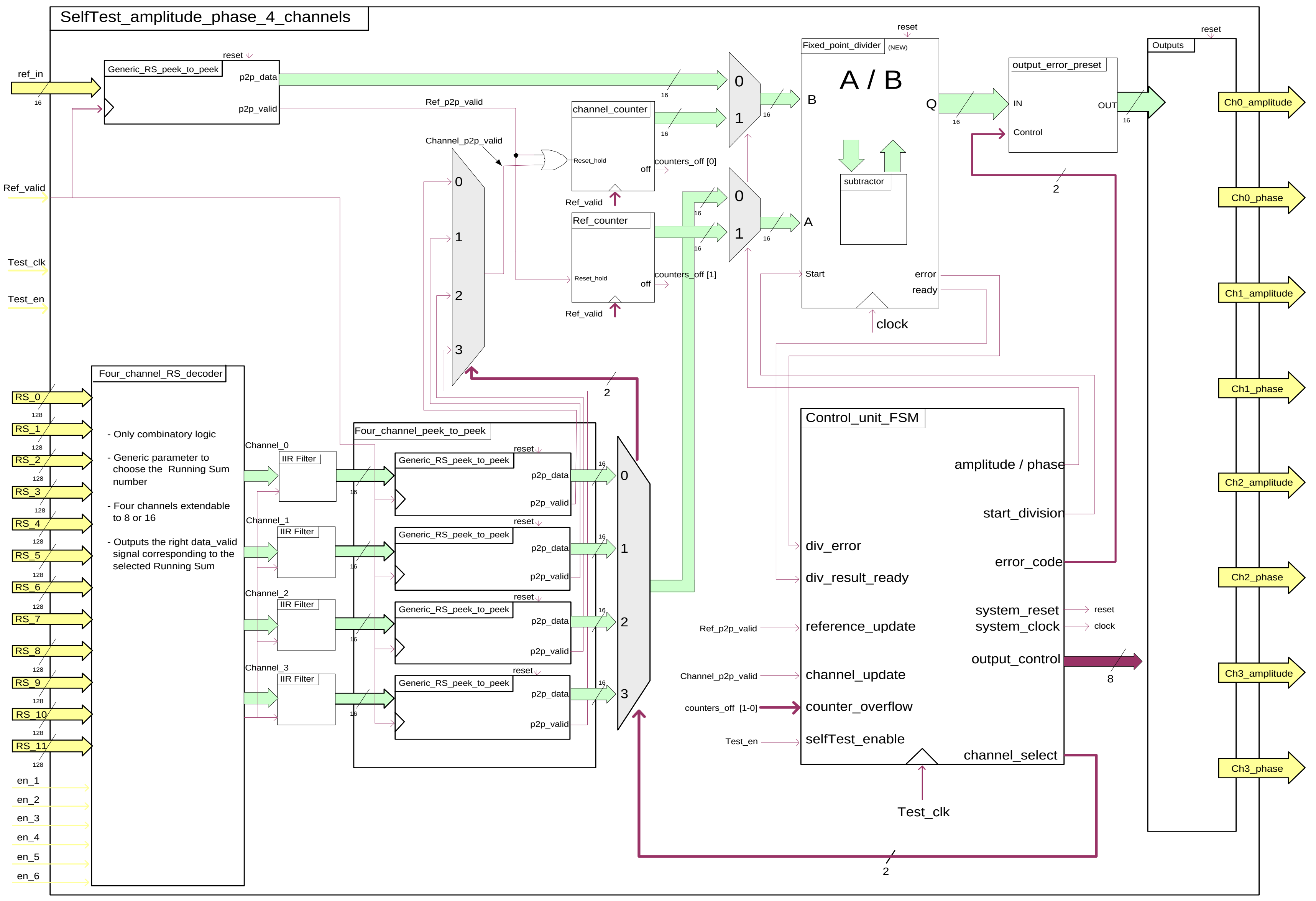
5 Conclusion

This technical student work took about four and a half month to be completed, with the active help of **Christos Zamantzas** and **Jonathan Emery** and under the supervision of **Bernd Dehning**. It has then been actively tested, and is completely functional in the present state, regarding the gain and phase values production. The magnitude comparators are not implemented yet because the circuit for production of the local beam permit signal has still to be discussed. The two different possibilities considered are the basic value comparison and the successive differential comparison with a threshold.

To ensure a valuable follow up of the system designed after the time limited technical student contract, this document has been written, in addition to a whole ”rapport de stage” (in French). This document is available through **Bernd Dehning** or **Jonathan Emery**. These two people are to be contacted in case of doubt or problem concerning this design.

The design of this self test functionality is focused on a delivery after 5 months. The approach of a harmonic analysis has thus been considered from the beginning, because it seemed to be the simplest way to measure whether the acquisition chain is valid or not. If a more precise characterization of the chain characteristics is needed, through a Fourier analysis of a pseudo random stimulation signal for example, only a few blocks are needed to be changed. This design is made in such a modular way that it can be changed easily to receiver a large range of measurement possibilities.

A Design RTL view



Four channel Amplitude / Phase extractor rev_6 - Beam Loss Monitoring - Erik Verhagen - Last update : 18/08/2006

B Butterworth IIR filter

B.1 Filter design

1. Canonical Form :

$$\boxed{H(s) = \frac{1}{1 + \sqrt{2}.\frac{s}{\omega_c} + (\frac{s}{\omega_c})^2}} \tag{B.1}$$

2. Poles and zeros :
   (a) No zeros
   (b) 2 analog poles :

$$\Delta = \frac{2}{\omega_c^2} - \frac{4}{\omega_c^2} = -\frac{2}{\omega_c^2} = \left(j.\frac{\sqrt{2}}{\omega_c}\right)^2 \Longrightarrow P_{1,2} = \frac{\frac{\sqrt{2}}{\omega_c} \pm j.\frac{\sqrt{2}}{\omega_c}}{2} = -\frac{\sqrt{2}}{2.\omega_c}.(1 \pm j)$$

3. Stability :

$$\boxed{\Re\{p_{1,2}\} = -\frac{\sqrt{2}}{2.\omega_c} < 0, \forall\, \omega_c} \qquad \Longrightarrow Always\ stable \tag{B.2}$$

4. Bilinear transform :

$$\boxed{p = \frac{2}{T_s}.\frac{1 - z^{-1}}{1 + z^{-1}}}$$

$$\begin{aligned} H(z) =& \frac{1}{\left[\frac{2}{T_s.\omega_c}.\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + \frac{\sqrt{2}}{\omega_c}.\left[\frac{2}{T_s}.\frac{1-z^{-1}}{1+z-1}\right] + 1} \\ =& \frac{(1+z^{-1})^2}{\left[\frac{2}{T_s.\omega_c}.(1-z^{-1})\right]^2 + \frac{\sqrt{2}}{\omega_c}.\left[\frac{2}{T_s}.(1-z^{-1}).(1+z^{-1})\right] + (1+z^{-1})^2} \\ =& \frac{1 + 2.z^{-1} + z^{-2}}{\left[\frac{2}{T_s.\omega_c}\right]^2.(1 - 2.z^{-1} + z^{-2}) + \left[\frac{2.\sqrt{2}}{T_s.\omega_c}\right].(1 - z^{-2}) + (1 + 2.z^{-1} + z^{-2})} \\ =& \frac{z^{-2} + 2.z^{-1} + 1}{b_1.z^{-2} + b_2.z^{-1} + C} = \frac{Y(z)}{X(z)} \end{aligned} \tag{B.3}$$

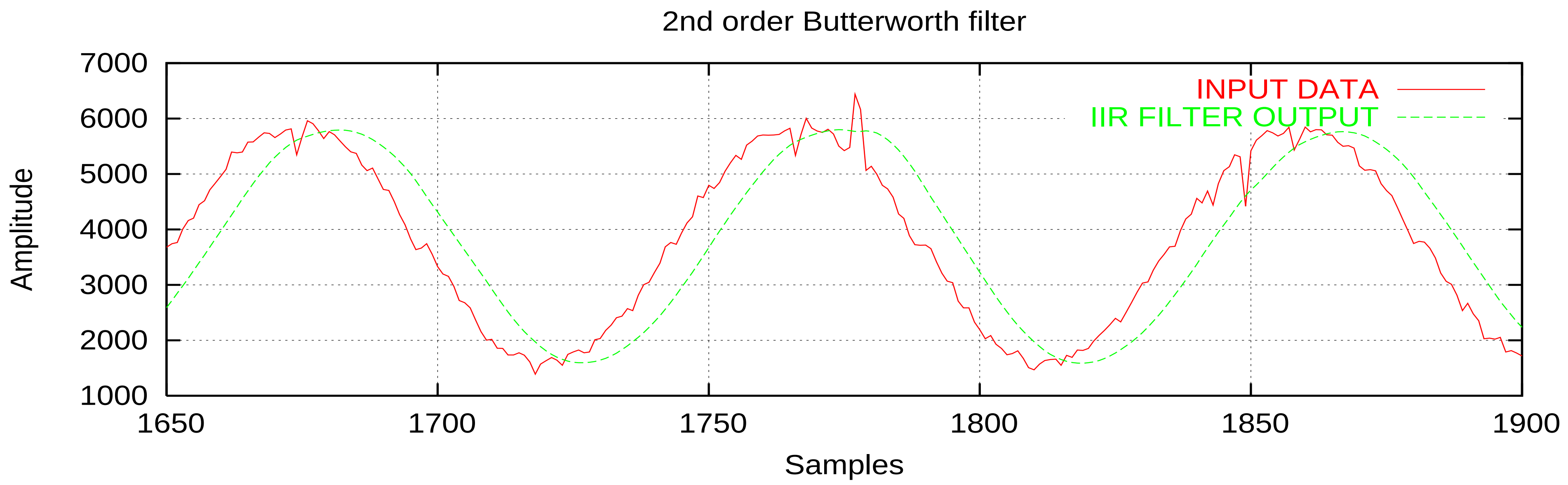
$$\boxed{b_1 = \left(\frac{2}{T_s.\omega_c}\right)^2 - \frac{2.\sqrt{2}}{T_s.\omega_c} + 1} \quad \boxed{b_2 = -\left(\frac{2.\sqrt{2}}{T_s.\omega_c}\right)^2 + 2} \quad \boxed{C = \left(\frac{2}{T_s.\omega_c}\right)^2 + \frac{2.\sqrt{2}}{T_s.\omega_c} + 1}$$

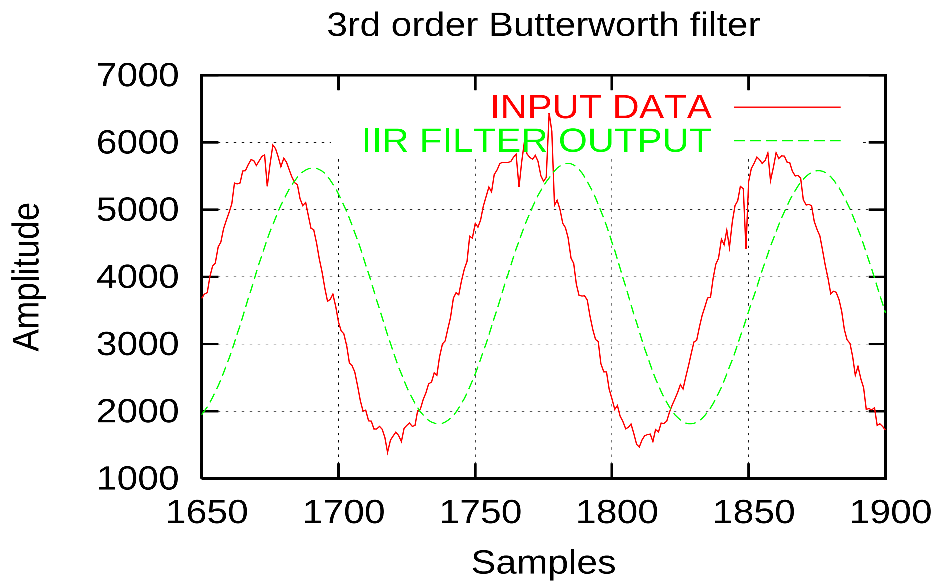
5. Digital equivalent ([Z]$^{-1}$ transform)

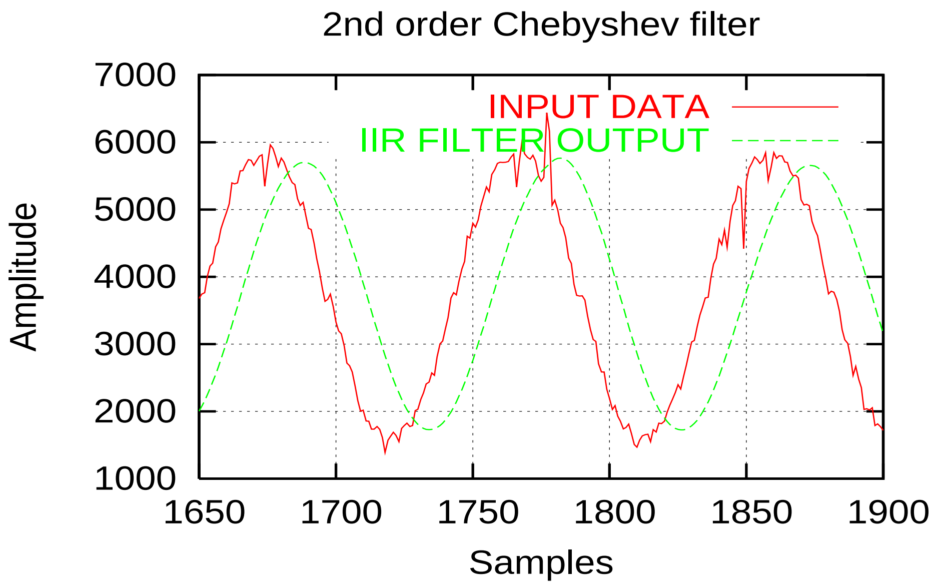
$$X(z).[z^{-2} + 2.z^{-1} + 1] = Y(z).[b_1.z^{-2} + b_2.z^{-1} + C]$$

$$\Longrightarrow \boxed{y(n) = \frac{x(n) + 2.x(n-1) + x(n-2) - b_2.y(n-1) - b_1.y(n-2)}{C}} \tag{B.4}$$

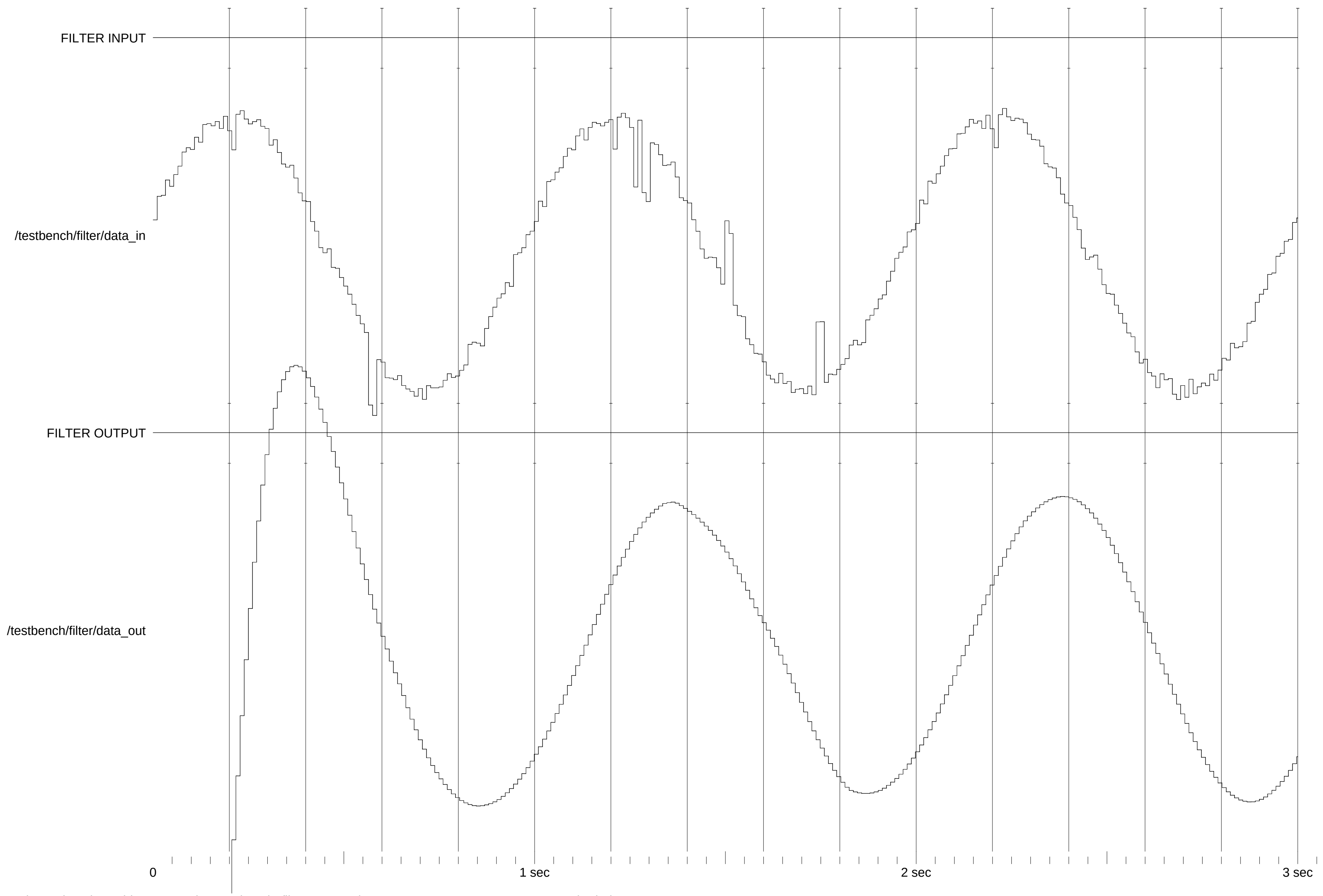
B.2 Filter comparison result (C)





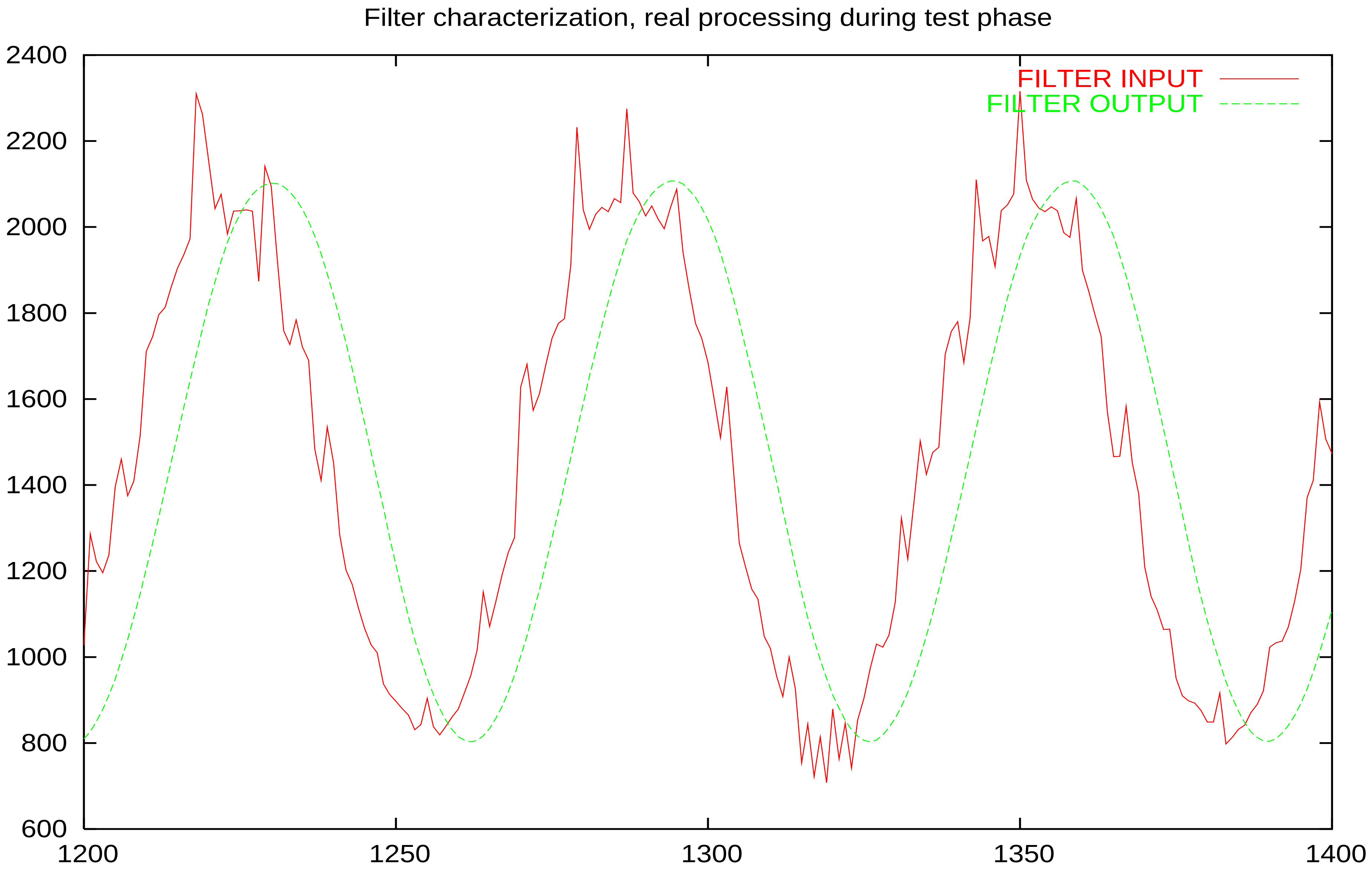


B.3 Filter simulation result (*VHDL*)



Entity:testbench Architecture:testing_testbench_filter Date: Fri Aug 18 10:45:51 AM W. Europe Standard Time 2006   Row: 1 Page: 1

B.4 Filter test result (*Signal tap*)


Filter characterization, real processing during test phase
FILTER INPUT
FILTER OUTPUT