U.Porto Journal of Engineering, 5:2 (2019) 38-45
ISSN 2183-6493
DOI: 10.24840/2183-6493_005.002_0005

Received: 7 December, 2018
Revised: 7 January, 2019
Published: 29 November, 2019

Evaluation of Stuck-at Faults in a FIR Filter Design

Helder H. Avelar[1]

[1]Departamento de Engenharia Eletrotécnica e Computadores, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal (up201809138@fe.up.pt) ORCID 0000-0001-5018-4651

The development of integrated circuits becomes more complex and gets more susceptible to manufacturing errors as technology nodes get smaller. Along with that, the observability of more complex chips gets reduced, making it harder and more expensive to test. Therefore, the impact of different faults must be observed on the design as early as possible, in order to reduce the time to market and the cost of the fault to the project.
Since many designs depend on Digital Signal Processing applications, this project proposes a case study of the implementation of a FIR filter design in Verilog and the analysis and comparison of its response by inserting stuck-at faults in its shift registers via test bench simulation.

Author Keywords. DSP, Digital Filters, Fault Tolerance, Stuck-at Faults.

Type: Research Article
Open Access Peer Reviewed CC BY

1. Introduction

Current integrated circuit (IC) design technology allows us to simply describe a digital device in hardware description language (HDL) and then synthesize it using electronic design automation to obtain a chip. This method, known as standard cell flow, has made the development of ICs very efficient and reliable. However, with the miniaturization of electronic components, dictated by Moore's Law (Moore 1965), the sensitivity of digital circuits to faults has increased considerably.

Faults are incorrect operations on systems, and may be caused by design errors, manufacturing defects, or external interference. They may be permanent or transient. Permanent faults are those where, once the component fails, it never works correctly again. Meanwhile, transient faults are those that have limited duration, caused by a temporary malfunction or by some external interference. Such faults may also be intermittent, occurring repeatedly for short intervals of time (Schivittz et al. 2016). There are two main models of permanent faults in digital systems, which will be considered in this work: Stuck-At-1, when a node is always at 1, and Stuck-At-0 when a node is always at 0 (Weste and Harris 2011).

Digital Signal Processing (DSP) is used to transform and manipulate analog and digital systems. With the rapid development of digital technologies, its use is increasingly broad, replacing analog signal processing in many applications. Digital systems have several advantages, such as the lower temperature sensitivity, aging and fault tolerance (Weste and Harris 2011).

One of the main applications of DSP is filtering, which makes the study of digital filters very important. There are two main classifications for this type of filter: filters with a finite impulse response (FIR) have a finite number of iterations, reducing to a finite sum for each output sample of the signal; In contrast, filters with an infinite impulse response (IIR) require the calculation of an infinite sum, using feedback.

One of the main reasons for the study of digital filters is their great popularity in DSPs. They are quickly replacing analog filters because of their ease of implementation, which can usually be done through algorithms and computational specifications (Jiang et al. 2015).

The increase in the number of permanent failures in the digital circuits makes an analysis of the effect of these failures on filters of this type relevant. Therefore, in this work, a FIR filter design was developed in a computational tool and implemented in a hardware description language. This circuit was be simulated, inserting faults in its operation, to analyze the variation of the results. The analysis was be carried out in two stages and in two different ways. First, an insertion of bit-shift faults in the filter coefficients will be done and its influence on the response of the circuit verified in both the frequency domain and the time domain. Then an insertion of Stuck-At-1 type faults will be made in different parts of the circuit, verifying their output in the time domain.

The objective of this work is to compare the errors obtained with insertion of faults at different points in the circuit, checking if there is a need for complex methods for fault tolerance in these systems and if there are parts of the circuit in which it is possible to abdicate these methods.

The presentation of this study will be done as follows: In section 2 will be presented more details about digital filters. Section 3 will be devoted to the methodology used in the project and section 4 to the analysis of the results obtained. Finally, the conclusions will be presented in section 5.

2. Digital Filters

Filters are responsible for the total or partial suppression of the characteristics of a signal and are therefore responsible for the filtering process in communication systems. Generally, this process is linked to the suppression of certain frequency bands that are interfering in the main signal, but not limited to that. There are several types of filters, the main ones being the filters that control the range of frequencies that must pass (Haykin and Veen 2003). The main disadvantage of using filters is loss of information inherently associated with them.

The most common filter is the linear time-invariant filter (LTI). An LTI filter interacts with its input signals by a process called linear convolution, represented by an asterisk, as shown in Equation 1, where *f* is the impulse response of the filter, *x* is the input signal and *y* is a convolution of the signals.

$$\mathrm{y} = \mathrm{f} * \mathrm{x} \tag{1}$$

Equations 2 and 3 present the formal definition of the convolution process, where *n* is the time index; then x[n] denotes a signal, while x[k] represents the value of x[n] at time k (Meyer-Baese 2007).

$$\mathrm{y[n]} = \mathrm{x[n]} * \mathrm{f[n]} = \sum_{\mathrm{k}} \mathrm{x[k]f[n-k]} \tag{2}$$

$$\mathrm{y[n]} = \mathrm{x[n]} * \mathrm{f[n]} = \sum_{\mathrm{k}} \mathrm{x[k]f[n-k]} \tag{3}$$

Digital LTI filters are generally classified as FIR or IIR. The FIR filter has a finite number of samples (n finite), reducing Equations 2 and 3 to a finite sum of products for each signal output sample. In contrast, the IIR requires the calculation of an infinite sum (*n = ∞*).

A constant coefficient FIR filter is a LTI digital filter. The output of a FIR filter for a given time input *x[n]* is given by a finite version of Equation 2, given by Equation 4, where *f[0] ≠ 0* to *f[L-*

1] ≠ 0 are the L coefficients of the filter and also correspond to the impulsive responses of the filter.

$$y[n] = x[n] * f[n] = \sum_{k=0}^{L-1} f[k]x[n-k] \quad (4)$$

For LTI systems, it is more convenient to express Equation 4 in Z- domain, by Equation 5, where F(z) is the transfer function of the FIR filter defined by Equation 6.

$$Y(z) = F(z)X(z) \quad (5)$$

$$F(z) = \sum_{k=0}^{L-1} f[k]z^{-k} \quad (6)$$

An important property of FIR filters is that they can perform a linear phase frequency response. Recognizing that a linear phase response corresponds to a constant delay, the problem of approaching the FIR filter design becomes much simplified. The L-th order of a FIR LTI filter can be interpreted as a set of adders and multipliers in a row of tap delays, according to Figure 1. One of the operands presented to each multiplier is the filter coefficient, known as the tap. The roots of the polynomial *F(z)* in Equation 6 define the zeroes of the filter. This filter is called the Direct FIR Filter (Meyer-Baese 2007).

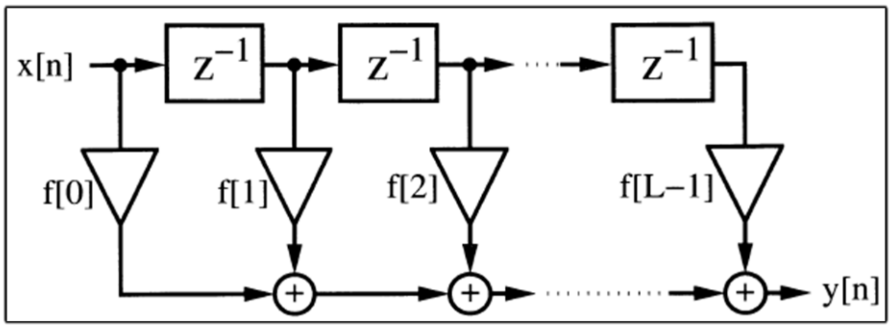

Figure 1: Direct FIR Filter

3. Methodology

A Direct FIR Filter design was developed to evaluate the effects of stuck-at faults in the system. Two different softwares were used in this analysis in complementary ways. For this reason, the study was divided into two parts. First a study of the FIR Filter in MATLAB and then the Verilog implementation was performed and simulated in Cadence Incisive.

3.1. Study of a FIR filter in MATLAB

MATLAB offers a complete toolbox for DSP analysis. In this case, Filter Design & Analysis tool (fdatool) was used. Fdatool allows the creation of a filter based on its characteristics, such as pass and stop frequencies, number of taps, signal attenuation and frequency of sampling. The toolbox also allows the reconstruction of a filter from its coefficients and the type of filter to be constructed, thus allowing to check if the response of the filter keeps unchanged.

3.2. Implementing the FIR filter in Verilog

Having the coefficients of the filter, the other data become unnecessary for its programming, which consists of a sum of products. First, a block diagram of the circuit was drawn (Figure 2), so that it can be described as shown in the diagram. The taps are shift registers, whose values are rotated each clock cycle, modifying the final product sum. Those can be made using flip-flops. It was chosen to describe the circuit in Verilog, due to the possibility of simulating the physical characteristics of the filter implemented. The simulation of the described hardware can be done by means of a test bench, which can also be implemented in Verilog. The bench test is responsible for generating input signals and checking for output signals from other hardware. For the analysis of the system, Cadence Incisive was used for elaboration and simulation of the code. Finally, SimVision was used for graphical simulation of the design.

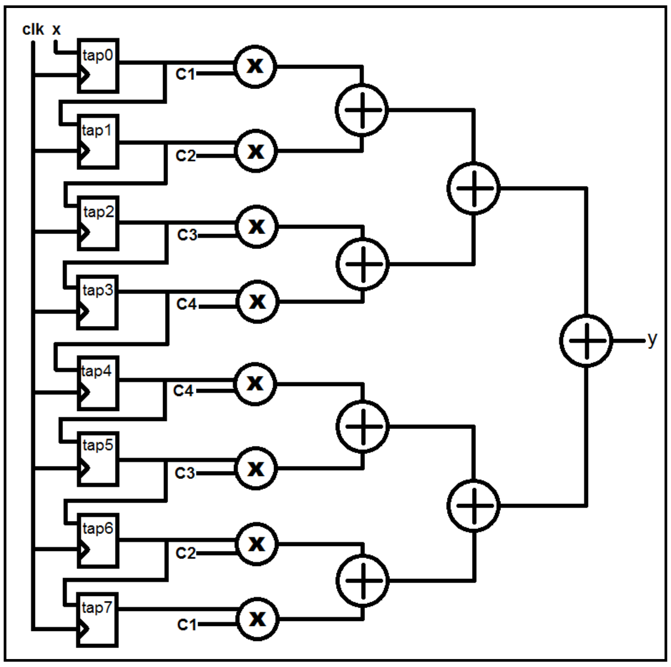

Figure 2: Block diagram of the circuit

4. Results and Discussion

A direct FIR filter was developed, whose frequency response is shown in Figure 3 and whose coefficients obtained in MATLAB are in Table 1. This filter has Fpass of 50 kHz and Fstop of 450 kHz. The sampling rate considered was 2 MHz.

To facilitate the implementation of the filter in Verilog, the coefficients were transformed into integer numbers. This transformation is done by the simple product of all coefficients by an integer constant. This process causes the output to be multiplied by this constant, so in the end of the process of filtering a division at the output is needed to correct the difference. The multiplier chosen was 4096, because it was a power of 2 that allowed to discard the fractional digits of the coefficients without causing significant loss of the characteristic of the filter. The use of a power of two will make it easier to bring the signal down to its correct value without the gain in hardware (division by powers of two can be made through shifting). Table 1 shows the fractional and integer coefficients for the filter. Since the largest value among the coefficients has 10 bits, that was the bit width considered for the coefficients. For data, 16 bits were used. For verification, the filter was reconstructed in MATLAB using its integer coefficients. It can be confirmed that the filter operates as previously described in MATLAB, as shown in Figure 4.

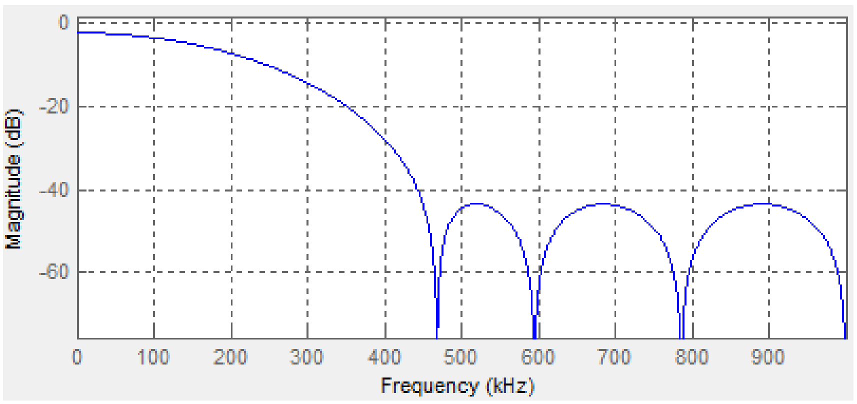

Figure 3: Frequency response of the Filter (Golden Model)

Coefficient Index	Obtained Value	Integer Value
0	0.0332	136
1	0.0996	408
2	0.1846	756
3	0.2451	1004
4	0.2451	1004
5	0.1846	756
6	0.0996	408
7	0.0332	136

Table 1: Filter Coefficients

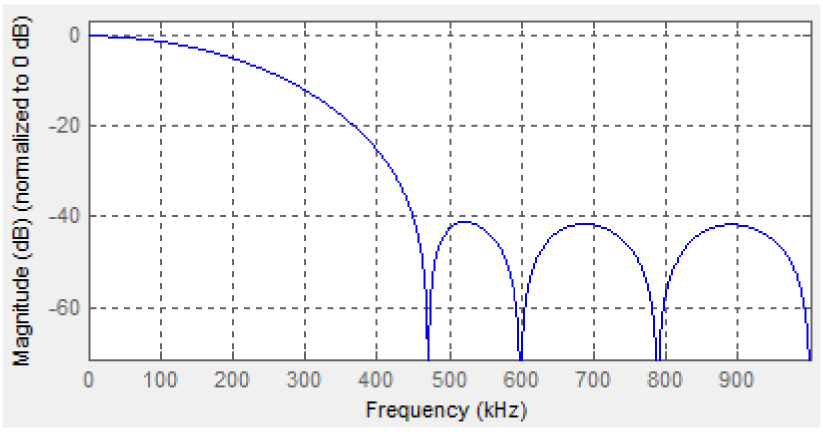

Figure 4: Frequency response of the reconstructed Filter with integer coefficients

Having the coefficients of the filter, it was possible to implement its Verilog model. The model allows an analysis in the time domain of the developed filter, verifying what occurs with data of different frequencies when filtered and when there are faults in the circuit. In Figure 5, the sine wave between 0 and 62 samples has a frequency of 40 kHz, so it passes completely without any attenuation. Between 63 and 92 samples, the wave has a frequency of 300kHz. It can be seen on the output that this wave is partially attenuated. Finally, between 93 and 123 clock cycles there is a high frequency wave, with 500kHz. As can be seen, this wave has maximum attenuation. From this point on, the low-frequency wave is added to the mid-wave and later to the high-frequency wave. Note again that the shape of the low frequency wave remains, that of the medium frequency is partially attenuated and the high frequency fully attenuated. After this test, the fault analysis step in the circuit is started.

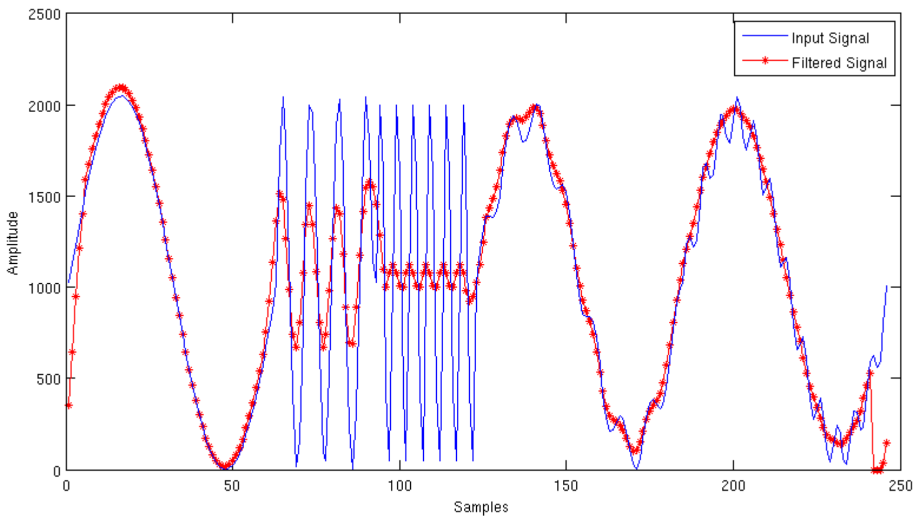

Figure 1: Input and filtered signal using the Golden Model of the filter

4.1. Analysis of the amplitude response of the filter with faults in different parts of the circuit

Single stuck-at-0 and stuck-at-1 faults were simulated in the coefficients and the inputs of the filter taps and the response in the time domain was observed using a bench test. The input waveforms used for verification of the golden model were reused at this stage and the analysis was done in two ways. The first was the verification of how such a fault influences the output of the circuit, depending on the bit in which it occurs in a single tap. Figure 6 shows the result obtained when the faults are inserted in different bits of tap 3, compared to the input. Tap 3 was chosen for this part of the experiment because it is in the middle of the filter. When the fault occurs on the least significant bits, its effects are attenuated by the filter. However, when the fault happens on bit 9 the error in the output becomes significant, because the noise frequency is too high, completely losing the characteristics of the original signal. This is because the fault only causes error when the bit in question has a one in the position where the fault is. In the first bits this oscillation has high frequency, which causes it to be filtered like any high frequency noise. As the bit index increases, this frequency decreases to medium and low and is no longer attenuated. Simulations were repeated for stuck-at-1 faults, and the results can be seen in Figure 7. It is important to observe the effect of the stuck-at-1 fault in bit 11. There is only the effect of shifting the curve in the *y* axis, because its maximum value has only 10 bits, which does not cause the curve to lose its format and frequency response when it contains faults in higher bits but causes this offset. Of course, this effect is dependent of the bit width of the input data. Another evaluation, shown in Figure 8, involved the analysis of stuck-at-0 faults in the same bit at different taps. The bit 8 was chosen because it is a central bit and has a strong influence on the response of the filter. Since the taps are shift registers, an error that occurs in one of them eventually propagates to the others, which causes smaller index taps to have greater influence on the filter response. Noting that FIR filters can have much higher orders, up to more than 100 taps, this fault propagation can have strong influence on the filter response.

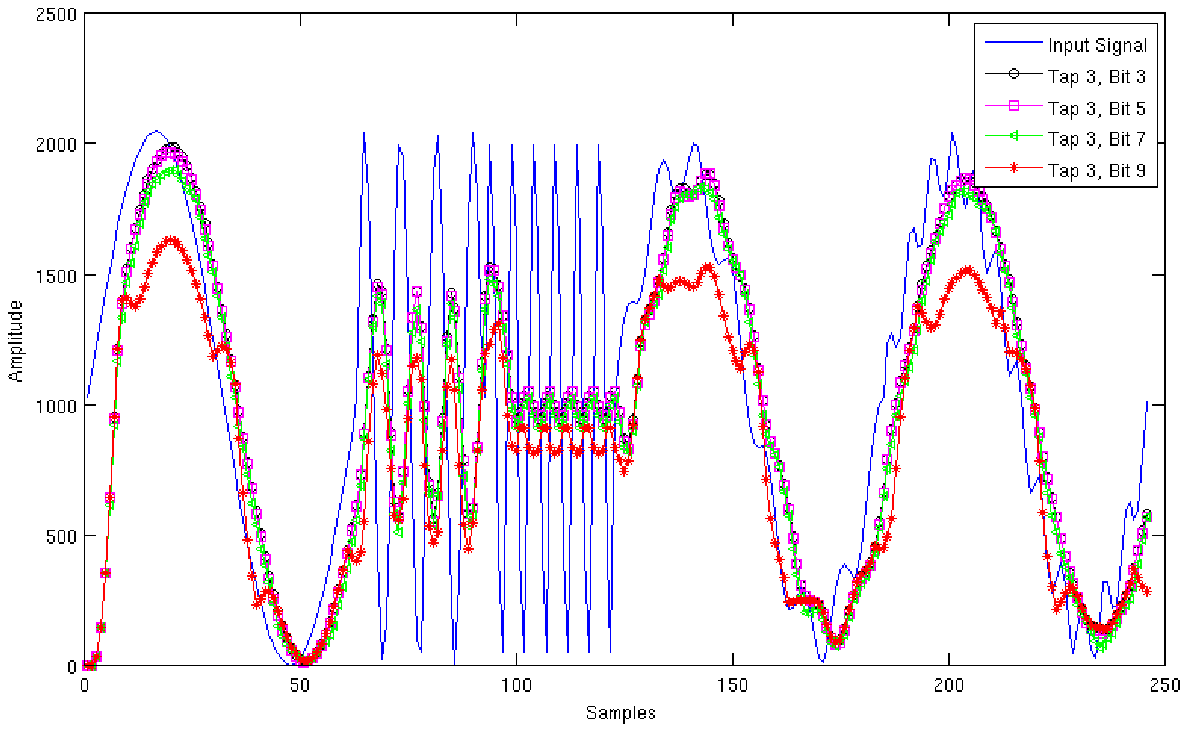

Figure 6: Effect of stuck-at-0 faults in different bits of the Tap3

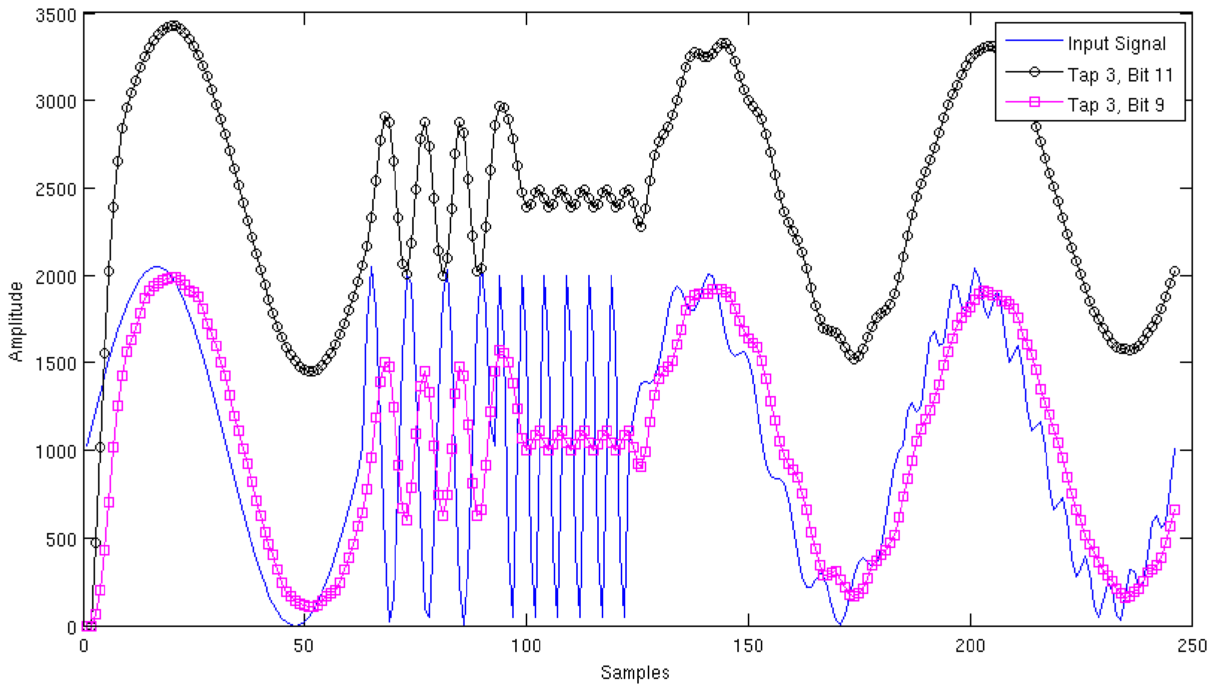

Figure 7: Effect of stuck-at-0 faults in different bits of the Tap3

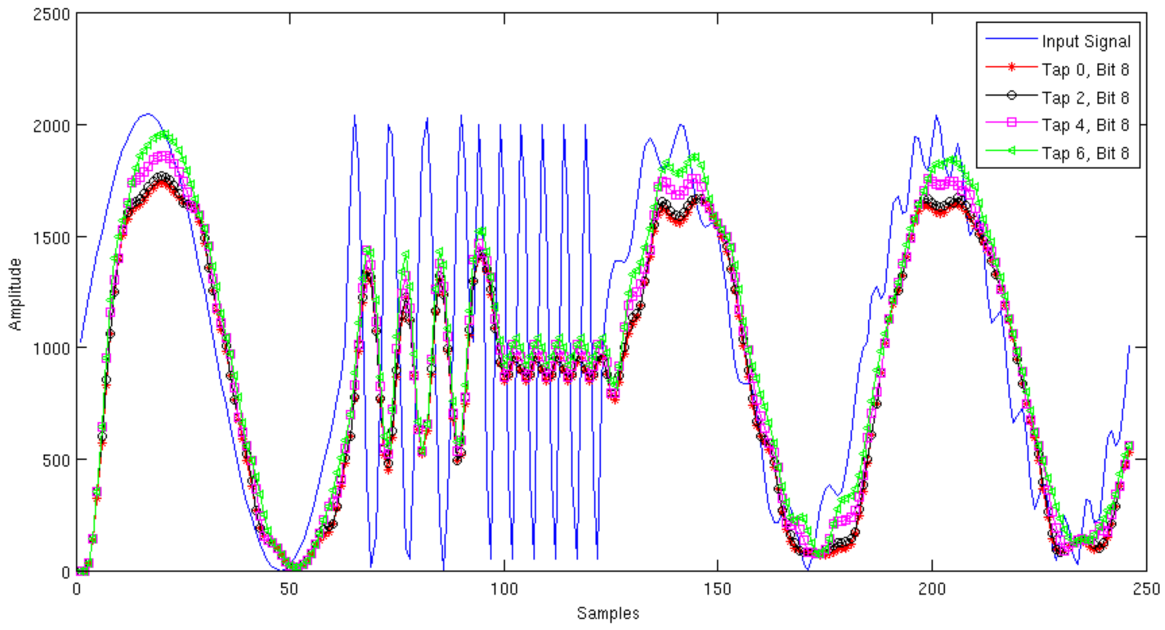

Figure 8: Effect of stuck-at-0 faults in the bit 8 of different taps of the filter

5. Conclusions

The study of the response of the FIR filters to different fault types proved to be very important for the practical development of a filter. After all, there are several variables involved in one filter and the relevance of the faults changes strongly, depending on where they occur.

To avoid faults, it is necessary to apply tools that increase the robustness of the filter, ensuring that any faults do not affect its operation. This type of tools demands area on the chip where the filter is developed and often even doubling the hardware. On the other hand, in situations where the filter itself masks the fault, this is not necessary, for example, when the faults generate high frequency noise that is filtered and thus masked.

In addition, there are faults that, despite generating quite relevant errors, can be easily identified. As an example, you can see data faults at taps. All data going through a tap will have to have passed through a previous place of the circuit, which means a parity check is enough to guarantee the robustness. Meanwhile, for adder and multiplier faults, error checking may require greater care. This work serves as a basis for more in-depth studies of the robustness of filters and several future works can be performed, such as a generalization of the problem for different kinds of filters.

References

Haykin, S., and B. V. Veen. 2003. *Sinais e sistemas*. Porto Alegre: Bookman.

Jiang, A., H. K. Kwan, Y. Zhu, X. Liu, N. Xu, and Y. Tang. 2015. "Design of sparse FIR filters with joint optimization of sparsity and filter order". *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, no. 1 (january): 195-204. https://doi.org/10.1109/TCSI.2014.2354771.

Meyer-Baese, U. 2007. *Digital signal processing with field programmable gate arrays*. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-72613-5.

Moore, Gordon E. 1965. "Cramming more components onto integrated circuits". *Electronics* 38, no. 8: 114.

Schivittz, R. B., D. T. Franco, C. Meinhardt, and P. F. Butzen. 2016. "A probabilistic model for stuck-on faults in combinational logic gates". In *2016 17th Latin-American Test Symposium (LATS)*, 39-44. https://doi.org/10.1109/LATW.2016.7483337.

Weste, N. H. E., and D. M. Harris. 2011. *CMOS VLSI design: A circuits and systems perspective*. 4th ed. Boston: Addison Wesley.